



Unleashing JonMon:

Deep Insights into Your Windows Activity

By: Jonny Johnson

Jonny Johnson (@jsecurity101)

■ Sr. Security Researcher of Adversarial Techniques and Capabilities @BinaryDefense

■ Interests

- Windows Internals
- Extracting and Exposing Telemetry
- Working on my Tacoma
- Brazilian Jiu Jitsu

■ Open-Source Author/Contributor

- The Defender's Guide
- TelemetrySource
- Atomic Test Harnesses

■ Formerly

- Sr. Threat Research @RedCanary
- Sr. Consultant @SpecterOps



The Problem

- Collection assumptions
- Interacting with telemetry mechanisms
- Understanding various event sources
- Attributes exposed through events

"If we are honest with ourselves, we have to admit that sometimes our assumptions and preconceived notions are wrong, and therefore, our interpretation of events is incorrect. This causes us to overreact, to take things personally, or to judge people unfairly." –

Elizabeth Thornton

The background is a deep blue gradient that transitions to a lighter teal and green at the bottom right. Overlaid on this are several white geometric shapes, primarily triangles and polygons, connected by thin white lines. Some of these shapes have small white dots at their vertices, creating a network-like or molecular structure. The overall aesthetic is modern and technological.

Let's take a trip to the past...

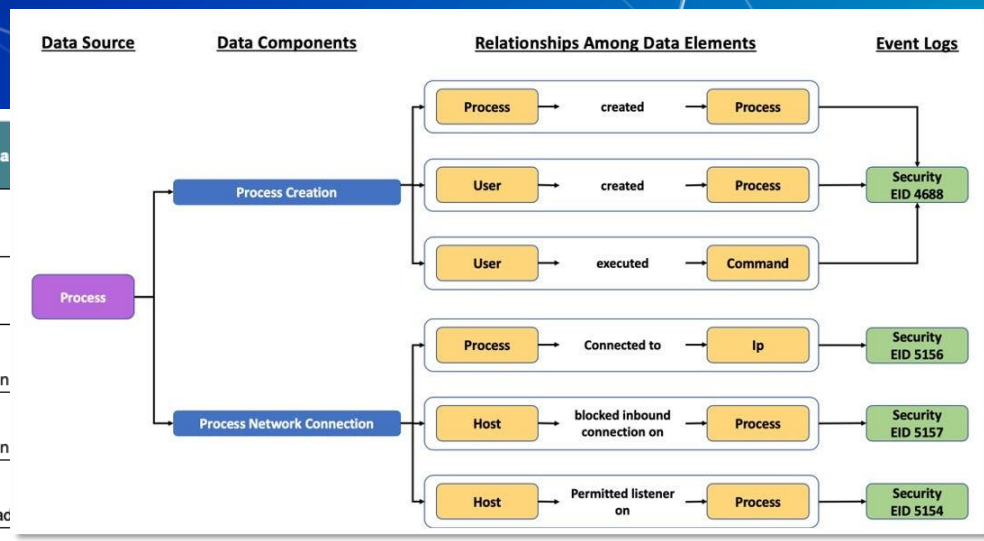
Event Mapping (data source to events)

■ In 2018 the Rodriguez brothers created mappings to help defenders understand object to action relationships

■ @Cyb3rWard0g

■ @Cyb3rPandaH

Data Source	Sub - Data Source
Process monitoring	process creation
Process monitoring	process creation
Process monitoring	process termination
Process monitoring	process termination
file monitoring	drivers load



Event Mapping (mechanism to event)

- TelemetrySource, previously API-to-EventId, was started in 2019 to build on previous research to understand how vendors were logging actions

EventId	Event Description	Operational Functions	Event Processing Functions	Id
1	THREATINT_ALLOCVM_REMOTE	NtAllocateVirtualMemoryEx → MmAllocateVirtualMemory → MiAllocateVirtualMemoryCommon → MiAllocateVirtualMemory, NtAllocateVirtualMemory → MiAllocateVirtualMemory, MiAllocateUserStack → MiAllocateVirtualMemory, MmStoreAllocateVirtualMemory → MiAllocateVirtualMemory, WbAllocateVirtualMemoryEx → MmAllocateVirtualMemory → MiAllocateVirtualMemoryCommon → MiAllocateVirtualMemory	EtwTiLogAllocExecVm	E
2	THREATINT_PROTECTVM_REMOTE	NtProtectVirtualMemory	EtwTiLogProtectExecVm	E
3	THREATINT_MAPVIEW_REMOTE	NtMapViewOfSectionEx → MiMapViewOfSectionExCommon, NtMapViewOfSection, PspMapView → MiMapViewOfSectionExCommon	EtwTiLogMapExecView	E
4	THREATINT_QUEUEUSERAPC_REMOTE	KeInsertQueueApc	EtwTiLogInsertQueueUserApc	E



Insert JonMon

Formerly known as Accio
(Thank you Olaf for the rename)

Purpose

- Allows researchers to test telemetry ideas
- Code base for defenders to understand telemetry technologies
- NOT meant for production environments – I am not a developer
- Push telemetry forward with ALL vendors
- Build better detection ideas based on telemetry
- Telemetry collection performance

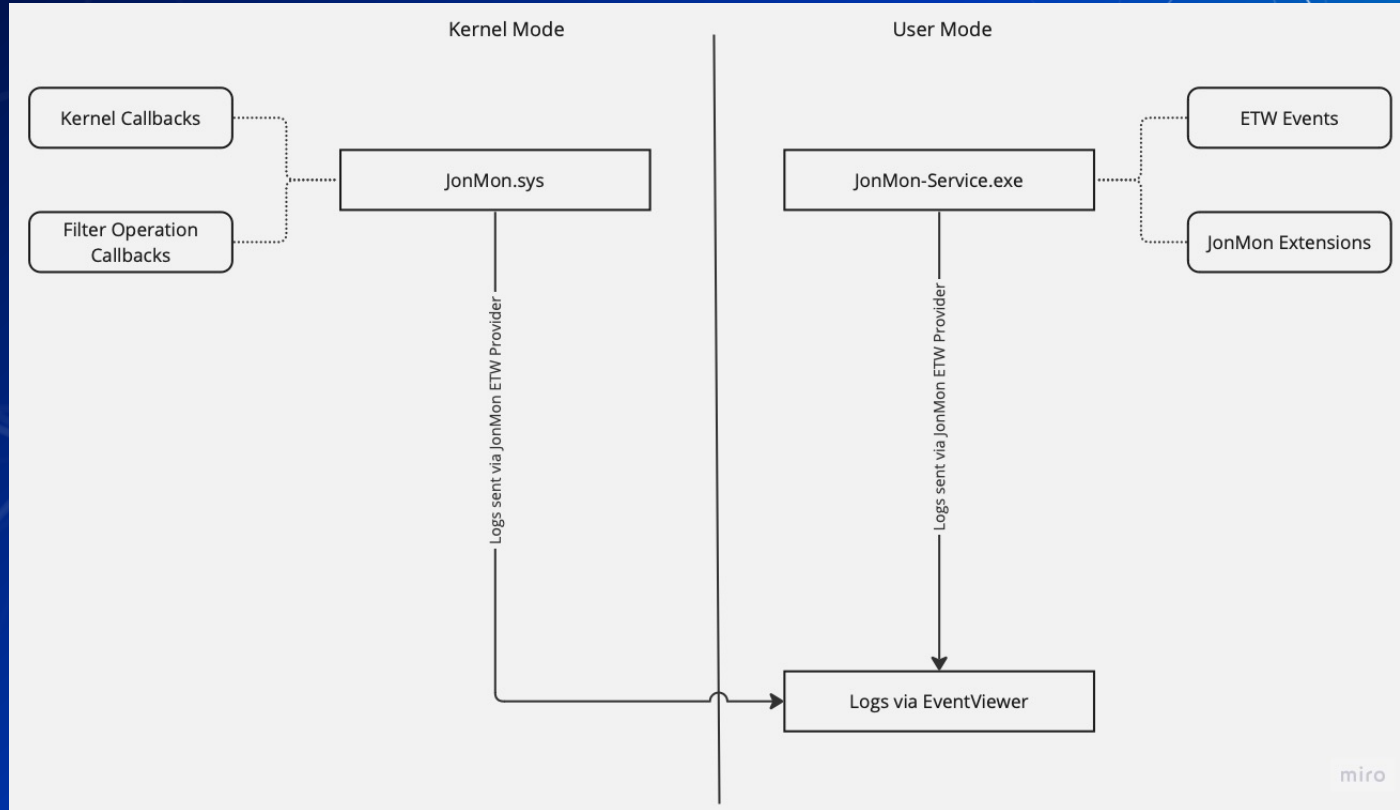
Disclaimers

- Still in beta
 - Pending performance feedback from the community
- Not meant to run in production environments
- Meant to be used as a research tool, not a full-time telemetry sensor

Internals

- Kernel Driver (JonMon.sys)
- Win32 Service (JonMon-Service.exe) - Runs as PPL
- Manifest ETW Provider (JonMon.man)
- User-Mode Exentions (JonMon-Ext1.dll)

Internals cont.d



Kernel Callbacks

EventID	EventType	Collection Mechanism
1, 8	Process Creation & Process Reparenting	PsSetCreateProcessNotifyRoutineEx
2	OpenProcess	ObRegisterCallback (Post-Callback)
3	DuplicateHandle	ObRegisterCallback (Post-Callback)
5, 6, 7, 14	Registry Events	RegistryCallback
4, 27	ImageLoad & DriverLoad	PsSetLoadImageNotifyRoutine
18	Create Remote Thread	PsSetCreateThreadNotifyRoutine

Event Tracing for Windows (ETW)

EventID	EventType	Collection Mechanism
30	Memory Read	Threat-Intelligence
29	Memory Write	Threat-Intelligence
13	Network (TCP)	Microsoft-Windows-Kernel-Network
15	DotNet Load	Microsoft-Windows-DotNETRuntime
11, 12	RPC Calls	Microsoft-Windows-RPC
28	DPAPI	Microsoft-Windows-Crypto-DPAPI
32	Remote Virtual Allocation	Threat-Intelligence

User-Mode Extensions

EventID	EventType	Collection Mechanism
31	Thread Token Impersonation	Process & Thread Query

Installation

- Requires TESTSIGNING to be ON
- Suggest DEBUG to be ON
 - Easier for debugging
 - Easier to help with issues

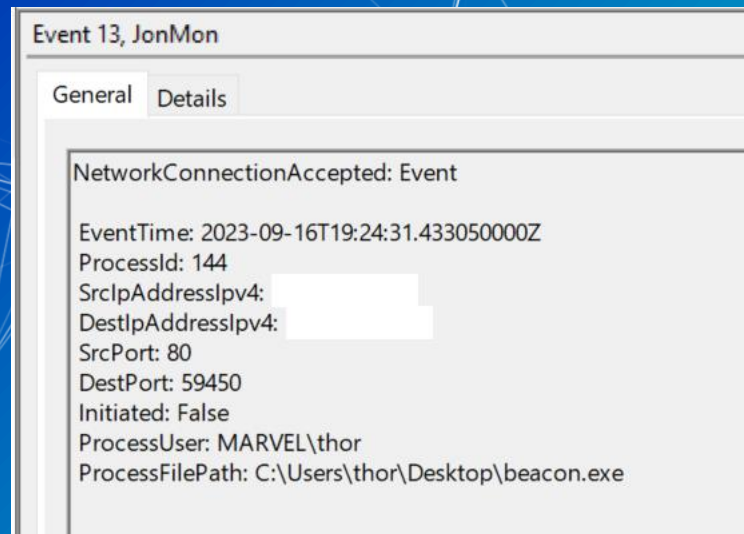
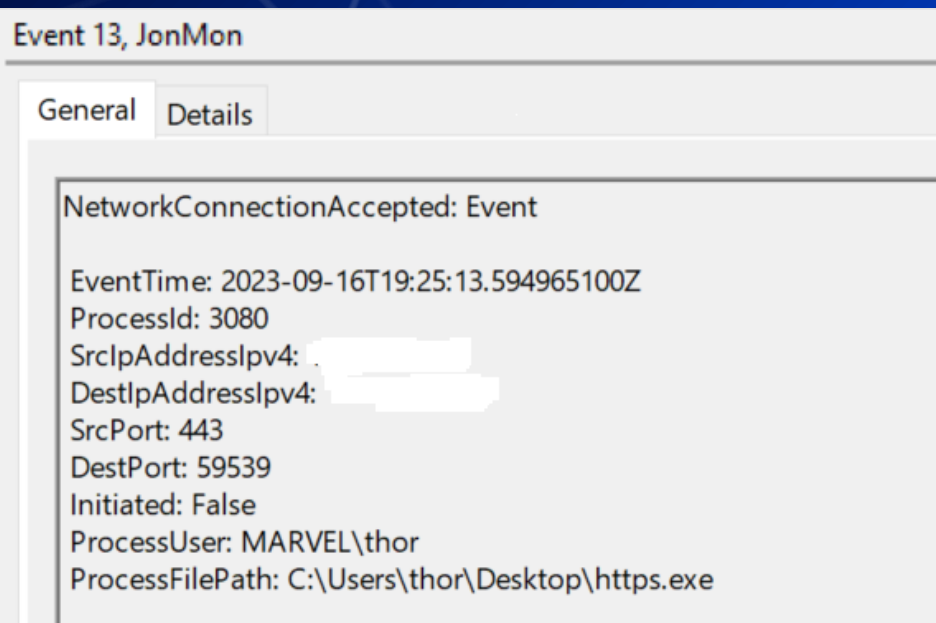
```
PS C:\Users\panther\Desktop\JonMon> .\JonMon-Service.exe -i
[*] Starting JonMon Installation Process....
[*] JonMon.dll copied
[*] Installing Manifest....
[*] Uninstalling Manifest....
[*] Manifest Uninstalled....
[*] Manifest Installed....
[*] JonMon.sys copied
[*] JonMon.inf copied
[*] JonMon-Ext1.dll copied
[*] Installing JonMonDrv Service....
[*] JonMonDrv Service Installed
[*] JonMon-Service.exe copied
[*] Creating Service JonMon....
[*] Service JonMon created successfully
[*] Starting Service JonMon....
[*] Service JonMon started successfully
```

Demos

The background of the slide is a gradient of blue, transitioning from a deep navy blue on the left to a lighter, teal-like blue on the right. Overlaid on this background is a complex network of thin white lines and small white dots. These lines and dots form various geometric shapes, including triangles, polygons, and larger, more intricate structures that resemble molecular models or network diagrams. The overall effect is a modern, tech-oriented aesthetic.

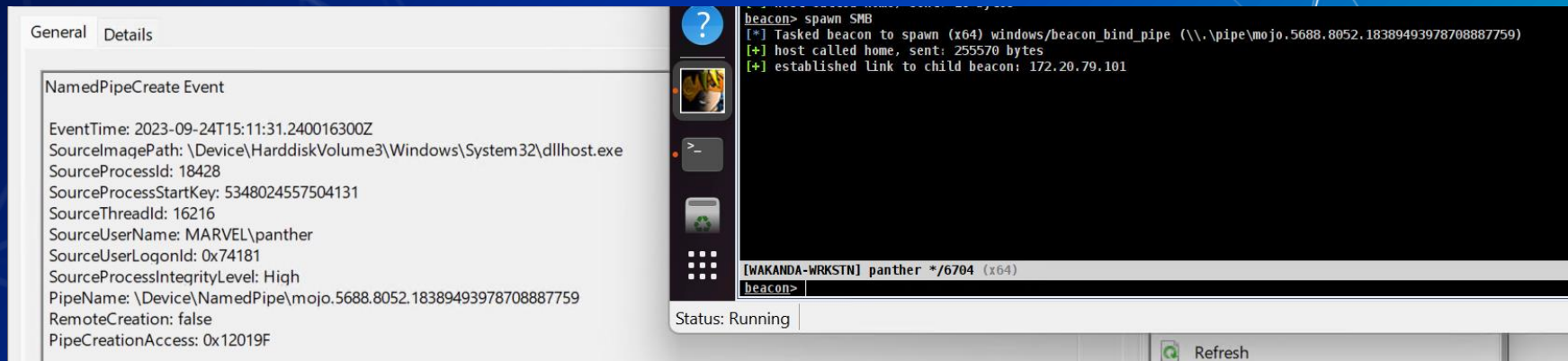
Cobalt Strike Beacon (TCP/IP)

■ Event ID 13 – TCP Connections



Cobalt Strike Beacon (Named Pipe)

■ Event ID 23 – Named Pipe Creation



The image displays a Windows Event Viewer window on the left and a Cobalt Strike console window on the right. The Event Viewer shows a 'NamedPipeCreate Event' with the following details:

- EventTime: 2023-09-24T15:11:31.240016300Z
- SourceImagePath: \Device\HarddiskVolume3\Windows\System32\dlhhost.exe
- SourceProcessId: 18428
- SourceProcessStartKey: 5348024557504131
- SourceThreadId: 16216
- SourceUserName: MARVEL\panther
- SourceUserLogonId: 0x74181
- SourceProcessIntegrityLevel: High
- PipeName: \Device\NamedPipe\mojo.5688.8052.18389493978708887759
- RemoteCreation: false
- PipeCreationAccess: 0x12019F

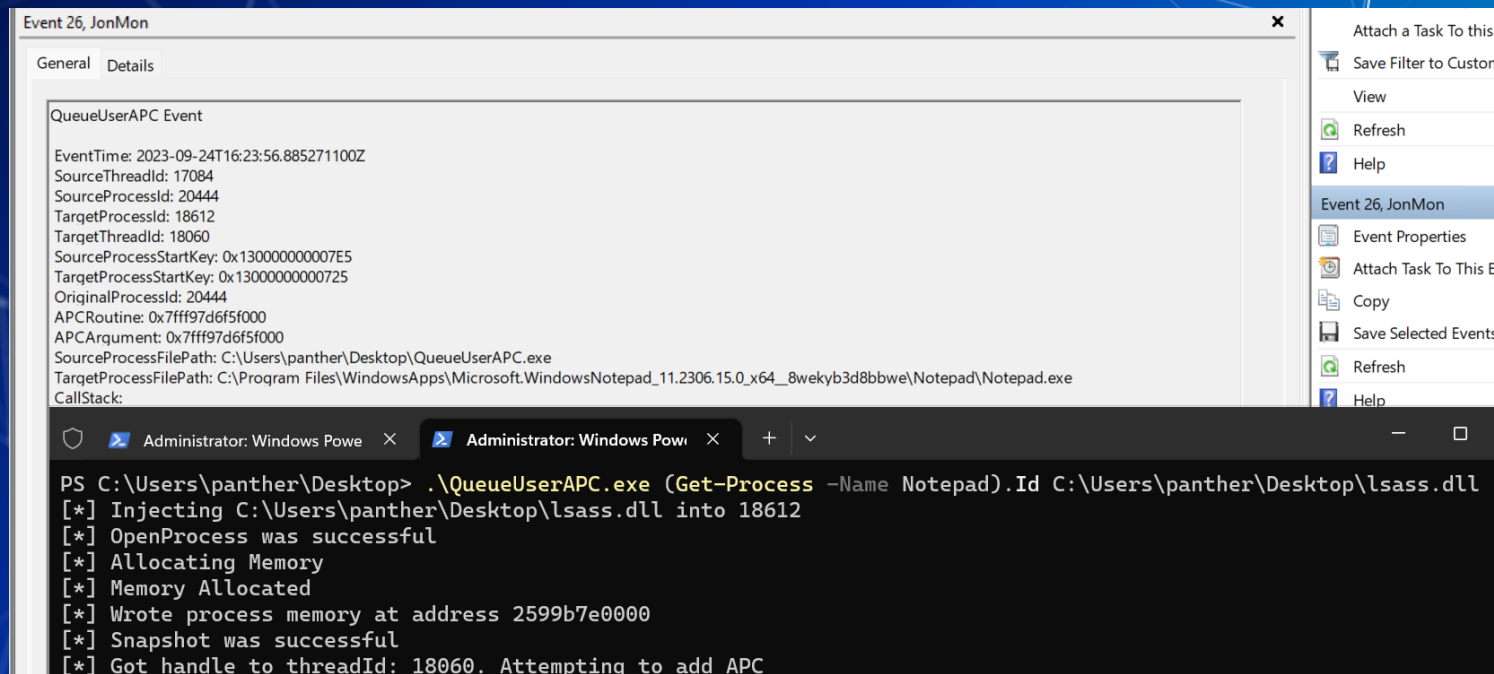
The Cobalt Strike console on the right shows the following output:

```
beacon> spawn SMB
[*] Tasked beacon to spawn (x64) windows/beacon_bind_pipe (\\.\pipe\mojo.5688.8052.18389493978708887759)
[+] host called home, sent: 255570 bytes
[+] established link to child beacon: 172.20.79.101
```

Below the console output, a status bar indicates: [WAKANDA-WRKSTN] panther */6704 (x64) beacon> Status: Running. A 'Refresh' button is visible at the bottom right of the console window.

QueueUserAPC

■ Event ID 26 – QueueUserAPC



The screenshot displays the Windows Event Viewer window for Event ID 26, titled "Event 26, JonMon". The "General" tab is selected, showing the following details:

- QueueUserAPC Event
- EventTime: 2023-09-24T16:23:56.885271100Z
- SourceThreadId: 17084
- SourceProcessId: 20444
- TargetProcessId: 18612
- TargetThreadId: 18060
- SourceProcessStartKey: 0x1300000000007E5
- TargetProcessStartKey: 0x130000000000725
- OriginalProcessId: 20444
- APCRoutine: 0x7fff97d6f5f000
- APCArument: 0x7fff97d6f5f000
- SourceProcessFilePath: C:\Users\panther\Desktop\QueueUserAPC.exe
- TargetProcessFilePath: C:\Program Files\WindowsApps\Microsoft.WindowsNotepad_11.2306.15.0_x64_8wekyb3d8bbwe\Notepad\Notepad.exe
- CallStack:

Below the event details, a Windows Command Prompt window is open, showing the execution of the QueueUserAPC.exe tool. The command used is:

```
PS C:\Users\panther\Desktop> .\QueueUserAPC.exe (Get-Process -Name Notepad).Id C:\Users\panther\Desktop\lsass.dll
```

The output of the command is as follows:

```
[*] Injecting C:\Users\panther\Desktop\lsass.dll into 18612
[*] OpenProcess was successful
[*] Allocating Memory
[*] Memory Allocated
[*] Wrote process memory at address 2599b7e0000
[*] Snapshot was successful
[*] Got handle to threadId: 18060. Attempting to add APC
```

RPC Calls

- Don't need to collect "everything"
- Focus on collecting known bad

Methods:

- EfsRpcOpenFileRaw (patched by Microsoft via CVE-2021-36942)
- EfsRpcEncryptFileSrv
- EfsRpcDecryptFileSrv
- EfsRpcQueryUsersOnFile
- EfsRpcQueryRecoveryAgents
- EfsRpcRemoveUsersFromFile
- EfsRpcAddUsersToFile

```
// MS-EFSR {D9A0A0C0-150F-11D1-8C7A-00C04FC297EB} || {C681D488-D850-11D0-8C52-00C04FD90F7E} ||
if ((wcsncmp(szInterfaceUUID, L"{C681D488-D850-11D0-8C52-00C04FD90F7E}") == 0) ||
    const wchar_t* InterfaceString = L"MS-EFSR";
    switch (procNum) {
    case 0:
    {
        const wchar_t* MethodString = L"EfsRpcOpenFileRaw";
        wchar_t* CallStack = GetCallStack(EventRecord, extendedData, hProcess);
        BOOL WriteEvent = WriteRPCEvent(EventRecord, EventHeader, RPCEvent, (wch
        delete[] CallStack;
        break;
```


RPC Calls

■ Event ID 11 – Client

Event 11, JonMon

General Details

RPCClientCall Event

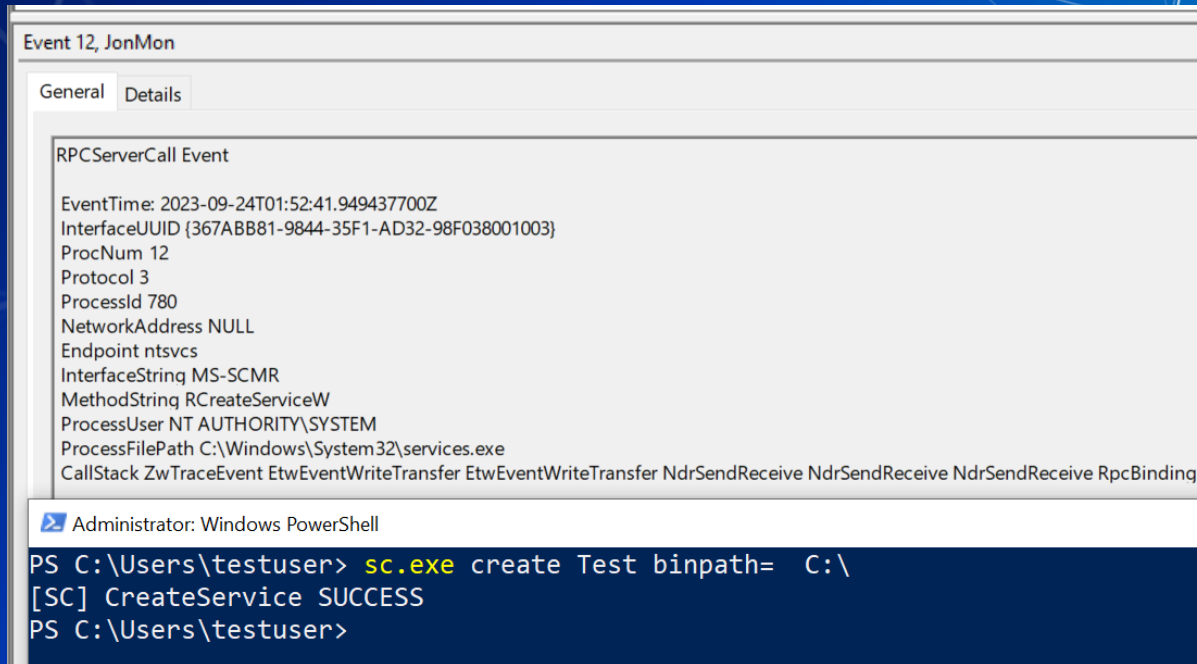
EventTime: 2023-09-24T00:38:05.508786900Z
InterfaceUUID: {367ABB81-9844-35F1-AD32-98F038001003}
ProcNum: 12
Protocol: 3
ProcessId: 9252
NetworkAddress: NULL
Endpoint: ntsvcs
InterfaceString: MS-SCMR
MethodString: RCreateServiceW
ProcessUser: DESKTOP-K0LULNE\TestUser
ProcessFilePath: Unknown - process potentially died
CallStack: ZwTraceEvent EtwEventWriteTransfer EtwEventWriteTransfer NdrSendReceive NdrSendReceive NdrSendReceive NdrpClientCall2 NdrClientCall2 CreateServiceW

Administrator: Windows PowerShell

```
PS C:\Users\TestUser> sc.exe create TestService binpath= C:\Windows\System32\cmd.exe  
[SC] CreateService SUCCESS
```

RPC Calls

■ Event ID 12 - Server



Event 12, JonMon

General Details

RPCServerCall Event

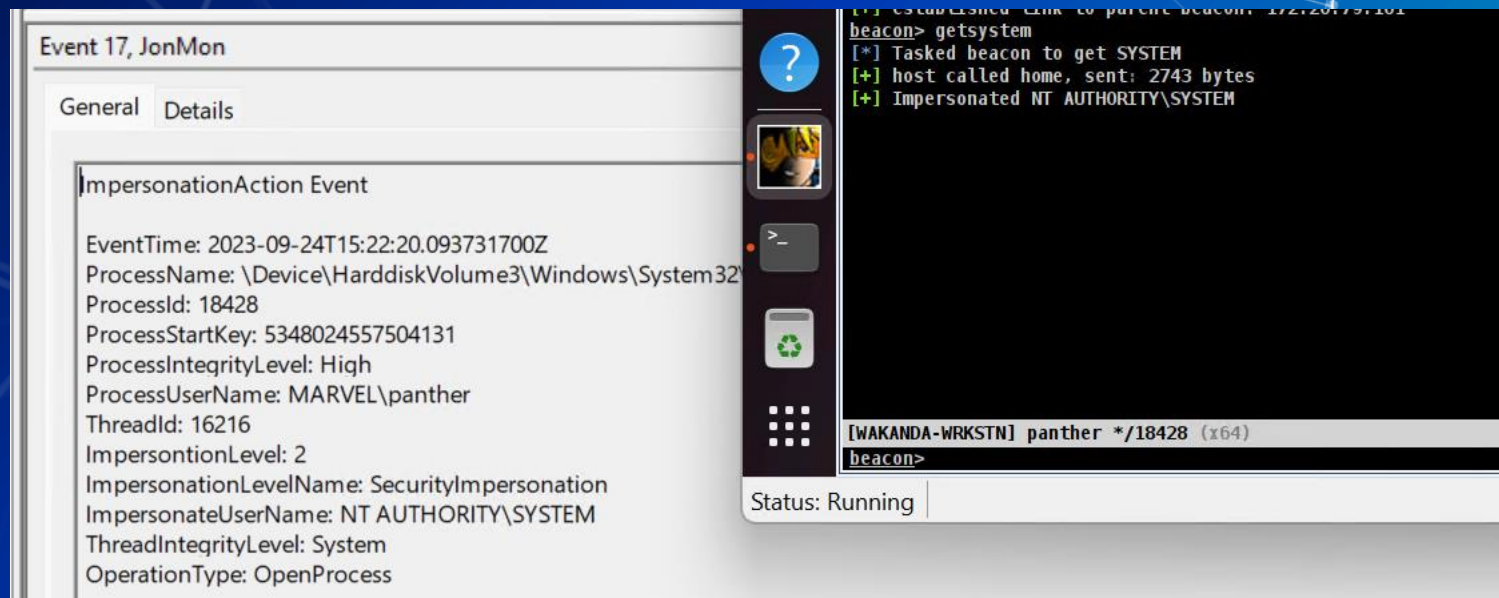
EventTime: 2023-09-24T01:52:41.949437700Z
InterfaceUUID {367ABB81-9844-35F1-AD32-98F038001003}
ProcNum 12
Protocol 3
ProcessId 780
NetworkAddress NULL
Endpoint ntsvcs
InterfaceString MS-SCMR
MethodString RCreateServiceW
ProcessUser NT AUTHORITY\SYSTEM
ProcessFilePath C:\Windows\System32\services.exe
CallStack ZwTraceEvent EtwEventWriteTransfer EtwEventWriteTransfer NdrSendReceive NdrSendReceive NdrSendReceive RpcBinding

Administrator: Windows PowerShell

```
PS C:\Users\testuser> sc.exe create Test binpath= C:\[SC] CreateService SUCCESS
PS C:\Users\testuser>
```

Token Impersonation

■ Event ID 17 - PsReferenceImpersonationToken



The image displays a Windows Event Viewer window on the left and a terminal window on the right, illustrating a token impersonation event.

Event Viewer Details:

- Event 17, JonMon**
- General** | Details
- ImpersonationAction Event**
- EventTime:** 2023-09-24T15:22:20.093731700Z
- ProcessName:** \Device\HarddiskVolume3\Windows\System32
- ProcessId:** 18428
- ProcessStartKey:** 5348024557504131
- ProcessIntegrityLevel:** High
- ProcessUserName:** MARVEL\panther
- ThreadId:** 16216
- ImpersonationLevel:** 2
- ImpersonationLevelName:** SecurityImpersonation
- ImpersonateUserName:** NT AUTHORITY\SYSTEM
- ThreadIntegrityLevel:** System
- OperationType:** OpenProcess

Terminal Window:

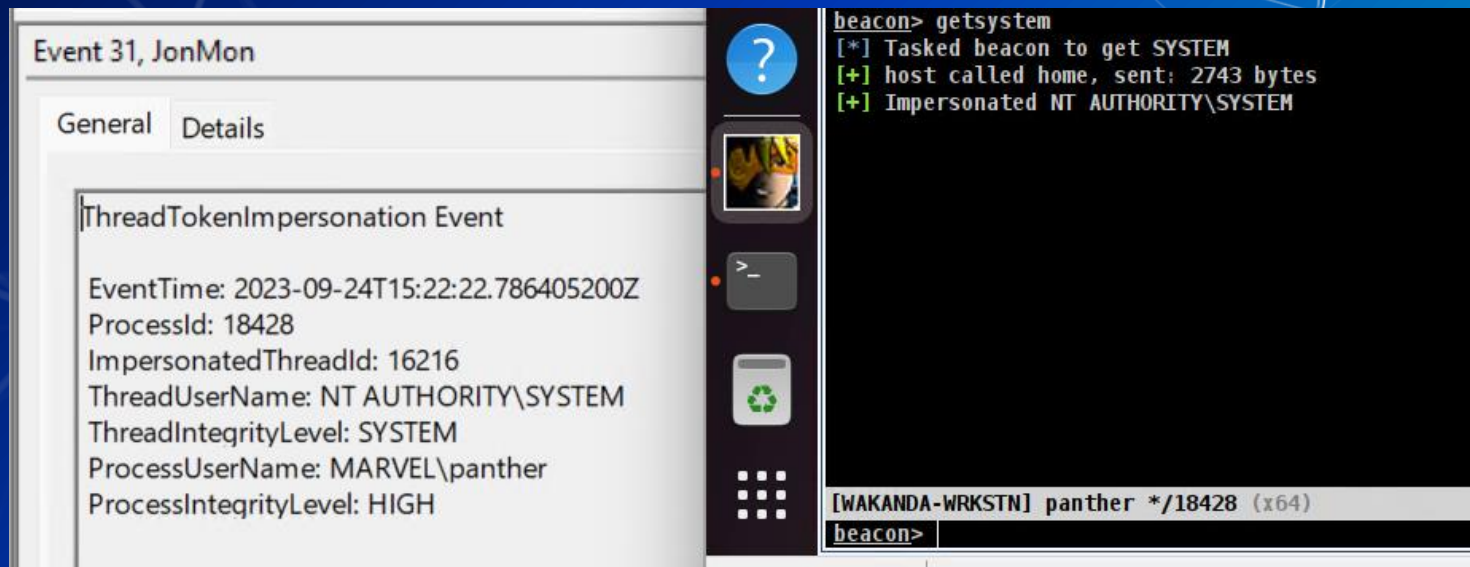
```
[*] Established link to parent beacon: 172.20.179.101
beacon> getsystem
[*] Tasked beacon to get SYSTEM
[+] host called home, sent: 2743 bytes
[+] Impersonated NT AUTHORITY\SYSTEM
```

Taskbar:

- Icons: Question mark, Wakaanda character, Command prompt, Recycle bin, App launcher.
- [WAKANDA-WRKSTN] panther */18428 (x64)**
- beacon>**
- Status:** Running

Token Impersonation

■ Event ID 31 – Thread Token Impersonation



The image displays a Windows Event Viewer window showing Event ID 31, titled "Event 31, JonMon". The "General" tab is selected, showing the event name "ThreadTokenImpersonation Event". The event details are as follows:

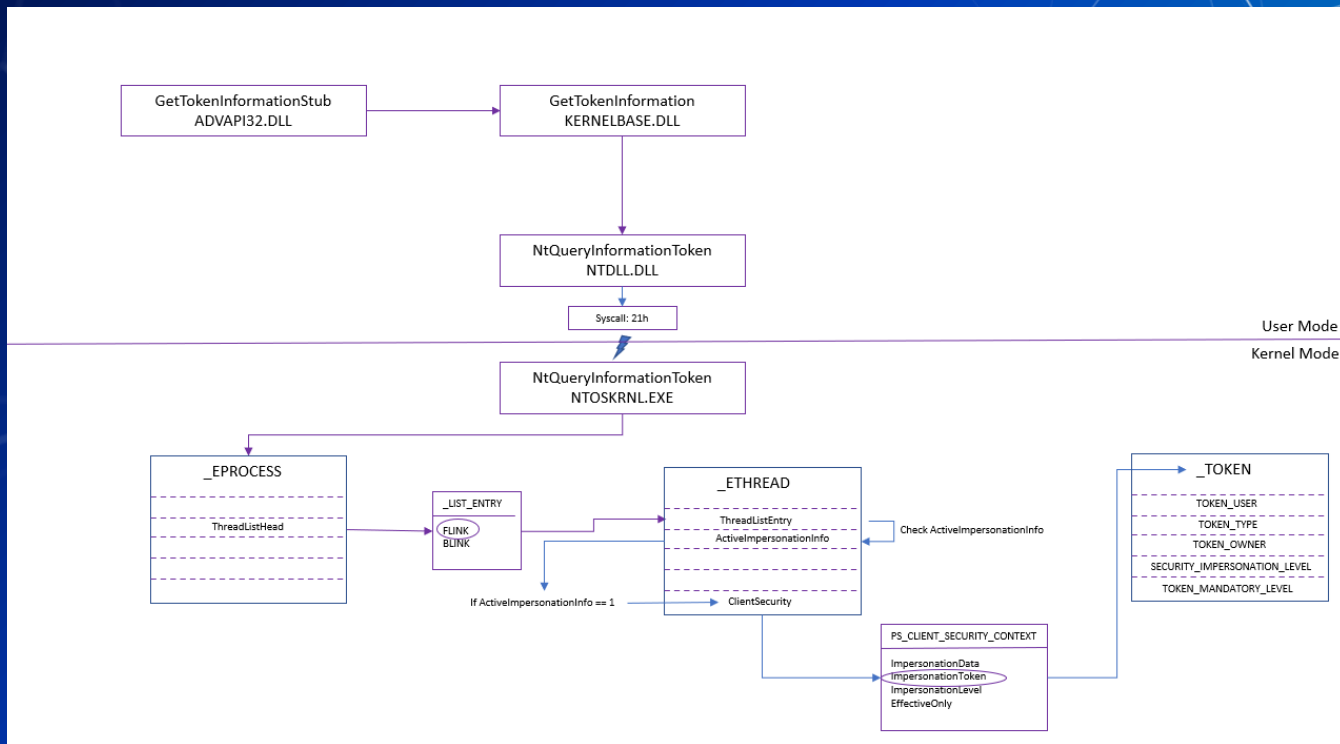
- EventTime: 2023-09-24T15:22:22.786405200Z
- ProcessId: 18428
- ImpersonatedThreadId: 16216
- ThreadUserName: NT AUTHORITY\SYSTEM
- ThreadIntegrityLevel: SYSTEM
- ProcessUserName: MARVEL\panther
- ProcessIntegrityLevel: HIGH

To the right of the event details is a vertical toolbar with icons for help, a thumbnail, a command prompt, a recycle bin, and a grid. Further right is a terminal window with the following output:

```
beacon> getsystem
[*] Tasked beacon to get SYSTEM
[+] host called home, sent: 2743 bytes
[+] Impersonated NT AUTHORITY\SYSTEM
```

At the bottom of the terminal, a status bar shows "[WAKANDA-WRKSTN] panther */18428 (x64)" and the prompt "beacon>".

Token Impersonation



WMI Event Subscription

■ Event ID 25 – WMIFilterToConsumerBinding



General Details

WMIFilterToConsumerBinding Event

EventTime: 2023-09-24T22:06:39.773834700Z

Namespace: //./root/subscription

ESS: ActiveScriptEventConsumer--test

Consumer: ActiveScriptEventConsumer="ActiveScriptEventConsumer--test"

PossibleCause: Binding EventFilter:

instance of __EventFilter

```
{
  CreatorSID = {1, 5, 0, 0, 0, 0, 5, 21, 0, 0, 0, 145, 61, 145, 136, 103, 150, 18, 116, 206, 74, 72, 89, 233, 3, 0, 0};
  EventNamespace = "root\cimv2";
  Name = "ActiveScriptEventConsumer--test";
  Query = "SELECT * FROM __InstanceModificationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.CommandLine LIKE '%powershell%'";
  QueryLanguage = "WQL";
};
```

Perm. Consumer:

instance of ActiveScriptEventConsumer

```
{
  CreatorSID = {1, 5, 0, 0, 0, 0, 5, 21, 0, 0, 0, 145, 61, 145, 136, 103, 150, 18, 116, 206, 74, 72, 89, 233, 3, 0, 0};
  Name = "ActiveScriptEventConsumer--test";
  ScriptingEngine = "VBScript";
  ScriptText = "Dim shl \n Set shl = CreateObject(\"Wscript.Shell\") \n Call shl.Run(\"\"\"cmd.exe\"\"\") \n Set shl = Nothing \n WScript.Quit";
};
```

Administrator: Windows PowerShell

```
>> Query="SELECT * FROM __InstanceModificationEvent WITHIN 1 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.CommandLine LIKE '%powershell%'";
PS C:\Users\testuser\Desktop\JonMon> $Filter=New-CimInstance -Namespace root/subscription -ClassName __EventFilter -Property $FilterArgs
PS C:\Users\testuser\Desktop\JonMon>
PS C:\Users\testuser\Desktop\JonMon>
PS C:\Users\testuser\Desktop\JonMon> $ConsumerArgs = @{name='ActiveScriptEventConsumer--test';
>> ScriptingEngine='VBScript';
>> ScriptText='Dim shl
>> Set shl = CreateObject("Wscript.Shell")
>> Call shl.Run("""cmd.exe""")
>> Set shl = Nothing
>> WScript.Quit';}
```

Drivers Loads

■ Event ID 27– Driver Load

Operational Number of events: 105,482 (!) New events available

Filtered: Log: JonMon/Operational; Source: ; Event ID: 27. Number of events: 2

Level	Date and Time	Source	Event ID	Task Category
Information	9/16/2023 9:21:10 PM	JonMon	27	None
Information	9/16/2023 9:21:10 PM	JonMon	27	None

Event 27, JonMon

General Details

DriverLoad Event

EventTime: 2023-09-16T21:21:10.232907700Z
ModulePath: \Device\HarddiskVolume3\Program Files\SystemInformer\SystemInformer.sys

Lessons Learned

- Kernel development isn't for the faint of heart
- Make the kernel do as little work/processing as possible
- Telemetry isn't necessarily easy to obtain
- Event consistency
- Understanding telemetry mechanisms helps improve Windows Internals knowledge
- Metadata isn't just "given"

ROADMAP

Config file

1

Move from
Manifest ETW to
Tracelogging

3

OSSEM

5

Collaboration
Templates

2

Minifilter multi-
threading

4

..AND MORE!!!

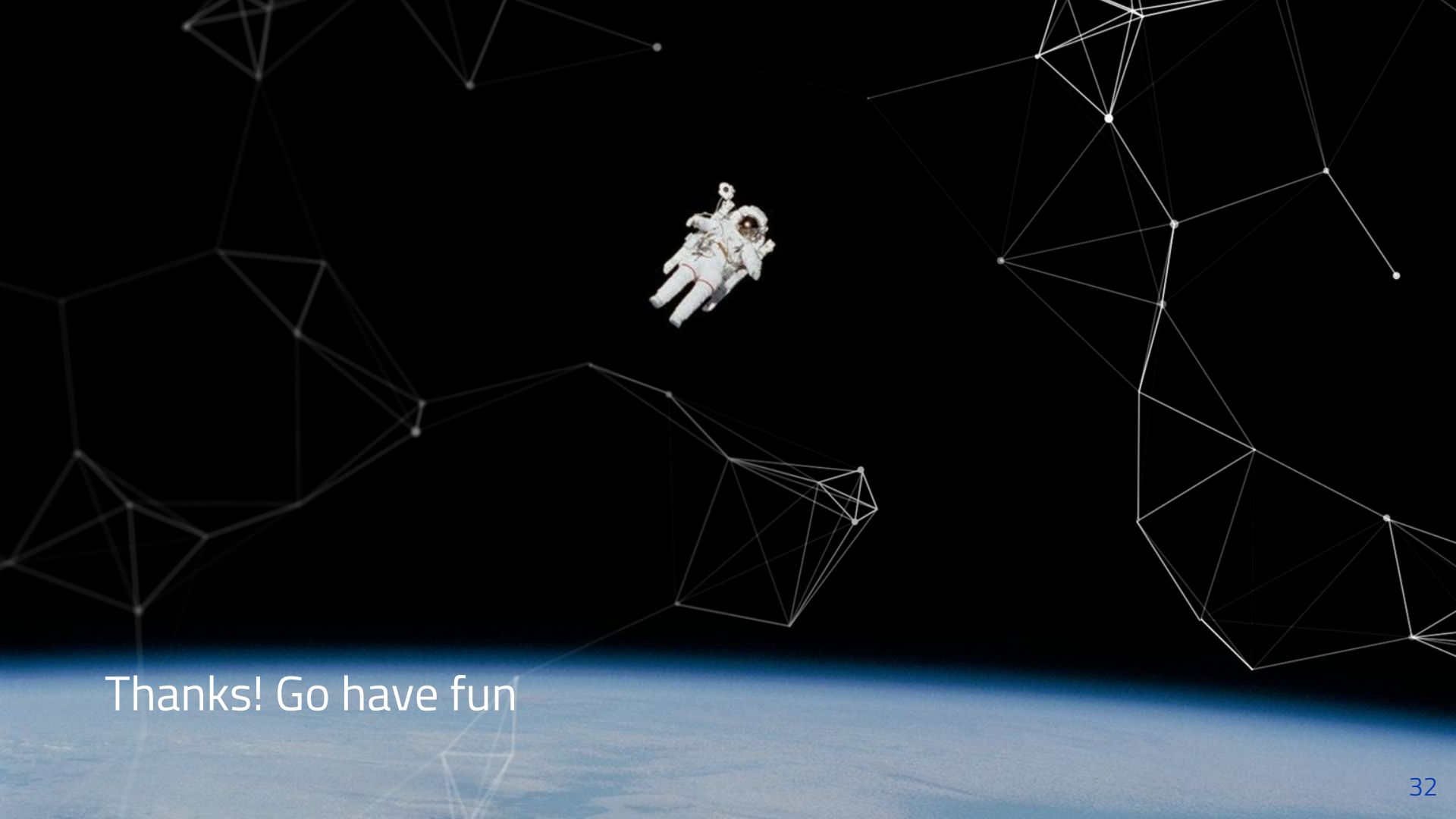
6

Bad News.....

- Code isn't quite ready yet..
- SIKE!!!!!!!!!!!!
- GO TO:



<https://github.com/jsecurity101/JonMon>



Thanks! Go have fun