SIGPwny

FA2024 Week 12 • 2024-11-19

Attacking Hardened Environments

Ronan Boyarski

Table of Contents

Application Whitelisting

AppLocker

- Defines rules to allow what can run on a system
- **NOT** a security boundary
  - Considered a defense-in-depth feature
- Designed to block unwanted software and some malware
  - Appears to be more designed to stop low-tier threats
- Default AppLocker rules (example)
  - Allow members of the local **Administrators** group to run everything
  - Allow members of the **Everyone** group to run apps that are located in C:\Windows
  - Allow members of the **Everyone** group to run apps that are located in the Program Files folder

Defeating AppLocker

- Put malware in C:\Windows\Tasks or C:\Windows\Temp
  - Bypasses rule number 2 (see previous slide)
- Bypass with DLLs
  - Default protection doesn't protect against running DLLs
    - `rundll32 sus.dll,main`
- Bypass with third party scripting engine
  - Check if something like Python is installed
- Execute shellcode w/LOLBAS
  - Example for arbitrary C# execution w/**msbuild.exe**

PowerShell
Constrained Language Mode

PowerShell CLM

- Language mode of powershell designed to only support what's necessary for day-to-day tasks
  - Restricts usage of WinAPI
  - **Not** a security boundary
- Enumerate
  - `$ExecutionContext.SessionState.LanguageMode`
- Designed as another defense-in-depth measure to be stacked with things like AppLocker

Bypassing PowerShell CLM

- Reuse the **msbuild** bypass
  - This lets us run arbitrary C#
  - If needed, we can call PowerShell from C# by hosting a custom runspace
- LOLBAS
  - `InstallUtil.exe /logfile= /LogToConsole=false /U malware.dll`
- Alternative: don't use powershell :)

Windows Defender Application Control

Windows Defender Application Control

- Can be configured to block unknown applications in a very robust manner
  - Can check by file name, hash, path, signing status, and more
- **IS** considered a security boundary
- No known bypasses (bypasses are considered CVEs)
- Must be attacked by reversing & attacking the policy in place
- Base policies can be found at `C:\Windows\schemas\CodeIntegrity\ExamplePolicies`, with a .p7b file extension
- Alternatively can be read from GPO

Windows Defender Application Control

- Download the .p7b file and reverse it with CIPolicyParser.ps1
- Avoid denied LOLBAS
- Search for allow lists
  - Wildcards are great
  - Look for wildcarded writable folders
  - Look for checking only the file name
  - Check if signing your binary lets it pass

```
<Deny ID="ID_DENY_D_0208" FileName="davsvc.dll" MaximumFileVersion="65355.
<Deny ID="ID_DENY_D_0209" FileName="DBGHOST.Exe" MinimumFileVersion="2.3.0.
<Deny ID="ID_DENY_D_020A" FileName="DBGSVC.Exe" MinimumFileVersion="2.3.0.
<Deny ID="ID_DENY_D_020B" FileName="dnx.Exe" MaximumFileVersion="65355.653
<Deny ID="ID_DENY_D_020C" FileName="dotnet.exe" MaximumFileVersion="65355.
<Deny ID="ID_DENY_D_020D" FileName="fsi.exe" MaximumFileVersion="65355.653
<Deny ID="ID_DENY_D_020E" FileName="fsiAnyCpu.exe" MaximumFileVersion="653
<Deny ID="ID_DENY_D_020F" FileName="infdefaultinstall.exe" MaximumFileVersi
<Deny ID="ID_DENY_D_0210" FileName="InstallUtil.exe" MaximumFileVersion="65
<Deny ID="ID_DENY_D_0211" FileName="jscript9.dll" MinimumFileVersion="11.0.
<Deny ID="ID_DENY_D_0212" FileName="kd.Exe" MaximumFileVersion="65355.65355
<Deny ID="ID_DENY_D_0213" FileName="kd.Exe" MaximumFileVersion="65355.65355
<Deny ID="ID_DENY_D_0214" FileName="kill.exe" MaximumFileVersion="65355.653
<Deny ID="ID_DENY_D_0215" FileName="lxrun.exe" MaximumFileVersion="65355.65
<Deny ID="ID_DENY_D_0216" FileName="LxssManager.dll" MaximumFileVersion="65
<Deny ID="ID_DENY_D_0217" FileName="mfc40.dll" MaximumFileVersion="65355.65
```

```
<FileRules>
  <Allow ID="ID_ALLOW_A_0001" FilePath="C:\Program Files\7-Zip\*" />
  <Allow ID="ID_ALLOW_A_0002" FileName="7zipInstall.exe" MinimumFileVersion="22.1.0.0" />
</FileRules>
```

Attack Surface Reduction

Attack Surface Reduction

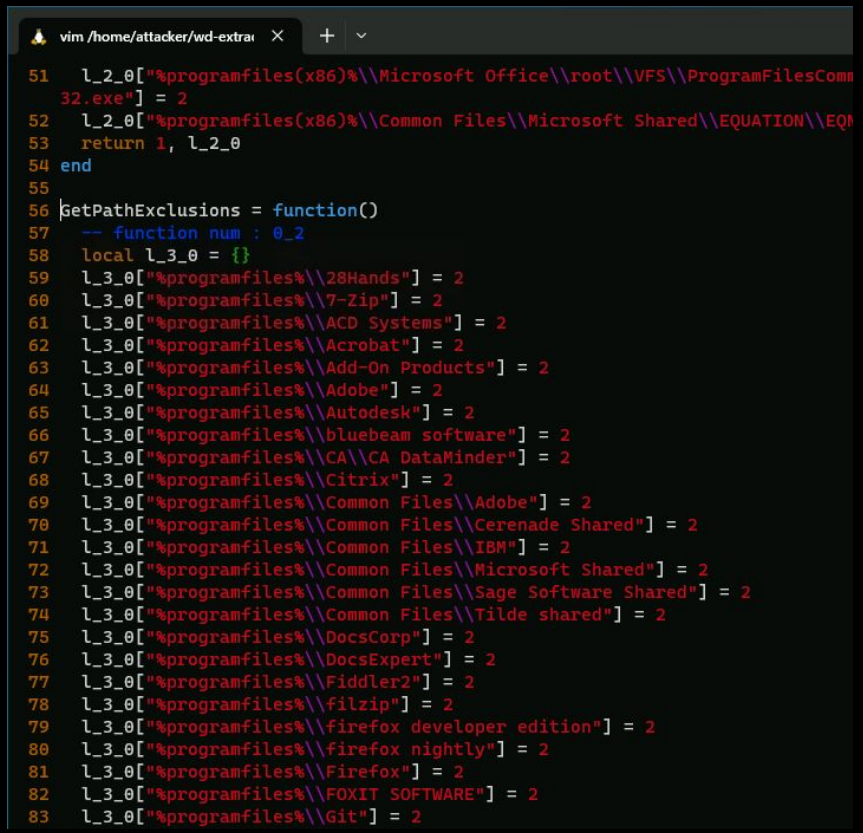
- Feature of Windows Defender that blocks common methods used for execution
- Example Rules:
  - Block all Office Applications from creating child processes
  - Block Win32 API calls from Office macros
  - Block Office applications from injection code into other processes
  - Block executable files from running unless they meet a prevalence, age, or trusted list criterion
  - Block unsigned processes that run from USB
  - Block JavaScript or VBScript from launching executable content
  - Block Process Creations from WMI (lateral movement!)

Attack Surface Reduction

- We can find enabled rules if we're on the device, but ASR largely blocks methods of getting in, so I won't even go over it here
- We need to reverse engineer ASR exclusions and make our processes fit those
  - We can use a script called **wd-extract.py** to find the exclusions on our local machine
  - We'll get a bunch of lua files which we can grep through to find the matching rule

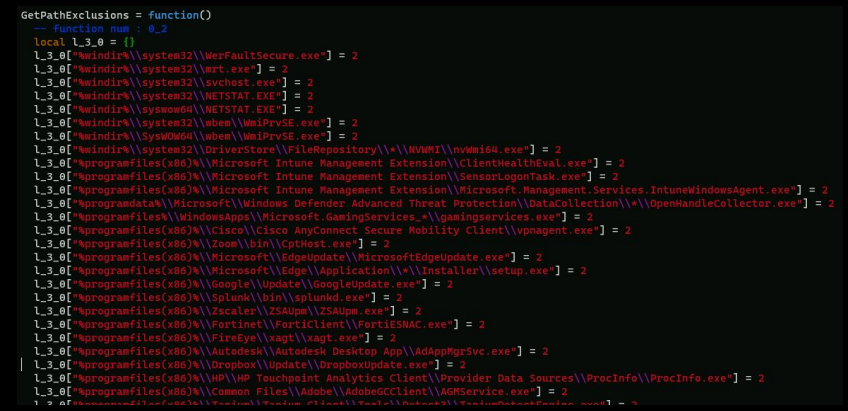
Attack Surface Reduction

- This is an example for spawning processes from an Office document
- There are hundreds of path exclusions
- Look for one with a writable directory (like %AppData%)
- So, if we need to, we can drop our malicious EXE to a whitelisted path and run it directly to get around this
- Regardless, getting around ASR is a pain



Attack Surface Reduction - LSASS

- If configured to "Block Credential Stealing from LSASS", you won't be able to get a handle to LSASS with read access
- The solution is to spawn and inject our LSASS dumper into whitelisted process that is allowed to get read access to LSASS
  - An easy one here is svchost.exe



Summary

- To combat defense in depth, we need to start practicing offense in depth
  - This is a huge departure from "run tools and pwn shit"
- Application Whitelisting (AppLocker)
  - Evade through LOLBAS & writable directories
- PowerShell Constrained Language Mode
  - Evade through LOLBAS
- Windows Defender Application Control
  - Evade through exploiting weak policies & exclusions (see here)
- Attack Surface Reduction
  - Evade through reverse engineering default exclusions

Next Meetings

`2024-11-19` • **This Thursday**

- Running Networking Devices
- Next meeting is on 12/3 (Introduction to Offensive Development)
  - I will likely be late as I'm giving a talk for ENG 298 the hour before