SYSMONX

Augmented Drop-In Replacement of Sysmon

https://github.com/marcosd4h/sysmonx

Marcos Oviedo (**@marcosd4h**)

Agenda

- About Me
- **WHY** SysmonX
- **HOW** this can be done
- **WHAT** SysmonX brings to us
- Demo

About Me

- **Marcos Oviedo - @marcosd4h**
- Infosec is my Passion!
- I truly believe on pushing the limits of open source tools
- Speaker at last 2 Arsenal Tracks at Blackhat USA, and last Defcon 27 Demo Labs + Defcon 27 BTV
- Software Architect at McAfee
- BSides Córdoba Argentina Organizer – bsidescordoba.org
- Reachable at moviedo [at] gmail.com

WHY SysmonX

WHY SysmonX

- Sysmon is widely used for monitoring and threat visibility, with numerous solutions designed around its events

- Sysmon has gaps in visibility and functionality
  - Ability to easily add new event visibility (ETW, WMI, etc)
  - Ability to add metadata to filter on events
  - Ability to correlate multiple events together
  - Poor signal-to-noise ratio in captured events
  - Weak against multiple subversion techniques

- Rapidly responding to and resolving these challenges in a dynamic threat landscape is ready made for an open-source community driven approach

WHY SysmonX (contd)

Threat Detection Engineering is getting complex as threat landscape evolves

Blueteamers rely on Sysmon for visibility and struggle with its limitations

Blueteamers require a way to contribute not only with content but also with new ways to collect data

SysmonX is a framework that can be used by the community to drive threat detection engineering content and visibility

HOW this can be done

Creating a new version of Sysmon

- If we want project to be adopted, we would have to provide current Sysmon value + more

- This implicitly means that alternative version of Sysmon has to be feature compatible + maintain the same user experience (Same input – Same output)
    - Filtering
    - Reporting

- Also, reported events should contain the same forensic data

- OK, but where does Sysmon gets its data anyway?

Creating a new version of Sysmon (contd)

- Sysmon uses a driver + a windows service to collect operating system events and operate on them

- Sysmon also have a utility in charge of creating the sysmon windows service and deploying the windows driver

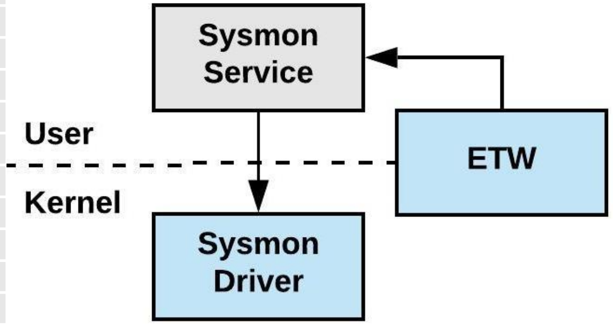
- Once created and running, the sysmon driver will register for Kernel Callbacks notifications and it will register Minifilter IRP Function Codes Handlers
  - This will allow driver to receive notifications on major OS events

- The sysmon windows service will also consume ETW data + and will register for WMI notifications

- The sysmon windows service will collect this information and will report it accordingly

Creating a new version of Sysmon (contd)

SYSMON EVENTS				
Event ID		Event Detail	Event Data Source	
1	ProcessCreate	Process Create	Driver - Kernel Callback	PsSetCreateProcessNotifyRoutine()
2	FileCreateTime	File creation time	Driver - Minifilter	Minifilter Function Codes Callback Handlers
3	NetworkConnect	Network connections	Service - ETW	Windows Kernel Trace -net
5	ProcessTerminate	Process terminated	Driver - Kernel Callback	PsSetCreateProcessNotifyRoutine()
6	DriverLoad	Driver Loaded	Driver - Kernel Callback	PsSetLoadImageNotifyRoutine()
7	ImageLoad	Image loaded	Driver - Kernel Callback	PsSetLoadImageNotifyRoutine()
8	CreateRemoteThread	CreateRemoteThread detected	Driver - Kernel Callback	PsSetCreateThreadNotifyRoutine()
9	RawAccessRead	RawAccessRead detected	Driver - Minifilter	Minifilter Function Codes Callback Handlers
10	ProcessAccess	Process accessed	Driver - Kernel Callback	ObRegisterCallbacks()
11	FileCreate	File created	Driver - Minifilter	Minifilter Function Codes Callback Handlers
12	RegistryEvent	Registry object added or deleted	Driver - Kernel Callback	CmpCallback()
13	RegistryEvent	Registry value set	Driver - Kernel Callback	CmpCallback()
14	RegistryEvent	Registry object renamed	Driver - Kernel Callback	CmpCallback()
15	FileCreateStreamHash	File stream created	Driver - Minifilter	Minifilter Function Codes Callback Handlers
17	PipeEvent	Named pipe created	Driver - Minifilter	Minifilter Function Codes Callback Handlers
18	PipeEvent	Named pipe connected	Driver - Minifilter	Minifilter Function Codes Callback Handlers
19	WmiEvent	WMI filter	Service - WMI	WMI COM APIs
20	WmiEvent	WMI consumer	Service - WMI	WMI COM APIs
21	WmiEvent	WMI consumer filter	Service - WMI	WMI COM APIs
22	DNSQuery	DNS query	Service - ETW	Microsoft-Windows-DNS-Client



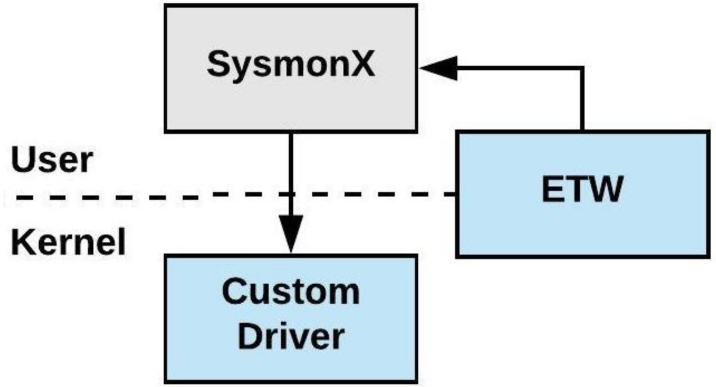
SysmonX approaches – First Iteration

OK, so we need a custom driver then? Yes, but doing it for first SysmonX version is problematic

- We will have to go through WHQL Signing process (Time and Money)
- We will have to produce a bug-free version of the driver from scratch (Requires a lot of expertise)

Can we just use ETW information to get the data we needed?

- Not really, there are information that it is not available through ETW (i.e Timestomping event)
- Using just ETW means that we are dropping security events and reporting only what’s possible (what is available through ETW). This breaks the drop-in compatible contract



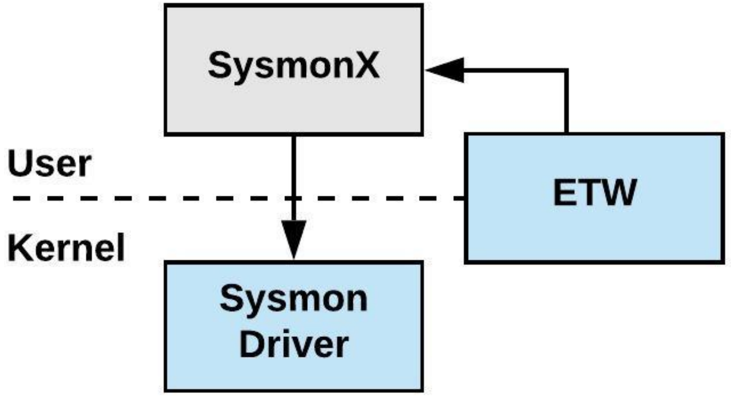
SysmonX approaches – Second Iteration

Can we just use the Sysmon driver instead of creating our own?

- Not really. There are license limitations specified on the Sysinternals Software License terms

License restrict us from

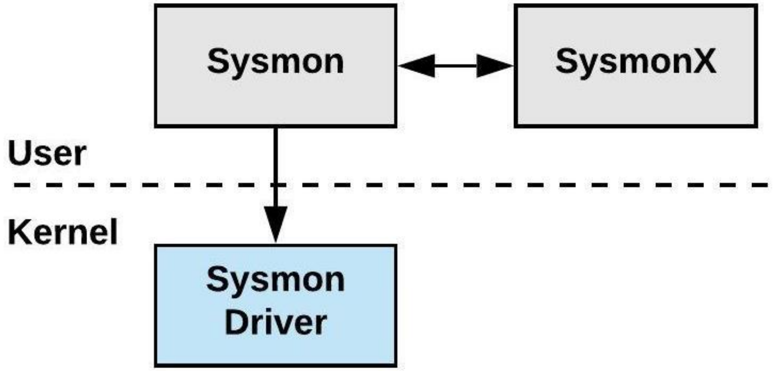
- Packing binary version of the sysmon utility (driver + service)
- Reverse Engineer components of the driver
- Public the software for others to copy



SysmonX approaches – Third Iteration

What if we deploy sysmon on behalf of the user and we just listen for its events?

- This is possible!
- SysmonX should download and install sysmon as requested by the user (no binary packaging)
- First time users would have to accept Sysmon EULA as expected
- Sysmon events can be listened through “Microsoft-Windows-Sysmon” ETW provider
- Sysmon will be configured as a sensor.
  - This is, to just report everything to SysmonX.
  - No filtering

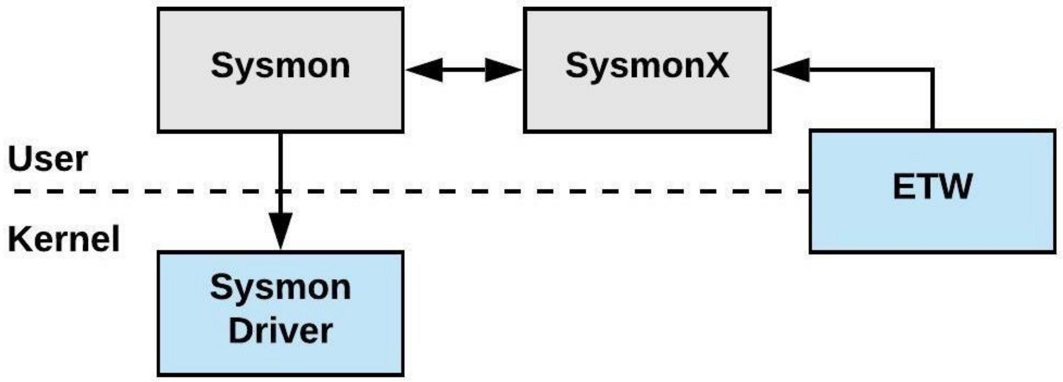


SysmonX approaches – Final Iteration

We can now listen for Sysmon events

What about new data sources? ETW to the rescue!

- SysmonX provides the foundation to listen and filter on numerous ETW events
- SysmonX also provides an architecture to extend sysmon functionality in several ways



WHAT SysmonX brings to us

SysmonX project goals

More visibility of threat activity

- Extend the Sysmon data collection sources and create new security events

Additional extensible threat detection.

- Extend the Sysmon ability to correlate events. Effectively enabling new logical operations between events and the creation of advanced detection capabilities

Improved signal to noise

- Enable the false positive reduction by narrowing down suspicious events through dedicated scanners

More resilient

- React to known subversion and evasion techniques that impact Sysmon

Community driven

- Leverage open source and community to improve security outcomes

SysmonX project goals (contd)

SysmonX is meant to become a framework that empowers the community to

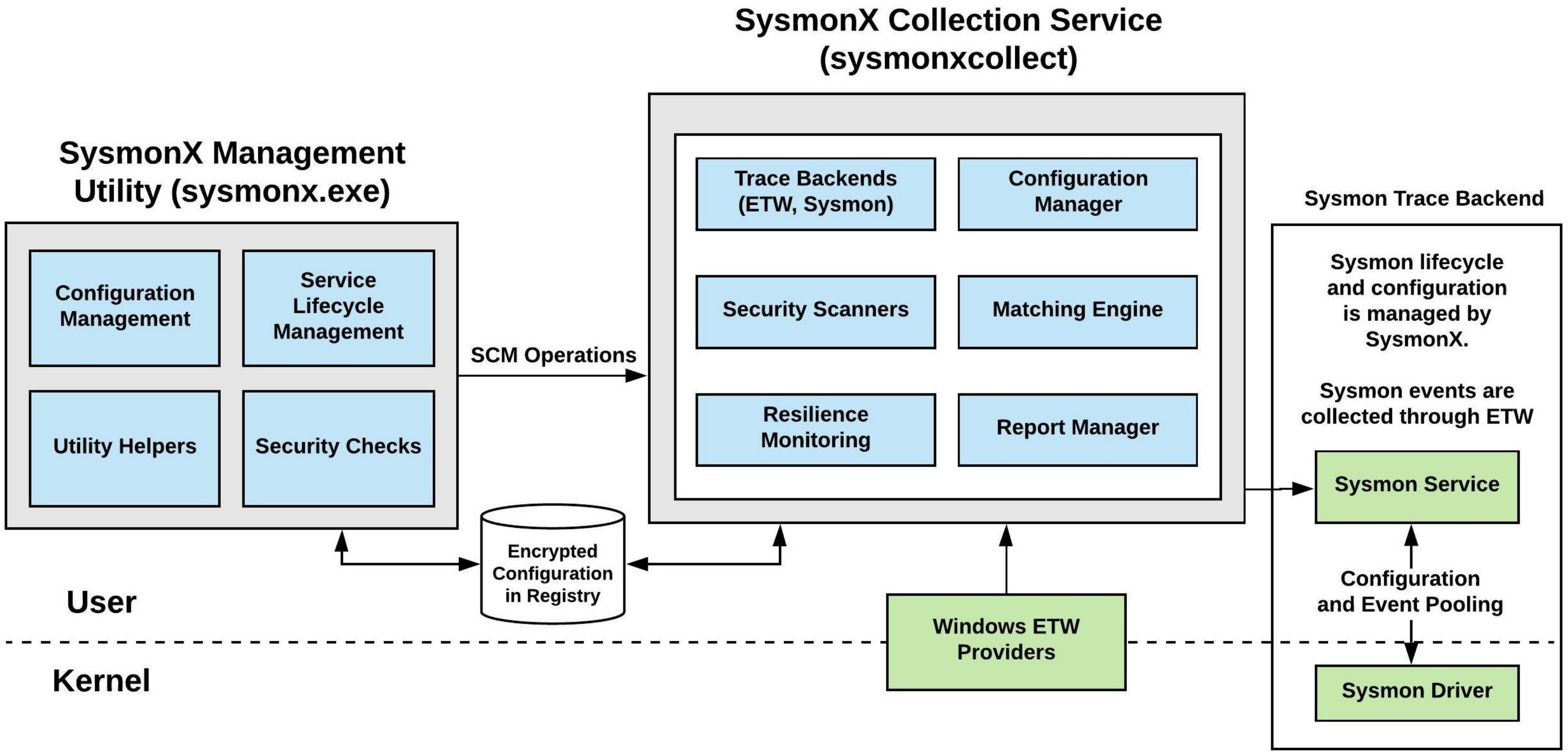
- Create new detection content
- Create new visibility
- Have a common place for detection engineering

SYSMONX

Augmented and community-driven version of Sysmon

https://github.com/marcosd4h/sysmonx

SysmonX current Architecture


SysmonX Collection Service (sysmonxcollect)
SysmonX Management Utility (sysmonx.exe)
Configuration Management
Service Lifecycle Management
Utility Helpers
Security Checks
SCM Operations
Trace Backends (ETW, Sysmon)
Configuration Manager
Security Scanners
Matching Engine
Resilience Monitoring
Report Manager
Sysmon Trace Backend
Sysmon lifecycle and configuration is managed by SysmonX.
Sysmon events are collected through ETW
Sysmon Service
Configuration and Event Pooling
Sysmon Driver
Encrypted Configuration in Registry
Windows ETW Providers
User
Kernel

SysmonX Features Development Priority

Sysmon Utility

1. Configuration Management
2. Trace backend Management
3. Service deployment and management

SysmonX Collection Service

1. Configuration Management
2. Trace Backends Management
3. Report Management
4. Matching Engine ← **We are here**
5. New Event Data Events
6. New Event Data Sources
7. Security Scanners
8. Resilience Monitoring

SysmonX demo

SysmonX deployment

New Detection

- Ability to detect userspace injection techniques (eventing + memory inspection through built in scanner modules)

Extensible

- Ability to perform regex over security event fields

More visibility

- Ability to collect and filter on PowerShell events

DEMO TIME

Questions?

@marcosd4h
moviedo [at] gmail.com

SYSMONX

Augmented and community-driven version of Sysmon

https://github.com/marcosd4h/sysmonx

References

- https://ackroute.com/post/2017/08/08/sysmon-enumeration-overview/
- https://i.blackhat.com/us-18/Wed-August-8/us-18-Graeber-Subverting-Sysmon-Application-Of-A-Formalized-Security-Product-Evasion-Methodology-wp.pdf
- http://www.programmersought.com/article/777836183/
- http://www.programmersought.com/article/128236140/
- https://www.codercto.com/a/87154.html
- https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/irp-major-function-codes
- https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
- https://docs.microsoft.com/en-us/sysinternals/license-terms