

# Trawling the Security Data Marsh for Windows Logs

Mason Willinger

Paul Poputa-Clean

# About Us

---

## Mason Willinger

- Microsoft / SANS certs
- Directory Services / Identity background
- Set-ScriptingLanguage -Name PowerShell
- Banjoist wannabe



## Paul Poputa-Clean - @p4ulpc

- SANS certs
- Big Data amateur (know just enough to be dangerous)
- Fluent in Romanian, English, Python
- Love most things on 2 wheels or out in the wilderness



# Why Are We Here – Problem

No self-respecting attacker is using malware

Better detection drives attackers to be better

How do we detect malware-less intrusions without an ETDR?

- Windows PowerShell (4103, 4104, 4105, 4106)
- WMI logs (5861)

## APT BINGO



Aliens



# Windows Logging 101

---

## Windows Event Forwarding

- **AGENTLESS, FREE**, underutilized
- Built-in to Windows
- Kerberos
- Almost stable

## Event Collectors

- Need at least one server, but can be distributed
- Subscription definitions

## Group Policy

- Easiest way to configure many endpoints
- Enhanced PowerShell logging!



# Windows Logging 102

## Where to collect logs?

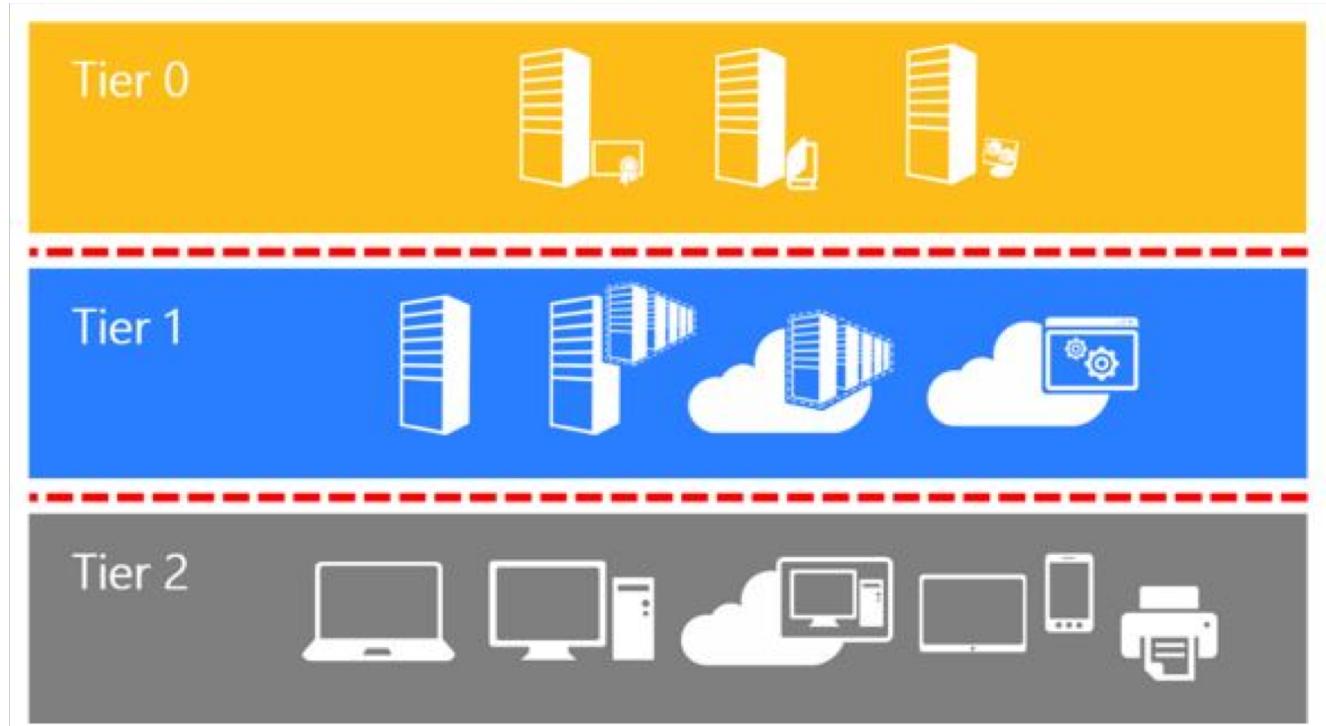
- Domain controllers
- Servers
- Client computers

## Why?

- Tier 0 assets – authentication and authorization nexus
- Workhorses and data storage
- Interface with the rest of the world

## How?

- Event collectors
- Group Policy



# Stage 1 - Event Log

The screenshot shows the Windows Event Viewer interface. The left pane displays a tree view of logs: Event Viewer (Custom Views, Windows Logs (Application, Security, Setup, System, Forwarded Events), Applications and Services Logs). The right pane shows a list of 'Forwarded Events' with 18,151,807 entries. The first few entries are:

Level	Date and Time	Source	Event ID	Task Category	Log	Computer
Information	9/25/2018 5:01:29 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:29 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:29 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:29 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:18 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:18 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:18 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:18 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:07 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:07 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:07 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	
Information	9/25/2018 5:01:07 PM	Microsoft Windows security auditing.	4688	Process Creation	Security	

Below the table, a specific event is selected: Event 4688, Microsoft Windows security auditing. The 'General' tab is selected, showing details about a new process creation:

A new process has been created.

**Subject:**

Security ID:	S-1-5-18
Account Name:	
Account Domain:	CORP
Logon ID:	0x3e7

**Process Information:**

New Process ID:	0x969e4
New Process Name:	C:\Windows\System32\conhost.exe
Token Elevation Type:	TokenElevationTypeDefault (1)
Creator Process ID:	0x348
Process Command Line:	

# Stage 2 - PowerShell

---

View all events in the ForwardedEvents log:

```
PS C:\> Get-WinEvent -LogName ForwardedEvents
```

Pull event 4104 from the ForwardedEvents log:

```
PS C:\> Get-WinEvent -FilterHashtable @{Logname="ForwardedEvents"; ID=4104}
```

Search for events containing the string "USB" in the ForwardedEvents log:

```
PS C:\> Get-WinEvent -FilterHashtable @{Logname="ForwardedEvents"} | `  
    Where {$_.Message -like "*USB*"}  
`
```

'grep'-style search for lines of events containing the case insensitive string "USB" in forwarded events:

```
PS C:\> Get-WinEvent -FilterHashtable @{Logname="ForwardedEvents"} | fl | findstr /i USB
```

Pull all errors (level=2) from forwarded events:

```
PS C:\> Get-WinEvent -FilterHashtable @{Logname="ForwardedEvents"; level=2}
```

# Stage 3 - Kibana

- Start looking for interesting things:
  - Encoded commands: event\_id:4103 AND "-encodedCommand"

```
Severity = Informational
Host Name = ConsoleHost
Host Version = 5.1.16299.492
Host ID = 1c9c81d2-0539-41d4-93d0-982cf2ac789
Host Application = c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe -noprofile -nologo -noninteractive -EncodedCommand aQBmACgAIAAoAGcAZQb0AC0AZQb4AGUAYwB1AHQAaQbVAG4AcABvAGwAaQbJAHKAIAAtAHMAYwBvAHAAZQAgAGwAbwBjA
GEAbAbtAGEAYwBaAgkAbgBLACKAIAAtAGUAcQAgACIAdQBuAGQAQb8mAGkAbgBLAGQAIgAgACkAIAB7ACAAJgAgACIAJABLAG4AdgA6AHcAaQBuAGQAaQByAFwAcwB5AHMAdABL
AG0AMwAyAFwAdwBpAG4AZABvAHcAcwBwAG8AdwBLAHIAcwBoAGUAbABsAFwAdgAxAC4AMABcAHAAbwB3AGUAcgBzAGgAZQ
BsAGwALgBLAHgAZQAIiACAALQBuAG8AcAByAG8AzgBpAGwAZQAgAC0AbgBvAGwAbwBnAG8AIAAtAG4AbwBuAGkAbgB0AGUAcgBhAGMAdABpAHYAZQAgAC0AZQb4AGUAYwB1AHQAaQbVAG4AcABvAGwAaQbJAHKAIABhAGwAbABzAGkAzwBuAGUAZAAgAC0AYwBvAG0AbQbHAG4AZAAgAHMAZQb0AC0AZQb4AGU
AYwB1AHQAaQbVAG4AcABvAGwAaQbJAHKAIABhAGwAbABzAGkAzwBuAGUAZAAgAdSAlAAgACYAIAAiACQAZQBuAHYA0gB3AGkAbgBkAGkAcgBcAHMAeQbZAhcAbwB3ADYANABcAhcAaQBuAGQAbwB3AHM
AcBvAhcAZQByAHMAaAbLAGwAbABcAHYAMQauADAAXABwAG8AdwBLAHIAcwBoAGUAbABsAC4AZQb4AGUAIgAgAC0AbgBvAHAAcgbVAGYAAqBsaGUAIAAtAG4AbwBsAG8AzwBvACAALQBuAG8AbgBpAG4AdABLAHIAYQbjAHQAaQb2AGUAIAAtAGUAeAbLAGMAdQb0AGkAbwBuAHAAbwBsAGkAYwB5ACAAYQbsAGwAcwBpAGcAbgBLAGQAIAAtAGMAbwBtAG0AYQb
uAGQAIABzAGUAdAAtAGUAcBLAGMAdQb0AGkAbwBwAHAAbwBsAGkAYwB5ACAAYQbsAGwAcwBpAGcAbgBLAGQAIAAtAGMAbwBtAG0AYQb
uAGQAIABzAGUAdAAtAGUAcBLAGMAdQb0AGkAb
Engine Version = 5.1.16299.492
Runspace ID = fda63e13-473e-412a-8c3c-2aae0fbfd11c7
Pipeline ID = 1
Command Name =
Command Type = Script
Script Name =
Command Path =
Sequence Number = 20
```

- Decoded Script Blocks (pick the time window for previous events and the right host): event\_id:4104

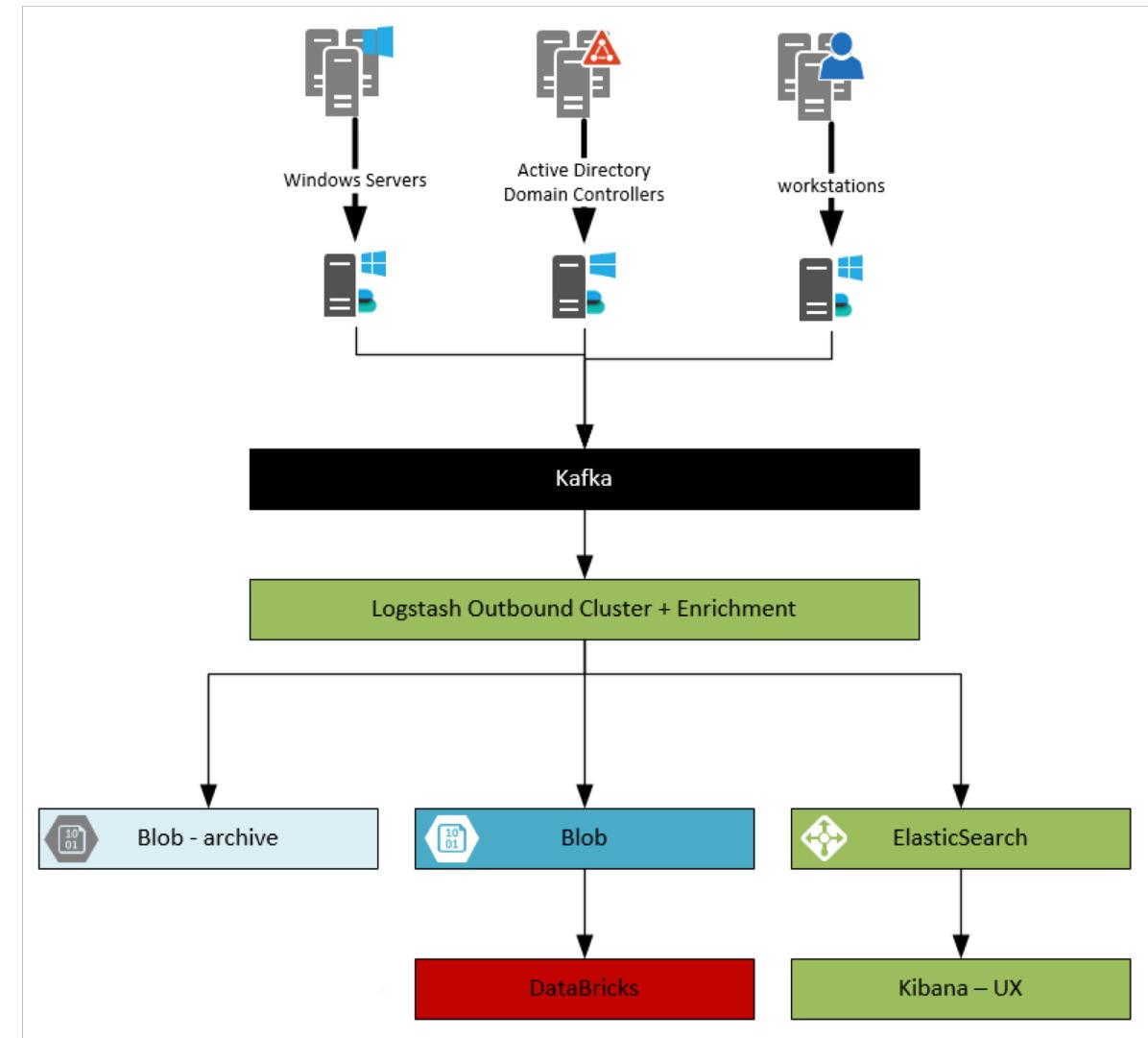
event\_data.ScriptBlockText

```
$global:?

if( (get-executionpolicy -scope localmachine) -eq "undefined" ) { & "$env:windir\system32\windowspowershell\v1.0\powershell.exe"
-noprofile -nologo -noninteractive -executionpolicy allsigned -command set-executionpolicy allsigned ; &
"$env:windir\syswow64\windowspowershell\v1.0\powershell.exe" -noprofile -nologo -noninteractive -executionpolicy allsigned -
command set-executionpolicy allsigned }
```

# Lambda Architecture for Windows Logs

- Three main collectors:
  - Domain Controllers
  - Servers
  - Client Computers
- Winlogbeats sends events directly to Kafka queue
- Logstash cluster picks up from Kafka and forwards to Elastic Search and saves JSON + gzipped JSON to blob (binary store)
- Using Databricks to selectively analyze JSON from blob



# Stage 4 – DataBricks (i.e Play Pretend Data Science)

```
1 wdf = spark.read.json("wasbs://[REDACTED].core.windows.net/[REDACTED]/2018-08-25").where((col('event_id') == 4103))  
2 wdf.groupBy('user.name').count().orderBy(col('count').desc()).show()
```

▼ (1) Spark Jobs

▼ Job 39 [View](#) (Stages: 2/2)  
Stage 51: 4339/4339 [i](#)  
Stage 52: 200/200 [i](#)

name	count
SYSTEM	10235011
xsvc	405741
xsvc	60467
xsvc	59696
xsvc	54975
xsvc	45365
xsvc	44984
xsvc	44936
xsvcs	34778
xsvcs	20956
xsvcs	18843
xsvc	7066
xsvcs	6544
xsvcs	5825
xco	5445
xco	5185
xsvc	4612
xsvc	4304
xsvcs	3467
xsvcs	2891

only showing top 20 rows

```
1 display(wdf.groupBy('user.name').count().orderBy(col('count').desc()).take(5))
```

▼ (10) Spark Jobs

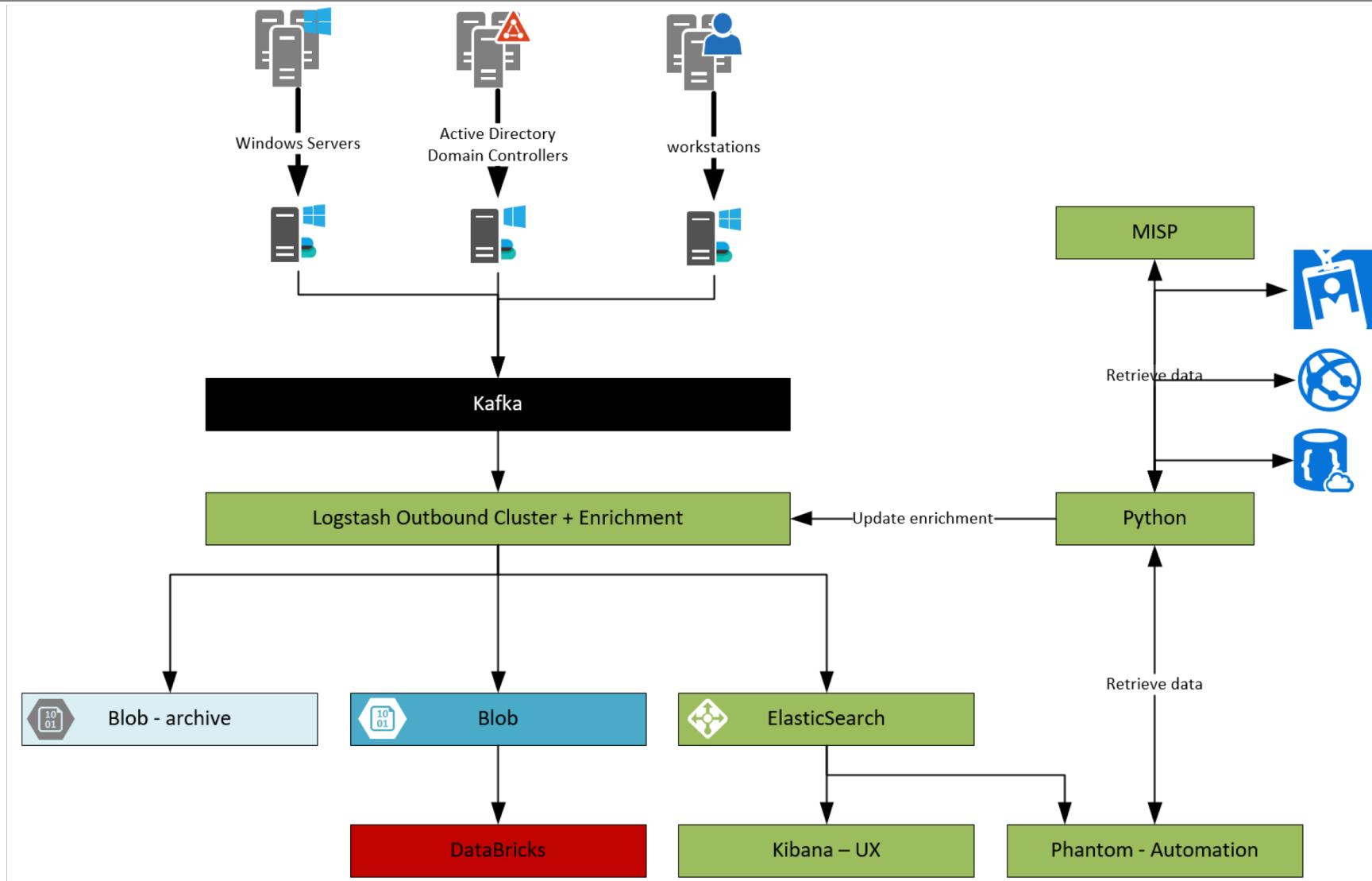
▼ Job 48 [View](#) (Stages: 2/2)  
Stage 69: 4339/4339 [i](#)  
Stage 70: 200/200 [i](#)

► Job 49 [View](#) (Stages: 1/1)  
► Job 50 [View](#) (Stages: 1/1)  
► Job 51 [View](#) (Stages: 1/1)  
► Job 52 [View](#) (Stages: 1/1)  
► Job 53 [View](#) (Stages: 1/1)  
► Job 54 [View](#) (Stages: 1/1)  
► Job 55 [View](#) (Stages: 1/1)  
► Job 56 [View](#) (Stages: 1/1)  
► Job 57 [View](#) (Stages: 1/1)

name

A donut chart titled "name" showing the distribution of user names. The largest segment is labeled "SYSTEM" at 95%. A small segment is labeled "xsvcs" at 4%, and another very small segment is labeled "xco" at 1%. Other user names like "xsvcs", "xsvcs" are represented by the remaining 1%.

# Analysis Lifecycle



# Lessons We Learned

---

Windows logging is really cool, but not easy:

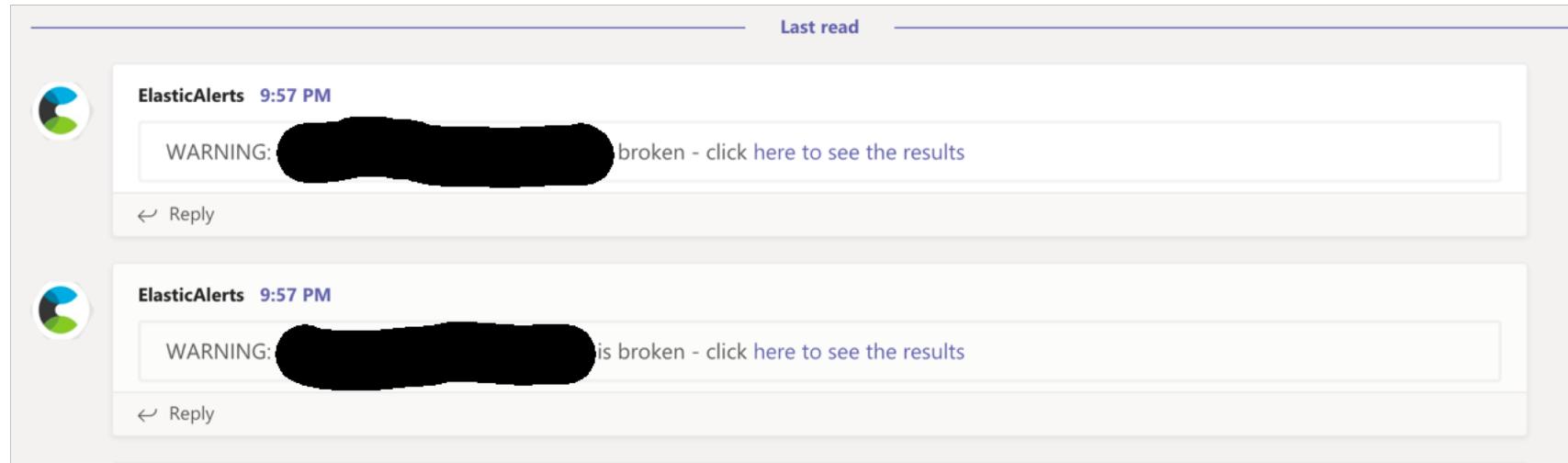
- Bi-directional communication requires synchronization between clients and WEF server
  - A few thousand clients will take a while to update, or sometimes never will
- Sometimes services just decide to stop or zombie out
- WEF collects historical data as far as the clients have it (years), hundreds of indices will slow down ingestion
- Case sensitivity (thanks Microsoft)

Kafka is awesome if you know what you're doing :

- Awesome throughput – Less is more regarding number of topics
- Elastic queueing, but make sure you don't fill the disk before rolling over

ELK ingestion is an art:

- EPS depends on a lot of factors:
  - Number of concurrent indices
  - Index optimization
- Alerting is a fun one as well
  - Watcher versus logstash
  - Email versus webhooks



# Thank You!

---

Paul PC - @paulpc

[paul.poputa-clean@dvn.com](mailto:paul.poputa-clean@dvn.com)

Mason Willinger

[mason.willinger@dvn.com](mailto:mason.willinger@dvn.com)

More Info

<https://bit.ly/2NWwfSB>



# Bibliography

---

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

Setting up a source initiated subscription: [https://msdn.microsoft.com/en-us/library/bb870973\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/bb870973(v=vs.85).aspx)

Tiered Subscriptions: <https://blogs.msdn.microsoft.com/canberrapfe/2015/09/21/diy-client-monitoring-setting-up-tiered-event-forwarding/>

Palantir's take on WEF: <https://medium.com/@palantir/windows-event-forwarding-for-network-defense-cb208d5ff86f> (blog)  
and <https://github.com/palantir/windows-event-forwarding> (code)

Microsoft's suggestions for intrusion detection using WEF: <https://docs.microsoft.com/en-us/windows/security/threat-protection/use-windows-event-forwarding-to-assist-in-intrusion-detection>

Intro on using WEF: <https://hackernoon.com/the-windows-event-forwarding-survival-guide-2010db7a68c4>

neu5ron's work on WEF: <https://github.com/neu5ron/WinLogsZero2Hero>

HELK - collecting windows events with the ELK stack: <https://github.com/Cyb3rWard0g/HELK/tree/master/helk-logstash/pipeline>

elastic.co's configuration: <https://www.elastic.co/guide/en/beats/winlogbeat/current/configuration-winlogbeat-options.html>

adsecurity.org's BlackHat 2018 presentation on AD security, including noteworthy event IDs: <https://adsecurity.org/wp-content/uploads/2018/08/us-18-Metcalf-From-Workstation-To-Domain-Admin-Why-Secure-Administration-Isnt-Secure-Final.pdf>

FireEye article about PowerShell logging - [https://www.fireeye.com/blog/threat-research/2016/02/greater\\_visibilityt.html](https://www.fireeye.com/blog/threat-research/2016/02/greater_visibilityt.html)

Black Hills Information Security article on PowerShell logging for the Blue Team - <https://www.blackhillsinfosec.com/powershell-logging-blue-team/>

Chad Tilbury talk at SANS Security: [https://www.dropbox.com/s/9nqw3m2csh1cmu3/Investigating\\_WMI\\_Attacks\\_Tilbury.pdf?dl=0](https://www.dropbox.com/s/9nqw3m2csh1cmu3/Investigating_WMI_Attacks_Tilbury.pdf?dl=0)

SANS PowerShell Get-WinEvent cheat sheet - <https://wiki.sans.blue/#!Tools/Get-WinEvent.md>