Hunting for Windows “Features” and How to use Them

Chris Spehn
Adversary Simulation, IBM X-Force Red

X-Force Red

Once Upon A Time

“I expected you to use a 0day to bypass our EDR product” -Client

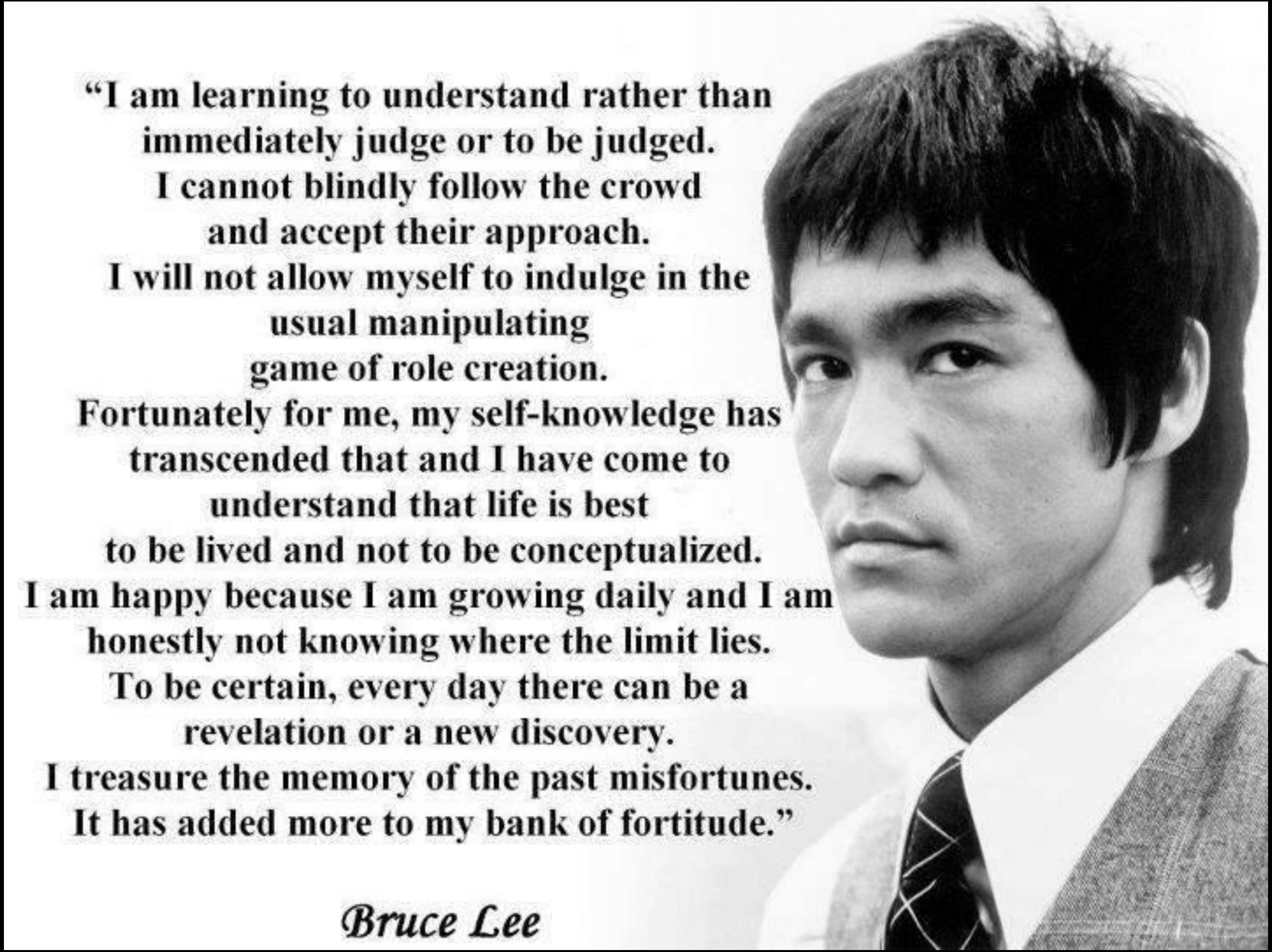
ARE YOU
SERIOUS RIGHT NOW
makeameme.org

WHAT THATS
IMPOSSIBLE
makeameme.org

I bet he's thinking of other women
How would I find and use a 0day on an adversarial simulation engagement?
imgflip.com

WHAT IF WE COULD
USE A WINDOWS "FEATURE"
imgflip.com

Who Am I?

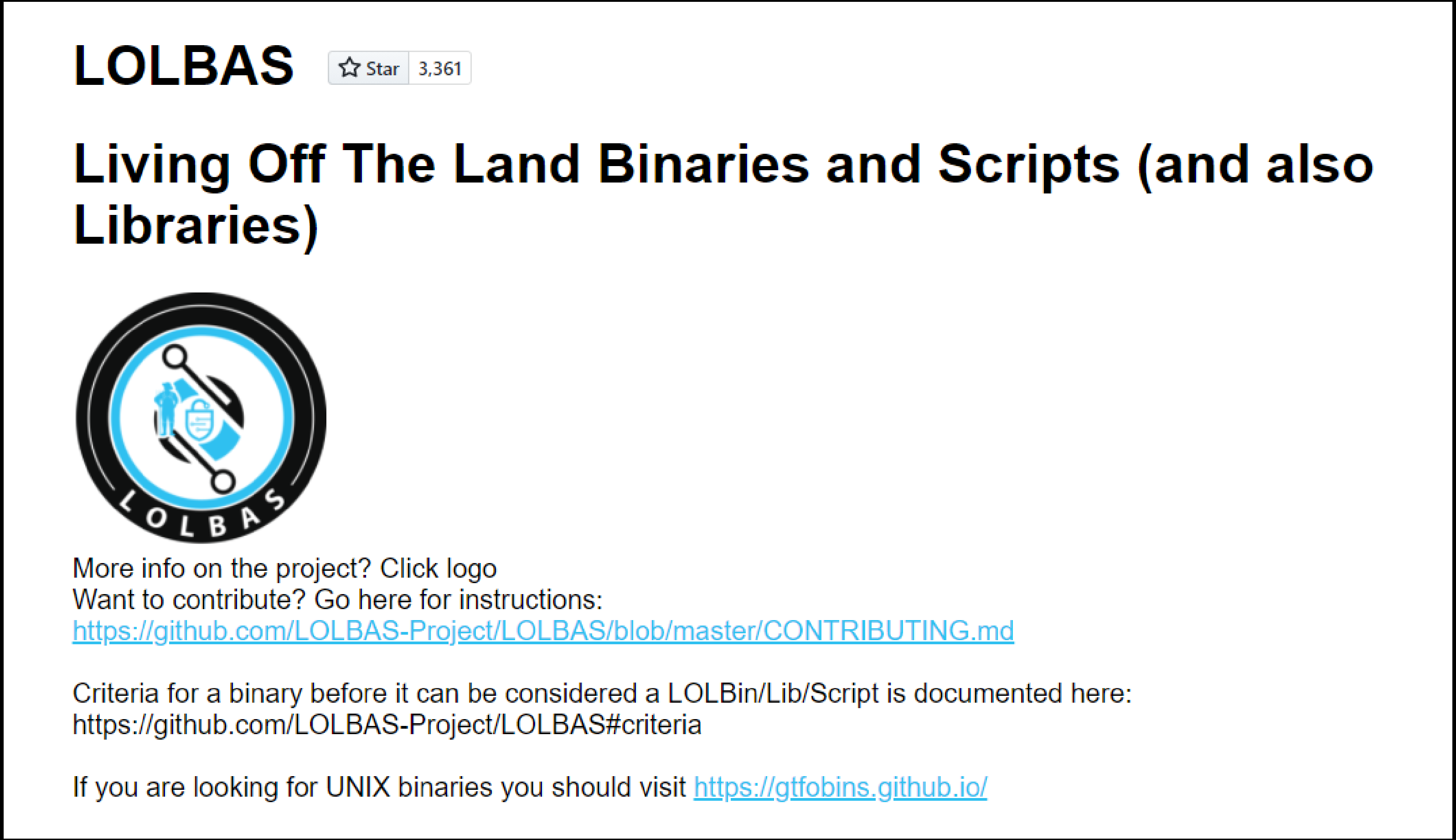


"I am learning to understand rather than immediately judge or to be judged.
I cannot blindly follow the crowd and accept their approach.
I will not allow myself to indulge in the usual manipulating game of role creation.
Fortunately for me, my self-knowledge has transcended that and I have come to understand that life is best to be lived and not to be conceptualized.
I am happy because I am growing daily and I am honestly not knowing where the limit lies.
To be certain, every day there can be a revelation or a new discovery.
I treasure the memory of the past misfortunes.
It has added more to my bank of fortitude."

Bruce Lee

Red Team Love

CHRIS SPEHN PRESENTS
GREATSCT
GOTTA CATCH'EM AWL



LOLBAS

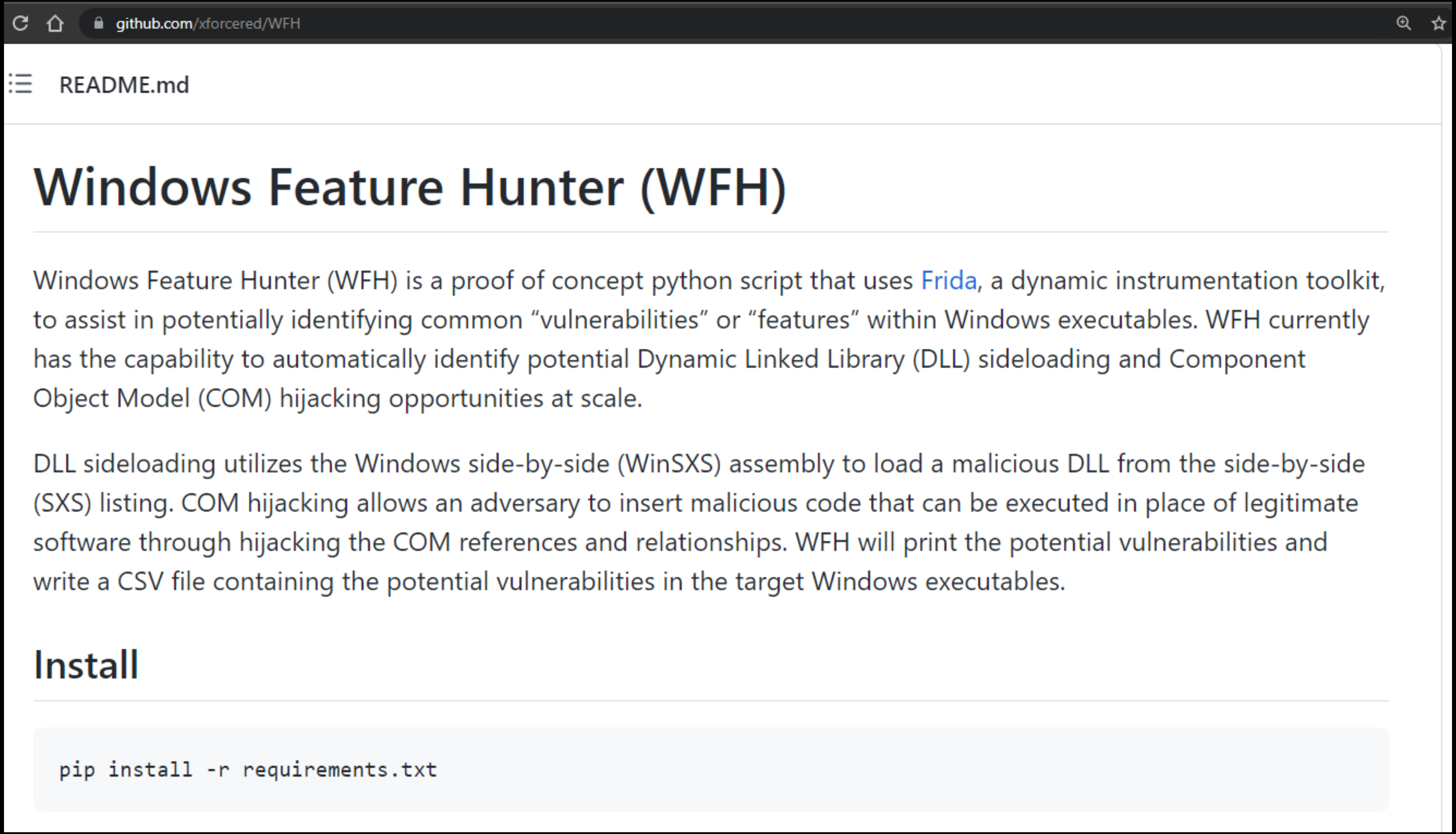
Star 3,361

Living Off The Land Binaries and Scripts (and also Libraries)

More info on the project? Click logo
Want to contribute? Go here for instructions:
https://github.com/LOLBAS-Project/LOLBAS/blob/master/CONTRIBUTING.md

Criteria for a binary before it can be considered a LOLBin/Lib/Script is documented here:
https://github.com/LOLBAS-Project/LOLBAS#criteria

If you are looking for UNIX binaries you should visit https://gtfobins.github.io/



Windows Feature Hunter (WFH)

Windows Feature Hunter (WFH) is a proof of concept python script that uses Frida, a dynamic instrumentation toolkit, to assist in potentially identifying common “vulnerabilities” or “features” within Windows executables. WFH currently has the capability to automatically identify potential Dynamic Linked Library (DLL) sideloading and Component Object Model (COM) hijacking opportunities at scale.

DLL sideloading utilizes the Windows side-by-side (WinSXS) assembly to load a malicious DLL from the side-by-side (SXS) listing. COM hijacking allows an adversary to insert malicious code that can be executed in place of legitimate software through hijacking the COM references and relationships. WFH will print the potential vulnerabilities and write a CSV file containing the potential vulnerabilities in the target Windows executables.

Install

```
pip install -r requirements.txt
```

Professional Experience

DISCOVER®

Trustwave®
SpiderLabs®

MasterCard®

®

Schnucks®

MANDIANT

X-Force Red

Windows “Features”

MITRE T1574.002: DLL Sideloading

Adversaries may execute their own malicious payloads by side-loading DLLs. Similar to DLL Search Order Hijacking, side-loading involves hijacking which DLL a program loads. But rather than just planting the DLL within the search order of a program then waiting for the victim application to be invoked, adversaries may directly side-load their payloads by planting then invoking a legitimate application that executes their payload(s).

Reference: https://attack.mitre.org/techniques/T1574/002/

Microsoft: DLL Sideloading

Microsoft also wrote a blog post where they define what is considered a vulnerability, saying that CWD scenarios would be addressed with a security fix, while PATH directory scenarios would not, “since there can’t be a non-admin directory in the PATH, [it] can’t be exploited.”

Reference: https://msrc-blog.microsoft.com/2018/04/04/triaging-a-dll-planting-vulnerability/

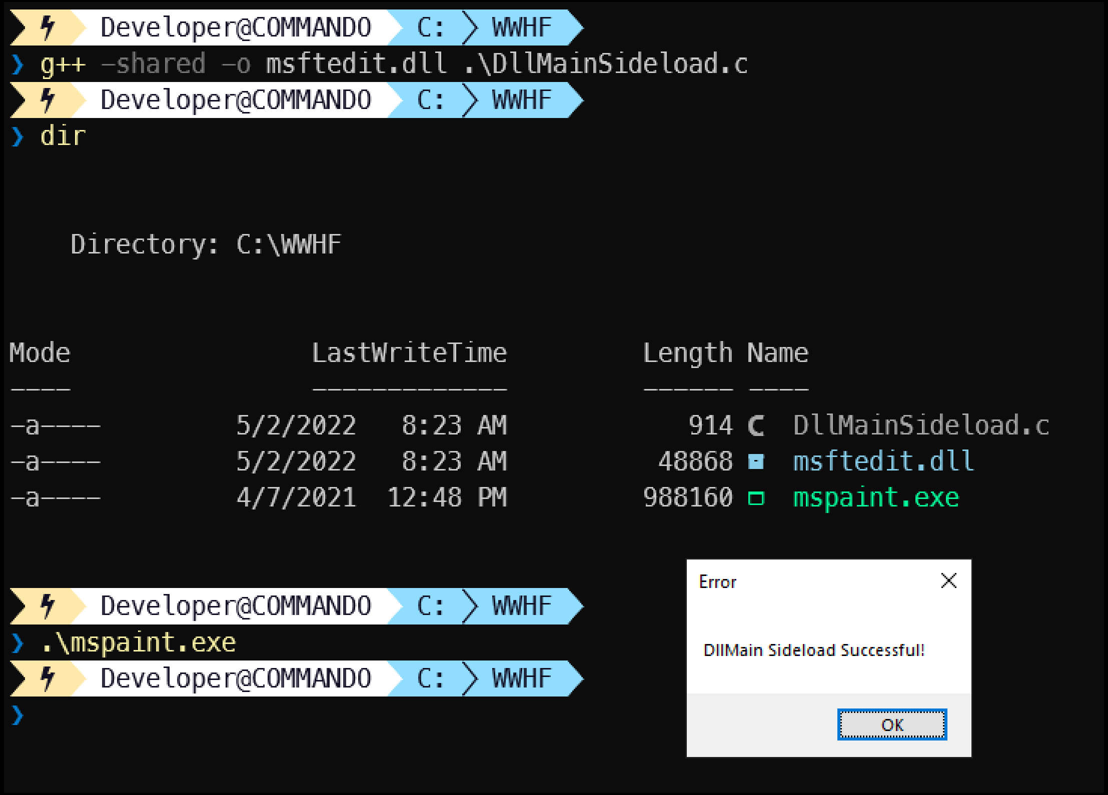
Microsoft: DLL Search Order

For example, suppose an application is designed to load a DLL from the user's current directory and fail gracefully if the DLL is not found. The application calls LoadLibrary with just the name of the DLL, which causes the system to search for the DLL. Assuming safe DLL search mode is enabled and the application is not using an alternate search order, the system searches directories in the following order:

1. The directory from which the application loaded.
2. The system directory.
3. The 16-bit system directory.
4. The Windows directory.
5. The current directory.
6. The directories that are listed in the PATH environment variable.

Reference: https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-library-security

MITRE T1574.002: DLL Sideloading



```
 ϟ  Developer@COMMANDO  C: 〉 WWHF
❯ g++ -shared -o msftedit.dll .\DllMainSideload.c
 ϟ  Developer@COMMANDO  C: 〉 WWHF
❯ dir


    Directory: C:\WWHF


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          5/2/2022   8:23 AM            914   DllMainSideload.c
-a----          5/2/2022   8:23 AM          48868   msftedit.dll
-a----          4/7/2021  12:48 PM         988160   mspaint.exe

 ϟ  Developer@COMMANDO  C: 〉 WWHF
❯ .\mspaint.exe
 ϟ  Developer@COMMANDO  C: 〉 WWHF
❯
```

Error

DllMain Sideload Successful!

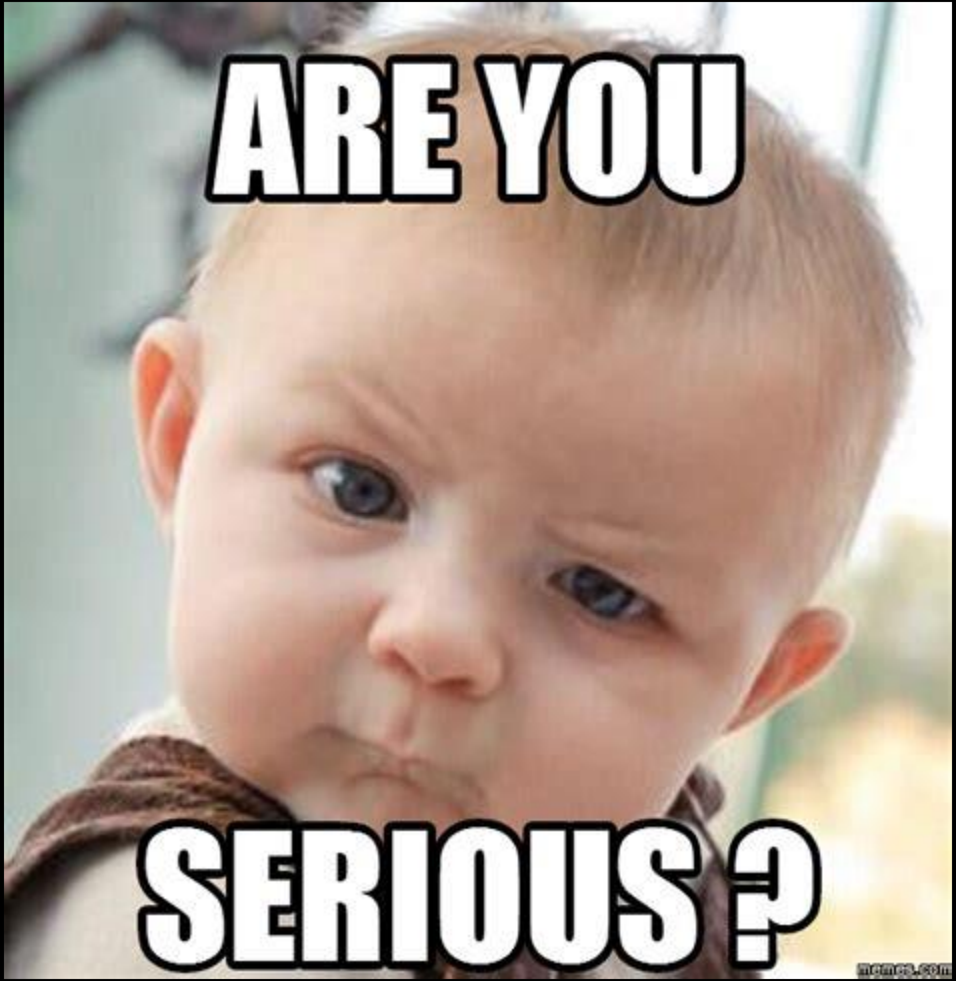
Developer Usage

WHEN YOU FIND OUT
IT'S A FEATURE, NOT A BUG
imgflip.com

Manual DLL Sideloading Identification

VERY
NICE

Automated DLL Sideloading Identification


ARE YOU
SERIOUS ?
memes.com

Detecting DLL Sideloading

GOTCHA
memegenerator.net

Finding DLL Sideloading Live

THAT'S A
MAYBE
makeameme.org

Real World Example

MISSION ACCOMPLISHED
imgflip.com

Questions?

Twitter: @ConsciousHacker

Discord: @Lopi#2914

References

https://github.com/xforcered/WFH

https://github.com/XForceIR/SideLoadHunter/tree/main/PS-SideLoadHunter

Hunting for Windows “Features” with Frida: DLL Sideloading

Hunting for Evidence of DLL Side-Loading With PowerShell and Sysmon