Hidden Treasure

Detecting Intrusions with ETW

Zac Brown
Senior Software Engineer, Microsoft

Story Time

(a short one, I promise)

Event 4688, Microsoft Windows security auditing.

General | Details

A new process has been created.

Creator Subject:

Security ID:	REDMOND\zbrown
Account Name:	zbrown
Account Domain:	REDMOND
Logon ID:	0x7581F57

Target Subject:

Security ID:	NULL SID
Account Name:	-
Account Domain:	-
Logon ID:	0x0

Process Information:

New Process ID:	0x4660
New Process Name:	C:\Windows\System32\bad.exe
Token Elevation Type:	%%1937
Mandatory Label:	Mandatory Label\High Mandatory Level
Creator Process ID:	0x4044
Creator Process Name:	C:\Windows\System32\w3wp.exe
Process Command Line:	"C:\Windows\System32\bad.exe"

Log Name:	Security		
Source:	Microsoft Windows security	Logged:	1/18/2017 2:00:18 PM
Event ID:	4688	Task Category:	Process Creation
Level:	Information	Keywords:	Audit Success

Forensic wishlist

What DNS lookups did it perform?

What IP addresses did it connect to?

How much data was transmitted?

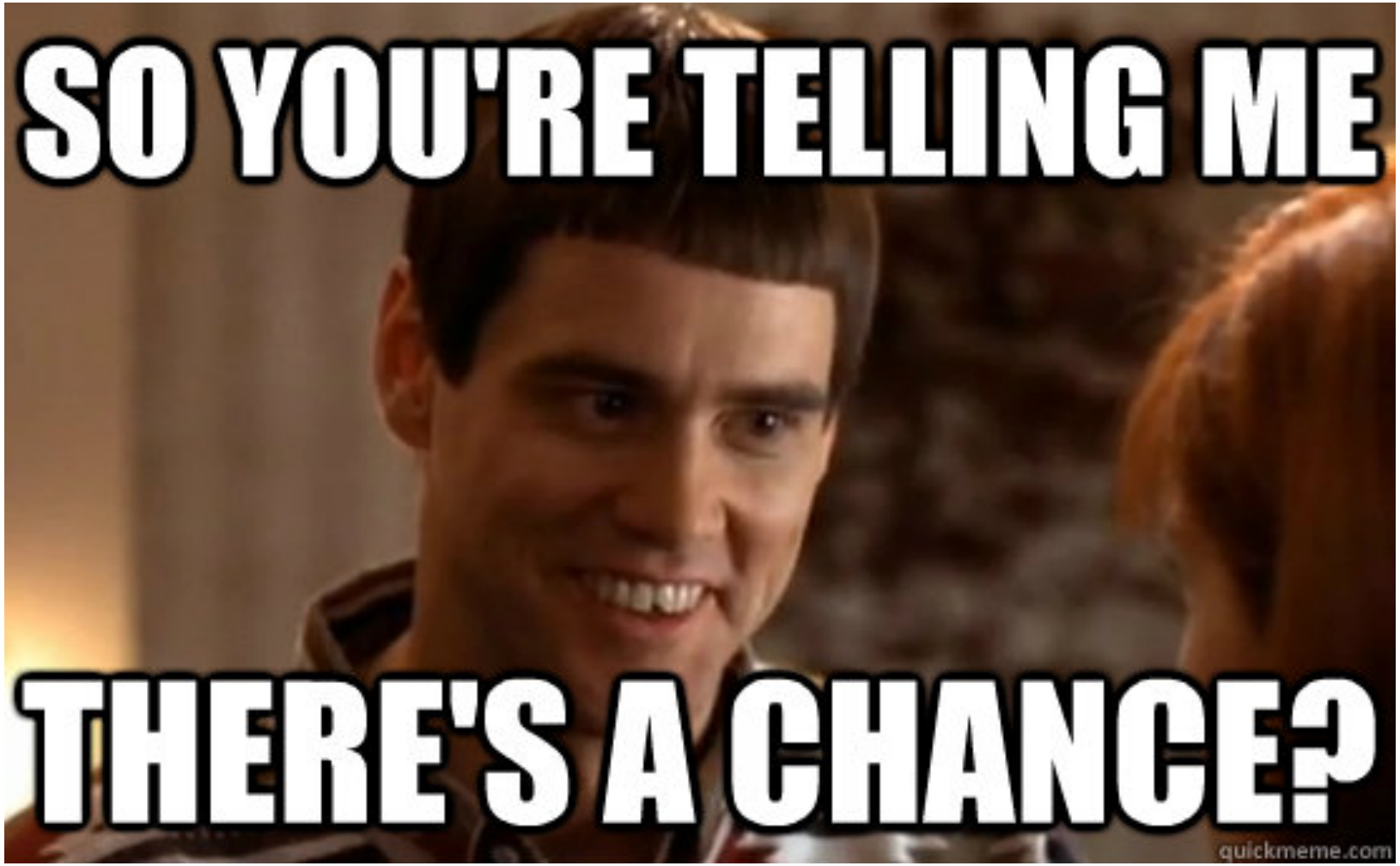
Is it “beaconing”?

What DLLs did it load?

Did it create threads in other processes?

What WMI operations did it perform?

What PowerShell functions were called?


SO YOU'RE TELLING ME
THERE'S A CHANCE?
quickmeme.com

ETW to the rescue

ETW is a built-in Windows tracing technology available since Windows 2000.

Originally for debug scenarios in Windows:

- Performance
- Power Management
- CLR behaviors (garbage collection, allocations)
- Tracing of various Windows subsystems

ETW visibility

Kernel mode providers

- Network, process, thread, memory, image load

User mode providers

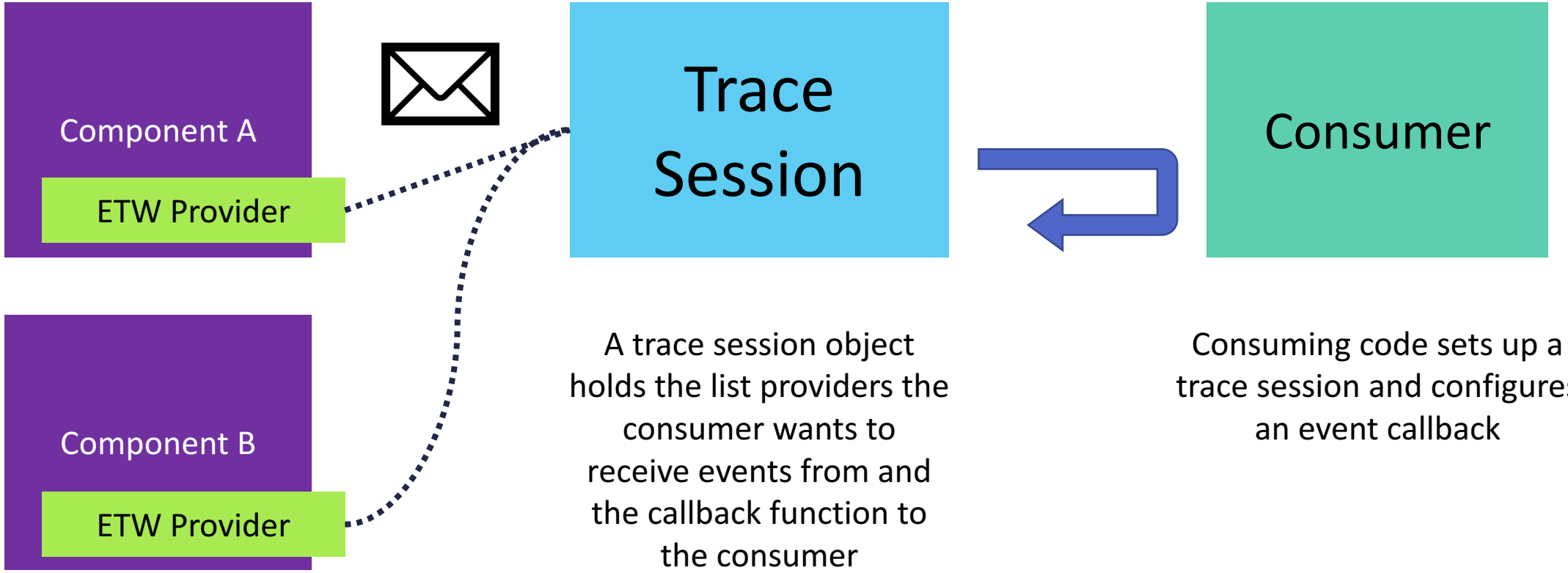
- PowerShell, WinINet, WMI, WinRM, DNS, RDP, Firewall, Defender, USB, .NET
- Over 1100 providers available on Windows 10

```
logman query providers > providers.txt
```

What’s ETW?

Consult your doctor before taking ETW.

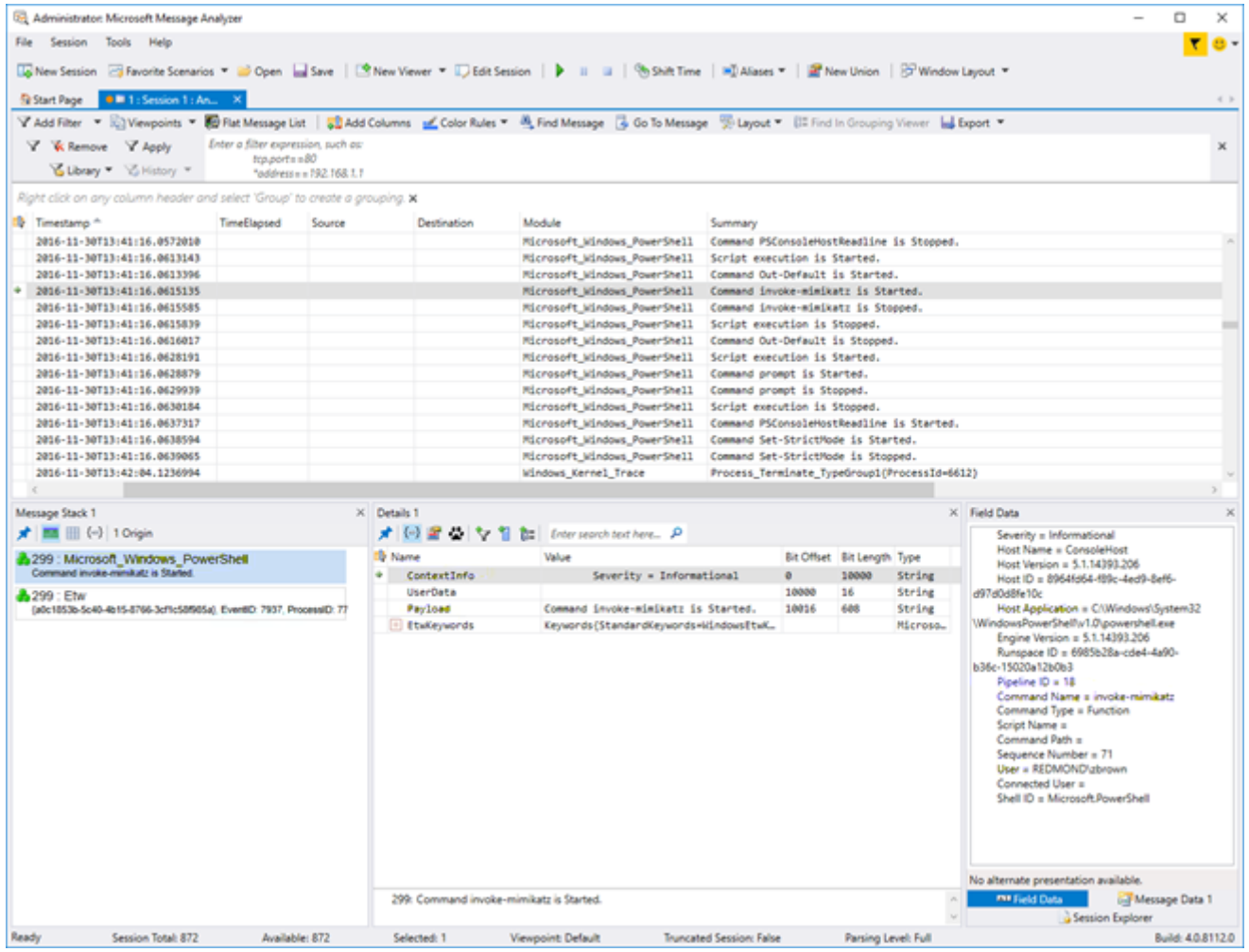
ETW overview



A trace session object holds the list providers the consumer wants to receive events from and the callback function to the consumer

Consuming code sets up a trace session and configures an event callback

What does an event look like?



What does an event look like?

Module	Summary
Microsoft_Windows_PowerShell	Command PSConsoleHostReadline is Stopped.
Microsoft_Windows_PowerShell	Script execution is Started.
Microsoft_Windows_PowerShell	Command Out-Default is Started.
Microsoft_Windows_PowerShell	Command invoke-mimikatz is Started.
Microsoft_Windows_PowerShell	Command invoke-mimikatz is Stopped.
Microsoft_Windows_PowerShell	Script execution is Stopped.
Microsoft_Windows_PowerShell	Command Out-Default is Stopped.
Microsoft_Windows_PowerShell	Script execution is Started.
Microsoft_Windows_PowerShell	Command prompt is Started.
Microsoft_Windows_PowerShell	Command prompt is Stopped.
Microsoft_Windows_PowerShell	Script execution is Stopped.
Microsoft_Windows_PowerShell	Command PSConsoleHostReadline is Started.
Microsoft_Windows_PowerShell	Command Set-StrictMode is Started.
Microsoft_Windows_PowerShell	Command Set-StrictMode is Stopped.

What does an event look like?

Details 1

Enter search text here...

Name	Value	Bit Offs
ContextInfo	Severity = Informational	0
UserData		10000
Payload	Command invoke-mimikatz is Started.	10016
EtwKeywords	Keywords{StandardKeywords=WindowsEtwK…	

Field Data

Severity = Informational
Host Name = ConsoleHost
Host Version = 5.1.14393.206
Host ID = 8964fd64-f89c-4ed9-8ef6-d97d0d8fe10c
Host Application = C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Engine Version = 5.1.14393.206
Runspace ID = 6985b28a-cde4-4a90-b36c-15020a12b0b3
Pipeline ID = 18
Command Name = invoke-mimikatz
Command Type = Function
Script Name =
Command Path =
Sequence Number = 71
User = REDMOND\zbrown
Connected User =
Shell ID = Microsoft.PowerShell

Event header also contains:

- Timestamp
- Process and Thread Id

How do you capture ETW events?

Log files

- Write events to ETL file format
- Doesn’t solve problem of volume

Real-time processing

- Custom code that handles events as they arrive

We are interested in real-time capture for intrusion detection.

Real-time ETW solutions

Win32 Event Tracing API

- cumbersome and inflexible
- program like it’s 1992

System.Diagnostics.Eventing

- unreliable
- poor performance (CPU & memory)

TraceEvent

- poor performance (memory)

Real-time ETW solutions

krabsetw - real-time ETW made easy

- managed and native (modern C++)
- flexible and intuitive
- performance and reliability
- built for Office 365 scale
- compatible with Win7+ and Win2008R2+

Example time

(Don’t worry, no live coding)

krabsetw DNS lookup example

```
public static void Run()
{
    var filter = new EventFilter(
        Filter.EventIdIs(3018) // cached lookup
        .Or(Filter.EventIdIs(3020))); // live lookup

    filter.OnEvent += (IEventRecord r) => {
        var query = r.GetUnicodeString("QueryName");
        var result = r.GetUnicodeString("QueryResults");
        Console.WriteLine($"DNS query ({r.Id}): {query} - {result}");
    };

    var provider = new Provider("Microsoft-Windows-DNS-Client");
    provider.AddFilter(filter);

    var trace = new UserTrace();
    trace.Enable(provider);
    Helpers.SetupCtrlC(trace); // Setup Ctrl-C to call trace.Stop();
    trace.Start();
}
```

1

2

3

4

```
DNS query (3020): www.nytimes.com - type:  5 nytimes.map.fastly.net;type:  6 ;
DNS query (3018): static01.nyt.com -
DNS query (3020): typeface.nyt.com - type:  5 nytimes.map.fastly.net;type:  6 ;
DNS query (3020): typeface.nyt.com - type:  5 nytimes.map.fastly.net;151.101.69.164;
DNS query (3020): www.nytimes.com - type:  5 nytimes.map.fastly.net;151.101.69.164;
DNS query (3018): www.googletagservices.com -
DNS query (3020): a1.nyt.com - type:  5 nytimes.map.fastly.net;151.101.69.164;
DNS query (3020): cdn.optimizely.com - 23.79.155.110;
DNS query (3018): www.googletagservices.com -
DNS query (3020): a1.nyt.com - type:  5 nytimes.map.fastly.net;type:  6 ;
DNS query (3018): cdnjs.cloudflare.com -
DNS query (3020): static01.nyt.com - type:  5 nytimes.map.fastly.net;151.101.69.164;
DNS query (3020): cdn.optimizely.com -
DNS query (3018): markets.on.nytimes.com -
DNS query (3018): cdnjs.cloudflare.com -
DNS query (3020): static01.nyt.com - type:  5 nytimes.map.fastly.net;type:  6 ;
DNS query (3018): www.google-analytics.com -
DNS query (3018): www.google-analytics.com -
DNS query (3020): int.nyt.com - type:  5 wildcard.nytimes.com.edgekey.net;type:  5 e5482.g.akamaiedge.net;23
DNS query (3018): markets.on.nytimes.com -
DNS query (3020): int.nyt.com - type:  5 wildcard.nytimes.com.edgekey.net;type:  5 e5482.g.akamaiedge.net;ty
DNS query (3018): s3.amazonaws.com -
DNS query (3018): s3.amazonaws.com -
DNS query (3020): cdnjs.cloudflare.com - 104.19.192.102;104.19.193.102;104.19.194.102;104.19.196.102;104.19.
DNS query (3020): www.googletagservices.com - type:  5 pagead46.l.doubleclick.net;216.58.193.98;
DNS query (3020): www.googletagservices.com - type:  5 pagead46.l.doubleclick.net;2607:f8b0:400a:800::2002;
DNS query (3020): cdnjs.cloudflare.com - 2400:cb00:2048:1::6813:c166;2400:cb00:2048:1::6813:c366;2400:cb00:2
DNS query (3020): www.google-analytics.com - type:  5 www-google-analytics.l.google.com;172.217.3.206;
DNS query (3020): www.google-analytics.com - type:  5 www-google-analytics.l.google.com;2607:f8b0:400a:800::
DNS query (3020): s3.amazonaws.com - type:  5 s3-1.amazonaws.com;52.216.17.131;
DNS query (3020): s3.amazonaws.com - type:  5 s3-1.amazonaws.com;type:  6 ;
```

krabsetw PowerShell DLL load example

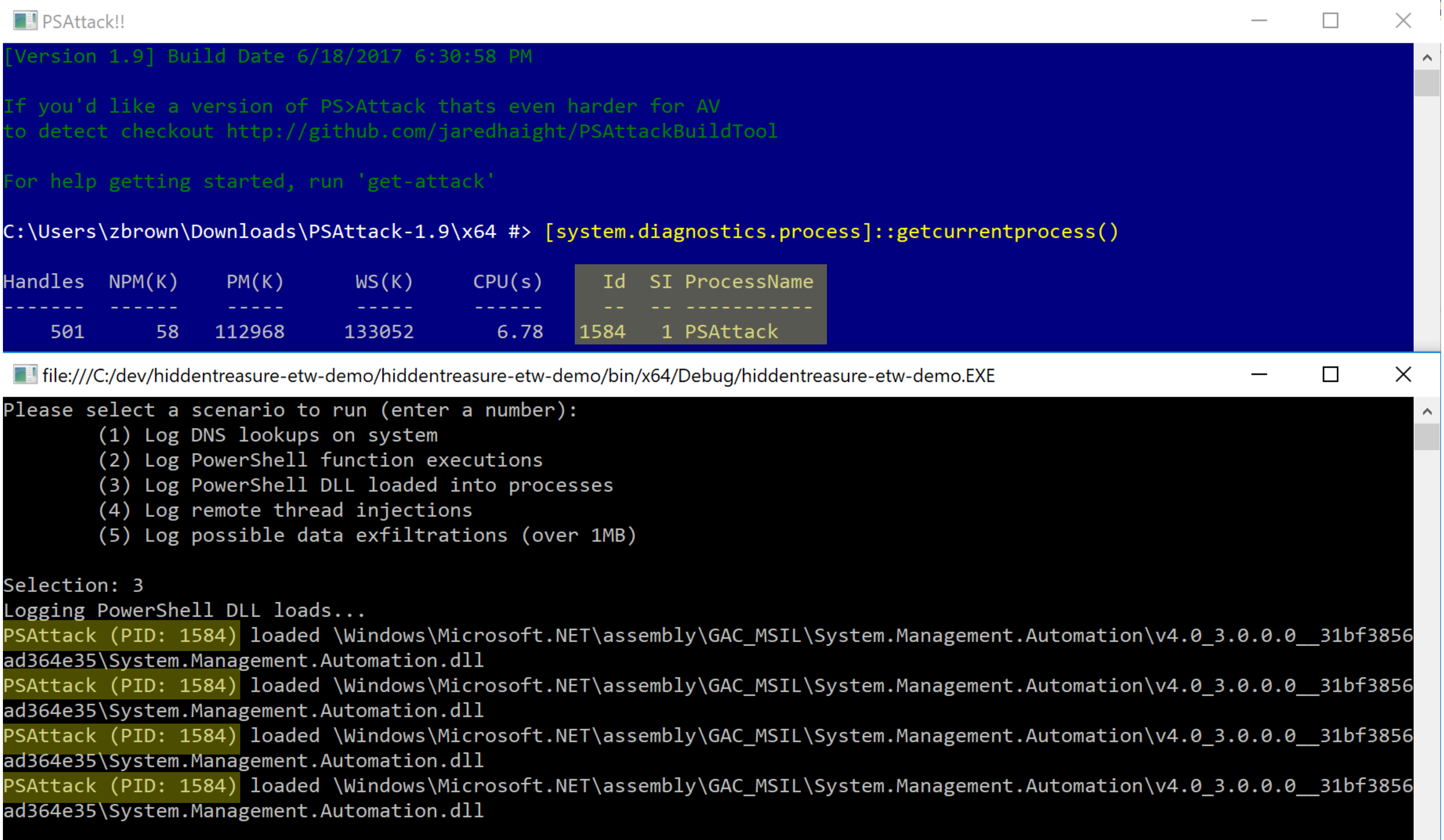
```
public static void Run()
{
    var filter = new EventFilter(Filter
        .EventIdIs(5)
        .And(UnicodeString.IContains("ImageName", @"\System.Management.Automation.dll")));

    filter.OnEvent += (IEventRecord r) => {
        var pid = (int)r.ProcessId;
        var processName = Process.GetProcessById(pid).ProcessName;
        var imageName = r.GetUnicodeString("ImageName");
        Console.WriteLine($"{processName} (PID: {pid}) loaded {imageName}");
    };

    var provider = new Provider("Microsoft-Windows-Kernel-Process");
    provider.AddFilter(filter);

    var trace = new UserTrace();
    trace.Enable(provider);
    Helpers.SetupCtrlC(trace); // Setup Ctrl-C to call trace.Stop();
    trace.Start();
}
```

1

2

3

4



```
[Version 1.9] Build Date 6/18/2017 6:30:58 PM

If you'd like a version of PS>Attack thats even harder for AV
to detect checkout http://github.com/jaredhaight/PSAttackBuildTool

For help getting started, run 'get-attack'

C:\Users\zbrown\Downloads\PSAttack-1.9\x64 #> [system.diagnostics.process]::getcurrentprocess()

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    501      58   112968     133052       6.78   1584   1 PSAttack
```

```
Please select a scenario to run (enter a number):
        (1) Log DNS lookups on system
        (2) Log PowerShell function executions
        (3) Log PowerShell DLL loaded into processes
        (4) Log remote thread injections
        (5) Log possible data exfiltrations (over 1MB)

Selection: 3
Logging PowerShell DLL loads...
PSAttack (PID: 1584) loaded \Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856
ad364e35\System.Management.Automation.dll
PSAttack (PID: 1584) loaded \Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856
ad364e35\System.Management.Automation.dll
PSAttack (PID: 1584) loaded \Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856
ad364e35\System.Management.Automation.dll
PSAttack (PID: 1584) loaded \Windows\Microsoft.NET\assembly\GAC_MSIL\System.Management.Automation\v4.0_3.0.0.0__31bf3856
ad364e35\System.Management.Automation.dll
```

krabsetw PowerShell command example

```
public static void Run()
{
    var filter = new EventFilter(Filter
        .EventIdIs(7937)
        .And(UnicodeString.Contains("Payload", "Started")));

    filter.OnEvent += (IEventRecord r) => {
        var method = r.GetUnicodeString("ContextInfo");
        Console.WriteLine($"Method executed:\n{method}");
    };

    var provider = new Provider("Microsoft-Windows-PowerShell");
    provider.AddFilter(filter);

    var trace = new UserTrace();
    trace.Enable(provider);
    Helpers.SetupCtrlC(trace); // Setup Ctrl-C to call trace.Stop();
    trace.Start();
}
```

1

2

3

4

PSAttack!!

```
[Version 1.9] Build Date 6/18/2017 6:30:58 PM

If you'd like a version of PS>Attack thats even harder for AV
to detect checkout http://github.com/jaredhaight/PSAttackBuildTool

For help getting started, run 'get-attack'

C:\Users\zbrown #> get-content foo.ps1 | write-host
function invoke-mimikatz {}
invoke-mimikatz
C:\Users\zbrown #> .\foo.ps1
```

C:\dev\hiddentreasure-etw-demo\hiddentreasure-etw-demo\bin\x64\Debug\hiddentreasure-etw-demo.exe

```
Method executed:
        Severity = Informational
        Host Name = PS ATTACK!!!
        Host Version = 3.0.0.0
        Host ID = 85c68b2d-55b4-48e7-8996-1a429003b1c5
        Host Application = C:\Users\zbrown\PSAttack.exe
        Engine Version = 5.1.15063.608
        Runspace ID = 32d88789-fdc9-45b7-9886-70c540518836
        Pipeline ID = 38
        Command Name = invoke-mimikatz
        Command Type = Function
        Script Name = C:\Users\zbrown\foo.ps1
        Command Path =
        Sequence Number = 443
        User = REDMOND\zbrown
        Connected User =
        Shell ID = Microsoft.PowerShell
```

krabsetw thread injection example

```
public static void Run()
{
    var filter = new EventFilter(Filter
        .EventIdIs(3));

    filter.OnEvent += (IEventRecord r) => {
        var sourcePID = r.ProcessId;
        var targetPID = r.GetUInt32("ProcessID");

        if (sourcePID != targetPID)
        {
            // This is where you'd check that the target process's
            // parent PID isn't the source PID.
            var createdTID = r.GetUInt32("ThreadID");
            var fmt = "Possible thread injection! - SourcePID: {0}, TargetPID: {1}, CreatedTID: {2}";
            Console.WriteLine(fmt, sourcePID, targetPID, createdTID);
        }
    };

    var provider = new Provider("Microsoft-Windows-Kernel-Process");
    provider.AddFilter(filter);

    var trace = new UserTrace();
    trace.Enable(provider);
    Helpers.SetupCtrlC(trace); // Setup Ctrl-C to call trace.Stop();
    trace.Start();
}
```

1

2

3

4

```
E:\code\hiddentreasure-etw-demo\hiddentreasure-etw-demo\bin\x64\Debug>.\hiddentreasure-etw-demo.exe
Possible thread injection! - SourcePID: 4, TargetPID: 7752, CreatedTID: 18816
Possible thread injection! - SourcePID: 4956, TargetPID: 5108, CreatedTID: 4276
Possible thread injection! - SourcePID: 4956, TargetPID: 18636, CreatedTID: 17904
Possible thread injection! - SourcePID: 4, TargetPID: 18636, CreatedTID: 10880
Possible thread injection! - SourcePID: 4, TargetPID: 5108, CreatedTID: 12624
Possible thread injection! - SourcePID: 4, TargetPID: 5108, CreatedTID: 12220
Possible thread injection! - SourcePID: 4, TargetPID: 4788, CreatedTID: 10992
Possible thread injection! - SourcePID: 4, TargetPID: 15752, CreatedTID: 18428
Possible thread injection! - SourcePID: 4, TargetPID: 840, CreatedTID: 4764
Possible thread injection! - SourcePID: 848, TargetPID: 13272, CreatedTID: 16788
Possible thread injection! - SourcePID: 13272, TargetPID: 16100, CreatedTID: 12748
Possible thread injection! - SourcePID: 4, TargetPID: 16708, CreatedTID: 3276
Possible thread injection! - SourcePID: 4, TargetPID: 2400, CreatedTID: 19288
Possible thread injection! - SourcePID: 4, TargetPID: 16100, CreatedTID: 13616
Possible thread injection! - SourcePID: 848, TargetPID: 18084, CreatedTID: 9028
Possible thread injection! - SourcePID: 4, TargetPID: 18084, CreatedTID: 3948
```

Forensic wishlist, revisited

What DNS lookups did it perform?

Microsoft-Windows-DNS-Client

What IP addresses did it connect to?

How much data was transmitted?

Is it “beaconing”?

Microsoft-Windows-Kernel-Network

What DLLs did it load?

Did it create threads in other processes?

Microsoft-Windows-Kernel-Process

What WMI operations did it perform?

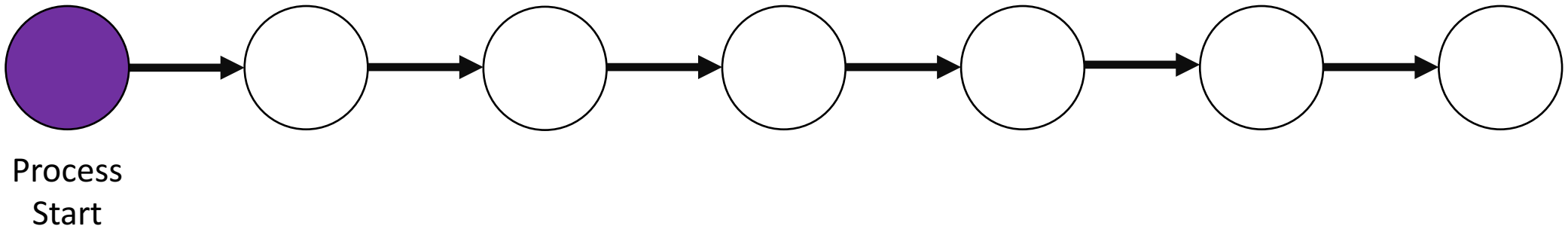
Microsoft-Windows-WMI

What PowerShell commands were called?

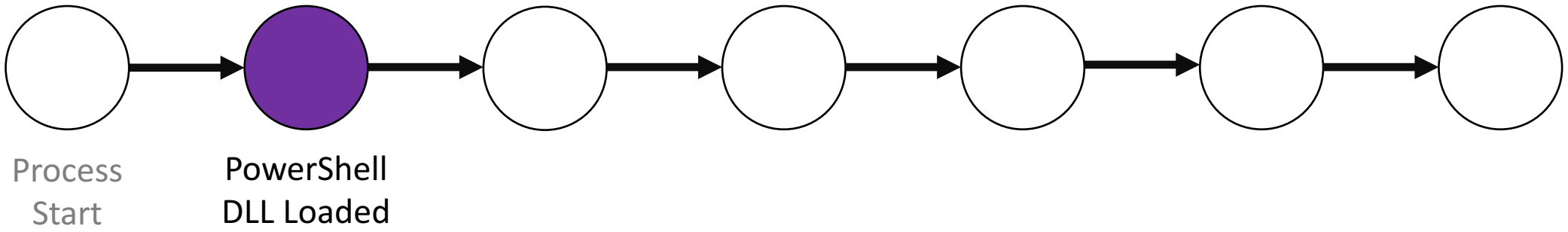
Microsoft-Windows-PowerShell

Revisiting bad.exe

Process Start



PowerShell DLL Loaded



Detection: PowerShell loaded in a process that's not powershell.exe/wsmprovhost.exe

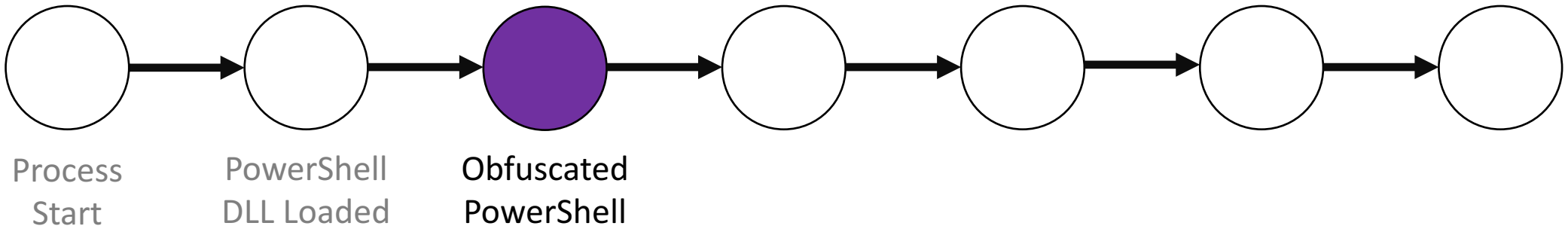
ETW Provider: Microsoft-Windows-Kernel-Process (user mode)

EventID: 5 (DLL loaded)

Example Data: System.Management.Automation.dll was loaded into PID 1234

Notes: heuristic detection, occasionally legitimate processes load PowerShell runtime

Obfuscated PowerShell



Detection: obfuscated PowerShell commands

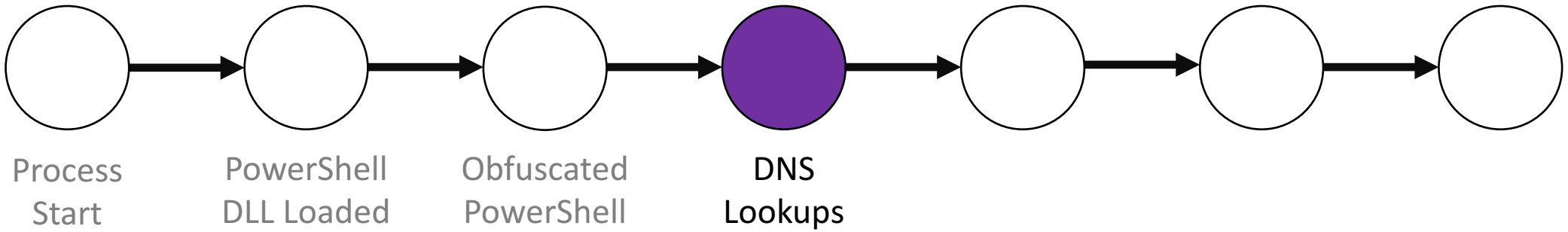
ETW Provider: Microsoft-Windows-PowerShell (user mode)

EventID: 7937 (command executed)

Example Data: PowerShell method named `ASDF1234`

Notes: requires statistical entropy analysis on method name

DNS Lookup



Detection: suspicious domain lookups

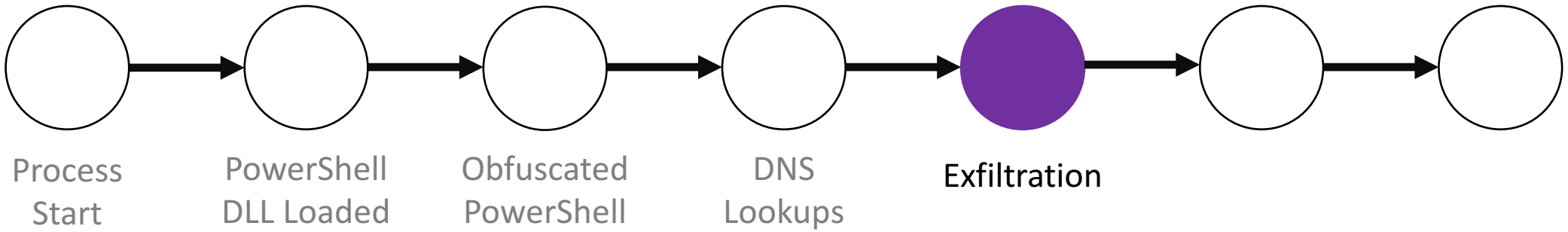
ETW Provider: Microsoft-Windows-DNS-Client (user mode)

EventID: 1016 (NXDOMAIN), 3018 (cache lookup), 3020 (live lookup)

Example Data: Machine `SuperServer` queried fanciestbear.com (ip: 1.2.3.4)

Notes: per-process lookups not available, use IP endpoints per process + reverse DNS lookup

Data Exfiltration



Detection: exfiltration of data to external endpoint

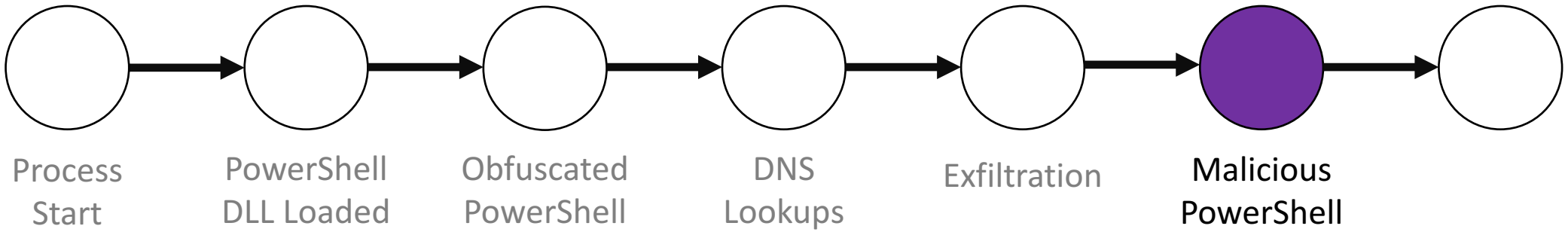
ETW Provider: Microsoft-Windows-Kernel-Network (user mode)

EventID: 10 (IPv4 send), 58 (IPv6 send)

Example Data: 248MB exfiltrated to known bad IP 1.2.3.4

Notes: requires aggregating bytes sent per process to endpoints

Malicious PowerShell



Detection: malicious PowerShell command executed

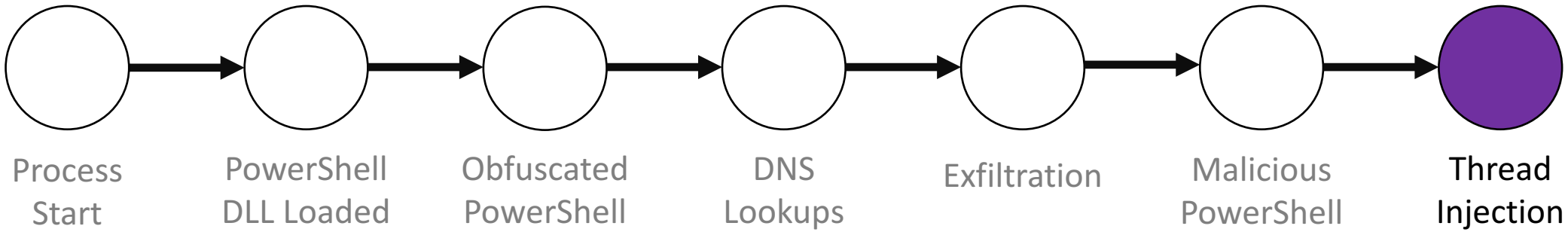
ETW Provider: Microsoft-Windows-PowerShell (user mode)

EventID: 7937 (command executed)

Example Data: Invoke-ReflectivePEInjection, Create-RemoteThread

Notes: ***any*** process that loads System.Management.Automation.dll (e.g. PSAttack.exe)

Remote Thread Injection



Detection: remote thread injection

ETW Provider: Microsoft-Windows-Kernel-Process (user mode)

EventID: 3 (thread created)

Example Data: PID 1234 created a thread with TID 910 in PID 5678 at 6:56PM

Notes: heuristic, process starts are initially remote thread injections

Forward everything to the SIEM?


WHARRGARBL

Event overload!

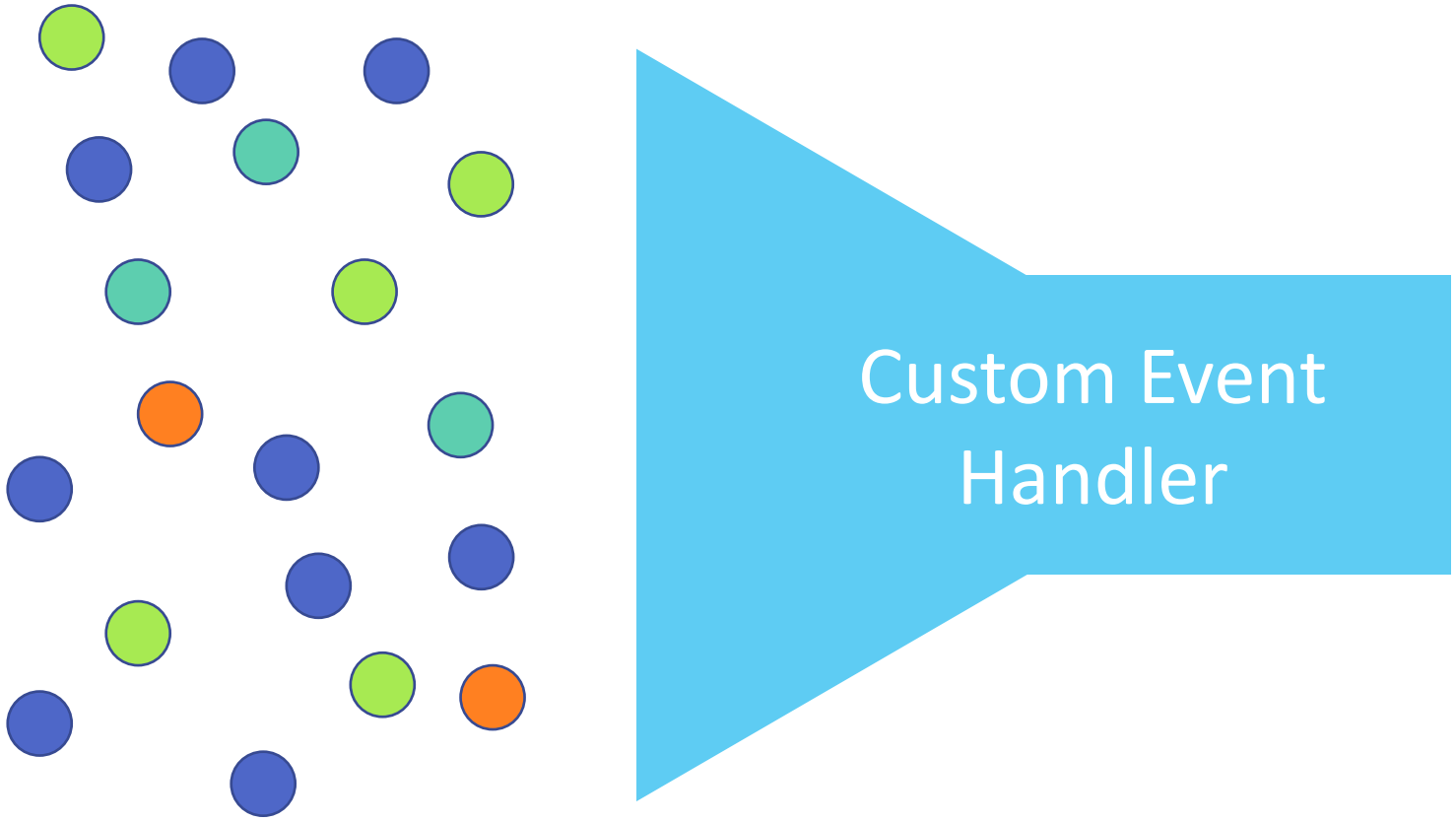
ETW is really for debugging

Could your SIEM handle every…

- Image load for every process started on every machine?
- Packet sent across your entire network?
- DNS resolution by any client?

Probably not

Reducing event volume



RAW → Reducer → To SIEM

Signals for reducing volume

Types of signals

Oracle

signal in deny list is alertable

signal not in allow list is alertable

Types of signals

Aggregation

accumulated signals over threshold is alertable

Types of signals

Heuristic

presence of signal or signals **MAY** be alertable

(this usually requires further analysis)

Types of signals

Statistic

statistical model generating a signal is alertable

Techniques applied

	Aggregation	Oracle	Statistics	Heuristic
Malicicious PowerShell	X	X		
Obfuscated PowerShell			X	
Data Exfiltration	X	X		
DNS Lookups		X		
Thread Injection				X
PowerShell DLL Loaded		X		X

Gotchas

Not everything can be sunshine and rainbows :(

Performance & Reliability

CPU usage is directly correlated to event volume

- pro tip: it’s probably network data – filter, filter, filter

Events can be dropped if you don’t pump fast enough

- pro tip: good filters are super important
- pro tip: send events to async thread(s) for processing

ETW occasionally writes events with bad/missing data

Tampering

Trace sessions can be stopped by administrators

Trace sessions can be modified by administrators

Malicious software can:

- write fake data to existing ETW providers
- write bad data to existing ETW providers

What did we find?

How does the Red team do?

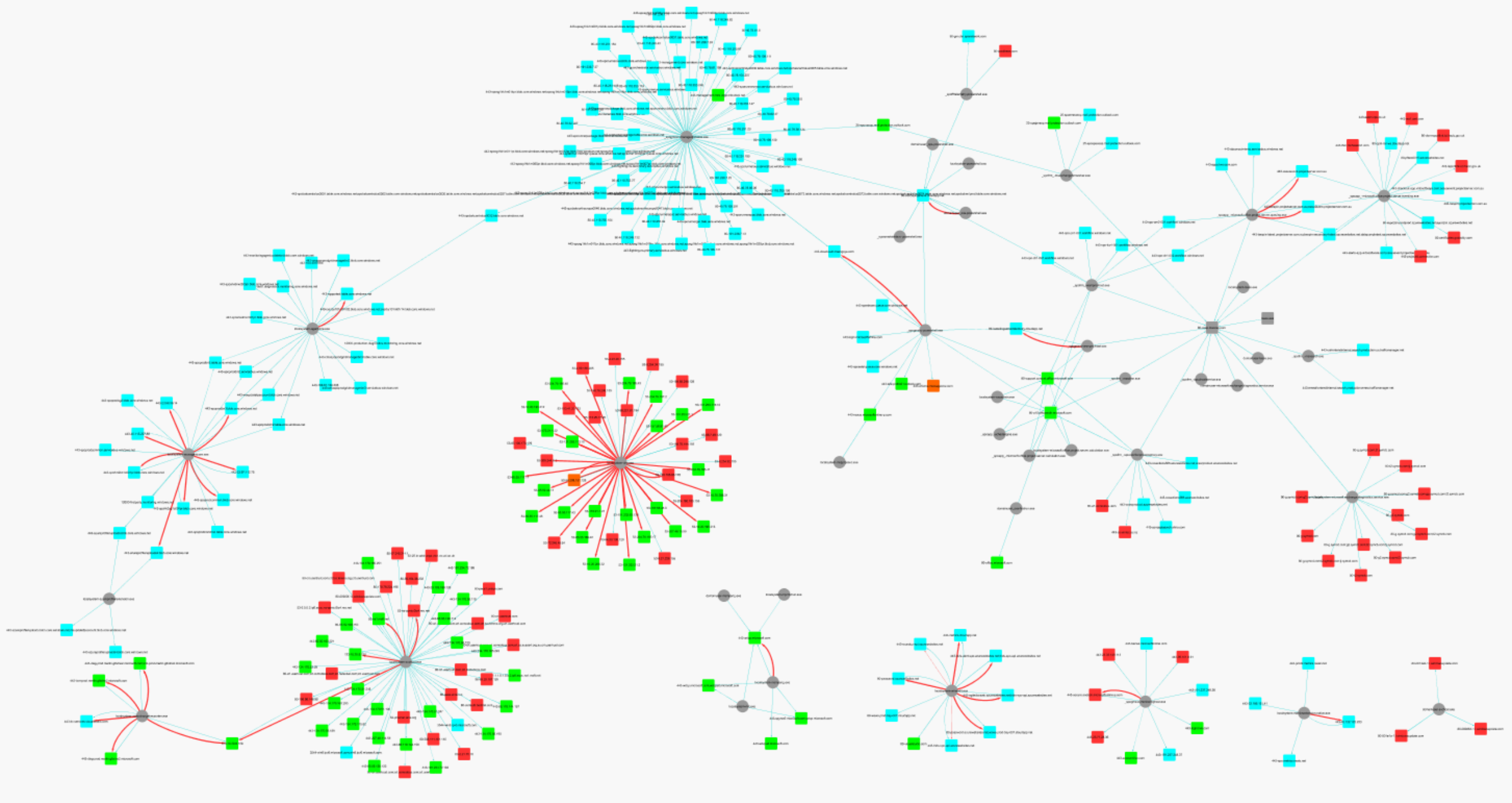
WMI activity

PowerShell command execution

PowerShell DLL loaded in anomalous process

New processes uploading to external endpoint

Beaconing processes (C2 communication)



But will it scale?

>100k machines across environment

>7TB of data per day across environment

>500B events per day across environment

After filtering, as of January 2017.

Wrap it up already

Is he still talking? Really?

How can you use ETW in your environment?

krabsetw

Open Source under the MIT license

NuGet packages for .NET and modern C++ APIs

compatible with Win7+ and Win 2008R2+

PowerShellMethodAuditor

Sample application using krabsetw to consume PowerShell ETW events

What’s next?

Signature-based tracing tool?

- Sigma or YARA for etw

Flat-C API for krabsetw?

- Enables FFI to languages like Go or Python

These don’t exist yet but I’d love your feedback.

Questions?

Get in touch with me:

@zacbrown

github.com/zacbrown

aka.ms/etw

aka.ms/etwdemo

aka.ms/PowerShellMethodAuditor

aka.ms/MessageAnalyzer