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

- 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

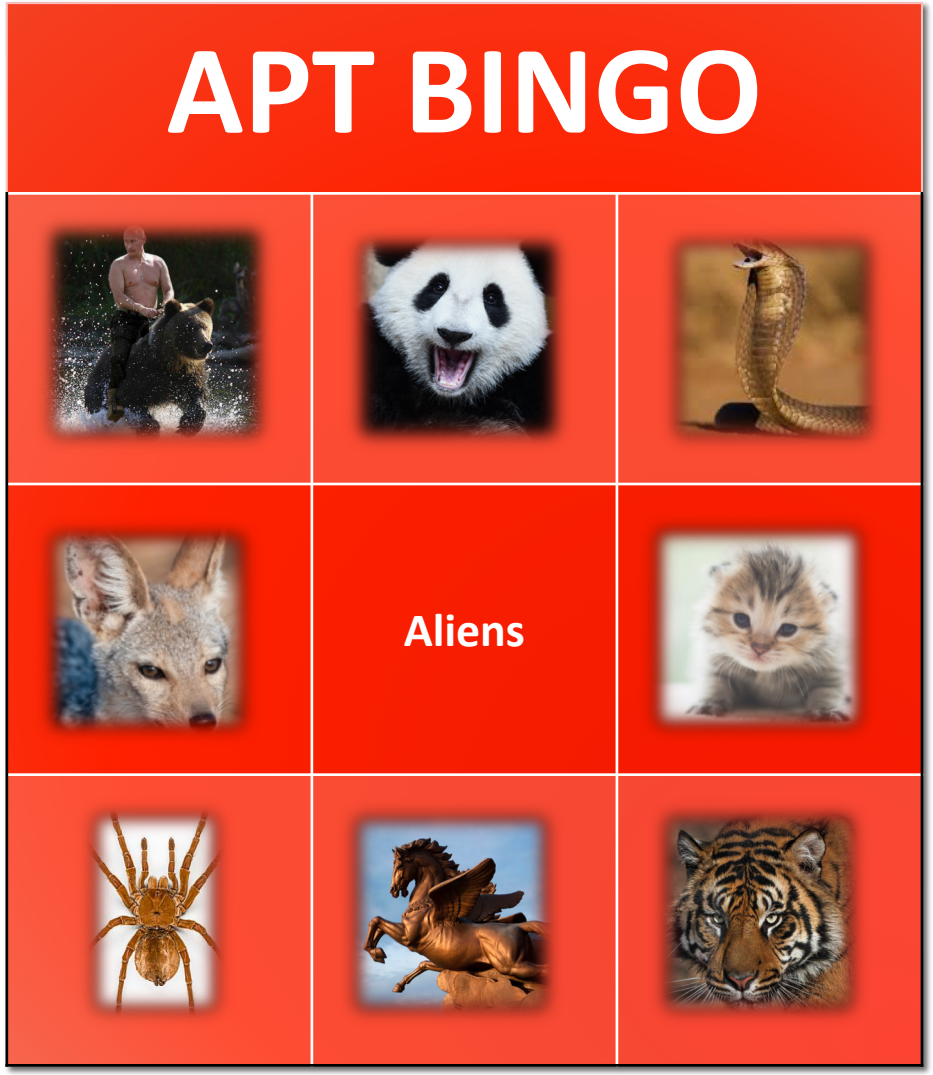
Why Are We Here – Problem

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)



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?

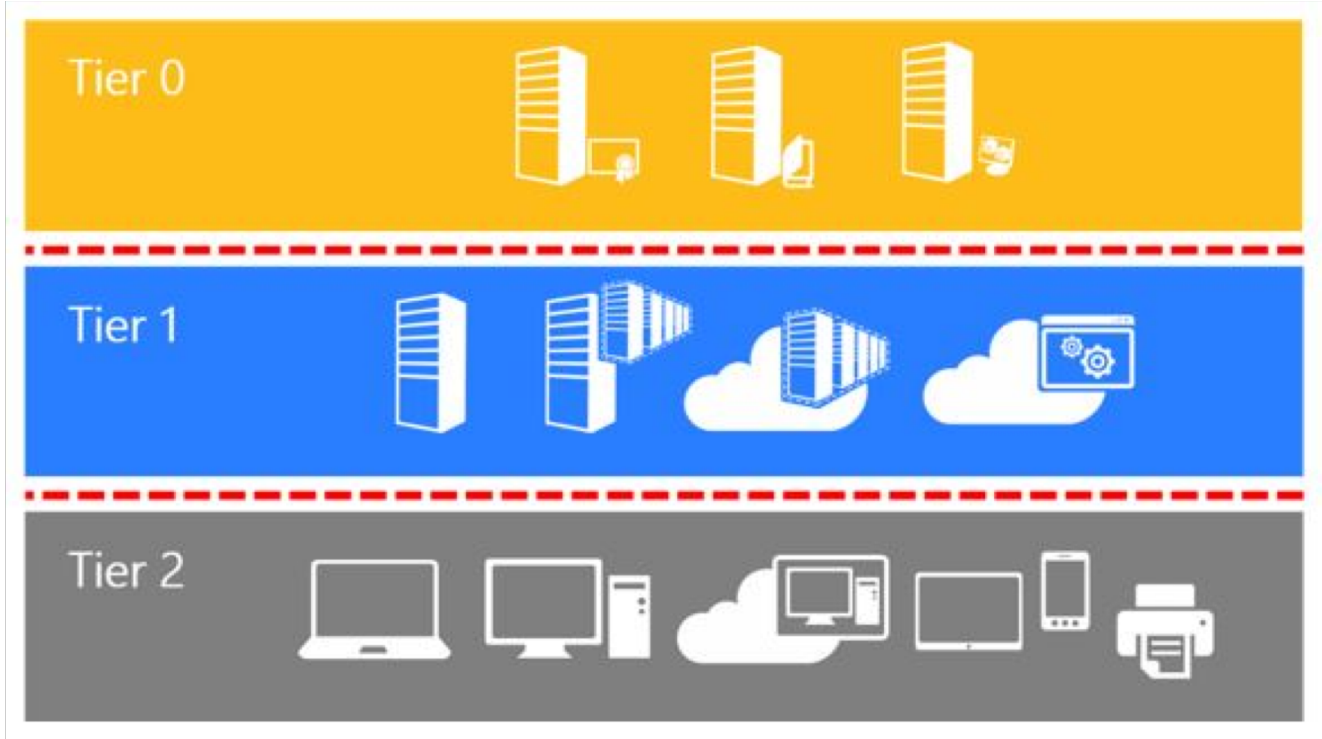
- Domain controllers
- Servers
- Client computers

Why?

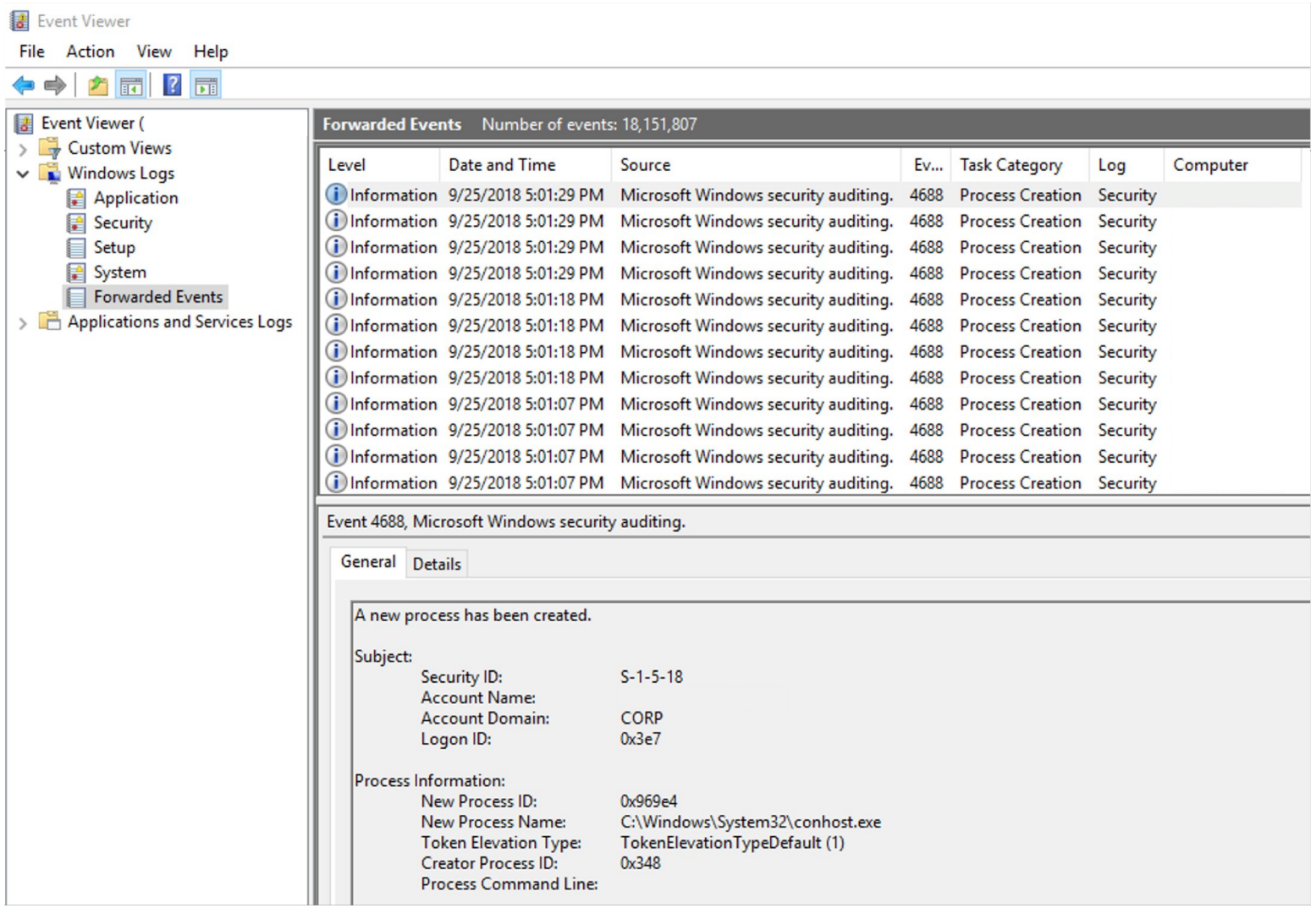
- Tier 0 assets – authentication and authorization nexus
- Workhorses and data storage
- Interface with the rest of the world

How?

- Event collectors
- Group Policy



Stage 1 - Event Log



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

Encoded commands: event_id:4103 AND ”-encodedCommand”

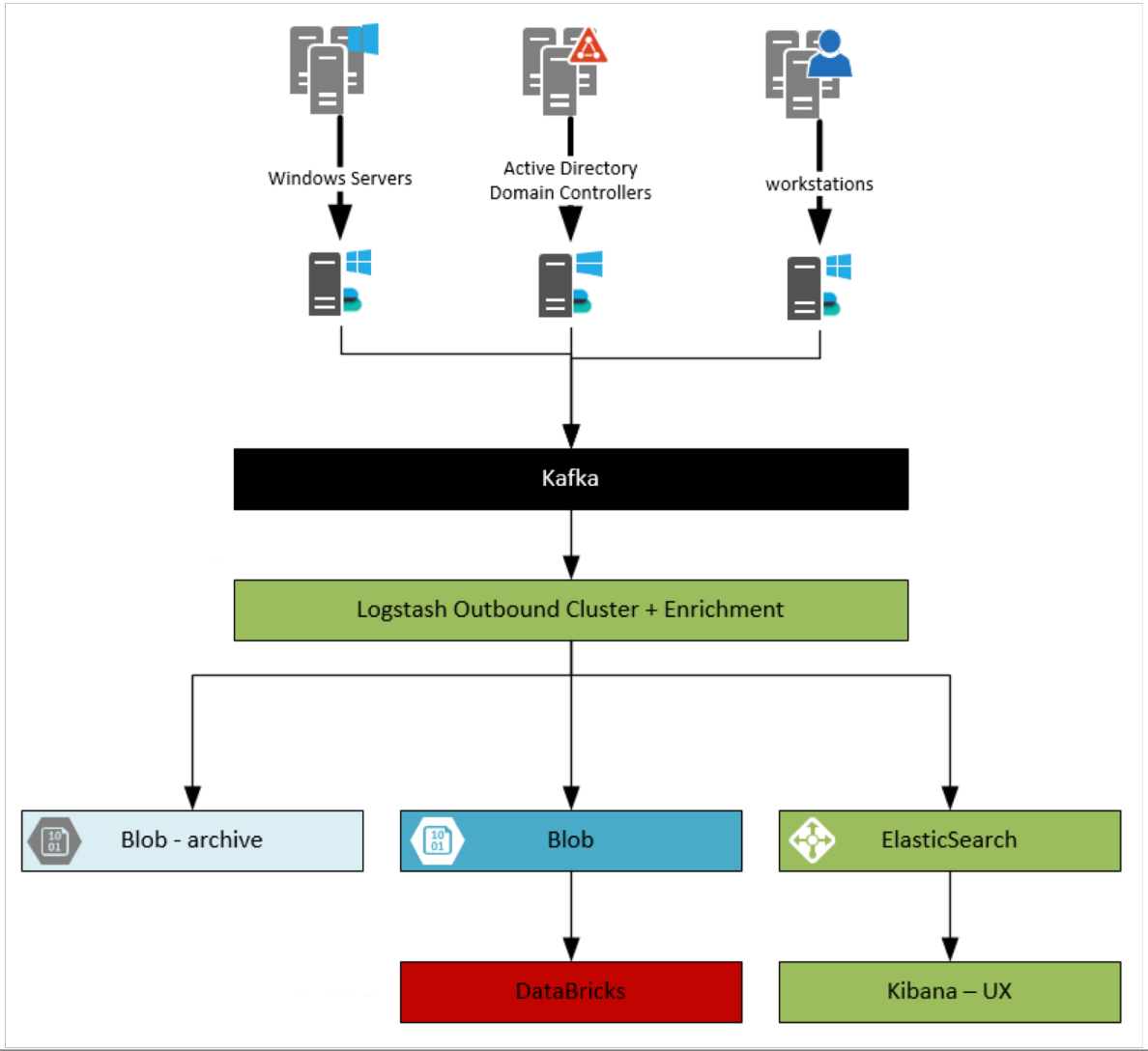
```
        Severity = Informational
        Host Name = ConsoleHost
        Host Version = 5.1.16299.492
        Host ID = 1c9c81d2-0539-41d4-93d0-982cfd2ac789
        Host Application = c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe -noprofile -nologo -noninteractive -EncodedCommand aQBmACgAIAAoAGcAZQB0AC0AZQB4AGUAYwB1AHQAaQBvAG4AcABvAGwAaQBjAHkAIAAtAHMAYwBvAHAAZQAgAGwAbwBjA
GEAbABtAGEAYwBoAGkAbgBlACkAIAAtAGUAcQAgACIAdQBuAGQAZQBmAGkAbgBlAGQAIgAgACkAIABZACAAIgAgACIAIABlAG4AdgA6AHcAaQBuAGQAaQBvAFwAcwB5AHMAdABlAG0AMwAyAFwAdwBpAG4AZABvAHcAcwBwAG8AdwBlAHIAcwBoAGUAbABsAFwAdgAxAC4AMABcAHAAbwB3AGUAcgBzAGgAZQ
BsAGwALgBlAHgAZQAiACAALQBuAG8AcAByAG8AZgBpAGwAZQAgAC0AbgBvAGwAbwBnAG8AIAAtAG4AbwBuAGkAbgB0AGUAcgBhAGMAdABpAHYAZQAgAC0AZQB4AGUAYwB1AHQAaQBvAG4AcABvAGwAaQBjAHkAIABhAGwAbABzAGkAZwBuAGUAZAAgAC0AYwBvAG0AbQBhAG4AZAAgAHMAZQB0AC0AZQB4AGU
AYwB1AHQAaQBvAG4AcABvAGwAaQBjAHkAIABhAGwAbABzAGkAZwBuAGUAZAAgADsAIAAgACYAIAAiACQAZQBuAHYAOgB3AGkAbgBkAGkAcgBcAHMAeQBzAHcAbwB3ADYANABcAHcAaQBuAGQAbwB3AHMAcABvAHcAZQByAHMAaABlAGwAbABcAHYAMQAuADAAXABwAG8AdwBlAHIAcwBoAGUAbABsAC4AZQB4
AGUAIgAgAC0AbgBvAHAAcgBvAGYAaQBsAGUAIAAtAG4AbwBsAG8AZwBvACAALQBuAG8AbgBpAG4AdABlAHIAYQBjAHQAaQB2AGUAIAAtAGUAeABlAGMAdQB0AGkAbwBuAHAAbwBsAGkAYwB5ACAAYQBsAGwAcwBpAGcAbgBlAGQAIAAtAGUAeABlAGMAdQB0AGkAbwBuAHAAbwBsAGkAYwB5ACAAYQBsAGwAcwBpAGcAbgBlAGQAIABzAGUAdAAtAGUAeABlAGMAdQB0AGkAbwBuAHAAbwBsAGkAYwB5ACAAYQBsAGwAcwBpAGcAbgBlAGQAIAAgAH0AIAA=
        Engine Version = 5.1.16299.492
        Runspace ID = fda63e13-473e-412a-8c3c-2aae0fbd11c7
        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)

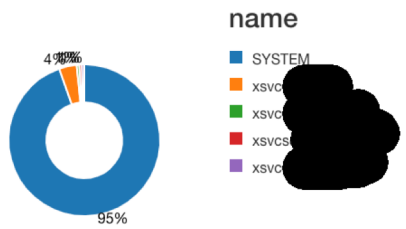
```
1 wdf = spark.read.json("wasbs://[redacted]core.windows.net/[redacted]/2018-08-25").where((col('event_id') == 4103))
2 wdf.groupBy('user.name').count().orderBy(col('count').desc()).show()
```

▾ (1) Spark Jobs
▾ Job 39 View (Stages: 2/2)
Stage 51: 4339/4339
Stage 52: 200/200

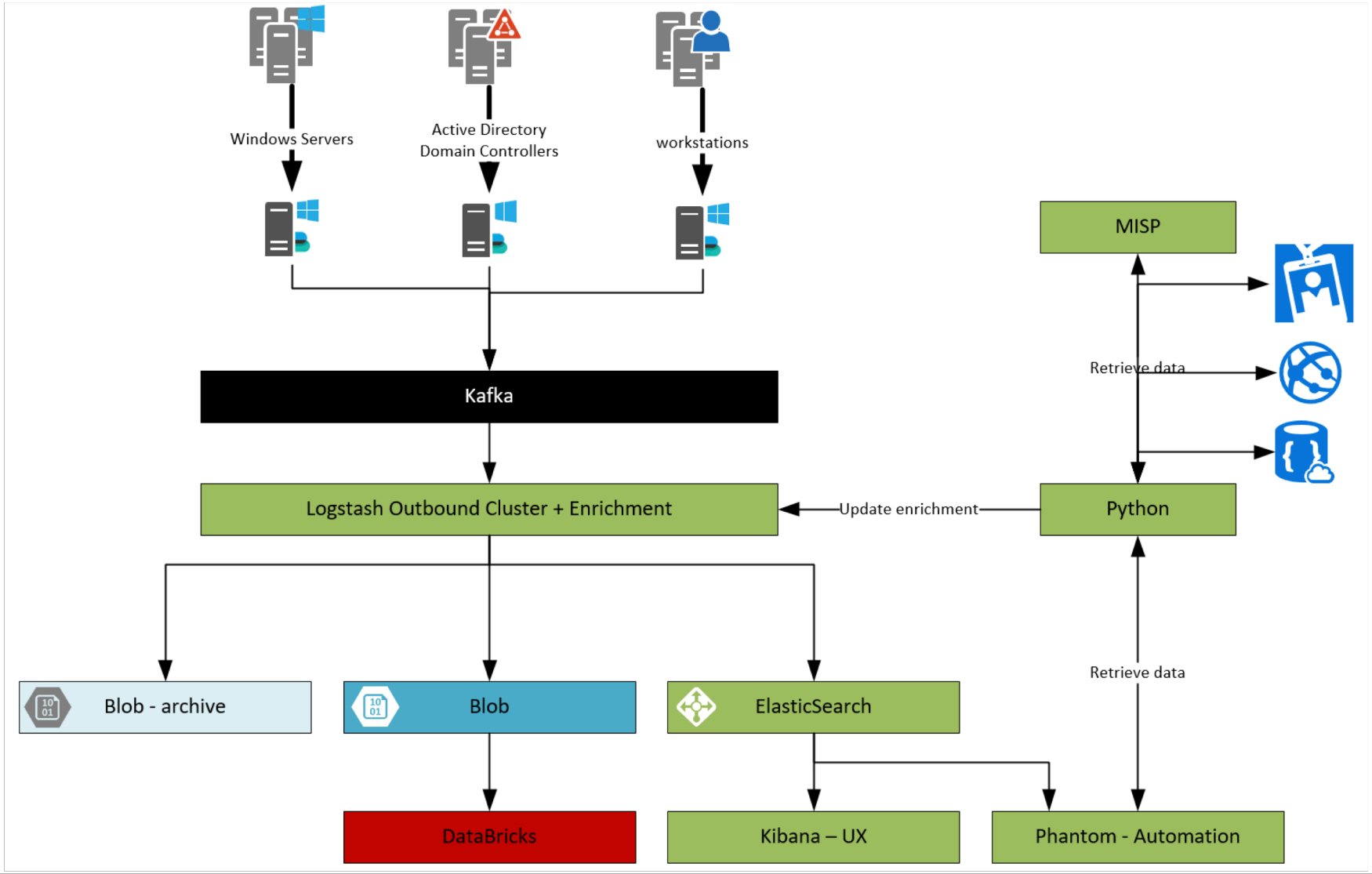
```
+----------------+--------+
|            name|   count|
+----------------+--------+
|          SYSTEM|10235011|
|      xsvc[redacted]|  405741|
|   xsvc[redacted]|   60467|
|xsvc[redacted]|   59696|
|xsvc[redacted]|   54975|
|  xsvc[redacted]|   45365|
|  xsvc[redacted]|   44984|
|  xsvc[redacted]|   44936|
|  xsvcs[redacted]|   34778|
|  xsvcs[redacted]|   20956|
|  xsvcs[redacted]|   18843|
|      xsvc[redacted]|    7066|
|  xsvc[redacted]|    6544|
|  xsvc[redacted]|    5825|
|[redacted]|    5445|
|     xco[redacted]|    5185|
|[redacted]|    4612|
|    xsvc[redacted]|    4304|
|      XSVC[redacted]|    3467|
|[redacted]|    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
Stage 70: 200/200
▸ 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)



Analysis Lifecycle



Lessons We Learned

Windows logging is really cool but it's not easy:

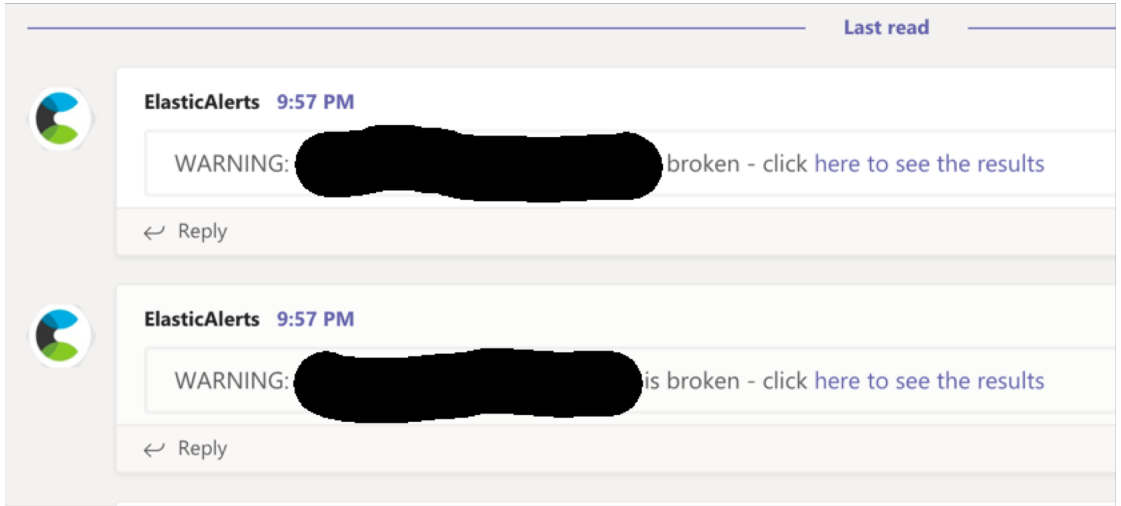
- Bi-directional communication requires synchronization between clients and WEF server
  - A few thousand clients will take a while to update, or sometimes never will
- Services just decide to stop or zombie out
- WEF collects historical data as far as the clients have it (years), hundreds of indices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

Mason Willinger
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

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

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

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