DIY Trojan horse or: how to get your malware past EDR

=----=[By Mark Steenberg (0x0vid)]=----=

#> whoami

Mark Staal Steenberg

Working as a senior consultant, conducting offensive engagements

Socials:
- https://github.com/0xOvid
- https://medium.com/@0x0vid
-https://www.linkedin.com/in/markstaalsteenberg/

--[0x1 - Structure

- Purpose
- What Is a Trojan Horse?
- Evading Static Analysis
- Evading Sandbox Analysis
- Evading Runtime Analysis
- Putting It All Together
- But How Do I Get Started?
- Release: farsidePacker
- Detection And Prevention
- Conclusion
- QA
- References

--[0x2 - Purpose

- Give an introduction to the magical world of EDR evasion
- Show how you can get started
- Show how this can be done

--[0x3 - What Is a Trojan Horse?

“What is a Trojan horse?

In computing, a Trojan horse is a program downloaded and installed on a computer that appears harmless, but is, in fact, malicious. Unexpected changes to computer settings and unusual activity, even when the computer should be idle, are strong indications that a Trojan is residing on a computer.”

--[0x4 - Stages of Analysis

- Static:
  - IOCs
  - Static signatures: Strings, bytes, hashes etc.
- Sandbox:
  - Execute in sandbox and monitor behavior
- Runtime:
  - Hooked system calls
  - Behavior post execution
  - Cloud behavioral analysis

--[0x5 - Evading Static Analysis (AV)

- Telemetry:
  - Static signatures
- Evasion:
  - Encryption: AES, XOR etc.
  - Obfuscation: Strings, code-flow
- How to spot
  - File getting deleted pre-execution

--[0x6 - Evading Sandbox Analysis

- Telemetry:
  - Run in VM and see what happens
- Evasion:
  - Check for signs of sandboxes
  - Wait – sandbox has limited resources cant do analysis for to long
- How to spot:
  - Pause before first print statement (in terminal)

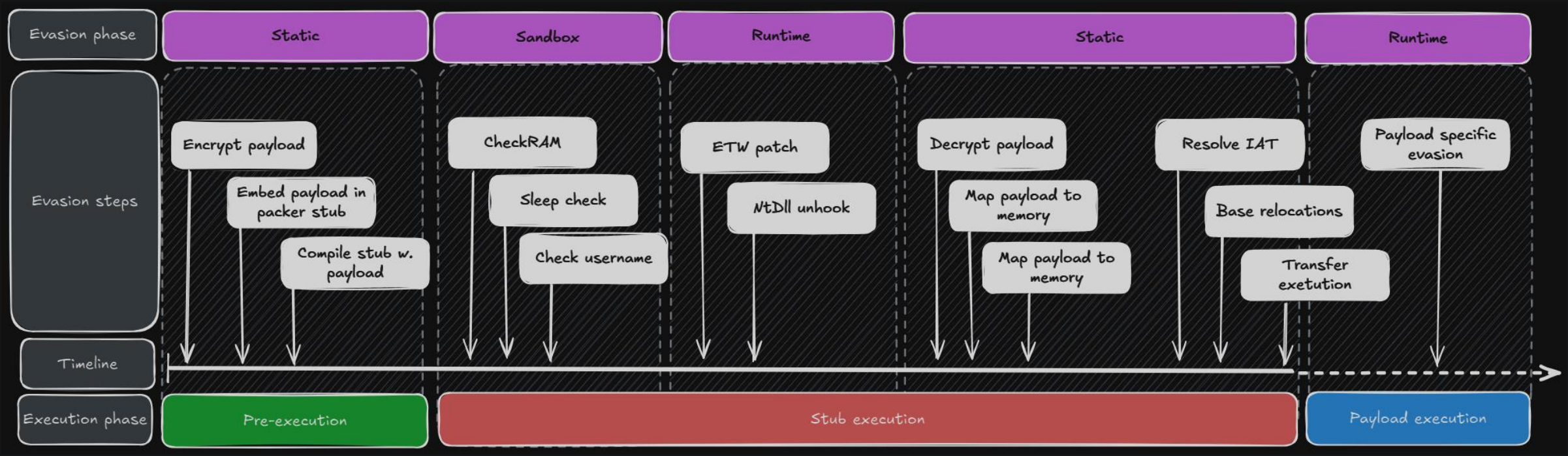
--[0x7 - Evading Runtime Analysis (EDR)

- Telemetry:
  - WinAPI function hooking
  - Event Tracing for Windows (ETW) & Threat Intelligence
  - Network monitoring
- Evasion:
  - Patch ETW
  - Remove/circumvent Hooks (Hells gate, Halos gate, NtDll unhooking etc.)
- How to spot:
  - File is executing and then gets detected/removed

--[0x8 – Putting It All Together

Hide malicious behavior in order of detection:

- Static -> Sandbox -> Runtime



--[0x9 - But How Do I Get Started?

--[0x9 … Continued

- Nim: Converting a pointer to a byte to a ulonglong to add it to a DWORD from a struct from an object to convert it to a pointer function pointer

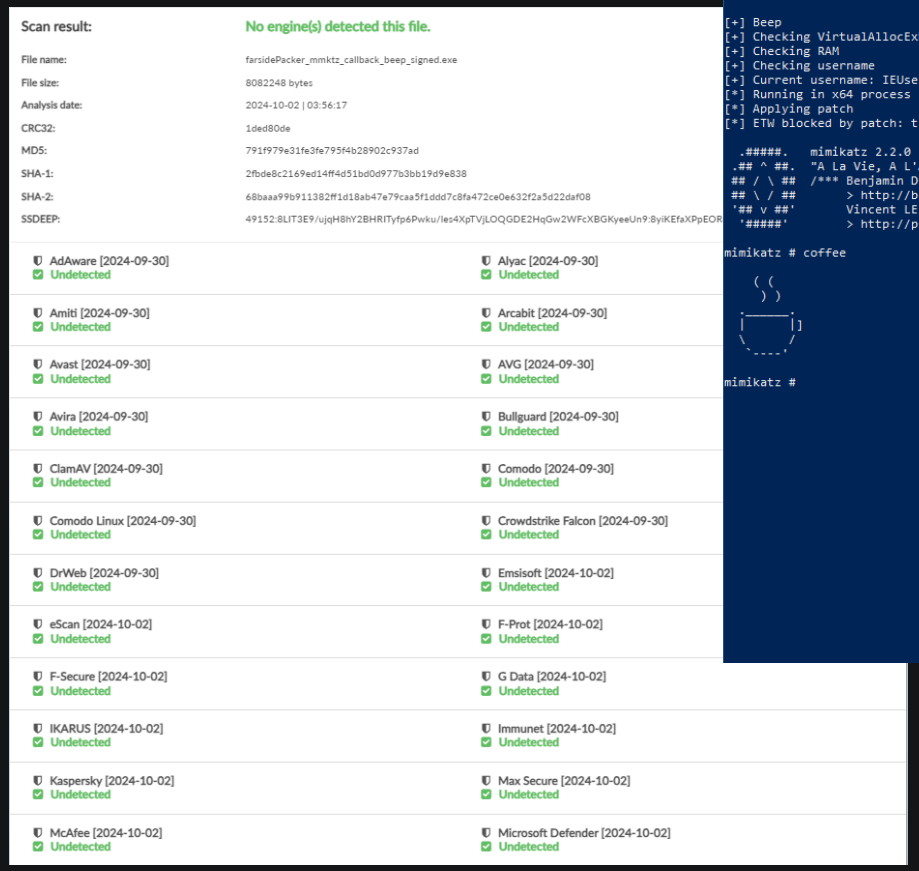
```
proc transferExecution*(ntHeaders: PIMAGE_NT_HEADERS, pImageBase: ptr BYTE, hProcess: HANDLE) =
    echo "[+] Executing PE in memory"
    var entryPoint = cast[LPTHREAD_START_ROUTINE](ntHeaders.OptionalHeader.AddressOfEntryPoint + cast[ULONGLONG](pImageBase))
    echo "\t|-> @ 0x", toHex(ntHeaders.OptionalHeader.AddressOfEntryPoint + cast[ULONGLONG](pImageBase))
    #@@executionType@@
```

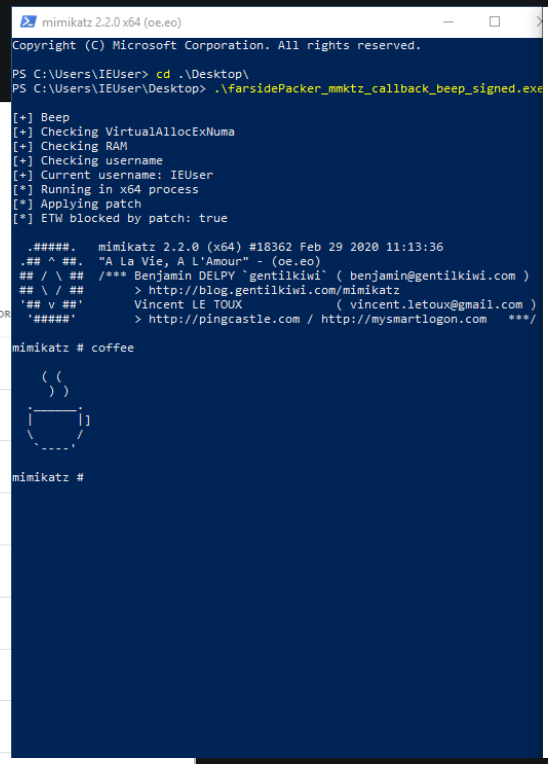
- Golang: Take the address of the first byte in an array, cast to an unsafe pointer, cast that to a uintptr

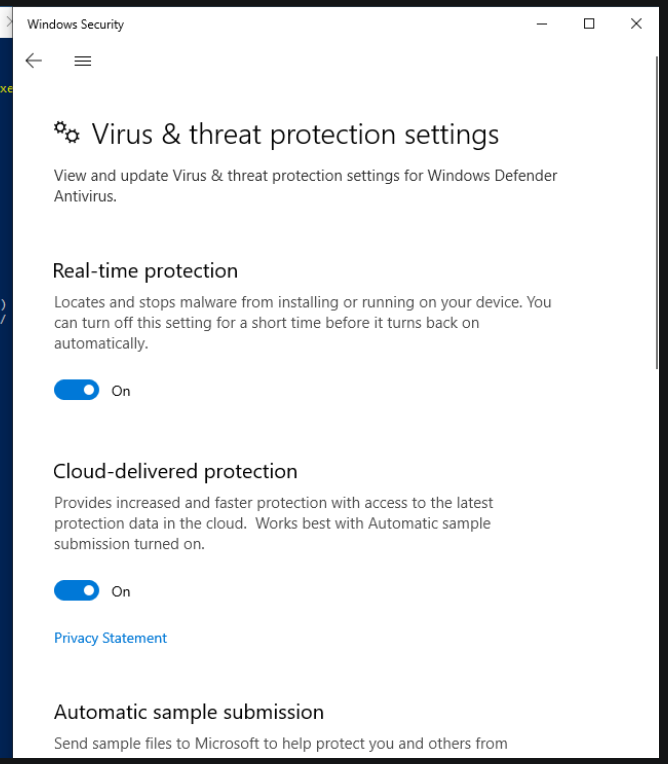
```
status, _, err := RtlCopyMemory.Call((uintptr)(unsafe.Pointer(&inBuffer[0])), (uintptr)(unsafe.Pointer(&os.Args[1])), uintptr(len(os.Args[1])))
```

--[0xA - Release: farsidePacker

- Here is how it can be done, use it to learn something!







https://github.com/0xOvid/farsidePacker

--[0xB - Detection And Prevention

Detection

- Suspicious API calls (defense evasion, RunPE, direct syscalls)
- Loading of ntdll twice
- Compound detections (LSASS handle by mimikatz, heap allocations by cobalt strike, network detections, defender for identity on DC)

Prevention

- Don’t let hackers execute code in your environment!
- AppLocker
- Attack Surface Reduction Rules
- Windows Defender Application Control (WDAC)
- Regularly update systems

… if you can run code then you can evade detection …

--[0xB … Continued

ASR:

```powershell
#Attack Surface Reduction Rules JSON File
$URL = "https://raw.githubusercontent.com/Kaidja/Defender-for-Endpoint/main/AttackSurfaceReductionRules.json"
#Convert ASR Rules from JSON
$ASRRules = (Invoke-WebRequest -Uri $URL -UseBasicParsing).Content | ConvertFrom-Json
foreach($Rule in $ASRRules){
            $ASRRuleName = $Rule.Name
            $ASRRuleGUID = $Rule.GUID
            Write-Output -InputObject "Working on $ASRRuleName. Setting the rule to Audit Mode"
            Add-MpPreference -AttackSurfaceReductionRules_Ids $Rule.GUID -AttackSurfaceReductionRules_Actions AuditMode
}
```

AppLocker:

Use AppLocker to achieve ML1

When the admin is deploying an AppLocker policy for user-based application control, the following rules can be used as a sample path-based implementation. This includes the rules, enforcement of rules and the automatic starting of the Application Identity service.

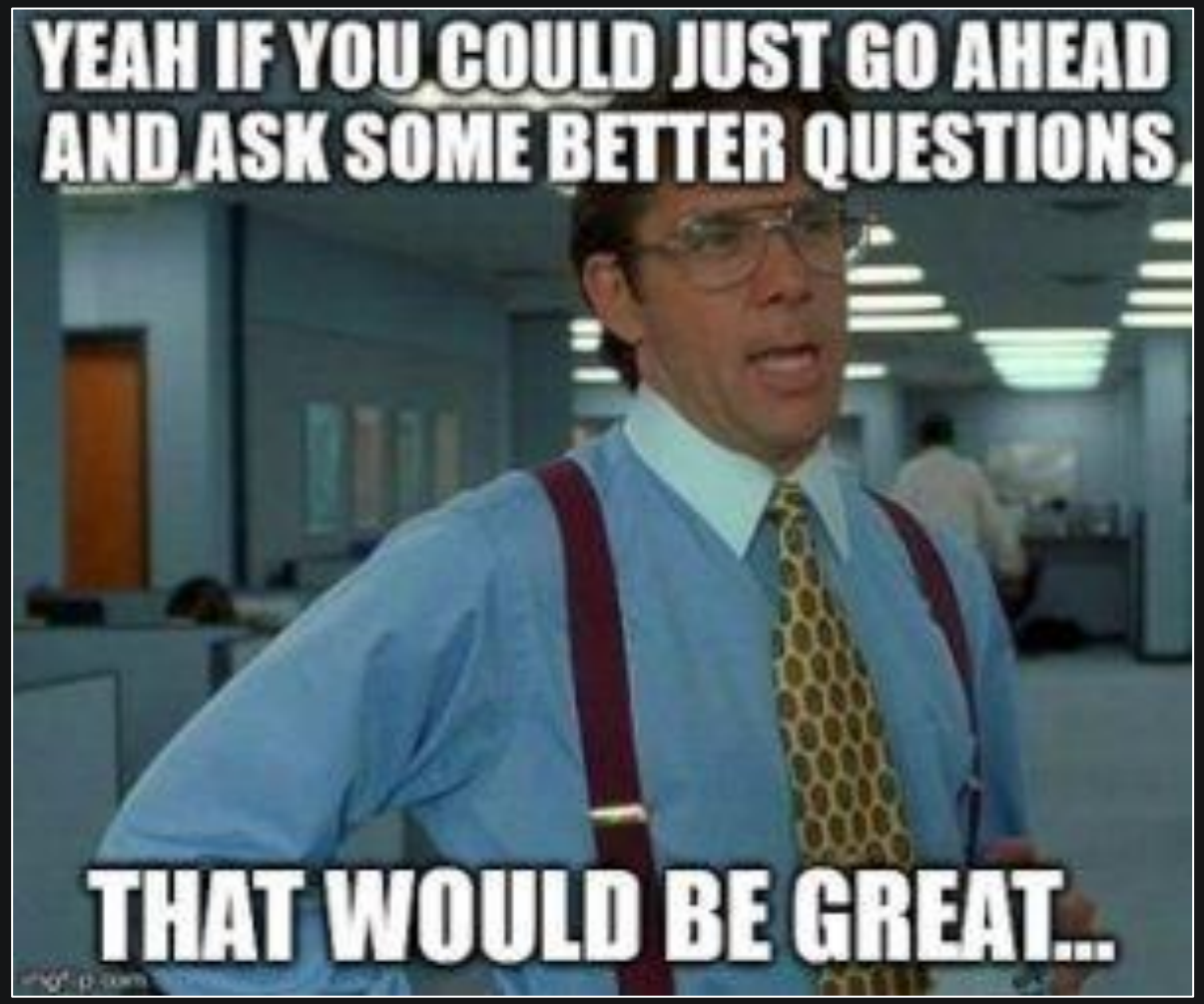
Suggestion is to block (as a minimum) the following paths:

- C:\Windows\Temp*.*
- %USERPROFILE%\AppData\Local*.*
  - Add exception for %USERPROFILE%\AppData\Local\Microsoft\WindowsApps
- %USERPROFILE%\AppData\Roaming*.*

… Also just block execution from the “downloads” folder … nobody needs to execute anything from there!

--[0xC - Conclusion

--[0xD - Q&A


YEAH IF YOU COULD JUST GO AHEAD
AND ASK SOME BETTER QUESTIONS
THAT WOULD BE GREAT...

--[0xE - References

- [001] — #HITB2022SIN EDR Evasion Primer For Red Teamers — Jorge Gimenez & Karsten Nohl — https://www.youtube.com/watch?v=CKfjLnEMfvI&ab_channel=HackInTheBoxSecurityConference
- [002] — The more predictable you are, the less you get detected — hiding malicious shellcodes via Shannon encoding — https://kleiton0x00.github.io/posts/The-more-predictable-you-are-the-less-you-are-able-to-get-detected/
- [003] — Building a custom Mimikatz binary — https://s3cur3th1ssh1t.github.io/Building-a-custom-Mimikatz-binary/
- [004] — What is encryption? — https://cloud.google.com/learn/what-is-encryption
- [005] — RunPE: How to hide code behind a legit process — https://www.adlice.com/runpe-hide-code-behind-legit-process/
- [006] — Process Hollowing, RunPE — https://unprotect.it/technique/process-hollowing-runpe/
- [007] — PE Injection — https://unprotect.it/technique/pe-injection/
- [008] — Joe sandbox — https://www.joesandbox.com/#windows
- [009] — ANY.RUN — https://any.run/
- [010] — Virus total — https://www.virustotal.com/gui/home/upload
- [011] — Checking Memory Size — https://unprotect.it/technique/checking-memory-size/
- [012] — RDTSCP — https://unprotect.it/technique/rdtscp/
- [013] — Windows API Hooking — https://www.ired.team/offensive-security/code-injection-process-injection/how-to-hook-windows-api-using-c++
- [014] — Event Tracing for Windows (ETW) — https://learn.microsoft.com/en-us/windows-hardware/drivers/devtest/event-tracing-for-windows--etw-
- [015] — Uncovering Windows Events — Jonathan Johnson — https://jsecurity101.medium.com/uncovering-windows-events-b4b9db7eac54
- [016] — Introduction into Microsoft Threat Intelligence Drivers (ETW-TI) — meekochii — https://research.meekolab.com/introduction-into-microsoft-threat-intelligence-drivers-etw-ti
- [017] — Sysmon — https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon
- [018] — Patching Event Tracing for Windows (ETW) in C# — https://www.phrack.me/tools/2023/04/10/Patching-ETW-in-C.html
- [019] — Direct Syscalls: A journey from high to low — Daniel Feichter @VirtualAllocEx — https://redops.at/en/blog/direct-syscalls-a-journey-from-high-to-low
- [020] — https://github.com/TierZeroSecurity/edr_blocker
- [021] — Nim-RunPE - S3cur3Th1sSh1t — https://github.com/S3cur3Th1sSh1t/Nim-RunPE/blob/main/NimRunPE.nim
- [022] — Helixo32/NimReflectiveLoader — https://github.com/Helixo32/NimReflectiveLoader/blob/main/src/RunRemoteDll.nim
- [023] — aaaddress1/RunPE-In-Memory — https://github.com/aaaddress1/RunPE-In-Memory/blob/master/RunPE-In-Memory/RunPEinMemory/fixIAT.hpp
- [024] — OffensiveNim /encrypt_decrypt_bin.nim — https://github.com/byt3bl33d3r/OffensiveNim/blob/master/src/encrypt_decrypt_bin.nim
- [025] — OffensiveNim — https://github.com/byt3bl33d3r/OffensiveNim
- [026] — nimcrypt2 — icyguider - https://github.com/icyguider/Nimcrypt2/tree/main
- [027] — Writing a Packer From Scratch in Nim — 0x0vid — https://medium.com/p/460b5b3692e0
- [028] - Simplifying Cyber Defense: How to Configure Attack Surface Reduction Rules with PowerShell - Kaido Järvemets - https://kaidojarvemets.com/simplifying-cyber-defense-how-to-configure-attack-surface-reduction-with-powershell/
- [029] - Applications that can bypass App Control and how to block them - https://learn.microsoft.com/en-us/windows/security/application-security/application-control/app-control-for-business/design/applications-that-can-bypass-appcontrol
- [030] - Essential Eight application control - https://learn.microsoft.com/en-us/compliance/anz/e8-app-control

Other references:

- Definition: What is a trojan horse — https://www.techtarget.com/searchsecurity/definition/Trojan-horse
- A dive into the PE file format — Introduction — 0xRick — https://0xrick.github.io/win-internals/pe1/
- PE Data Directories — https://offwhitesecurity.dev/malware-development/portable-executable-pe/nt-headers/optional-header/data-directories/
- VirtualAlloc — https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc
- Process Hollowing and Portable Executable Relocations — https://www.ired.team/offensive-security/code-injection-process-injection/process-hollowing-and-pe-image-relocations#relocation
- Reflective DLL Injection — https://www.ired.team/offensive-security/code-injection-process-injection/reflective-dll-injection#resolving-import-address-table