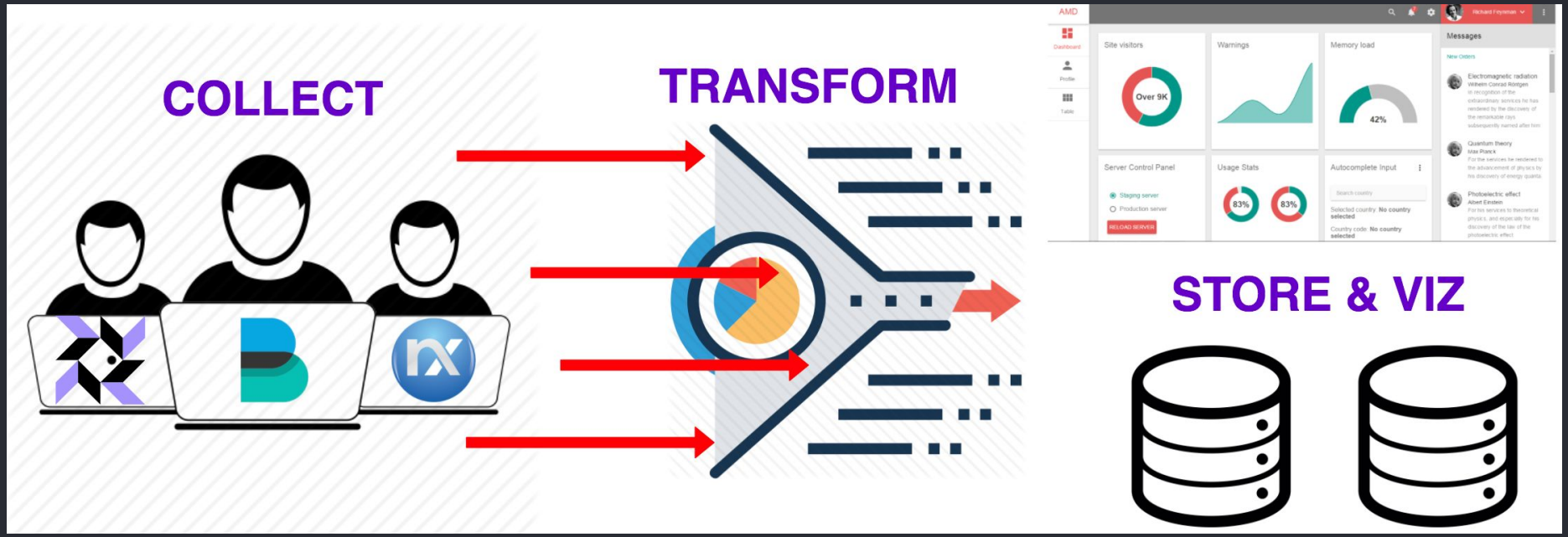
Perched
Endpoint Operator

Course Outline

- Challenges with Windows Logging
- Winlogbeat and Filebeat
- Intro to The Hunting ELK (HELK)
- EDR and PowerShell
- Google Rapid Response

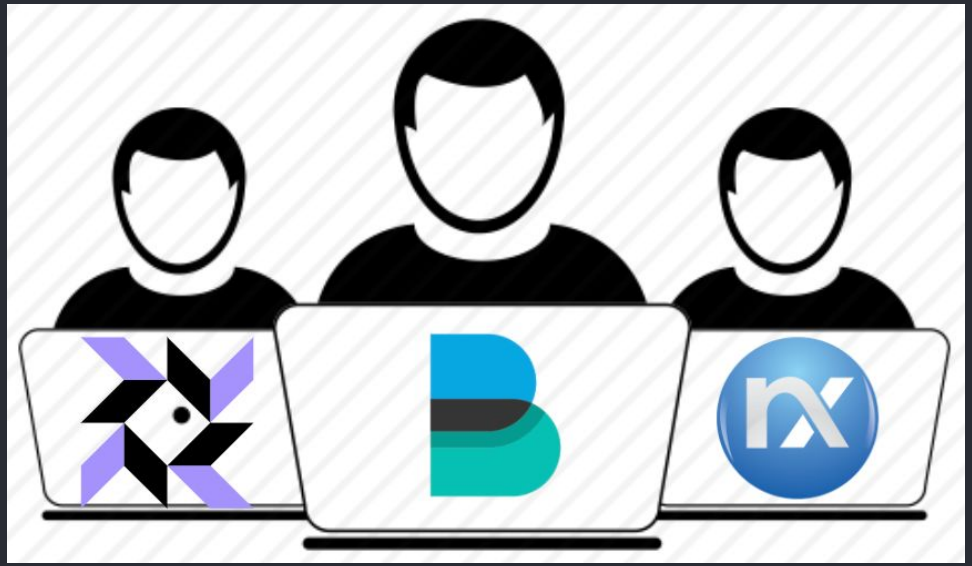
Challenges with Windows Logging

Current State



Current State - Collection

- The Right Data
  - PowerShell
  - Sysmon
  - Bro
  - OSQuery
- Host-Based Detection Techniques
  - Point-In-Time - Scripting
  - Historical - Windows Event Logs
  - Real-Time - EDR, Sysmon
- Challenges:
  - What do I do with all this data?
  - What else should I collect?
  - So much data!
    - bandwidth (ATMs)



https://posts.specterops.io/thoughts-on-host-based-detection-techniques-21d9c97082ce

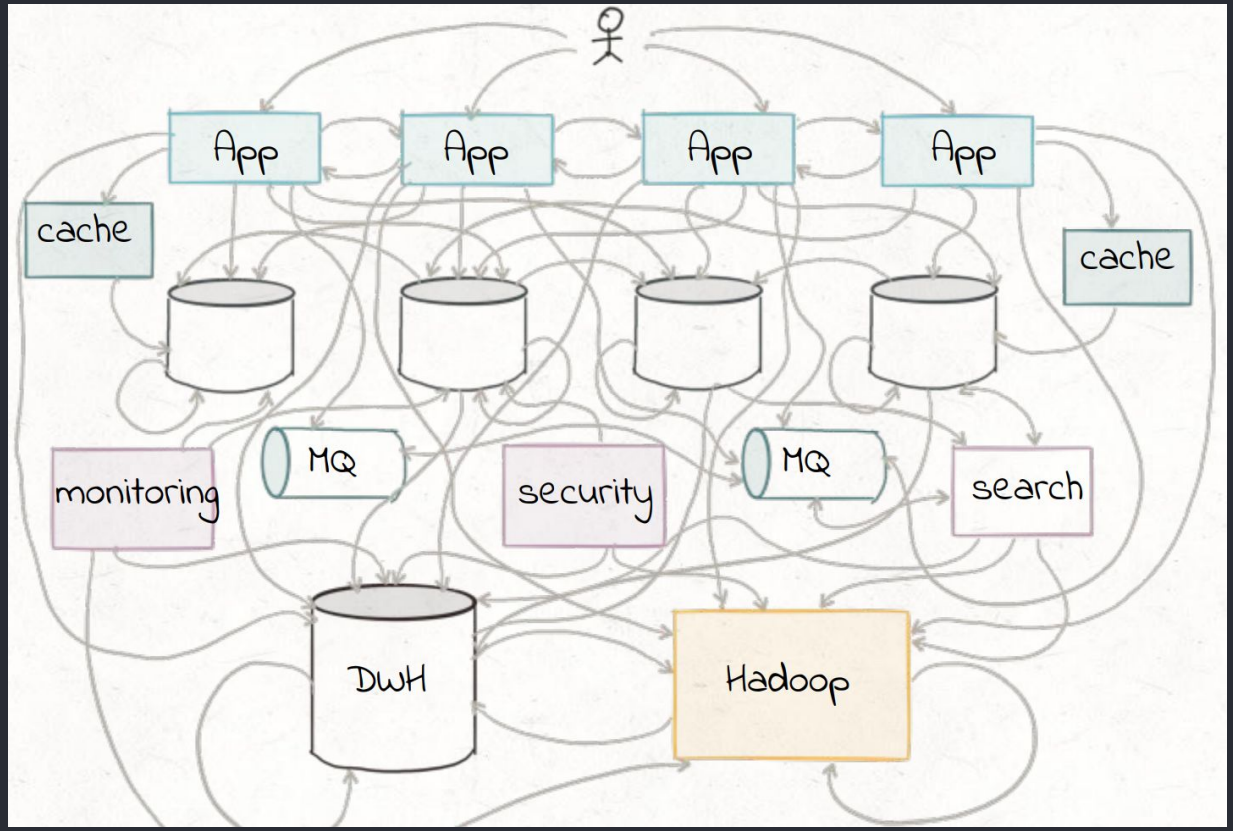
Current State - Transformation

- Real time vs Query run time
  - Streaming Data integration
  - Data models at query time
    - Batch integration

- Challenges:
  - More data more problems!
    - Different formats
    - Same field names but different values
    - Same values but different field names...
    - Unstructured data
  - Pivoting
    - Hunt, Response, alert triage, etc.

Current State - Storing & Visualizing

- Challenges:
  - Decentralized
  - Expensive
    - Big Data
  - Too many dashboards
    - No actionable data
  - Graphs
    - Millions of nodes!



https://qconsf.com/sf2016/system/files/keynotes-slides/etl_is_dead_long-live_streams.pdf

MITRE ATT&CK

Accessibility Features

Windows contains accessibility features that may be launched with a key combination before a user has logged in (for example, when the user is on the Windows logon screen). An adversary can modify the way these programs are launched to get a command prompt or backdoor without logging in to the system.

Accessibility Features

Technique

ID	T1015
Tactic	Persistence, Privilege Escalation
Platform	Windows
Permissions Required	Administrator
Effective Permissions	SYSTEM
Data Sources	Windows Registry, File monitoring, Process monitoring
CAPEC ID	CAPEC-558
Contributors	Paul Speulstra, AECOM Global Security Operations Center

Two common accessibility programs are `C:\Windows\System32\sethc.exe`, launched when the shift key is pressed five times and `C:\Windows\System32\utilman.exe`, launched when the Windows + U key combination is pressed. The sethc.exe program is often referred to as "sticky keys", and has been used by adversaries for unauthenticated access through a remote desktop login screen.[1]

MITRE ATT&CK

Access Token Manipulation

Windows uses access tokens to determine the ownership of a running process. A user can manipulate access tokens to make a running process appear as though it belongs to someone other than the user that started the process. When this occurs, the process also takes on the security context associated with the new token. For example, Microsoft promotes the use of access tokens as a security best practice. Administrators should log in as a standard user but run their tools with administrator privileges using the built-in access token manipulation command `runas`. [1]

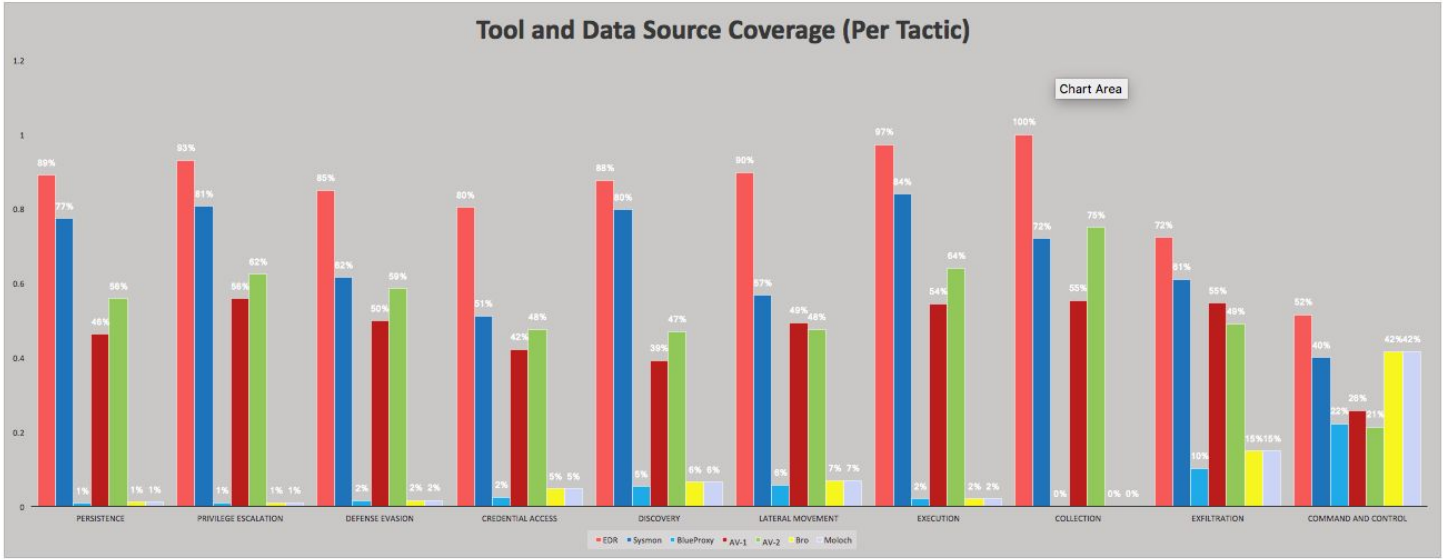
Adversaries may use access tokens to operate under a different user or system security context to perform actions and evade detection. An adversary can use built-in Windows API functions to copy access tokens from existing processes; this is known as token stealing. An adversary must already be in a privileged user context (i.e. administrator)

Access Token Manipulation

Technique

ID	T1134
Tactic	Defense Evasion, Privilege Escalation
Platform	Windows
Permissions Required	User, Administrator
Effective Permissions	SYSTEM
Data Sources	API monitoring, Access Tokens
Contributors	Tom Ueltschi @c_APT_ure, Travis Smith, Tripwire, Jared Atkinson, @jaredcatkinson, Robby Winchester, @robwinchester3

Tactics	EDR	Sysmon	BlueProxy	AV-1	AV-2	Bro	Moloch
Persistence	89%	77%	1%	46%	56%	1%	1%
Privilege Escalation	93%	81%	1%	56%	62%	1%	1%
Defense Evasion	85%	62%	2%	50%	59%	2%	2%
Credential Access	80%	51%	2%	42%	48%	5%	5%
Discovery	88%	80%	5%	39%	47%	6%	6%
Lateral Movement	90%	57%	6%	49%	48%	7%	7%
Execution	97%	84%	2%	54%	64%	2%	2%
Collection	100%	72%	0%	55%	75%	0%	0%
Exfiltration	72%	61%	10%	55%	49%	15%	15%
Command and Control	52%	40%	22%	26%	21%	42%	42%



https://posts.specterops.io/ready-to-hunt-first-show-me-your-data-a642c6b170d6

How To Collect Efficiently
& for Free Using Windows Event Forwarding (WEF)

- #1 Share your logs… With other departments..
  - Get buy in and prove your worth
- WEF requires no additional software and uses built in credentials
- Bandwidth issues? No problem
  - CPU throttling, rate limiting, pre-filter on endpoint, etc.
- Cost: Hard Drives are cheap

https://bit.ly/WinLogsZero2Hero

https://blogs.technet.microsoft.com/jepayne/2015/11/23/monitoring-what-matters-windows-event-forwarding-for-everyone-even-if-you-already-have-a-siem/

Transformation?

- Filtering, massaging the data..
- The need for a GLOBAL SCHEMA!!
- Automatic transformation
  - Might not work as expected without the right documentation and expertise.

Spending Your Life Parsing...

How we spend our days is, of course, how we spend our lives.
-Annie Dillard

One day, maybe we can agree on a naming and value schema/standard.

Until then, here we go...

Logs are the worst…

- Endless variations
  - Values
  - Field names
  - Casing
  - Data sets

- But what if someone/something breaks your parser?
  - CEF
    - ||||||||===========|=||==|==
  - JSON
    - {}{}{}{[][][}{}{}{}{[}{}{}{}{}{]{}{]{}}{}{{}}{}{}{}{{{}}}}}{{{{}}}}}}{}{}}{}{}{}}{}{
  - XML
    - <><><><<<>>>>><<<><><><><<<><

(Challenge) Values

- Firewalls
  - Which one is it “Allow” or “Permit”
- Network protocol (one of the worst, TippingPoint)
  - SMB? Not a protocol...
  - HTTP? Not a protocol...
- SSL/TLS Naming
  - Bro = “TLSv12”
  - Suricata = “TLS 1.2”
- Let’s not even mention casing..
  - TCP/Tcp/tcp
- Network Interface(s) -- but that's a VLAN too...
  - ethernet1/10.1999

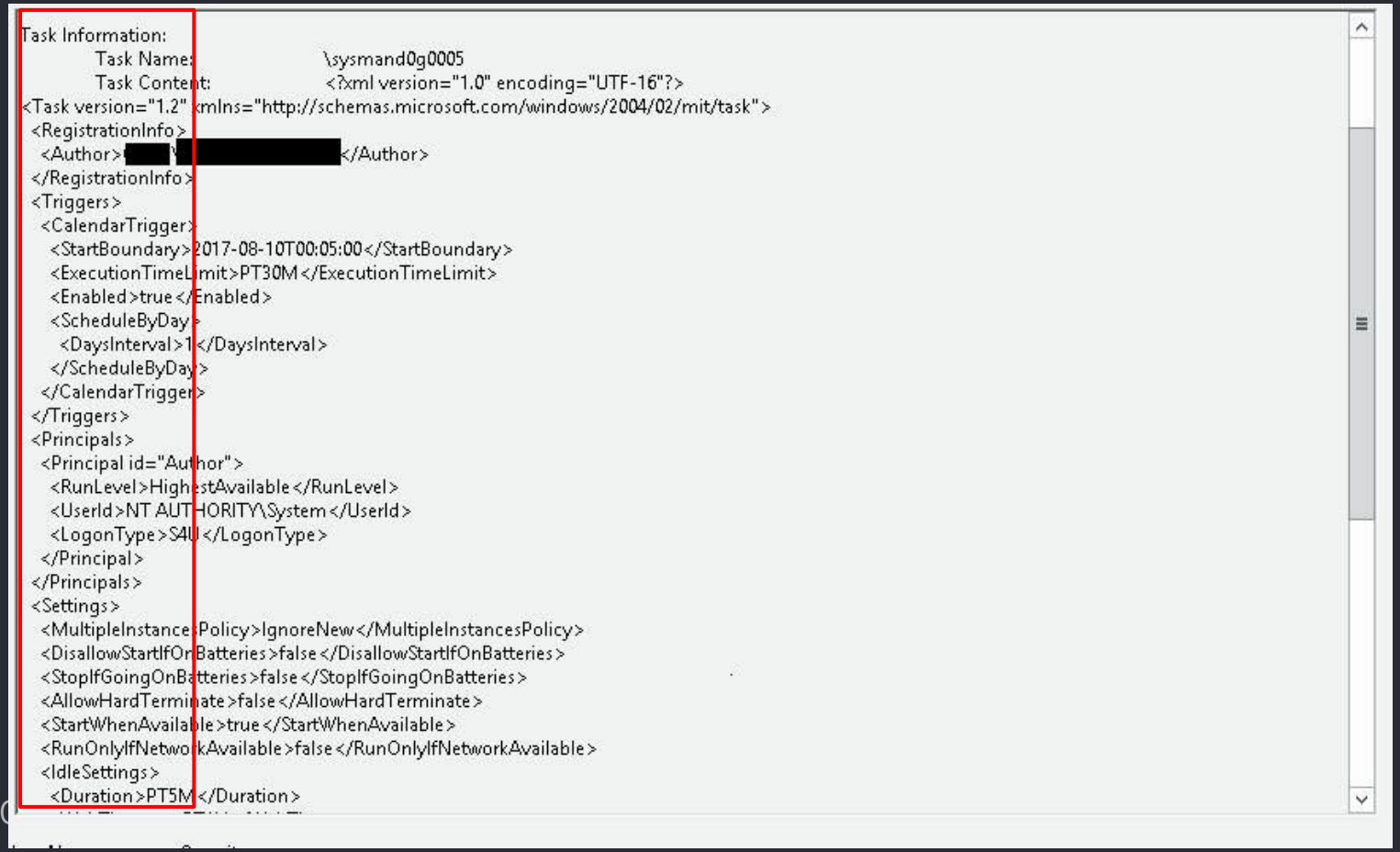
(Challenge) PowerShell 4103

```
CommandInvocation(Add-Content): "Add-Content"
ParameterBinding(Add-Content): name="Path"; value="C:\Windows\Temp\active-ad-devices.txt"
ParameterBinding(Add-Content): name="Value"; value="2018-03-01.15:00:02;1;19"


Context:
        Severity = Informational
        Host Name = ConsoleHost
        Host Version = 5.1.14409.1012
        Host ID = bcbe5[redacted]36-7d47-485f-ac51-bbc392a207a3
        Host Application = powershell.exe -ExecutionPolicy Bypass C:\Users\[redacted]Desktop\find-active-ad-computers.ps1 1
C:\Windows\Temp\active-ad-devices.txt
        Engine Version = 5.1.14409.1012
        Runspace ID = ba62158c-887a-497a-8767-ceae04525b09
        Pipeline ID = 1
        Command Name = Add-Content
        Command Type = Cmdlet
        Script Name = C:\Users\[redacted]\Desktop\find-active-ad-computers.ps1
        Command Path =
        Sequence Number = 20
        User = CORP\SYSTEM
        Connected User =
        Shell ID = Microsoft.PowerShell


User Data:
```


(Challenge) PowerShell 4104

```
Creating Scriptblock text (1 of 1):
Param(
 [int]$NumberOfHoursToQuery=1,
 [string]$path
)

$date = ([DateTime]::Now.AddHours(-$NumberOfHoursToQuery)).tofiletime()
$filter = "(&(objectclass=computer)(operatingsystem=*windows*)(lastlogon>=$date))"
$adsiResults = ([adsisearcher]$filter).FindAll()
$comps = $adsiResults `
 | select @{n='hostname';e={$_.properties.name}},
       @{n='lastlogon';e={$_.properties.lastlogon}} `
 | select hostname, @{n='lastlogon';e={[datetime]::FromFileTime($_.lastlogon)}}


$TimeRan = Get-Date -format "yyyy-MM-dd.HH:mm:ss"
$NumberOfDevices = "$($comps.count)"
# Append file
Add-Content $path "$TimeRan;$NumberOfHoursToQuery;$NumberOfDevices"



# Usage:
# powershell.exe –ExecutionPolicy Bypass " C:\Windows\Temp\find-active-ad-computers.ps1" 1 "C:\Windows\Temp\active-ad-
devices.txt"

# Below overwrites
#Set-Content -Value $out -path $path


ScriptBlock ID: 0c13bf3e-460c-480a-b025-c9c642ce1fd3
Path: C:\Users\[redacted]\Desktop\find-active-ad-computers.ps1
```


(Challenge) Scheduled Task 4698



(Challenge) Windows IP Fields

Number of Possible Field names… 18 and counting ?!

- IpAddress
- SourceIp
- DestinationIp
- ClientAddress
- ClientIPAddress
- LaunchedViaIPAddress
- ConnectedViaIPAddress
- MachineIpAddress
- ipAddress
- Value
- ClientIP
- IPString
- TargetServer
- Source
- DCIPAddress
- ClientIpAddress
- ServerIpAddress
- IPAddress
- Not even including all the other IPs found through “Message” fields that don’t have a designated IP field

“ClientIP”

(Microsoft-Windows-RemoteDesktopServices-RdpCoreTS/Operational:131)

- [10.10.10.10]:50231
- 10.10.10.10:49283
- Also IPv6 Variants

“Sent UpdateServer”

(System:8009 & SourceName:Microsoft-Windows-DNS-Client)

- 99.1.23.45:53
- <?>
- [::1]:53
- [2607:ff08:206::20]:53

Asked by:

0
Points

neu5ron (Nate Guagenti)
Joined Dec 2011
neu5ron (Nate ...
Show activity

Windows Event Log Channel with incorrect naming of field

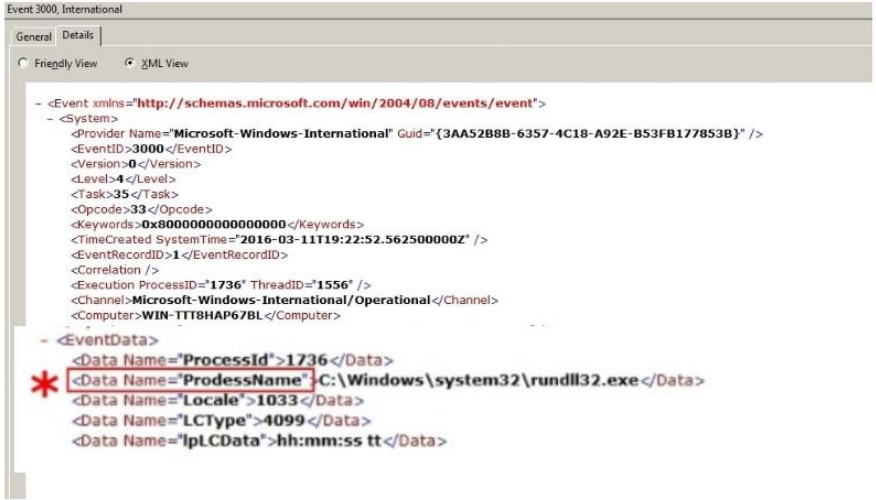
Security > MBSA - Microsoft Baseline Security Analyzer

Question

Channel "Microsoft-Windows-International/Operational" labels/names ProcessName as "ProdessName".
Specifically have seen in EventIDs: 3000 and 3003

Tested on Windows 7, Windows 10, Windows Server 2008R2, and Windows Server 2012

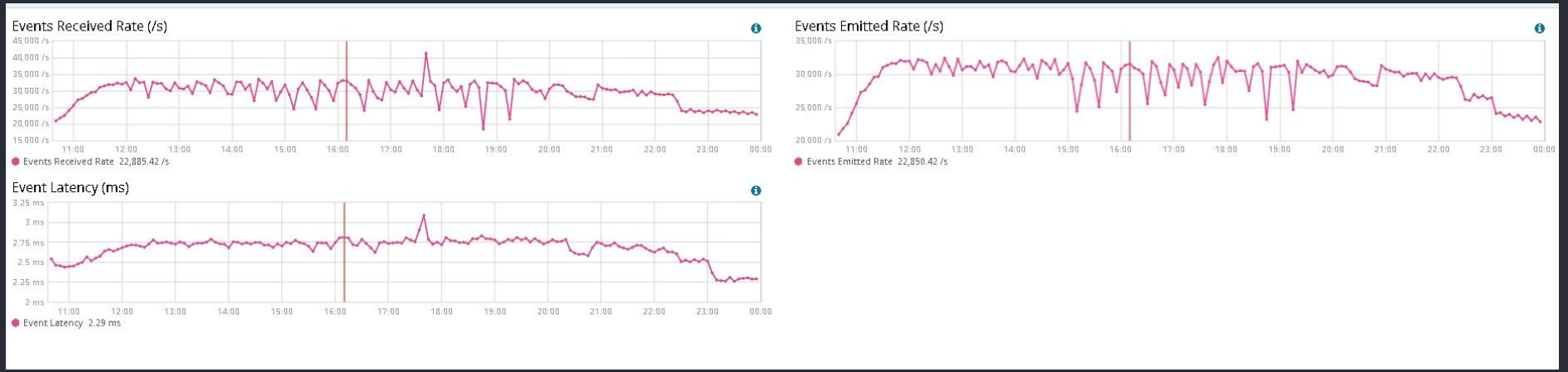
0
Sign in to vote



Won’t This Slow My Pipeline/Transform?

- Compare statement in Assembly is only a few bytes and little CPU overhead
- Log Queue like Kafka, takes burden off of Logstash
- Caching -- example: can make Geo/AS lookups faster

Won’t This Slow My Pipeline/Transform?



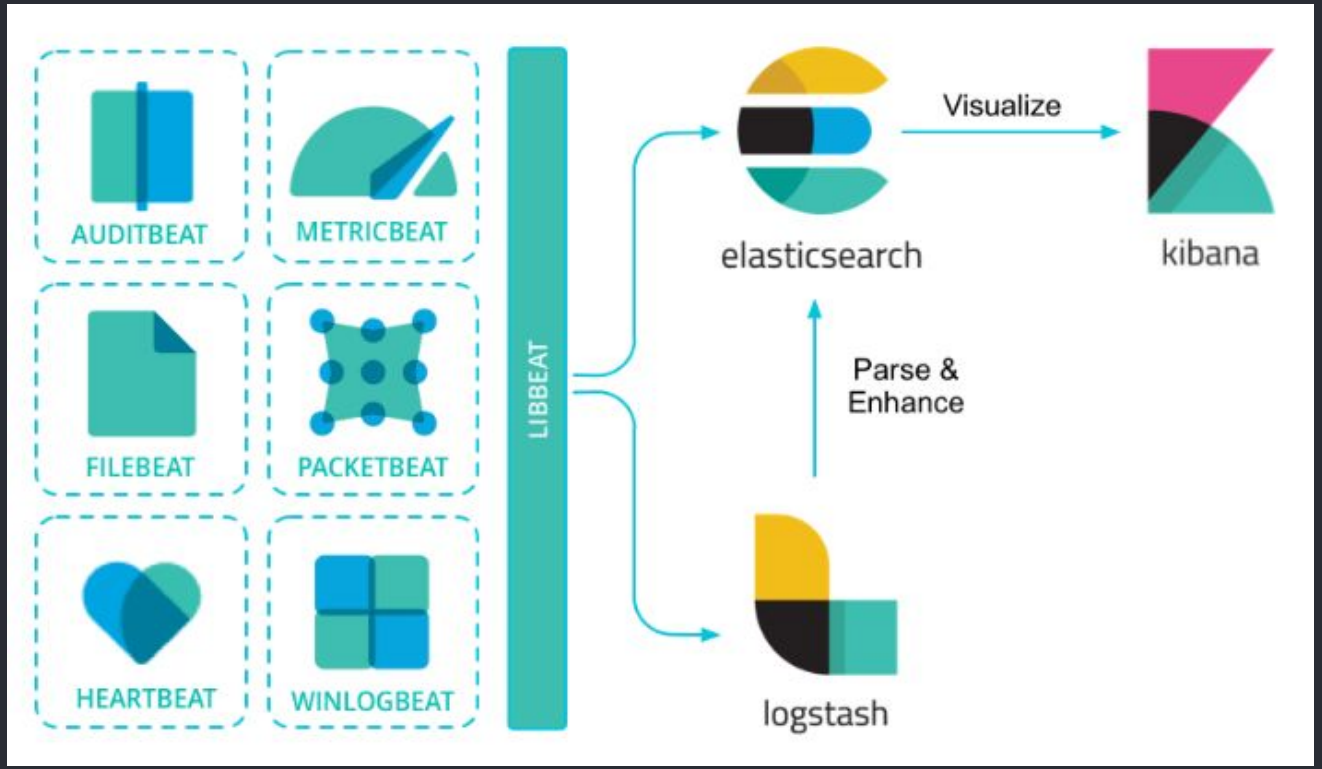
What now… Some prerequisites

- Do not let be limited by
  - Naming/schema and their assigned values
  - Value Length (bye powershell…)
  - Storage amount/size
  - Events per Second (EPS)
  - Log formatting -- CEF :(

- Data Table (Schema)
  - Of what original vendor/event/log calls a field that is mapped to what your database/SIEM now calls it

Winlogbeat & Filebeat for Endpoints

What are Beats?



Winlogbeat

- Ships Windows logs to Elasticsearch (or Logstash)
- Installed as a Windows Service (or ran as an “.exe”)
- Read position is persistent

Filebeat

- Generic log shipper
- Internal modules for parsing:
  - Apache
  - NGINX
  - MySQL
  - System
  - etc.

The Hunting ELK (HELK) Introduction

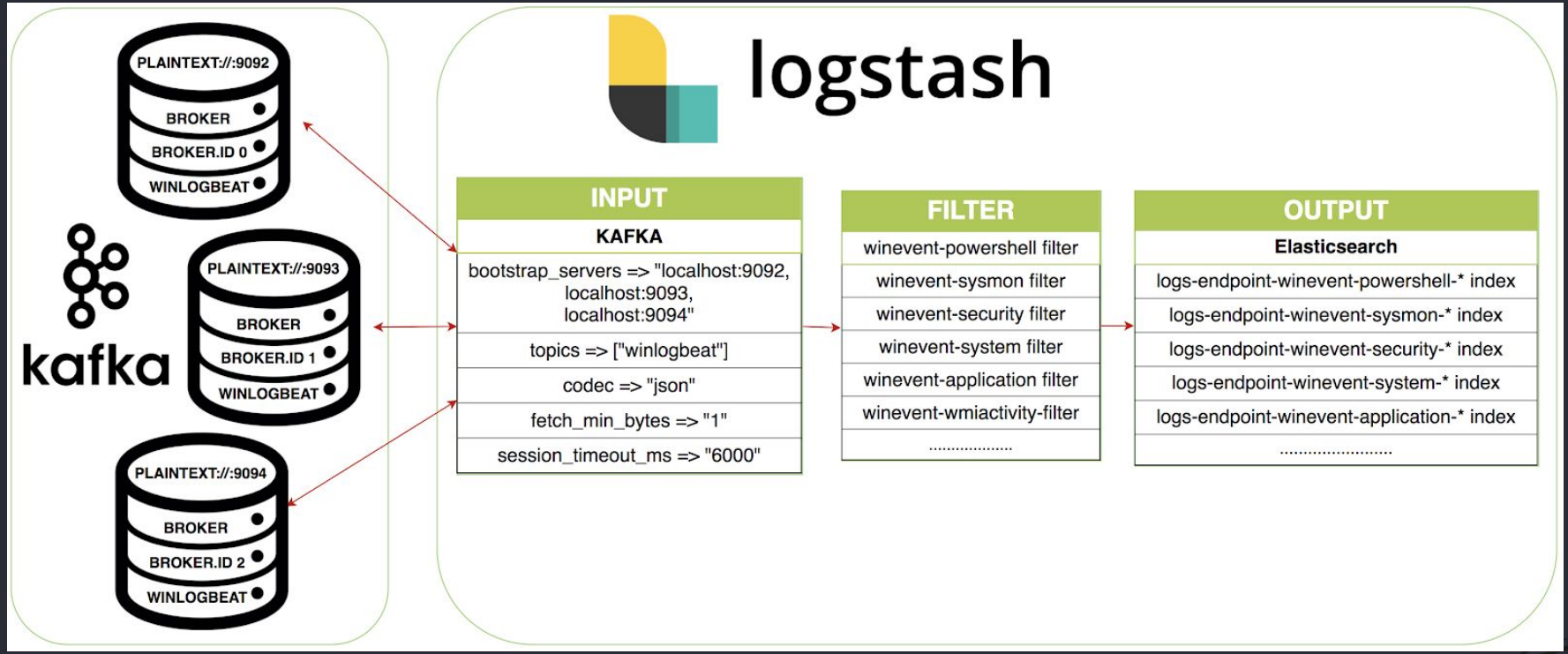
Perched | Endpoint Operator

Not official spokesperson for HELK

Roberto Rodriguez @Cyb3erWard0g

- Developer:
  - ThreatHunter-Playbook
  - Hunting ELK (HELK)
  - Invoke-ATTACKAPI
  - Mordor
- Adversary Detection Analyst @SpecterOps
- Former:
  - Capital One, Senior Threat Hunter

Data Transformation



https://github.com/Cyb3rWard0g/HELK/wiki/Logstash

PowerShell 4103 Parsed!

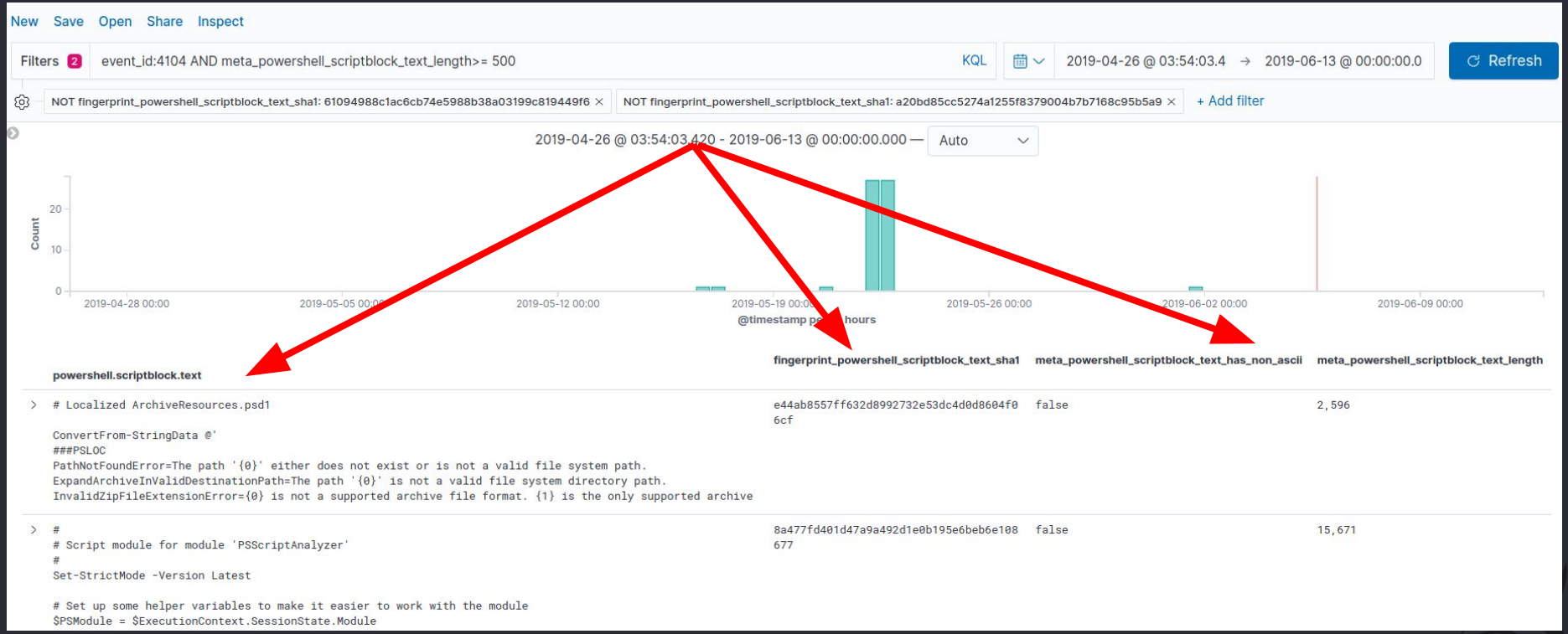
powershell.param.name.keyword: Ascending	Count
AllHosts	1
AutoLogonCount	1
BackupFile	1
BlockSizeBytes	1
CleanupUnneededContentFiles	1
ClientOnly	1
Directory	1
Dynamic	1
ExactMatch	1
ExePath	1
FoundName	1

param.value_nonalphanumeric
!=..,>=.(>=.->)
""""():".":_----_.::+$.($,$)+~~~~~~ [],+:
""--"$:\..-$:-$:_"-"$:\..-$:-$:_."-"$:\..-$:-$:_"--
""--"$:\..-$:--"$:\..-$:-."-"$:\..-$:-"--
"":
"":"",'\\
"":"()."\\:\\\\\..::\\+$.=[.]:: ($:_...\\+~~~~~~ (:)[],\\+:\\\\"\\
"":"-",

powershell.host.name	Count
ConsoleHost	6,355,657
Default Host	5,544,455
Chocolatey_PSHost	262,636
ServerRemoteHost	6,152
Windows PowerShell ISE Host	6,063
Visual Studio Code Host	4,750
Package Manager Host	650
Microsoft.BDD.Workbench.WorkbenchPSHost	177
WixHost	10

powershell.host.application
C:\Program Files\Docker\Docker\com.docker.service
C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.e: -NonInteractive -NoProfile -WindowStyle Hidden & C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules\Sml -Scenario Client
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
PowerShell -NoProfile -ExecutionPolicy bypass -command Impo 'C:\ProgramData\boxstarter\Boxstarter.Bootstrapper\boxstarte Boxstarter -RebootOk -NoPassword:$True
C:\WINDOWS\system32\WindowsPowerShell\v1.0\powershell.e: -NonInteractive -NoProfile -WindowStyle Hidden & C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules\Sml -Scenario Client
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.e

PowerShell 4104 Parsed!



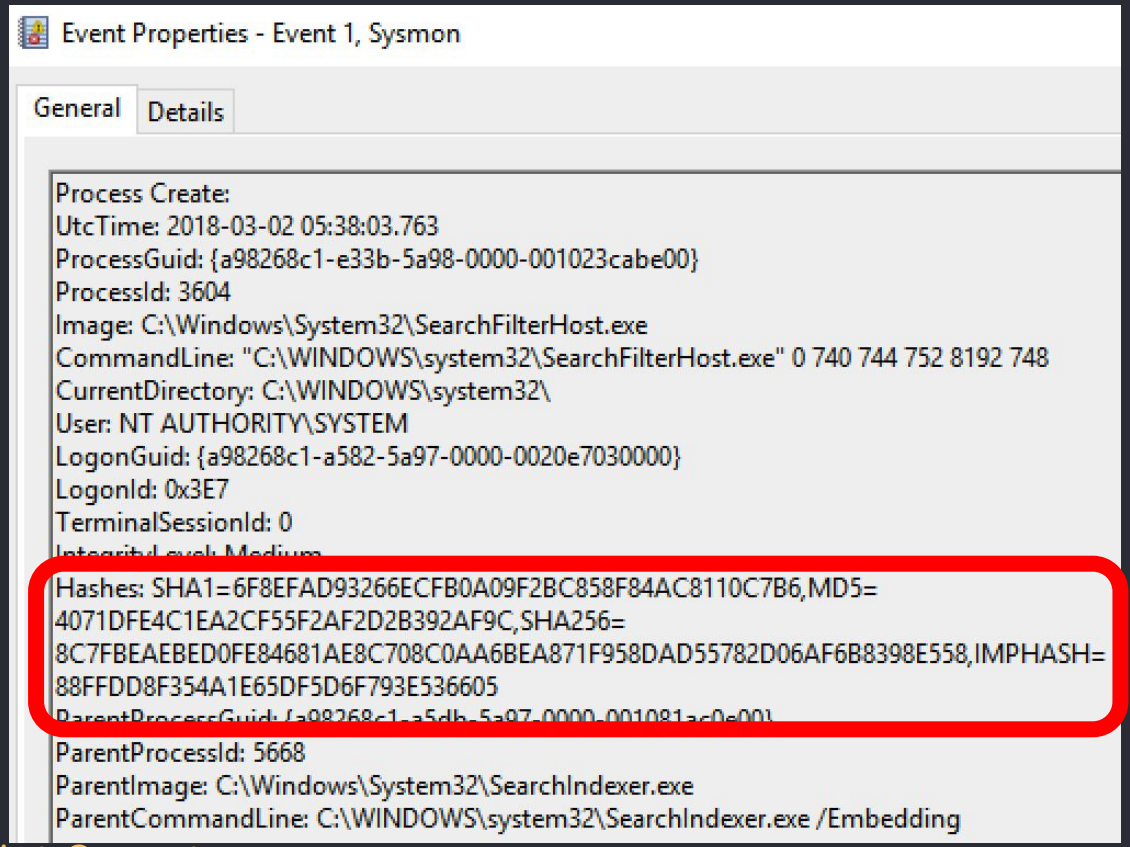
IP Routable/Private & RFC

```
# Private/RFC1918
if ip_address.start_with?( "10.", "192.168." )…
# (Local)link-local RFC3927
elsif ip_address.start_with?( "169.254." )…
# Loopback RFC1122-3.2.1.3
elsif ip_address.start_with?( "127." )…
# RFC1700
elsif ip_address.start_with?("0.")…
# IPv6 to IP4 anycast RFC3068
elsif ip_address.start_with?( "192.88.99." )…
# IPv6 to IP4 anycast RFC7535
elsif ip_address.start_with?( "192.31.196." )…
# IPv6 to IP4 anycast RFC7450, "Automatic Multicast Tunneling"
elsif ip_address.start_with?( "192.52.193" )…
#  Reserved RFC6890, RFC1122-3.2.1.3, RFC2544, RFC5737
elsif ip_address.start_with?( "0.", "192.0.0.", "192.0.1.", "192.0.2.", "192.18.
"203.0.113." )…
# Private/RFC-1918 -- continued -- 172.16.0.0-17.31.255.255
elsif ip_address.start_with?( "172." )…
# Private/RFC-1918 -- continued -- 100.64.0.1 - 100.127.255.254
elsif ip_address.start_with?( "100." )…
# The remaining possible NON public/routable IPs begin with 2 and are either mul
elsif ip_address.start_with?( "2" )…
# RFC1366, Public/Routable…
end
```


Disparate DataSets (ie: Bro <> Windows)

```
"mappings": {
  "properties": {
    "dst_ip_addr": {
      "type": "ip",
      "copy_to": "any_ip_addr"
    },
```

Sysmon Hashes Parser (Raw Log)



Sysmon Hashes Parser
(Logstash KV Filter Plugin)

```
kv {
  source => "[event_data][Hashes]"
  field_split => ","
  value_split => "="
  prefix => "hash_"
  transform_key => "lowercase"
}
```

Sysmon Hashes Parser (Results)

hash_imphash	88FFDD8F354A1E65DF5D6F793E536605
hash_md5	4071DFE4C1EA2CF55F2AF2D2B392AF9C
hash_sha1	6F8EFAD93266ECFB0A09F2BC858F84AC8110C7B6
hash_sha256	8C7FBEAEBED0FE84681AE8C708C0AA6BEA871F958DAD55782D06AF6B8398E558

Sysmon Hashes Visualization

hash_256	Count	uniq host	unique process_name
0915426F34077201AB2BB4FFE7F2E500BDEBB8B88B9DBA1FD5CAA96D8DB7D9E1	6	1	4
1342D275E058BA07524550BEACC7760E5CDDC0B3A32659F4C65346D3BE6515E3	6	1	4
32511D9B73CA53EE382AC4E727100885F562A1B0F1F4AD117ACEB506F887187C	6	1	4
556FF2B03495E2117223E5697B54253F30BD10ED3C67468947D79945168A624A	6	1	4
5F97027117657DAF3E268871C338F495D238A4F248E3327EAC5EA08F9E929701	6	1	4

Export: Raw Formatted

1 2 3 4 5 ... 10 »

Sysmon Hashes (EID 7 vs EID 1)

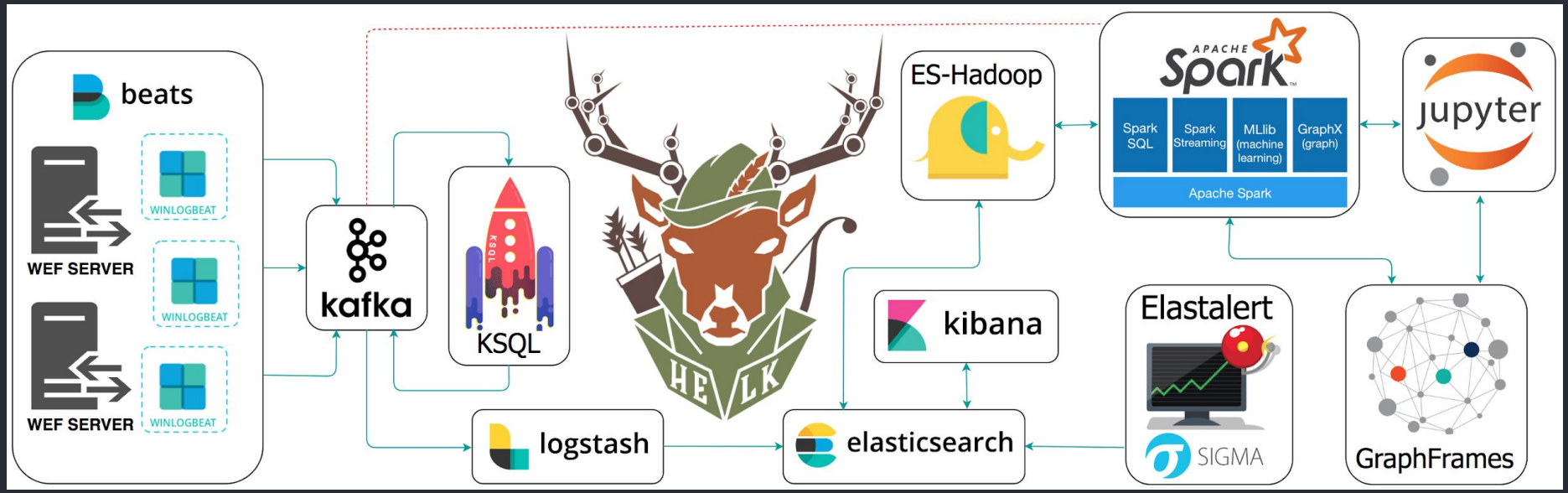
process_name	image_loaded	hash_sha256	event_id
SearchFilterHost.exe	C:\Windows\System32\kernel32.dll	7C76BB7AEC3C51164A49041E8A710C3C291BA1D30BAE07D126434A52A80D99E1	7
SearchProtocolHost.exe	C:\Windows\System32\kernel32.dll	7C76BB7AEC3C51164A49041E8A710C3C291BA1D30BAE07D126434A52A80D99E1	7
RuntimeBroker.exe	C:\Windows\System32\kernel32.dll	7C76BB7AEC3C51164A49041E8A710C3C291BA1D30BAE07D126434A52A80D99E1	7
HxTsr.exe	C:\Windows\System32\kernel32.dll	7C76BB7AEC3C51164A49041E8A710C3C291BA1D30BAE07D126434A52A80D99E1	7
SearchFilterHost.exe	C:\Windows\System32\kernel32.dll	7C76BB7AEC3C51164A49041E8A710C3C291BA1D30BAE07D126434A52A80D99E1	7
SearchProtocolHost.exe	C:\Windows\System32\kernel32.dll	7C76BB7AEC3C51164A49041E8A710C3C291BA1D30BAE07D126434A52A80D99E1	7

Global Schema

- Open Source Security Events Metadata (OSSEM)
  - https://github.com/Cyb3rWard0g/OSSEM

- Elastic Common Schema
  - https://www.elastic.co/guide/en/ecs/current/index.html

What The HELK? A Hunting ELK



HELK Scenario

- August 25th, 2018
- Domain Controller
- 2x Windows Machines (Executives)

Example 1 - Service Installs

Search Query:

- log_name:System AND event_id:7045

Event 7045:

- New service was installed

Example 2 - Mimikatz

PowerShell Dashboard:

- P.WinEvent.PowerShell

Base64 encoded PowerShell Application

Example 3 - Lateral Movement

Hunting Saved Search

- P.WinEvent. Channel:Security EID:4648 (Hunting)

Event 4648

- Attempted logon with explicit credentials

Example 4 - Rudimentary Log-ons

Search Query

- log_name:Security AND event_id:4672

Event 4672

- Account with “administrator equivalent” logs on

Example 5 - PowerShell Enumeration

Search Query

- log_name:Security AND event_id:4799

Event 4799

- A security-enabled local group membership was enumerated

Questions?

Endpoint Detection & Response

Baselining and Threat Hunting with PowerShell

Perched | Endpoint Operator

Course Outline

- EDR Overview
- Agent vs. Agentless Monitoring
- Baselining w/ PowerShell
- Threat Hunting w/ PowerShell
- Initial Incident Response with PowerShell

What is EDR?

- Detection
  - Collect
  - Record
  - Store
- Response
  - Investigate
  - Mitigate

Agent vs. Agentless Endpoint Monitoring

- Agent
  - Server + Endpoint Agent (Application)
  - e.g. Beats, GRR, Tanium, Carbon Black, etc.
- “Agentless”
  - No dedicated tools installed on the endpoints
  - e.g. log forwarding, PowerShell, manual response

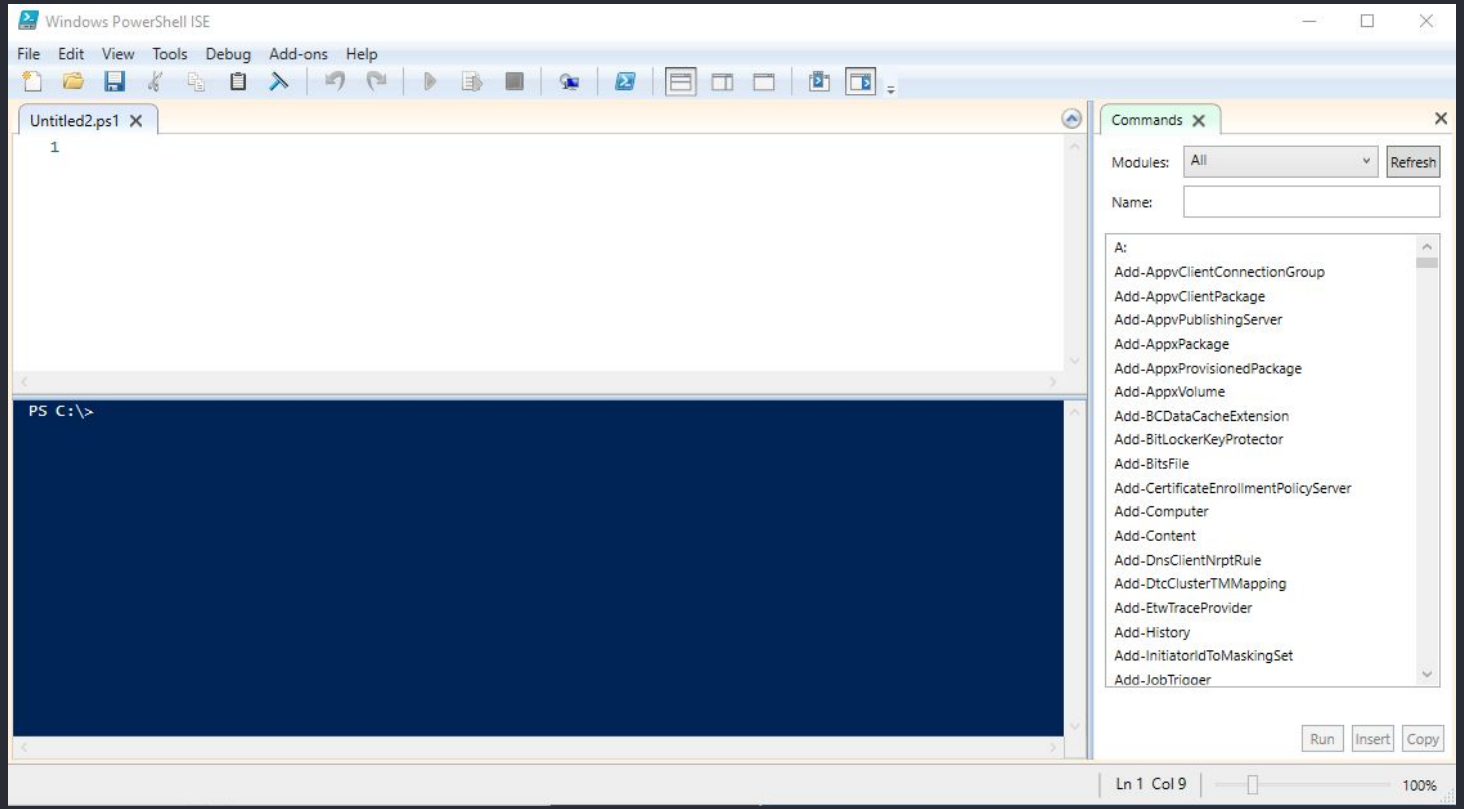
Baselining - What do you need to know?

- How many hosts do I have?
- What operating systems and versions?
- Are they connected to the domain?
- What things are different?

Exercise Disclaimer - PowerShell

- Task-based command line shell
- Scripting language built on .NET

PowerShell Integrated Scripting Environment (ISE)



How many hosts are alive?

- Ping-Sweep.ps1

```
1   # Ping-Sweep
2
3   1..255 | ForEach-Object{
4       Write-Output "172.25.26.$_"; ping -n 1 -w 100 172.25.26.$_
5   } | Select-String ttl
6
```

Exercise - Baselining with PowerShell

- How many hosts are alive on your subnet?

Exercise - Multi-threading PowerShell

```
1   # parallelSweep
2
3   workflow ParallelSweep {
4       foreach -parallel -throttlelimit 4 ($i in 1..255) {
5           ping -n 1 -w 100 172.25.26.$i
6       }
7   };
8   ParallelSweep | Select-String ttl
9
```

Exercise - Multithreading PowerShell

Making “jobs”

```powershell
# Making Jobs

foreach($i in 1..255){
    Start-Job -Name "Scanning Network" -ArgumentList $i -ScriptBlock{
        ping -n 1 -w 100 172.25.26.$i
    }
}

```

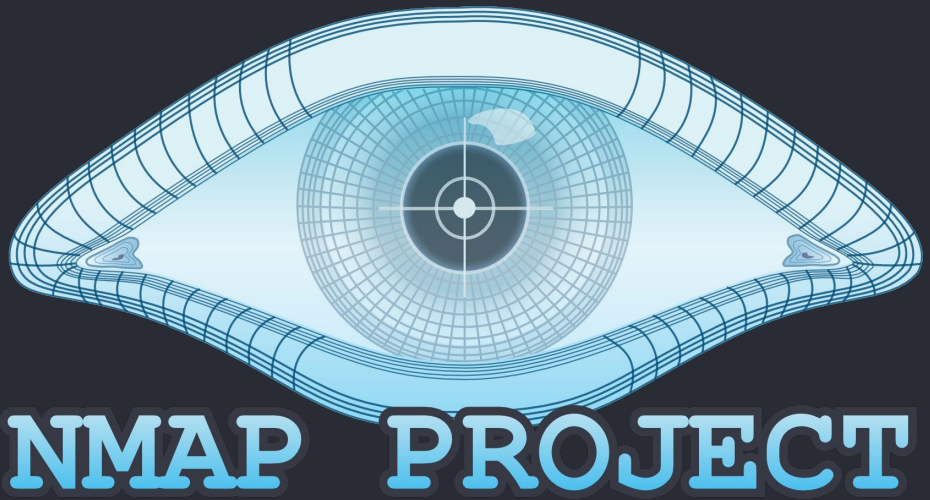

Port-Scanning

```
1   # Port-Scanning
2
3   Try{
4       $SMB_response = Test-NetConnection -ComputerName 172.25.26.101 `
5       -CommonTCPPort SMB -InformationLevel Quiet -ErrorAction Continue
6   }
7   Catch{
8      Write-Host "SMB Connection Failed"
9   }
10
```


PowerShell Objects and Properties

```
39          $properties = @{
40              IP4_address = $args[0]
41              SMB_Status = $SMB_response
42              HTTP_Status = $HTTP_response
43              WINRM_Status = $WINRM_response
44              RDP_Status = $RDP_response
45          }
46
47          New-Object -TypeName psobject -Property $properties
```


Exercise - Port-Scanning



Creating Reports

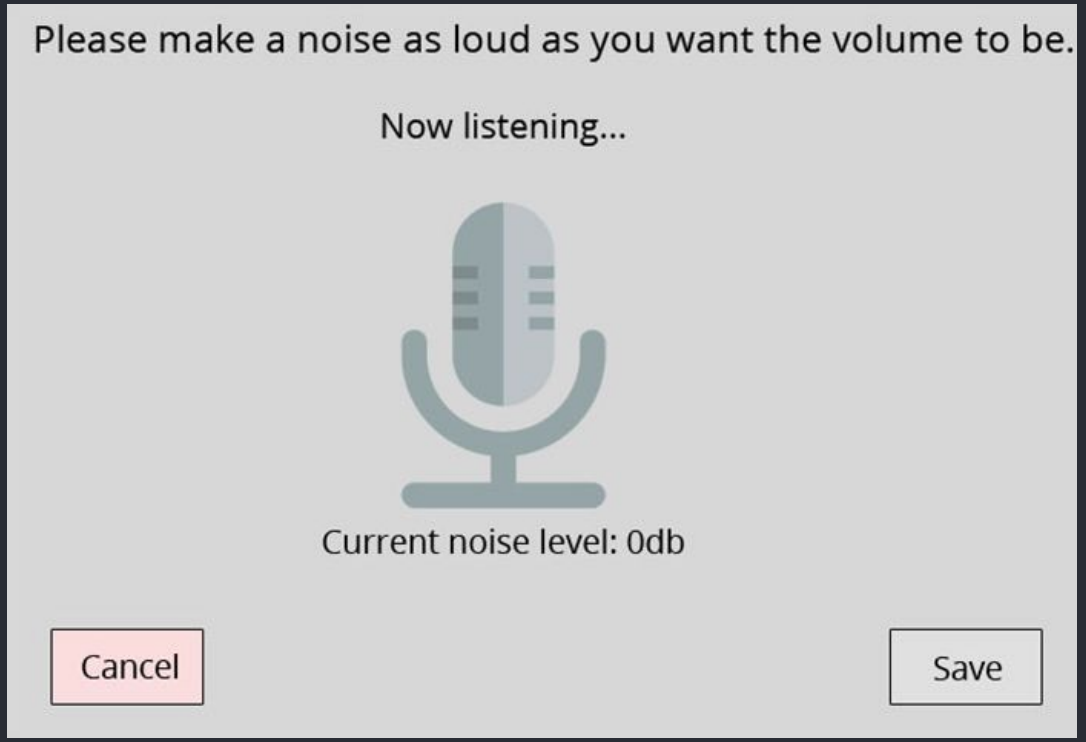
```
53    $report = Get-Job | Receive-Job -Wait -AutoRemoveJob
54
55    $date = Get-Date -Format yy_MM_dd_HHmmss
56    $current_dir = Get-Location
57
58    $report | Export-Clixml -path $current_dir\$date.xml
59    $report | ConvertTo-Html | Out-File $current_dir\$date.html
60    $report | ConvertTo-Json | Out-File $current_dir\$date.json
```


Exercise - Creating Reports

User Interaction

```
1   # User Interaction
2
3   $first_three = read-host "Enter the  first 3 octets of the target network"
4   $Starting_IP = read-host "Enter starting ip (last octet)"
5   $Ending_IP= read-host "Enter ending ip (last octet)"
6   $iprange = $Starting_IP..$Ending_IP
7
8   Foreach($ip in $iprange){
9       $I = $first_three+[string]$ip}
10
```


Exercise - User Interaction



Threat Hunting with PowerShell

- Can we compare objects from multiple computers?
- What kinds of information can I gather?
- Are there any computers running processes they shouldn’t?
- What things are different?

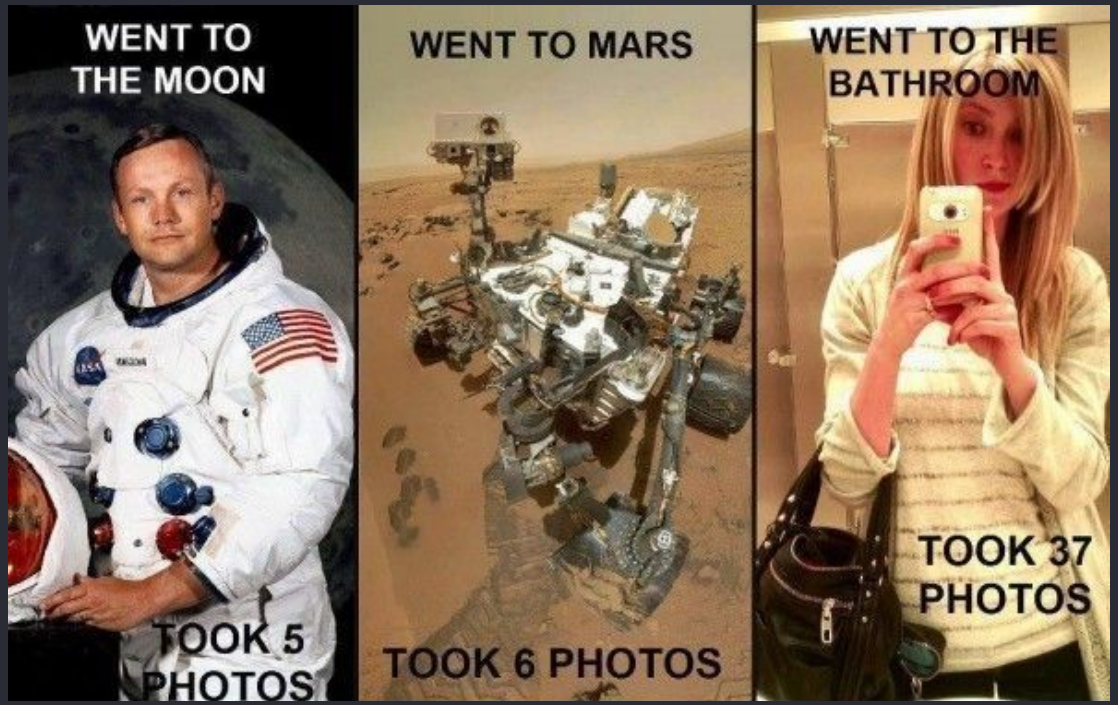
Compare Single Objects

```
PS C:\> $A = Get-Process

PS C:\> $B = Get-Process

PS C:\> Compare-Object $A $B
```


Output to Text Files

```
5   $OutFile = "$current_dir\Processes-$date.txt"
6
7   $A = Get-Process
8
9   $A | Out-File $OutFile
```


Exercise - Compare-Object



Comparing Multiple Files - Part 1/3

```
1   $date = Get-Date -Format yy_MM_dd_HHmmss
2
3   $current_dir = Get-Location
4
5   $OutFile = "$current_dir\Processes-$date.txt"
6
7   $A = Get-Process
8
9   $A | Out-File $OutFile
10  $A | Export-Clixml -Path $current_dir\Processes-$date.xml
```

Comparing Multiple Files - Part 2/3

```
4   $Path1 = "~\test1"
5
6   $OutFile = "$current_dir\Compare-$date.txt"
7
8   $A = Import-Clixml ~\baseline.xml
9
10  $Dir1 = Get-ChildItem -Path $Path1 -filter *.xml
```

Comparing Multiple Files - Part 3/3

```
12  foreach($File in $Dir1){
13      $B = Import-Clixml $File
14
15      "----------------------------------------" | Out-File $OutFile -Append
16      $File.Name | Out-File $OutFile -Append
17      "Comparison:" | Out-File $OutFile -Append
18
19      Compare-Object $A $B |
20      Format-Table -Auto |
21      Out-File $OutFile -Append
22  }
```


Exercise - Comparing Multiple Files

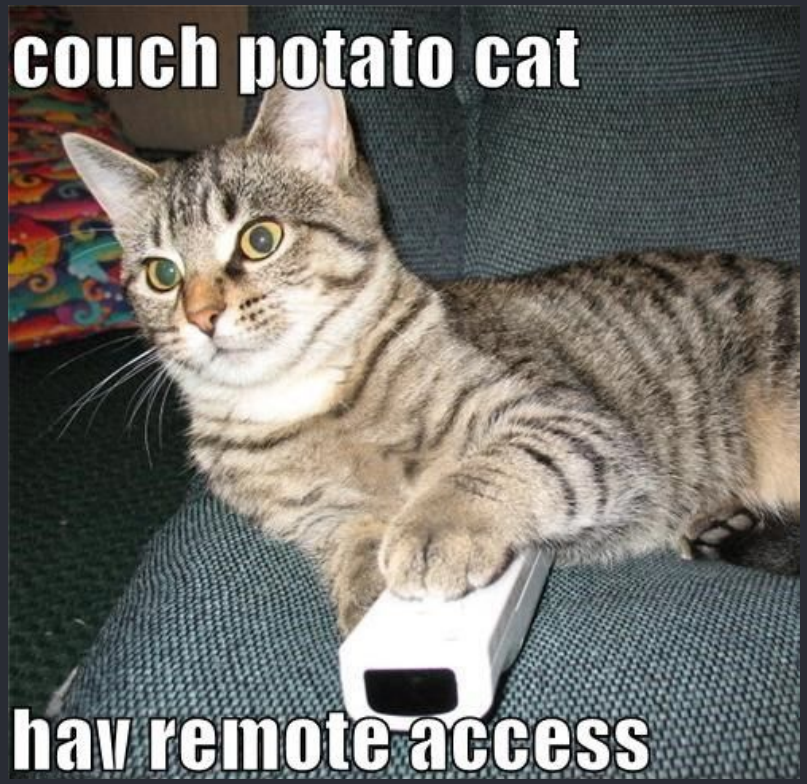
Running Scripts on Remote Computers

```
1  # Running Remote Scripts
2
3  Invoke-Command -FilePath C:\scripts\processes.ps1 -ComputerName 172.25.26.101
4
5  Invoke-Command -ComputerName 172.25.26.101 -Credential PERCHED\dcadmin -ScriptBlock{
6      Get-Process
7  }
8
```


Remote Background Jobs

```powershell
$s = New-PSSession -ComputerName DESKTOP-EF8F6KV, DESKTOP-FO7K018, DESKTOP-9LEI5SC

Invoke-Command -Session $s -ScriptBlock {Get-EventLog system} -AsJob

# Wait for the above jobs to finish!

$j = Get-Job

$j | Format-List -Property *

$results = $j | Receive-Job
```

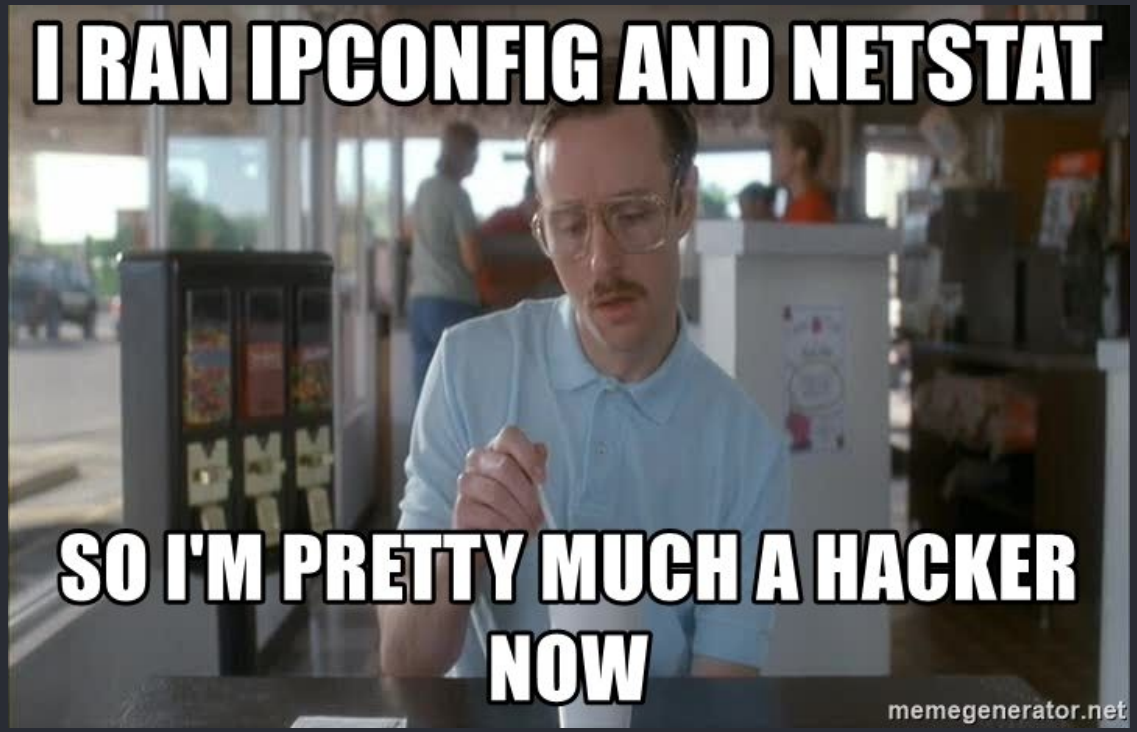
Exercise - Scripting Remotely



netstat

```
PS C:\Users\user02> netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       404
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       11568
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING       6252
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:7680           0.0.0.0:0              LISTENING       5404
  TCP    0.0.0.0:9030           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       612
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       1568
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1296
  TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING       512
  TCP    0.0.0.0:49669          0.0.0.0:0              LISTENING       708
  TCP    0.0.0.0:49697          0.0.0.0:0              LISTENING       684
  TCP    0.0.0.0:49743          0.0.0.0:0              LISTENING       708
  TCP    0.0.0.0:53493          0.0.0.0:0              LISTENING       13108
  TCP    0.0.0.0:54028          0.0.0.0:0              LISTENING       10640
  TCP    127.0.0.1:54093        0.0.0.0:0              LISTENING       8760
  TCP    127.0.0.1:54454        0.0.0.0:0              LISTENING       5620
  TCP    172.25.26.44:139       0.0.0.0:0              LISTENING       4
  TCP    172.25.26.44:53060     52.242.211.89:443      ESTABLISHED     3488
  TCP    172.25.26.44:53821     23.7.93.170:80         TIME_WAIT       0
  TCP    172.25.26.44:53823     23.200.143.59:443      CLOSE_WAIT      6088
  TCP    172.25.26.44:53838     172.25.26.10:135       TIME_WAIT       0
  TCP    172.25.26.44:53839     172.25.26.10:135       TIME_WAIT       0
```


Exercise - netstat



tasklist

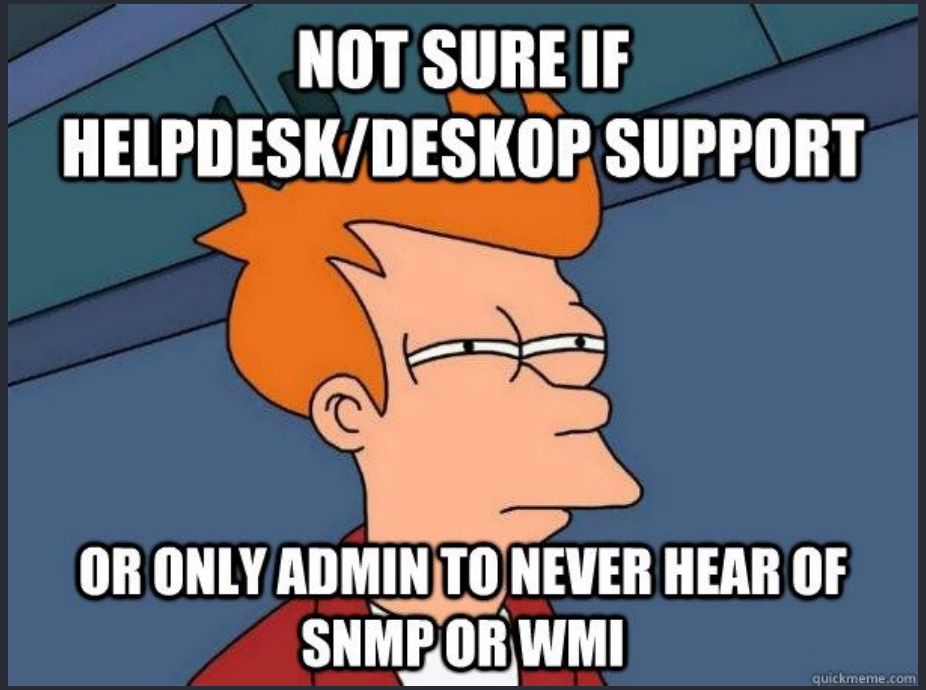
```
PS C:\Users\user02> tasklist /svc /fi "PID EQ 6252"

Image Name                     PID Services
========================= ======== ============================================
svchost.exe                   6252 CDPSvc
```


Exercise - tasklist

WMI

```
PS C:\Users\user02> Get-WmiObject -Class Win32_Service -Filter "name='CDPSvc'" | select *


PSComputerName          : DESKTOP-VF9R2P9
Name                    : CDPSvc
Status                  : OK
ExitCode                : 0
DesktopInteract         : False
ErrorControl            : Normal
PathName                : C:\WINDOWS\system32\svchost.exe -k LocalService -p
ServiceType             : Share Process
StartMode               : Auto
__GENUS                 : 2
__CLASS                 : Win32_Service
__SUPERCLASS            : Win32_BaseService
__DYNASTY               : CIM_ManagedSystemElement
__RELPATH               : Win32_Service.Name="CDPSvc"
__PROPERTY_COUNT        : 26
__DERIVATION            : {Win32_BaseService, CIM_Service, CIM_LogicalElement, CIM_ManagedSystemElement}
__SERVER                : DESKTOP-VF9R2P9
__NAMESPACE             : root\cimv2
__PATH                  : \\DESKTOP-VF9R2P9\root\cimv2:Win32_Service.Name="CDPSvc"
AcceptPause             : False
AcceptStop              : True
Caption                 : Connected Devices Platform Service
CheckPoint              : 0
CreationClassName       : Win32_Service
DelayedAutoStart        : True
Description             : This service is used for Connected Devices Platform scenarios
DisplayName             : Connected Devices Platform Service
```


Exercise - WMI



Questions?

Google Rapid Response (GRR)

Course Outline

- Overview
- Installation/Engineering
- Flows and Hunts
- Virtual File System
- Artifacts
- Additional Configurations

GRR Rapid Response (GRR)

- Incident Response framework focused on live forensics
- Server w/ agents on all the endpoints

Challenges at scale

- Joe saw something weird, check his machine
- Forensically acquire 25 machines for analysis
- Tell me if this machine is compromised, or all 10,000 of them!
- New APT report is released

GRR Server Overview

- Collection system
- Enterprise endpoint hunting
- Automated scheduling
- Export & output capabilities

GRR Client Overview

- Linux, OS X, and Windows
- Remote memory analysis w/ YARA
- Search capable for files or Windows registry
- SleuthKit (TSK)
- Detailed monitoring (CPU, memory, IO usage)

Securing access (important!)

- Root access to GRR server
- Direct write access to GRR datastore

Exercise - Installing GRR Client

Life of a GRR Client

- Initial key generation
- Heart beat
- Transaction log
- Memory limit
- CPU limit
- Advanced Settings

GRR Client Protection

- Obfuscation
  - service and binary names
  - registry keys
  - external watchers
- Enrollment
  - intel loss
  - “Well Known” flows

Investigating with GRR

- Virtual File System
- Flows
- Hunts
- Artifacts

Client-Server Communication

- Clients poll on 10-minute intervals
- Enters “fast-poll” mode after requested work
- Poll is an HTTP request
  - Signed and encrypted payload

Searching for Clients

- Hostname
- FQDN
- MAC Address
- IP Address
- User
- Label
- Time of Last Data Update

Exercise - Searching & Interpreting Client Results

GRR Flows

- Server-side code entities that call client actions
- CAEnroler
- Interrogate
- File Finder

Exercise - Running a Basic Flow

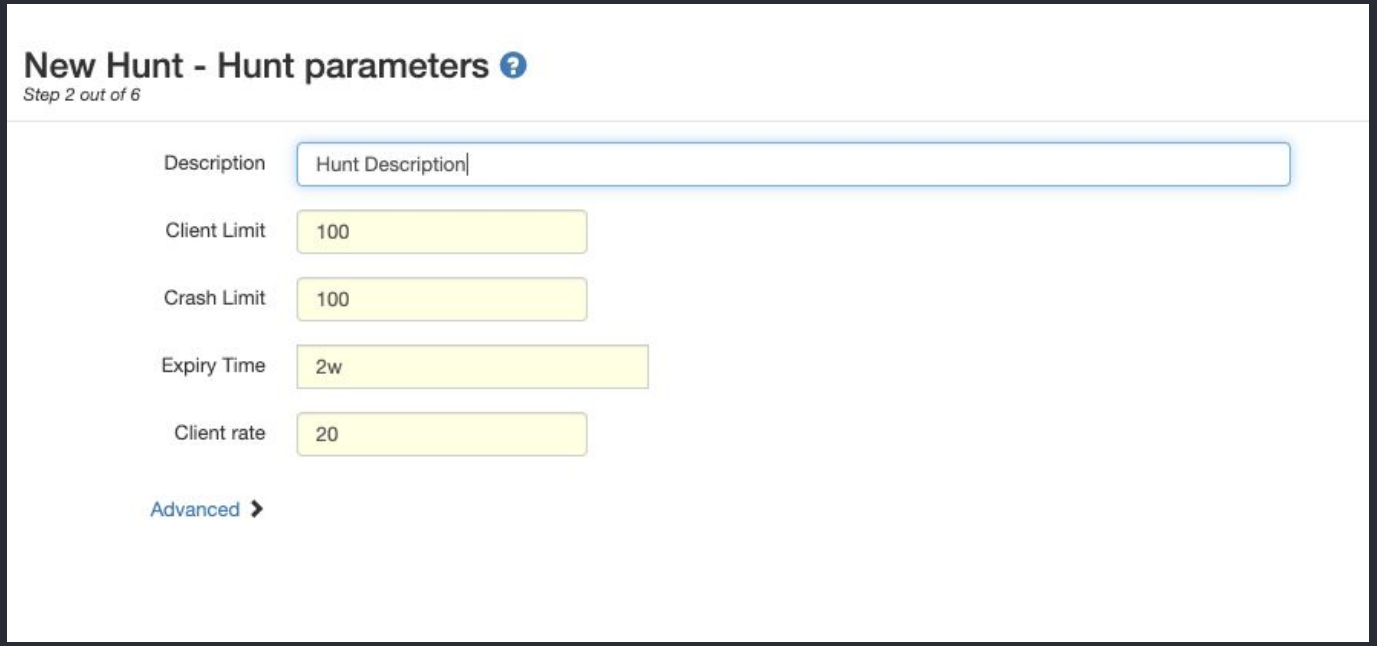
Virtual File System

- Unified view of the server's data store
- VFS Tree
  - fs/os
  - fs/tsk
  - registry
- Refresh and Timelines

Understanding Hunts

- Hunt = Many Flows!
- Use an existing and tested flow

Hunt Parameters



Hunt Limits

- Individual Client Limits
  - 600 cpu seconds / client
  - 100 MB of network traffic / client
- Limits of Average Resource Usage (>1,000 clients)
  - 1000 results on average / client
  - 60 cpu seconds on average / client
  - 10 MB of network traffic on average / client

Exercise - Hunt

Artifacts

- Framework for grouping collection
- Location and format varies across systems
- ArtifactCollectorFlow

Cron Jobs

- Default jobs perform periodical cleanup and maintenance tasks.
- Can setup cron jobs for hunts

Importing the NSRL

- National Software Reference Library (NSRL) by NIST
- Reduces the scope of analysis
- Whitelisting approach

Questions?