


black hat[®]
EUROPE 2021

november 10-11, 2021

BRIEFINGS

Veni, **No Vidi, No Vici**: Attacks on **ETW** Blind **EDR** Sensors

Claudiu Teodorescu

Igor Korkin

Andrey Golchikov

The Binary Team

- Claudiu “to the rescue” Teodorescu – @cteo13
 - Digital Forensics, Reverse Engineering, Malware & Program Analysis
 - Instructor of Special Topics of Malware Analysis Course on BlackHat USA
 - Speaker at DEF CON, BSidesLV, DerbyCon, ReCon
- Igor Korkin – @IgorKorkin
 - PhD, Windows Kernel Researcher
 - Author of MemoryRanger
 - igorkorkin.blogspot.com
- Andrey “red plait” Golchikov – @real_redp
 - More than 20 years in researching operating system security and reversing Windows Internals
 - Author of EtwCheck
 - redplait.blogspot.com

Agenda

- Event Tracing for Windows (ETW)
 - Architecture and features
 - Applying ETW for security: academic and practical results
- Attacks on ETW blind the whole class of EDR solutions
 - Overview of existing attacks on ETW
 - Attacks on Process Monitor and Windows Defender
- EtwCheck detects attacks on ETW
- MemoryRanger prevents some kernel attacks on ETW

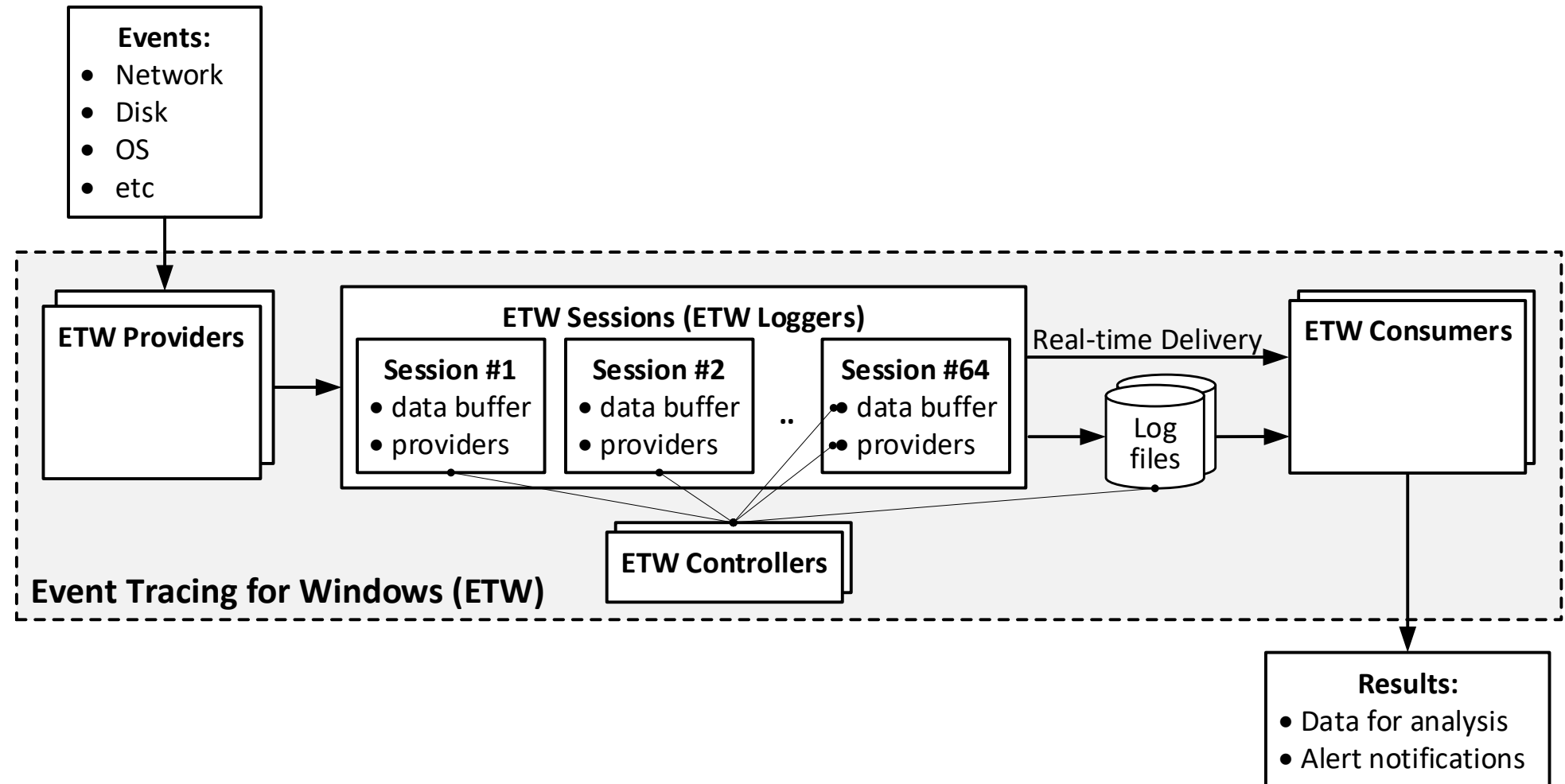
Event Tracing for Windows

ETW is a built-in diagnostic feature to log events from OS kernel, drivers and apps.

Windows 11: ETW can collect more than 50,000 events from about 1000 providers.

ETW features:

- System-wide
- Adjustable
- High speed logging
- Standardized
- Already Available
- Continual Features Growth
- Does not require rebooting
- Does not require driver installation
- Does not require any hardware features



ETW Providers: APPs, DLLs, OS Kernel and Drivers

Apps and services	DLLs	OS kernel and Drivers
<ul style="list-style-type: none"> • OS system <ul style="list-style-type: none"> • Smss • Wininit • Services • Security and AV <ul style="list-style-type: none"> • NisSrv • MsMpEng • SgrmBroker • Common Apps: <ul style="list-style-type: none"> • Explorer • Notepad • Msedge • Mspaint 	<ul style="list-style-type: none"> • OS system <ul style="list-style-type: none"> • Ntdll • KernelBase • Shell32 \ Advapi32 \ SetupAPI • Security and Encryption <ul style="list-style-type: none"> • Wintrust \ Amsi \ Wscapi • Rsaenh \ Ncrypt \ Bcrypt \ Crypt32 • Firewallapi \ FwpucInt • Network and WMI <ul style="list-style-type: none"> • Urlmon\ WinHTTP • Webio \ Wbemcomn • Iertutil \ DnsApi 	<ul style="list-style-type: none"> • OS system <ul style="list-style-type: none"> • Nt • Win32kbase \ Win32kfull • Security and Encryption <ul style="list-style-type: none"> • Cng \ FileCrypt • KSecdd • MsSecFlt • Tbs • Network and Devices <ul style="list-style-type: none"> • NTFS \ CimFS \ WciFS • VwifiFlt \ CldFlt \ BindFlt • HTTP \ TcpIp \ NetIO • Partmgr \ VolMgr \ NDIS • Disk \ CDrom \ ClassPNP

All these OS components can supply various ETW events

Using ETW for Malware Hunting



- Process
 - Process create and exit
 - Thread create and exit
 - Image loads, including o
- Network
 - ETW network tracing
- Profiling
 - Toolhelp thread snapshot

The first example of using ETW to trace malware activity:

(2012) [Malware Hunting with the Sysinternals Tools](#) by Mark Russinovich

2021: more than 10 market-leading EDR solutions use ETW for threat hunting

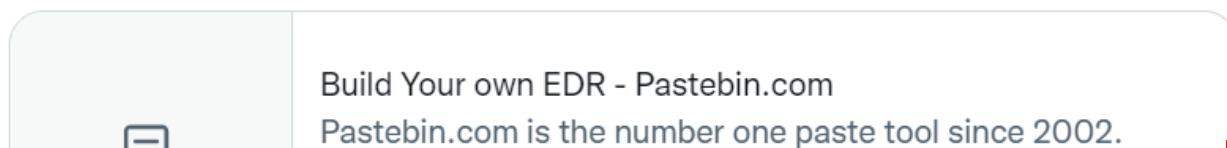
Disclaimer: This analysis is not designed to promote any products and solutions, it is meant to inform the cybersecurity community. The analysis of the products and solutions uses only materials freely available online.

Ideas of building EDRs based on ETW



Alex Ionescu
@aionescu

Build your own EDR with Microsoft's Threat Intelligence ETW channel: pastebin.com/6VGHjGjH cc @mattifestation @subTee @enigma0x3



Nasreddine Bencherchali
@nas_bench

Quick thread on how @symantec EDR uses event enrichment rules and ETW to detect "malicious activity"

Example : Brute Force attempt using event 4625

(1/)

[#detection](#) [#symantec](#) [#edr](#) [#BlueTeam](#)

11:55 PM · Aug 11, 2021 · Twitter Web App

<https://t.me/learningnets>



Matt Graeber
@mattifestation

I had a fun malware analysis session w/ a co-worker today and we were doing some maldoc analysis. With how much insight can now be gleaned via AMSI, we turned it into a dynamic analysis tool. Here are the steps we used to start a trace and dump events: [gist github.com/mgraeber-rc/51](https://gist.github.com/mgraeber-rc/51) 1/3



Black Hat ✓
@BlackHatEvents

Event Tracing for Windows (ETW) is a built-in feature, originally designed to perform software diagnostics, and nowadays ETW is essential for Endpoint Detection & Response (EDR) solutions. See a demo from [@roberpupum](#) at [#BHEU Arsenal](#) informatech.co/3jC89uu

9:33 PM · Sep 6, 2021 · Sprout Social

~100 research projects use ETW data

- [Tracing Adversaries: Detecting Attacks with ETW](#) by Matt Hastings (@_mhastings_) Dave Hull(@davehull)
- [Hidden Treasure: Detecting Intrusions with ETW](#) by Zac Brown (@zacbrown)
- [Hunting for Memory-Resident Malware](#) by Joe Desimone (@dez_)
- [Using Sysmon and ETW For So Much More](#) by David Kennedy (@HackingDave)



- Windows Low-Level System [Monitoring Data Collection](#)
- Project [MARPLE](#)
- Project [APTShield](#)



- MITRE-built ETW-based [Security Sensor](#)



- SiSyPHuS Project analyzed [Windows Logging Capabilities](#)

ETW based academic papers: 2019-2021

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

procmonML: Generating evasion resilient host-based behavioral analytics from tree ensembles

Joseph W. Mikhail*, Jamie C. Williams, George R. Roelke

The MITRE Corporation, United States

Malware Characterization Using Behavioral Components

Chaitanya Yavvari, Arnur Tokhtabayev, Huzefa Rangwala, and Angelos Stavrou

Computer Science Department, George Mason University, Fairfax, VA, USA
{cyavvari,atokhtab,astavrou}@gmu.edu, rangwala@cs.gmu.edu

PROBLEMCHILD: DISCOVERING ANOMALOUS PATTERNS BASED ON PARENT-CHILD PROCESS RELATIONSHIPS

Bobby Filar
Elastic
filar@elastic.co

Tactical Provenance Analysis for Endpoint Detection and Response Systems

Wajih Ul Hassan
University of Illinois at Urbana-Champaign
whassan3@illinois.edu

Adam Bates
University of Illinois at Urbana-Champaign
batesa@illinois.edu

Daniel Marino
NortonLifeLock Research Group
daniel.marino@nortonlifelock.com

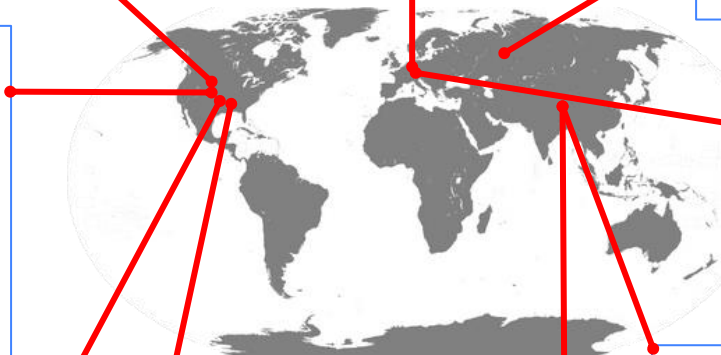
 UNIVERSITY OF AMSTERDAM

Detecting Fileless Malicious Behaviour of .NET C2 Agents using ETW

Alexander Bode abode@os3.nl
Niels Warnars nwarnars@os3.nl

REVEALING CRYPTOCURRENCY MINING MALWARE VIA ETW

Kazakov O.A.⁴, Korkin I.Y.⁵



Peeler: Profiling Kernel-Level Events to Detect Ransomware

Muhammad Ejaz Ahmed, Hyounghick Kim, Seyit Camtepe, and Surya Nepal

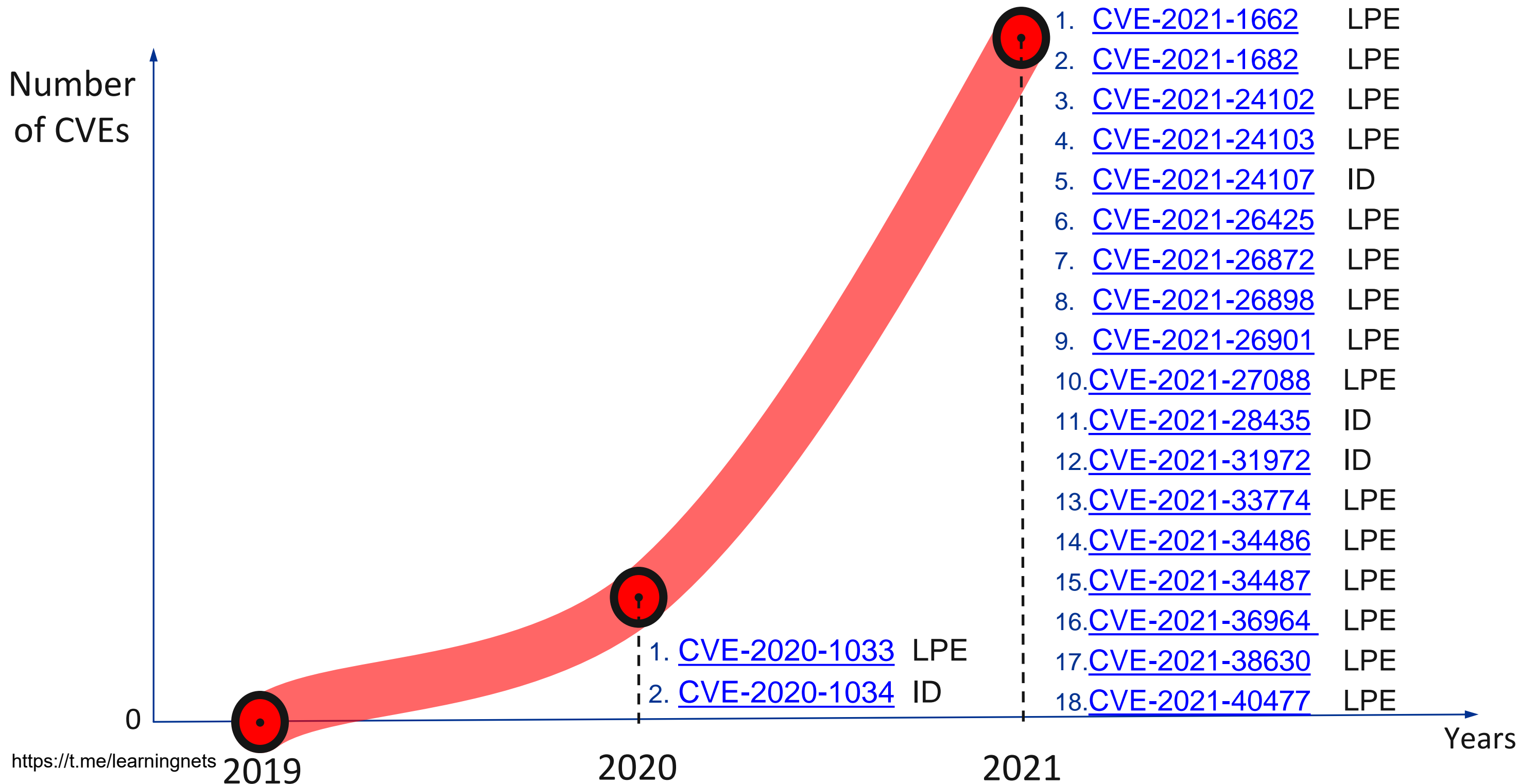
RATSCOPE: Recording and Reconstructing Missing RAT Semantic Behaviors for Forensic Analysis on Windows

Runqing Yang *, Xutong Chen *, Haitao Xu, Yueqiang Cheng, Chunlin Xiong, Linqi Ruan, Mohammad Kavousi, Zhenyuan Li, Liheng Xu, and Yan Chen, *Fellow, IEEE*

CONAN: A Practical Real-time APT Detection System with High Accuracy and Efficiency

Chunlin Xiong, Tiantian Zhu, Weihao Dong, Linqi Ruan, Runqing Yang, Yueqiang Cheng, Yan Chen, *Fellow, IEEE*, Shuai Cheng, and Xutong Chen

ETW is under the microscope of bug hunters



ETW IS HELPFUL FOR DEFENSE
BUT, IT HAS SOME DRAWBACKS

ETW Undocumented features

- ETW was originally designed for internal use:
 - ETW API has opaque structure
 - ETW includes several undocumented providers
 - Some ETW providers deliver events which are undocumented (ETW templates about events are stored in WEVT_TEMPLATE resource inside PE file)
- ETW can be used for malicious purposes:
 - ETW can be used as a [file](#), [registry](#), [network](#), [process](#), [thread](#), [drivers](#), and [keystroke sniffer](#) without installing kernel driver or registering callback functions
 - ETW can [flood HDD](#)
 - Some [ETW providers](#) are available only for apps with enabled Protected Process Light (PPL), but [malware can patch PPL Level](#) without BSOD
 - ETW can help to [detect sandbox](#) detonations
 - [No footprints](#): WinAPI do not return information about which app is consuming events.

ETW CAN BE BYPASSED

Disabling ETW is widely discussed



Grzegorz Tworek
@Ogtweet

Ever wanted to stop Eventlog-Security logger? Me too... And now I know it is not about ACLs. The way to stop unstoppable logger is to issue "TRUE" as the last parameter in EtwpStopTrace() call. And user can call it only through NtTraceCont "FALSE". 🙄



Joe Desimone
@dez_

Disable those pesky user mode etw loggers

```
void DisableEtw()  
{  
    // Disable any usermode etw loggers in the current process  
    DWORD dwOld = 0;  
    void * pEventWrite = GetProcAddress(GetModuleHandle(L"ntdll.dll"), "EtwEventWrite");  
    &dwOld);
```

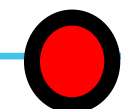


red plait
@real_redp

how you can find structure _TlgProvider_t and dynamically disable #etw tracing in any running processes: redplait.blogspot.com/2020/07/tlgpro...

5:38 PM · Jul 24, 2020 · Twitter Web App

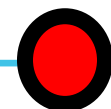
Malware Examples of evading ETW-based logging



2018

[APT Slingshot](#)

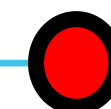
- renames ETW-logs to avoid leaving traces
- cyber-espionage platform



2019

[Ransomware LockerGoga](#)

- disables ETW to bypass host-based sensors
- Sends Norsk Hydro Into Manual Mode



2020-2021

[APT41 Group](#)

- disables ETW to evade detection
- Targets Indo-Pacific countries

Defense Evasion (post-exploitation) Frameworks:

- [SharpSploit](#) disable ETW monitoring for current process
- [ScareCrow](#) – payload creation framework bypasses EDR
- [EDR Evasion](#) – about 10 examples of blocking ETW logging

[MITRE ATT&CK – Impair Defenses](#)

- Indicator Blocking
- Disable Cloud Logs

Malware Examples of evading ETW-based logging

MITRE | ATT&CK®

Impair Defenses: Indicator Blocking (ID T1562.006)

- re
 - av
 - cy
 - pl
- An adversary may attempt to block indicators or events typically captured by sensors from being gathered and analyzed.
 - This could include maliciously redirecting [1] or even disabling host-based sensors, such as Event Tracing for Windows (ETW), [2] by tampering settings that control the collection and flow of event telemetry. [3]

Defense Evasion (post-exploitation) Frameworks:

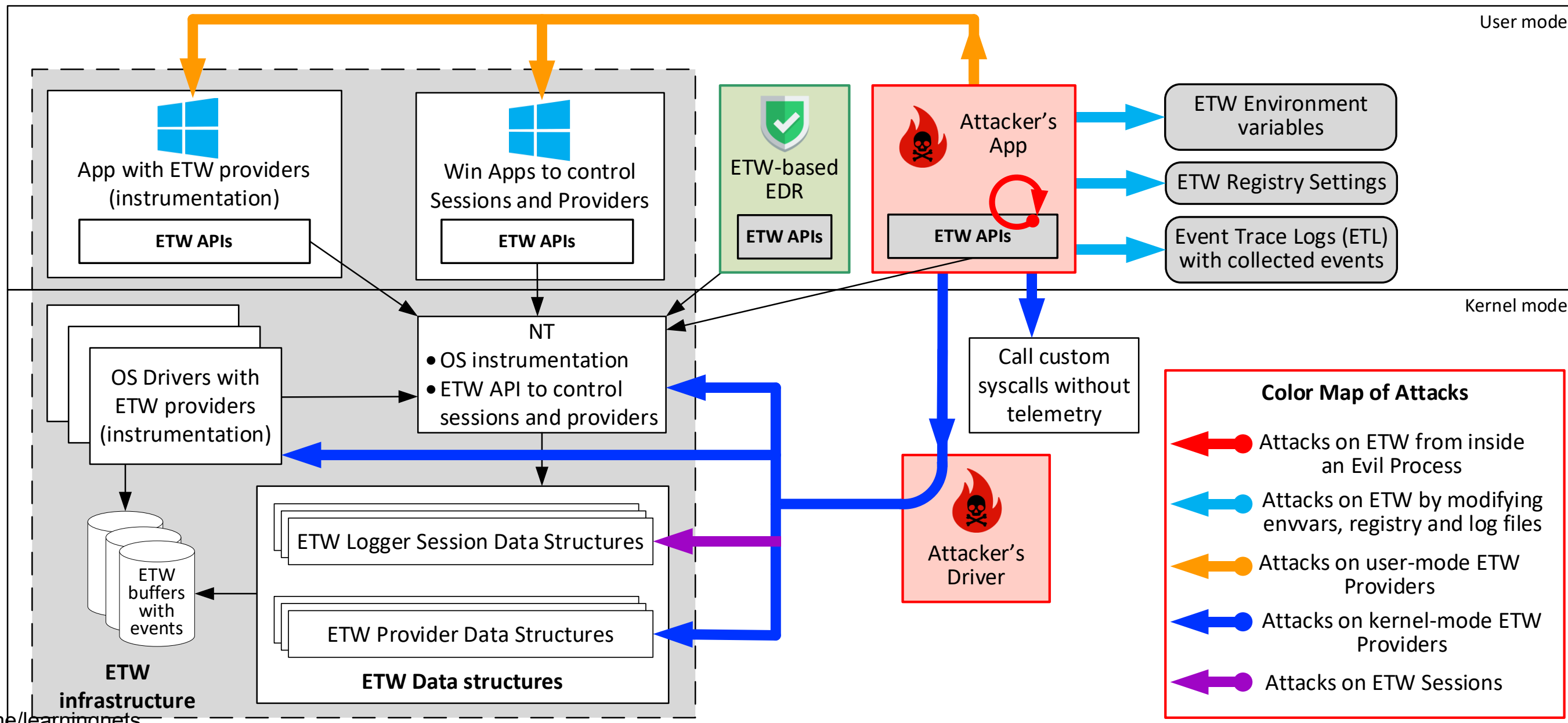
- [SharpSploit](#) disable ETW monitoring for current process
- [ScareCrow](#) – payload creation framework bypasses EDR
- [EDR Evasion](#) – about 10 examples of blocking ETW logging

MITRE ATT&CK – Impair Defenses

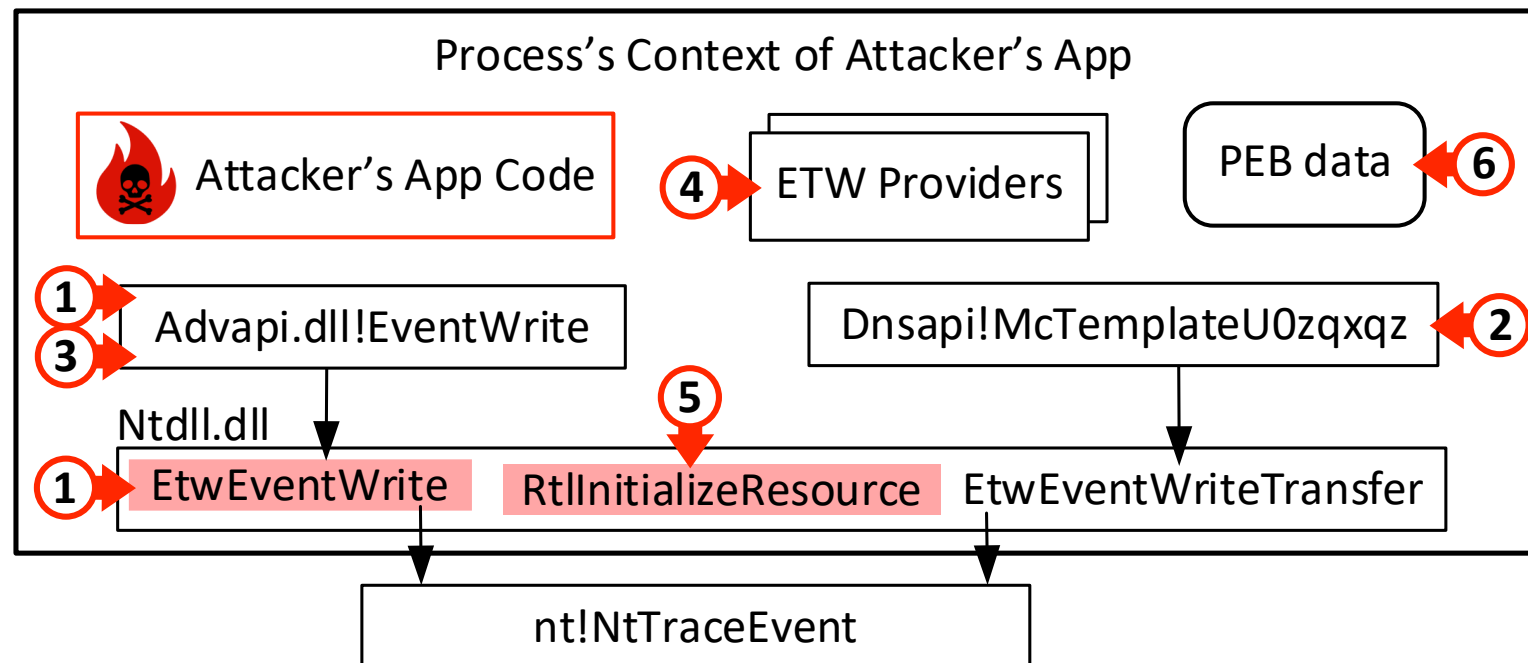
- Indicator Blocking
- Disable Cloud Logs

ATTACKS ON ETW – THE BIG PICTURE

Threat Modeling ETW

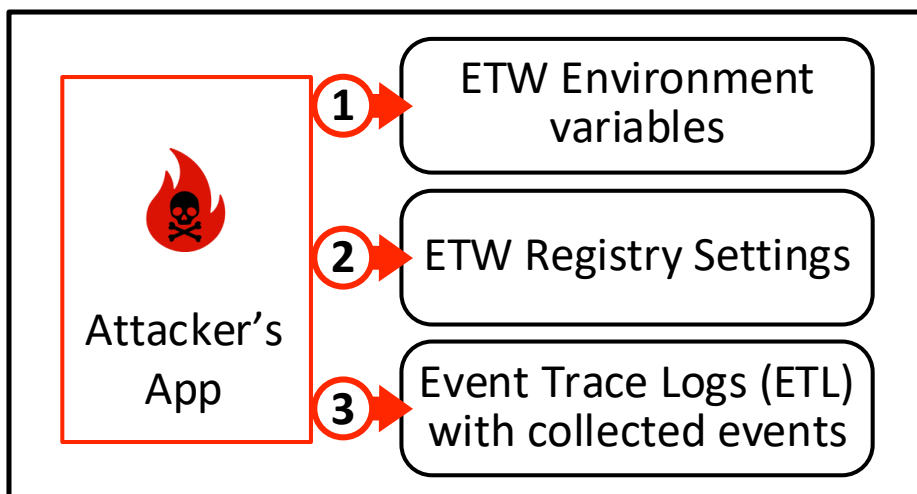


Local Attacks on ETW from inside an evil app



Attacks	References	Techniques
1 5 Block logging events about malware user-mode activity	1a , 1b , 1c , 1d , 1e , 1f , 5	Change the control flow using: <ul style="list-style-type: none"> Import Address Table (IAT) Hooking Inline hooking\Splicing Function patching with RET Hardware Breakpoints
2 Block logging events from <i>Microsoft-Windows-DNS-Client</i> provider	2	
3 Send bogus ETW events	3	
4 Disable Suspicious ScriptBlock Logging via PowerShell	4a , 4b	Set <code>m_enabled</code> in <code>PSEtwLogProvider</code> ETW Provider to FALSE
6 Fake source process image file and fake the list of loaded modules	6	Overwrite fields inside PEB

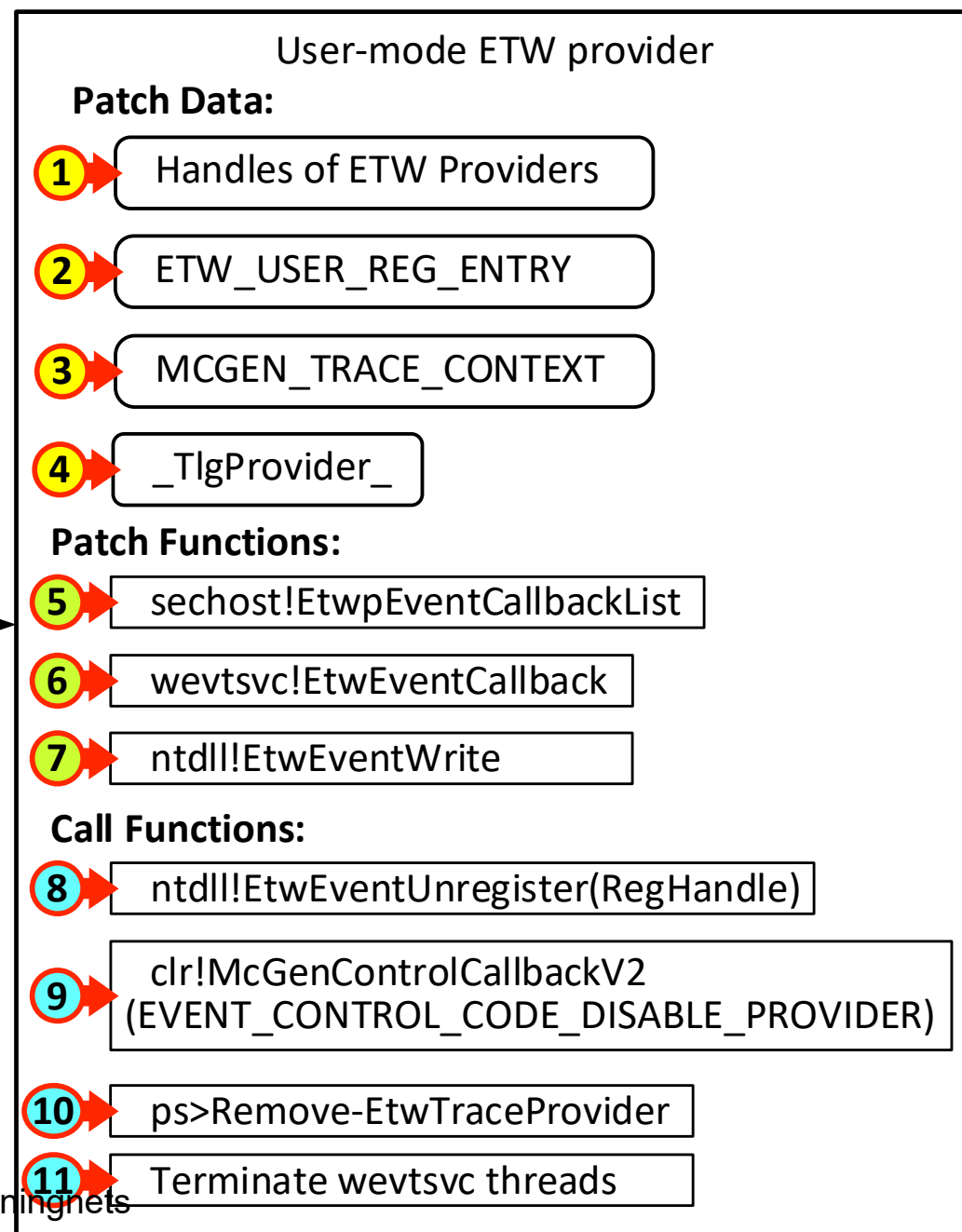
Attacks on ETW: EnvVars, Registry, and Log Files



Attacks	Techniques	References
① Disable Runtime Event Provider in .NET apps	Set the variables to zero : <ul style="list-style-type: none"> ▪ COMPlus_ETWFlags ▪ COMPlus_ETWEnabled 	1a , 1b
③ Tamper with ETL file	Slingshot APT avoids leaving traces of its activity by renaming the ETW-logs	3

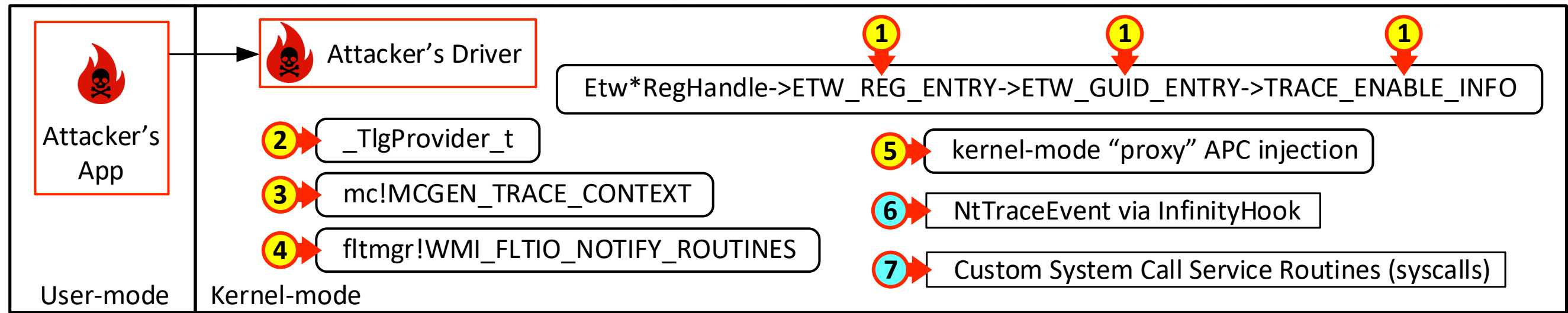
Attack Techniques		References
① ②	Disable Runtime Event Provider in .NET apps by modifying environment variables or by patching registry "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment"	1a , 1b , 1c , 1d
②	Blind services.exe: value " TracingDisabled " in Software\Microsoft\Windows NT\CurrentVersion\Tracing\SCM\Regular Blind rpcrt4.dll: value " ExtErrorInformation " in "HKLM\Software\Policies\Microsoft\Windows NT\Rpc"	2
②	Blind Microsoft-Windows-PowerShell provider: by zeroing value " EnableProperty " in HKLM\System\CurrentControlSet\Control\WMI\Autologger\EventLog-Application\{GIUD}\ Blind Autologger events: by key deletion "HKLM\SYSTEM\CurrentControlSet\Control\WMI\Autologger\AUTOLOGGER_NAME\{PROVIDER_GUID}" Blind Autologger events: by zeroing value " Enable " in "HKLM\SYSTEM\CurrentControlSet\Control\WMI\Autologger\AUTOLOGGER_NAME\{PROVIDER_GUID}"	2
②	Patch security descriptors for ETW logger sessions: "HKLM\System\CurrentControlSet\Control\Wmi\Security"	2

Attacks on user-mode ETW providers



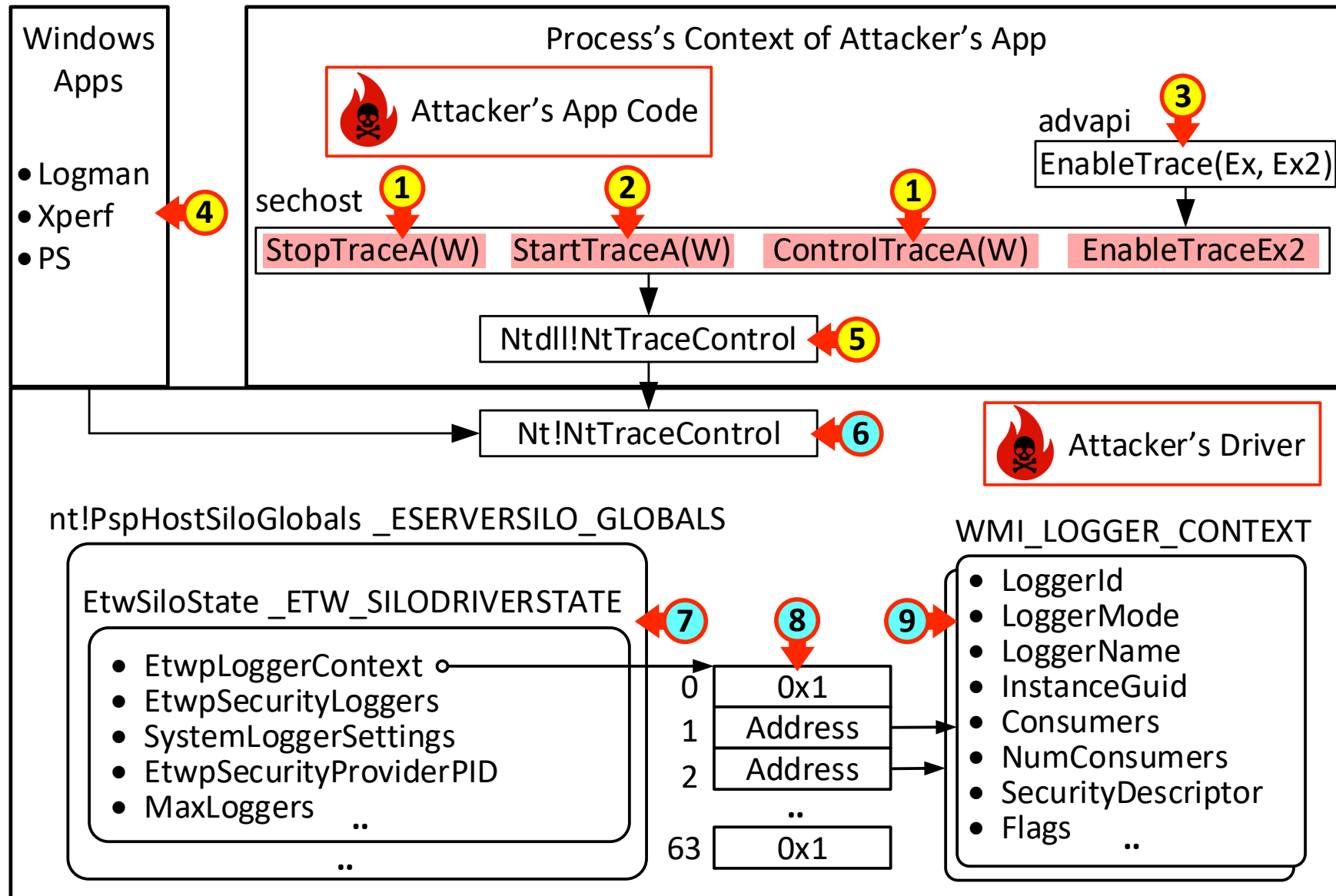
Attacks	Technique	Links
Disable ETW Provider for all processes globally	Patch Data: <ol style="list-style-type: none"> 1 ▪ ETW handles ▪ Trace level, EventsEnableBits 	1a , 1b
Disable ETW Provider for the one process	Patch Data: <ol style="list-style-type: none"> 2 ▪ ETW_USER_REG_ENTRY.Callback 3 ▪ MCGEN_TRACE_CONTEXT fields IsEnabled, level 	2 , 3
	<ol style="list-style-type: none"> 4 ▪ _TlgProvider_t fields LevelPlus1, KeywordAny, KeywordAll 	4a , 4b
	Patch Code with RET: <ol style="list-style-type: none"> 5 ▪ sechost!EtwEventCallbackList 7 ▪ ntdll!EtwEventWrite 	5a , 5b , 5c 7
	Call functions: <ol style="list-style-type: none"> 8 ▪ ntdll!EtwEventUnregister(Handle) 9 ▪ clr!McGenControlCallbackV2() 10 ▪ Remove-EtwTraceProvider 11 ▪ Terminate threads (wevtsvs) 	8a , 8b 9a , 9b 10a , 10b 11a , 11b , 11c
Block specific events in one process	Hook Function: <ol style="list-style-type: none"> 6 ▪ wevtsvc!EtwEventCallback 	6a , 6b

Attacks on kernel-mode ETW providers



No	Attacks	Technique	Links
1	Disable tracing	<ul style="list-style-type: none"> ▪ Zeroing <code>TRACE_ENABLE_INFO</code> <code>ProviderEnableInfo</code> fields <code>IsEnabled</code> and <code>Level</code> ▪ Zeroing <code>ETW_GUID_ENTRY.ProviderEnableInfo</code> (e.g. <code>EtwPsProvRegHandle</code>) 	1a , 1b
	Hijack gen. events	▪ Patching <code>ETW_REG_ENTRY->PETW_GUID_ENTRY</code> <code>GuidEntry</code>	
2	Disable tracing	<ul style="list-style-type: none"> ▪ Zeroing <code>LevelPlus1</code>, ▪ Patching <code>EnableCallback</code> ▪ Patching <code>RegHandle->ETW_REG_ENTRY.ProviderEnableInfo</code> ▪ Patching <code>ETW_REG_ENTRY->ETW_GUID_ENTRY</code> 	2a , 2b , 2c
	Hijack gen. events	Patching <code>RegHandle->ETW_REG_ENTRY</code>	
3	Disable tracing	Patching <code>IsEnabled</code> and <code>Level</code>	3a , 3b , 3c
4		Patching data structures designed for filter operations	4
5		Kernel APC injection can blind <i>Microsoft-Windows-Threat-Intelligence</i> sensor \ fake process name	5
6		InfinityHook helps to redirect the control flow. https://t.me/learningnets	6a , 6b , 7
		7 – Use custom syscalls to avoid being logged.	

Attacks on ETW sessions



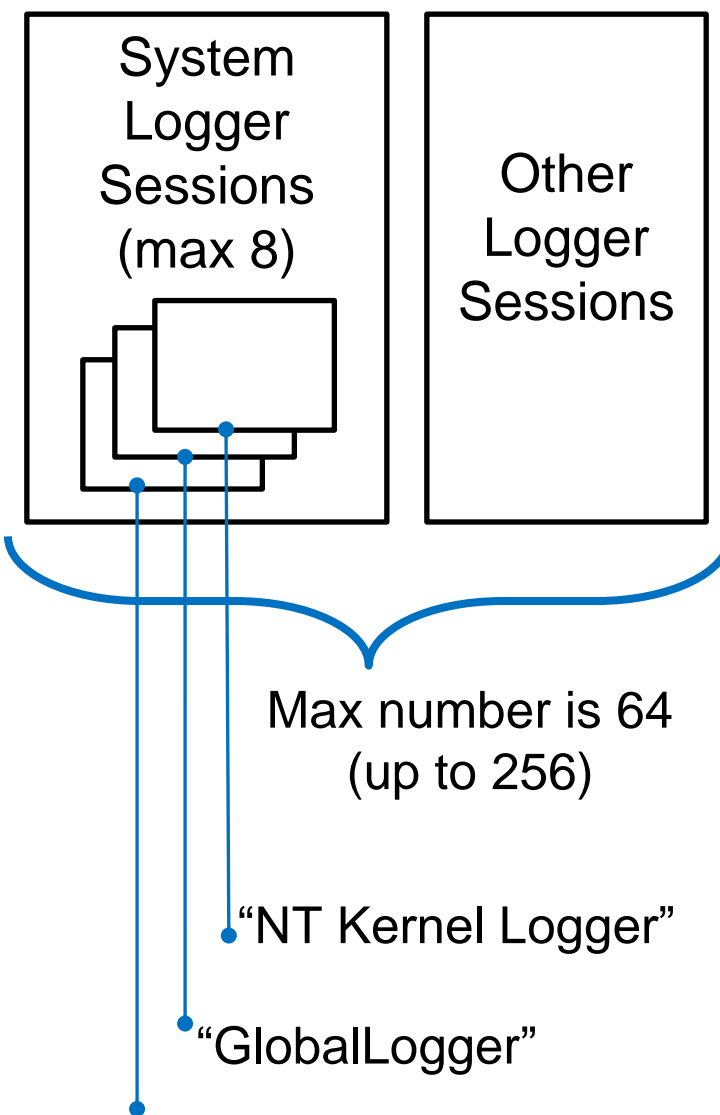
Attack	Technique	Link
Stop legal session	1 <ul style="list-style-type: none"> ▪ StopTrace ▪ ControlTrace(EVENT_TRACE_CONTROL_STOP) 	1a , 1b , 1c
	5 <ul style="list-style-type: none"> ▪ NtTraceControl 	5
	6 <ul style="list-style-type: none"> ▪ NtTraceControl 	6
	4 <ul style="list-style-type: none"> ▪ PS>Stop-EtwTraceSession ▪ logman stop LoggerName -ets ▪ xperf -stop LoggerName 	4a , 4b , 4c , 4d
Start fake session with legal name	8 <ul style="list-style-type: none"> ▪ Patch EtwpLoggerContext array 	new
	2 5 6 <ul style="list-style-type: none"> ▪ StartTrace / NtTraceControl 	2a , 2b
CVE-2020-1034	4 <ul style="list-style-type: none"> ▪ PS>Start-EtwTraceSession ▪ logman start LoggerName -ets ▪ xperf -start LoggerName -on Flag 	4e , 4f , 4g
	5 <ul style="list-style-type: none"> ▪ NtTraceControl (EtwReceiveNotification) 	5
Remove Provider	3 <ul style="list-style-type: none"> ▪ EnableTraceEx2(EVENT_CONTROL_CODE_DISABLE_PROVIDER) 	3
	4 <ul style="list-style-type: none"> ▪ PS>Remove-EtwTraceProvider 	4k
Change settings	7 <ul style="list-style-type: none"> ▪ Patch MaxLoggers ▪ Patch EtwpSecurityLoggers ▪ Patch SystemLoggerSettings 	7
Modify secure level	9 <ul style="list-style-type: none"> ▪ Patch Logger ▪ Patch Flags.SecurityTrace ▪ Patch SecurityDescriptor.Object 	new

Summary of ETW Attacks

Type of Attack	Number of different techniques
Attacks from inside AN EVIL process	6
Attacks on ETW env variables, registry, and files	3
Attacks on user-mode ETW providers	11
Attacks on kernel-mode ETW providers	7
Attacks on ETW sessions	9
All in all	36

These numbers are likely to increase, attacking ETW research is ongoing

ETW Sessions



Max number is 64
(up to 256)

• "NT Kernel Logger"

• "GlobalLogger"

• "Circular Kernel Context Logger"

<https://t.me/learningnets>

- ETW Session features:
 - ETW session is a global object with a unique name.
 - Each session can provide events for several consumer apps
 - There are no documented ways to find which app is receiving events from which session. But **EtwCheck** can do it.
- Special Logger Sessions:
 - **NT Kernel Logger Session** receives data from ETW providers implemented by ntoskrnl.exe and OS core drivers
 - Windows supports only one active Logger Session at a time.
 - Windows allows to start (stop) Logger Session for any app with admin privileges.

Process Monitor – a tool from Windows Sysinternals suite

- [Process Monitor](#) is an advanced monitoring tool that shows in real-time activity on:

- file system,

← based on file system minifilter

- Registry

← based on registry minifilter

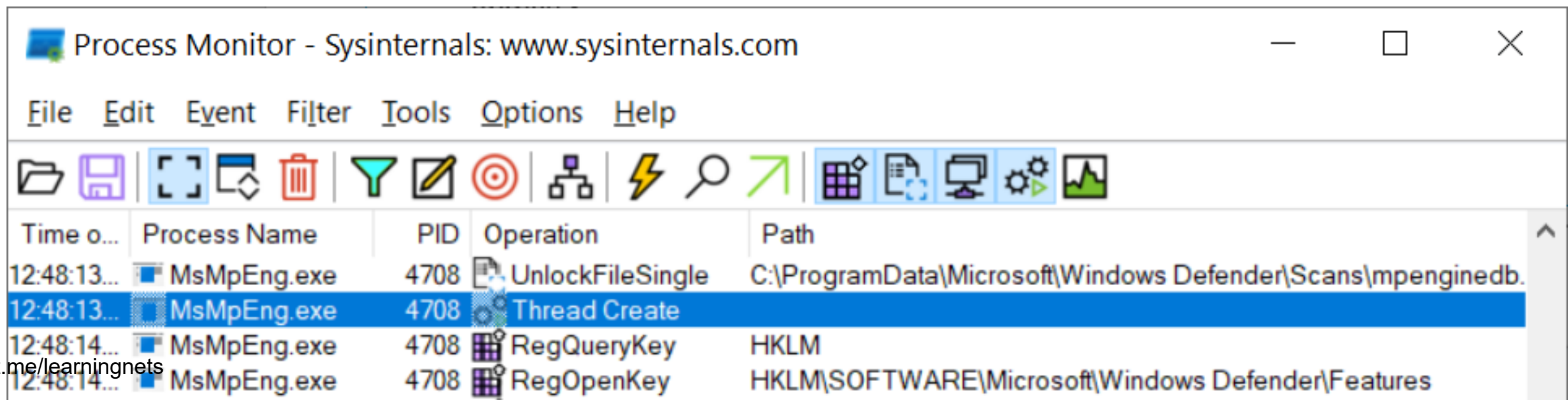
- Process/Thread

← based on process/thread callbacks

- Network

← based on ETW

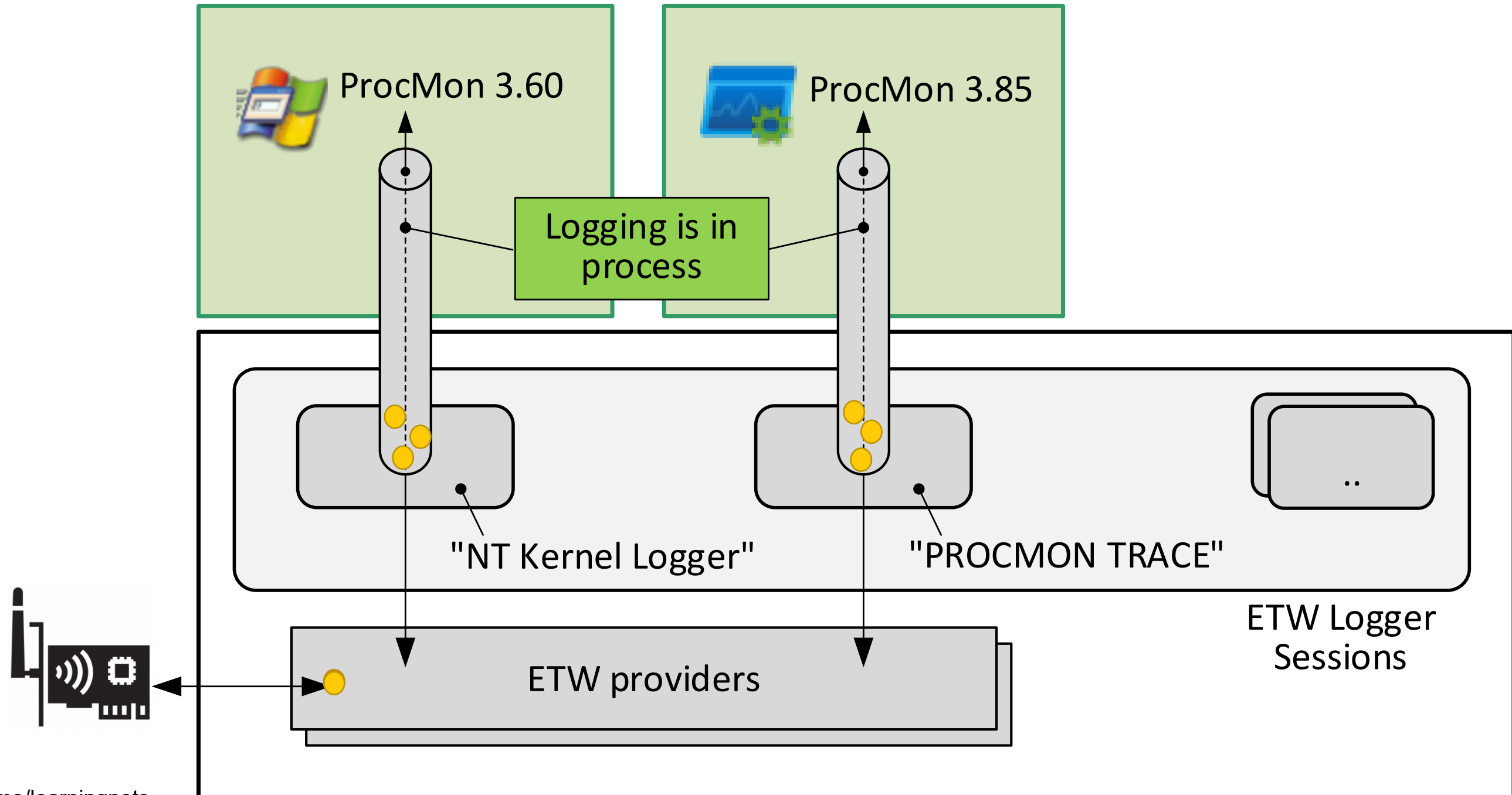
- Process Monitor uses the same technologies as many EDR solutions.



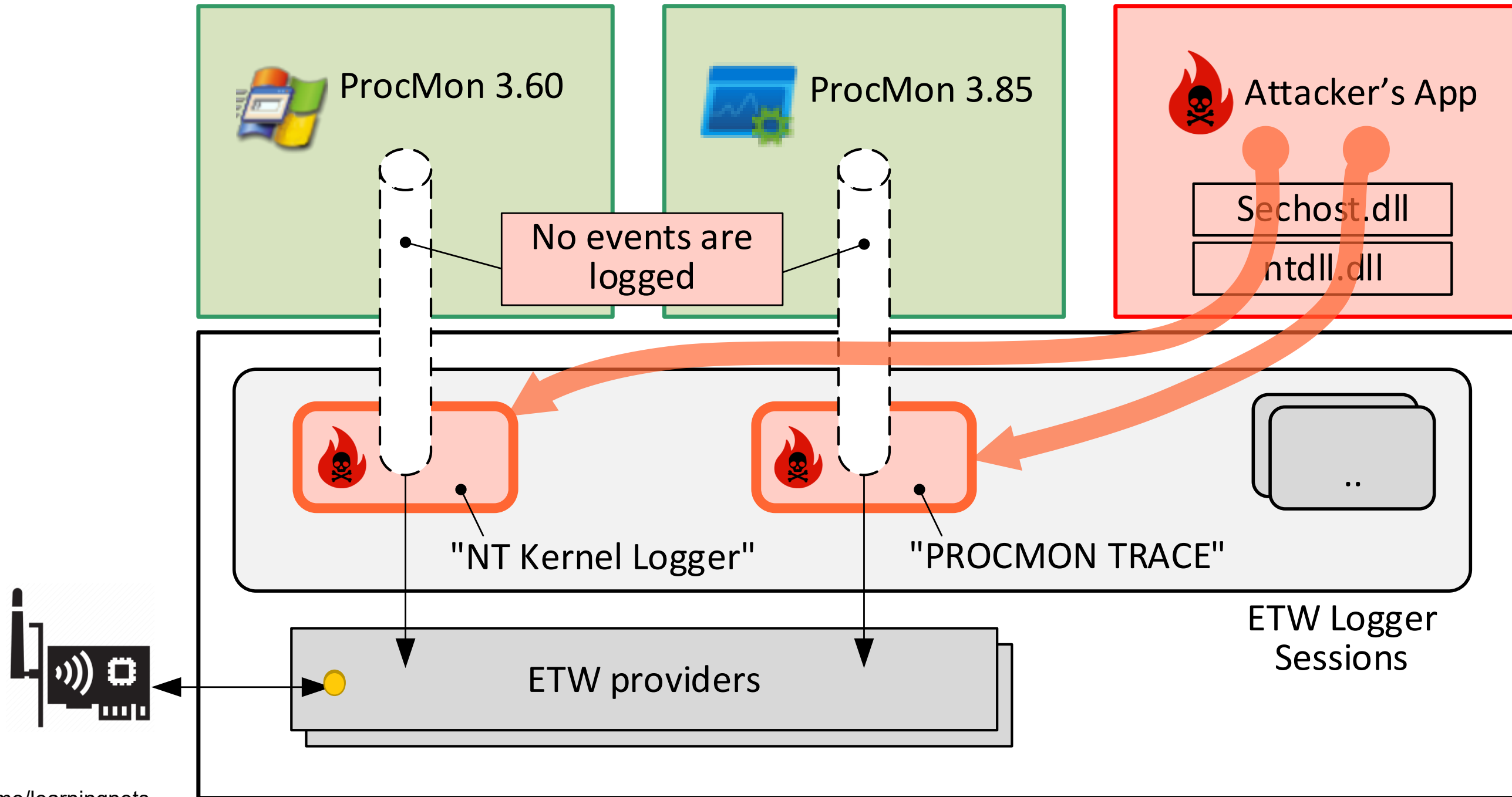
The screenshot shows the Process Monitor application window with the following menu items: File, Edit, Event, Filter, Tools, Options, Help. The toolbar contains icons for file operations, filtering, and monitoring. The main window displays a table of events:

Time o...	Process Name	PID	Operation	Path
12:48:13...	MsMpEng.exe	4708	UnlockFileSingle	C:\ProgramData\Microsoft\Windows Defender\Scans\mpengine.db.
12:48:13...	MsMpEng.exe	4708	Thread Create	
12:48:14...	MsMpEng.exe	4708	RegQueryKey	HKLM
12:48:14...	MsMpEng.exe	4708	RegOpenKey	HKLM\SOFTWARE\Microsoft\Windows Defender\Features

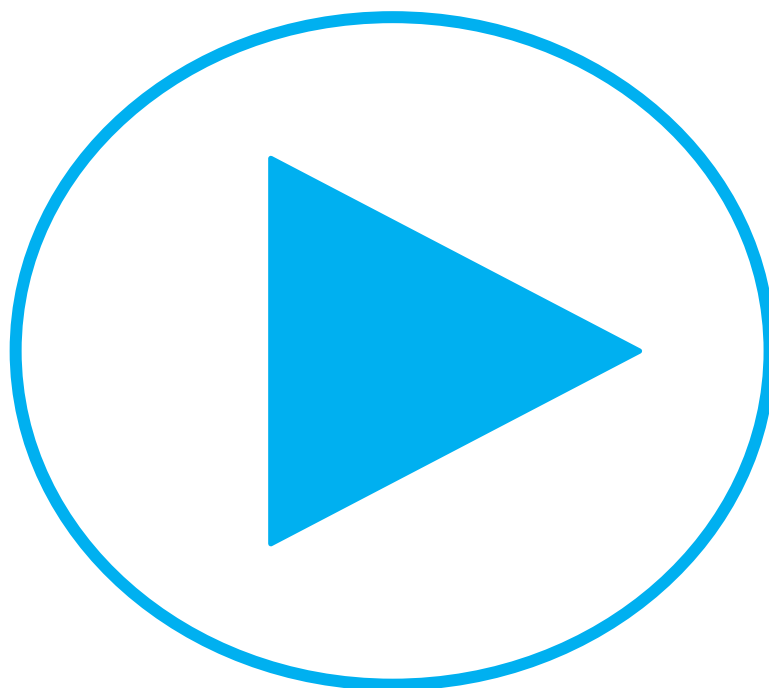
Process Monitor uses ETW to sniff network events



ETW Hijacker blinds Process Monitor

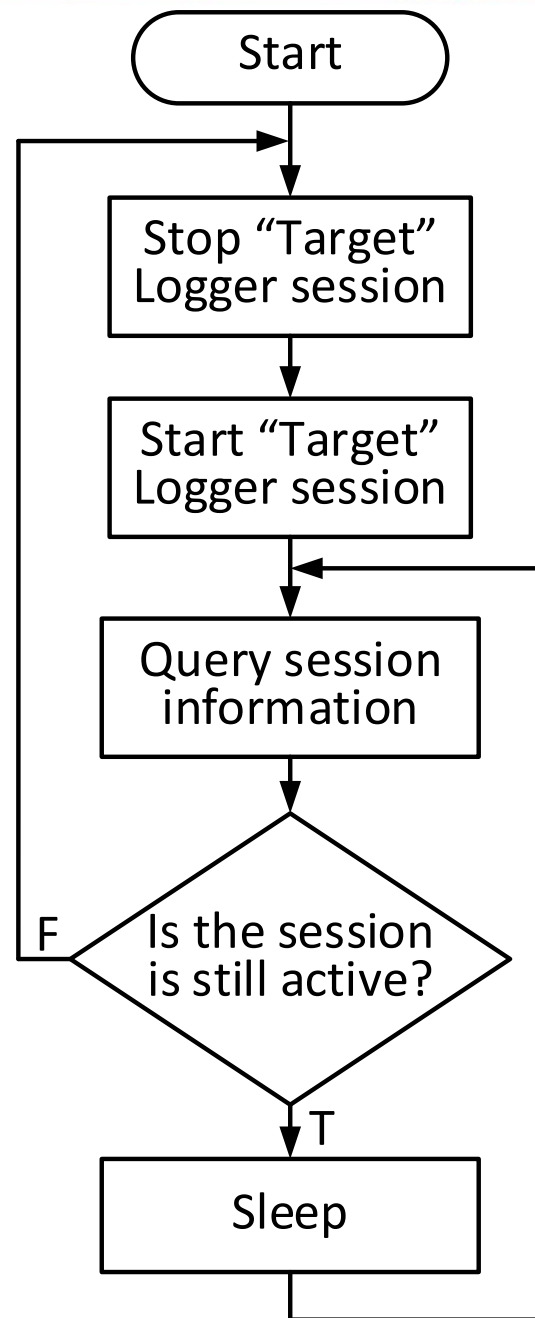


ETW Hijacker blinds Process Monitor



The online version is here – <https://www.binarly.io/posts>

ETW HIJACKER



Source code is here
<https://t.me/learningnets>
github.com/binary-io

The image shows two screenshots of a Windows Command Prompt window. The top screenshot shows the command `C:\tmp>etw_hijacker_console.exe` being executed. The bottom screenshot shows the command `C:\tmp>logman stop "NT Kernel Logger" -ets & logman start "NT Kernel Logger" -ets` being executed.

It is a super advanced tool that can extract various kernel data:

- Basic information:
 - OS system details
 - Active Processes and Parent PID, List of loaded drivers
- Internal information:
 - Kernel Processor Control Block
 - Kernel ETW handles (EtwpNetProvRegHandle etc)
 - Etwp*NotifyRoutines (EtwpDiskIoNotifyRoutines etc)
 - WMI guidentries, regentries,
 - EtwSiloState, EtwpHostSiloState
 - The whole list of WMI_LOGGER_CONTEXT with their **SecurityDescriptors**, Flags and consumers PID.
- Checks the integrity of DRV_OBJs, Major_Function[] arrays for various drivers.

Download EtwCheck from here <https://github.com/binary-io>

<https://t.me/learningsnets>

Output fragment with WMI_LOGGER_CONTEXTs

```
WMI_LOGGER_CONTEXTs at FFFF828D61D2A980 size 560 index 15
ctx[2] at FFFF828D61DD9840 GetCpuClock 0000000000000000
  LoggerMode: 2800480
  Flags: 10023
  SecurityDescriptor: FFFFCB86458F9EE0
  SD: O:BAG:BAD:(A;;0x120fff;;;SY)(A;;0x120fff;;;BA)(A;;0x100f
  LoggerStatus 0 NumConsumers 0
  LoggerName: Circular Kernel Context Logger
ctx[3] at FFFF828D61D3B040 GetCpuClock 0000000000000000
  LoggerMode: 188001C0
  Flags: F
  SecurityDescriptor: FFFFCB86459F6560
  SD: O:BAG:BAD:(A;;0x1800;;;WD)(A;;0x120fff;;;SY)(A;;0x120fff
  LoggerStatus 0 NumConsumers 1
  LoggerName: Eventlog-Security
  read 1 clients:
    [0] at FFFF828D675A6910 EPROCESS FFFF828D674DC080 PID 1300
ctx[4] at FFFF828D61D2C040 GetCpuClock 0000000000000000
  LoggerMode: 18800180
  Flags: 400F
  SecurityDescriptor: FFFFCB86459E00A0
  SD: O:BAG:BAD:(A;;0x120fff;;;SY)(A;;WP;;;SY)
  LoggerStatus 0 NumConsumers 1
  LoggerName: DefenderApiLogger
  read 1 clients:
    [0] at FFFF828D67DF7E60 EPROCESS FFFF828D67987080 PID 2428
ctx[5] at FFFF828D61D30040 GetCpuClock 0000000000000000
  LoggerMode: 188001C0
  Flags: 400F
  SecurityDescriptor: FFFFCB86459E00A0
  SD: O:BAG:BAD:(A;;0x120fff;;;SY)(A;;WP;;;SY)
  LoggerStatus 0 NumConsumers 1
  LoggerName: DefenderAuditLogger
  read 1 clients:
    [0] at FFFF828D67DF8080 EPROCESS FFFF828D67987080 PID 2428
```

Internals of a Secure ETW:

Query and **Stop** Defender's ETW Sessions

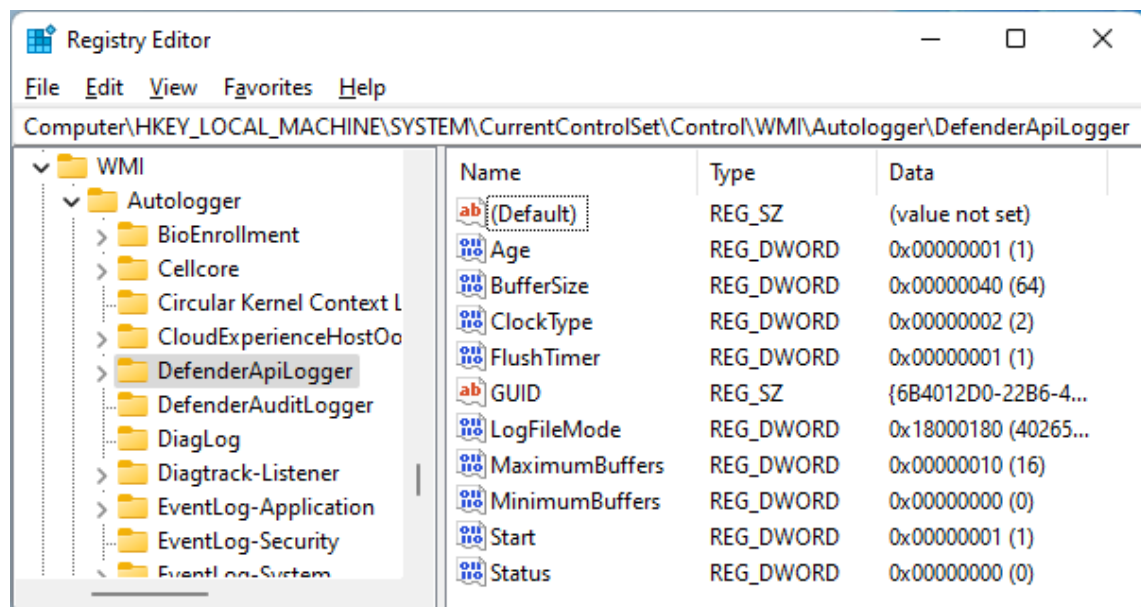
Windows Defender uses two ETW logger sessions

- List of Defenders ETW sessions:

LoggerId	LoggerName	InstanceGuid
4	DefenderApiLogger	{6B4012D0-22B6-464D-A553-20E9618403A2}
5	DefenderAuditLogger	{6B4012D0-22B6-464D-A553-20E9618403A1}

- Launched on Startup as a Autologger

- Can be disabled by [zeroing registry values](#):

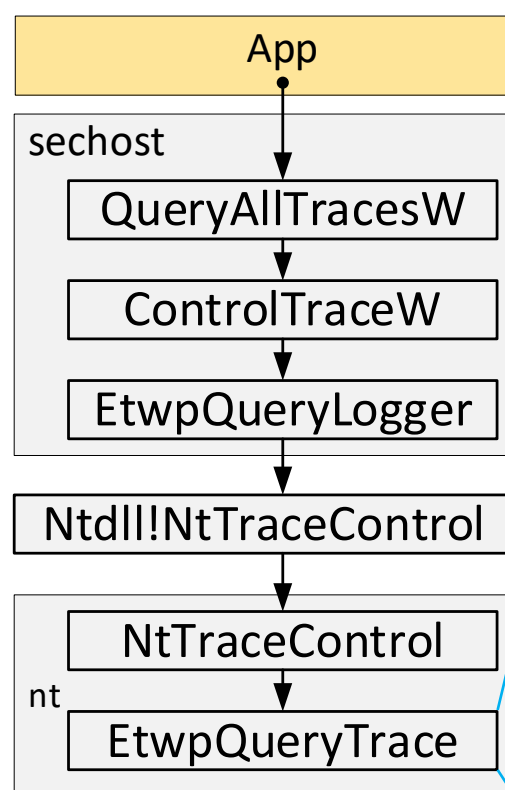


```
reg add
"HKLM\System\CurrentControlSet\Control\
WMI\Autologger\DefenderApiLogger" /v
"Start" /t REG_DWORD /d "0" /f
```


Query Secure ETW Sessions

Analysis of QueryAllTracesW() to get info about Defenders sessions

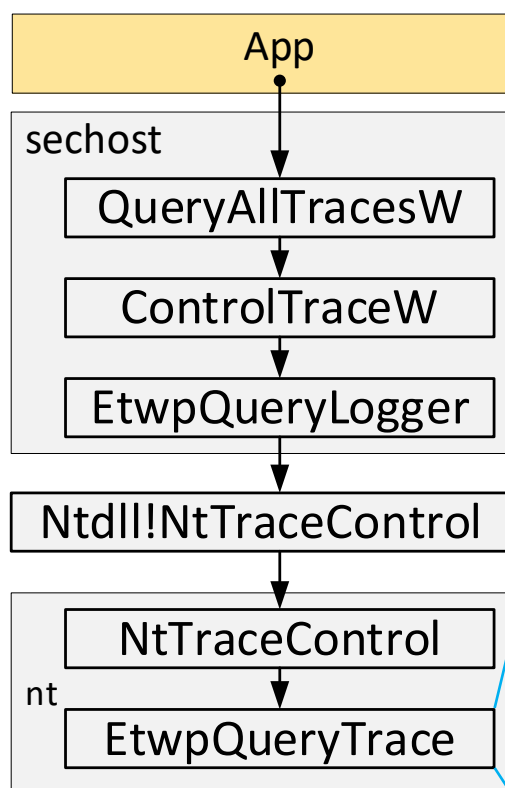
- Why apps even with admin privileges cannot get info about Defenders Sessions?



```
1 NTSTATUS EtwpQueryTrace(OUTPUT * outputInfo)
2 {
3     WMI_LOGGER_CONTEXT * pcontext = 0;
4     result = EtwpValidateLoggerInfo(..);
5     if ( result < 0 ) { goto EXIT; }
6     result = EtwpAcquireLoggerContext (&pcontext);
7     if ( result < 0 ) { goto EXIT; }
8     result = EtwpCheckLoggerControlAccess(DesiredAccess, pcontext);
9     if ( result >= 0 )
10    {
11        if (pcontext->flags.SecurityTrace)
12        {
13            result = EtwCheckSecurityLoggerAccess(Process);
14            if ( result < 0 ) { goto EXIT; }
15        }
16        result = EtwpGetLoggerInfoFromContext(outputInfo, pcontext);
17    }
18 EXIT:
19     EtwpReleaseLoggerContext(pcontext);
20     return result;
21 }
```

Analysis of QueryAllTracesW() to get info about Defenders sessions

- Why apps even with admin privileges cannot get info about Defenders Sessions?



```

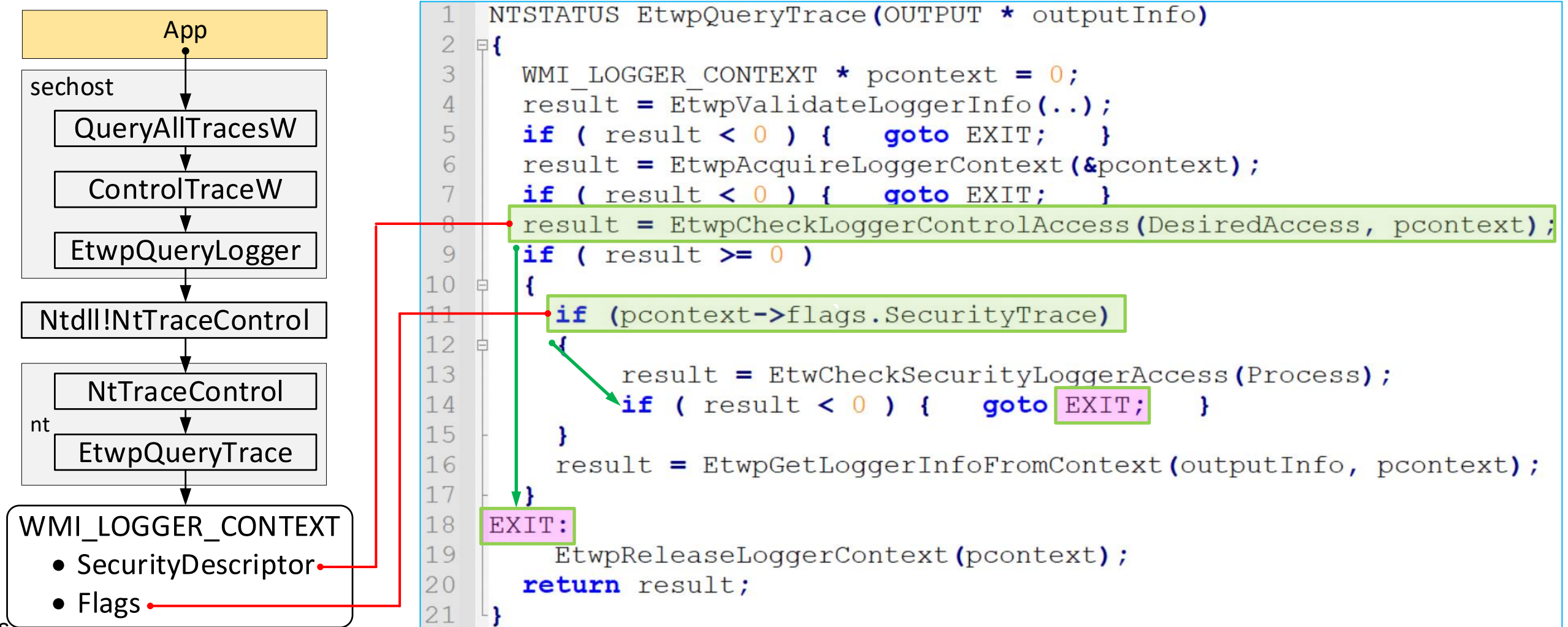
1  NTSTATUS EtwpQueryTrace(OUTPUT * outputInfo)
2  {
3      WMI_LOGGER_CONTEXT * pcontext = 0;
4      result = EtwpValidateLoggerInfo(..);
5      if ( result < 0 ) { goto EXIT; }
6      result = EtwpAcquireLoggerContext (&pcontext);
7      if ( result < 0 ) { goto EXIT; }
8      result = EtwpCheckLoggerControlAccess(DesiredAccess, pcontext);
9      if ( result >= 0 )
10     {
11         if (pcontext->flags.SecurityTrace)
12         {
13             result = EtwCheckSecurityLoggerAccess(Process);
14             if ( result < 0 )
15             {
16                 result = EtwpGetLogger...
17             }
18         }
19     }
20     EXIT:
21     EtwpReleaseLoggerContext(pcontext);
22     return result;
23 }
  
```

EtwpAccessCheck() → SeAccessCheck()

For sessions with **enabled SecurityTrace** we have an additional security check **RtlTestProtectedAccess()**

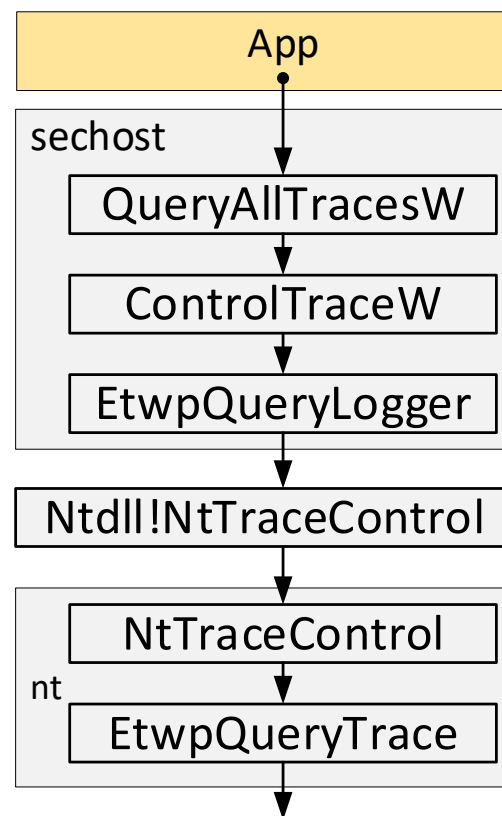
Analysis of QueryAllTracesW() to get info about Defenders sessions

- Why apps even with admin privileges cannot get info about Defenders Sessions?



Analysis of QueryAllTracesW() to get info about Defenders sessions

- Why apps even with admin privileges cannot get info about Defenders Sessions?

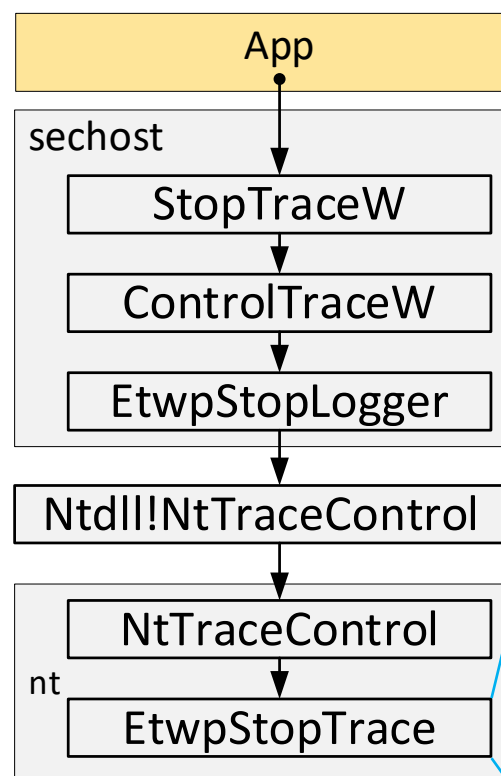


```
1 NTSTATUS EtwpQueryTrace(OUTPUT * outputInfo)
2 {
3     WMI_LOGGER_CONTEXT * pcontext = 0;
4     result = EtwpValidateLoggerInfo(..);
5     if ( result < 0 ) { goto EXIT; }
6     result = EtwpAcquireLoggerContext (&pcontext);
7     if ( result < 0 ) { goto EXIT; }
8     result = EtwpCheckLoggerControlAccess(DesiredAccess, pcontext);
9     if ( result >= 0 )
10    {
11        if (pcontext->flags.SecurityTrace)
12        {
13            result = EtwCheckSecurityLoggerAccess(Process);
14            if ( result < 0 ) { goto EXIT; }
15        }
16        result = EtwpGetLoggerInfoFromContext(outputInfo, pcontext);
17    }
18 EXIT:
19     EtwpReleaseLoggerContext(pcontext);
20     return result;
21 }
```

Stop Secure ETW Sessions

Analysis of StopTraceW() to stop Defenders sessions

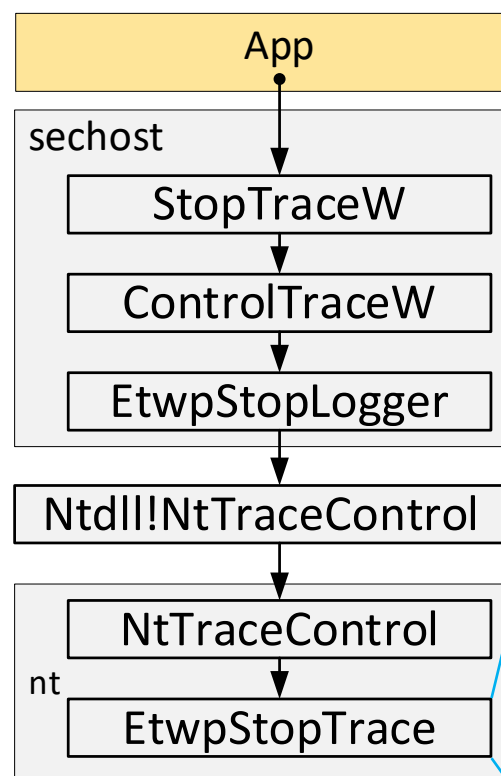
- Why apps even with admin privileges cannot stop Defenders Sessions?



```
1 NTSTATUS EtwpStopTrace(..)
2 {
3     WMI_LOGGER_CONTEXT * pcontext = 0;
4     result = EtwpValidateLoggerInfo(..);
5     if ( result < 0 ) { goto EXIT; }
6     result = EtwpAcquireLoggerContext(&pcontext);
7     if ( result < 0 ) { goto EXIT; }
8     if ((pcontext->LoggerMode & ETW_NONSTOPPABLE_MODE) != 0)
9     {
10         result = EtwpCheckLoggerControlAccess(DesiredAccess, pcontext);
11         if ( result >= 0 )
12         {
13             result = EtwpStopLoggerInstance(..);
14         }
15     }
16 EXIT:
17     EtwpReleaseLoggerContext(pcontext);
18     return result;
19 }
```

Analysis of StopTraceW() to stop Defenders sessions

- Why apps even with admin privileges cannot stop Defenders Sessions?



```

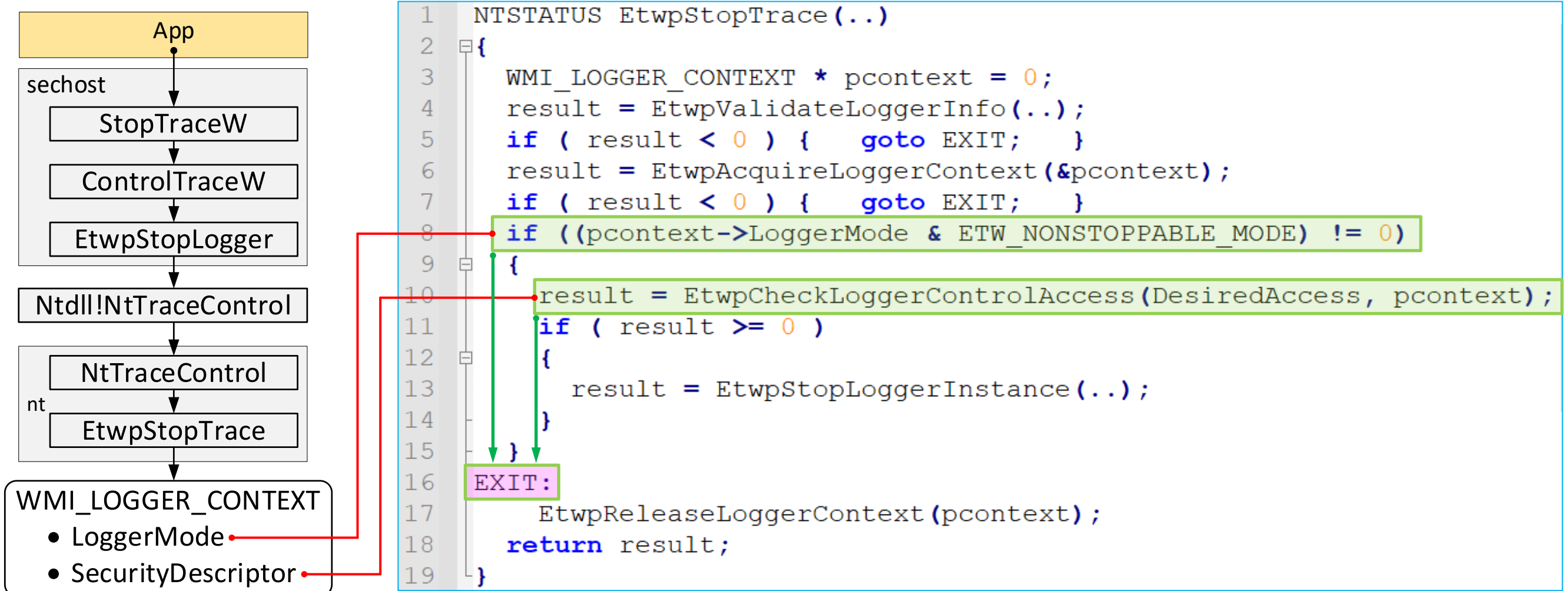
1  NTSTATUS EtwpStopTrace(..)
2  {
3      WMI_LOGGER_CONTEXT * pcontext = 0;
4      result = EtwpValidateLoggerInfo(..);
5      if ( result < 0 ) { goto EXIT; }
6      result = EtwpAcquireL
7      if ( result < 0 ) { goto
8      if ((pcontext->LoggerMode & ETW_NONSTOPPABLE_MODE) != 0)
9      {
10         result = EtwpCheckLoggerControlAccess(DesiredAccess, pcontext);
11         if ( result >= 0 )
12         {
13             result = EtwpStopLoggerInstance(..);
14         }
15     }
16 EXIT:
17     EtwpReleaseLoggerContext(pcontext);
18     return result;
19 }
  
```

Only "stoppable" sessions can be stopped

EtwpAccessCheck() → SeAccessCheck()

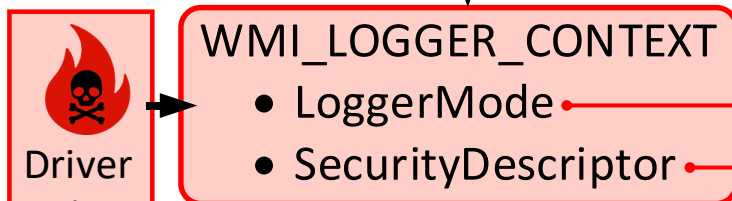
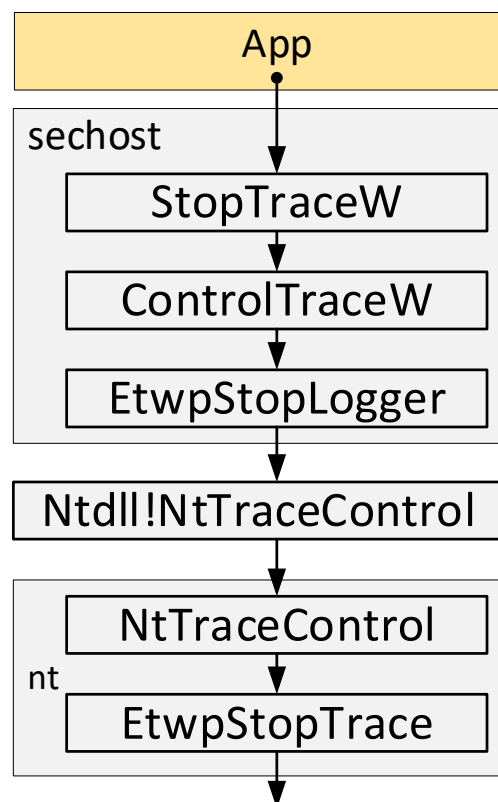
Analysis of StopTraceW() to stop Defenders sessions

- Why apps even with admin privileges cannot stop Defenders Sessions?



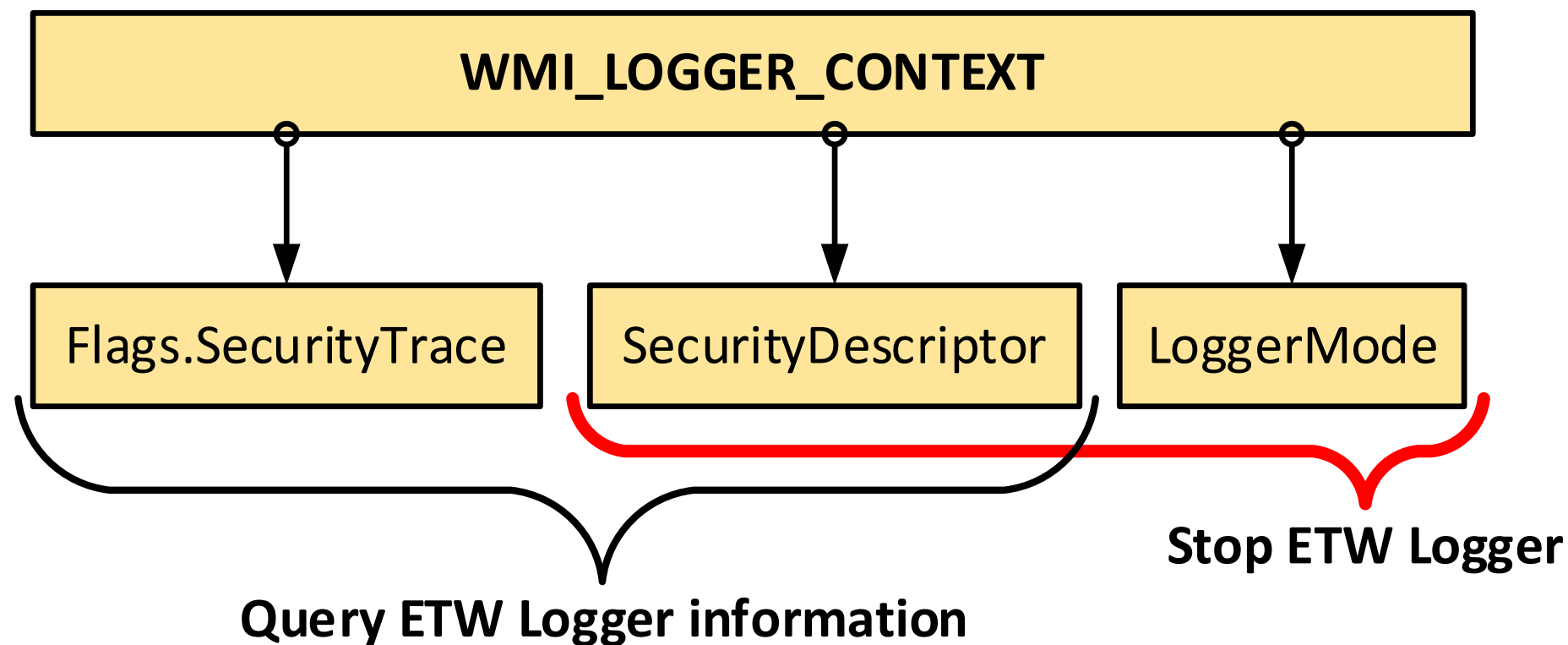
Analysis of StopTraceW() to stop Defenders sessions

- Why apps even with admin privileges cannot stop Defenders Sessions?



```
1 NTSTATUS EtwpStopTrace(..)
2 {
3     WMI_LOGGER_CONTEXT * pcontext = 0;
4     result = EtwpValidateLoggerInfo(..);
5     if ( result < 0 ) { goto EXIT; }
6     result = EtwpAcquireLoggerContext (&pcontext);
7     if ( result < 0 ) { goto EXIT; }
8     if ((pcontext->LoggerMode & ETW_NONSTOPPABLE_MODE) != 0)
9     {
10        result = EtwpCheckLoggerControlAccess(DesiredAccess, pcontext);
11        if ( result >= 0 )
12        {
13            result = EtwpStopLoggerInstance(..);
14        }
15    }
16 EXIT:
17     EtwpReleaseLoggerContext (pcontext);
18     return result;
19 }
```

Summary of the Attack



Query Info and Stop DefenderApiLogger



The online version is here – <https://www.binarly.io/posts>

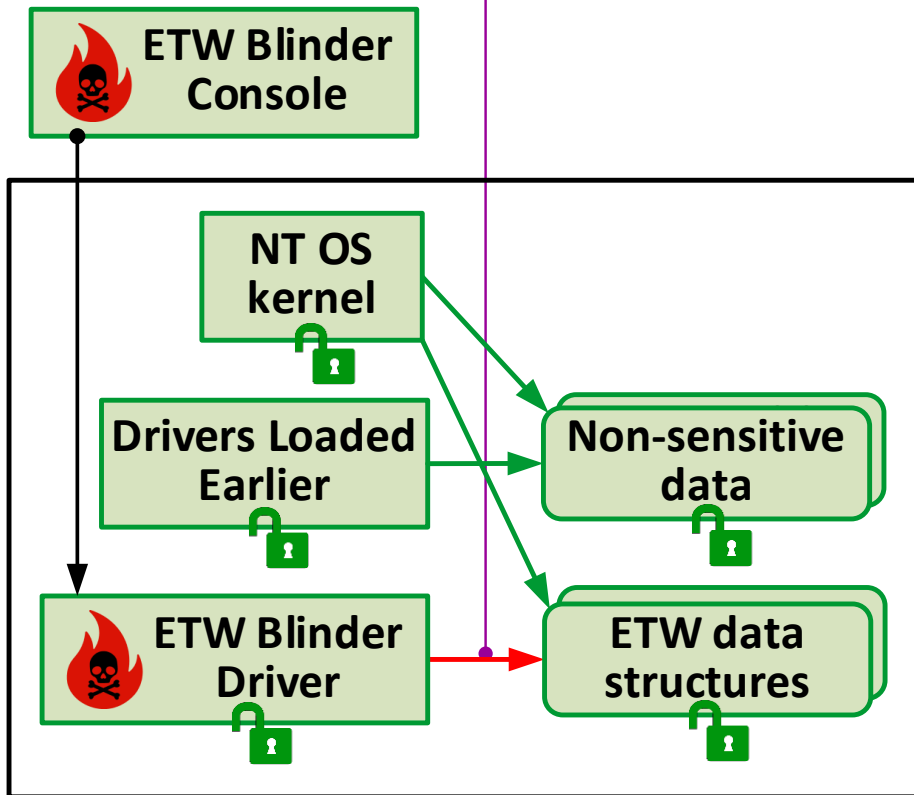
PatchGuard does not protect ETW

- Kernel Patch Protection (KPP, PatchGuard) is a Windows built-in security feature designed to provide integrity of critical kernel data:
 - EPROCESS,
 - DRV_OBJ
 - etc

- Unfortunately, ETW data structures are NOT protected

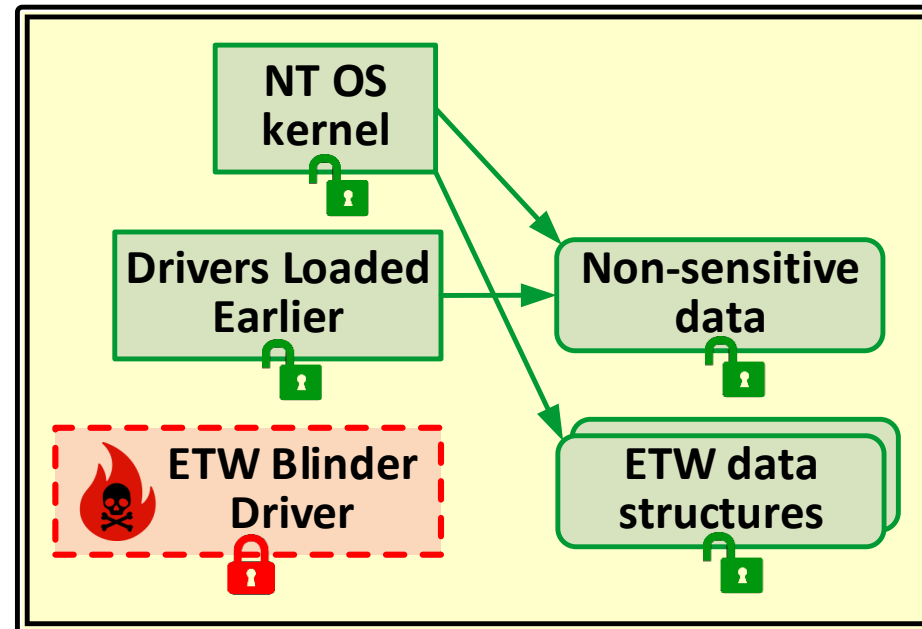
MemoryRanger can prevent patching ETW session structures

Patching ETW data

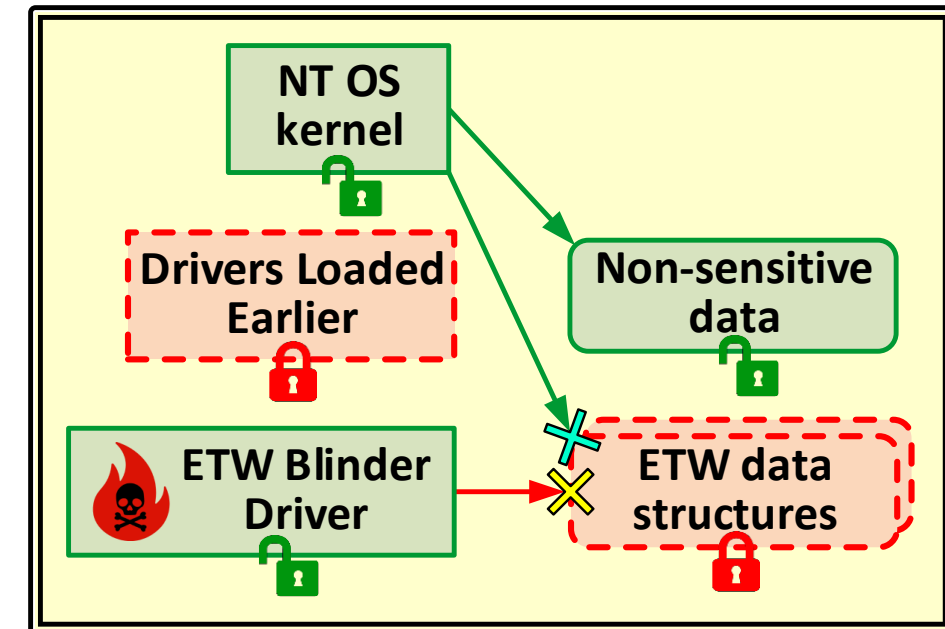


Situation without MemoryRanger

Default enclave for OS and driver loaded before



Allocated enclave for ETW Blinder Driver



MemoryRanger


- ✗ – MemoryRanger traps illegal access to the sensitive data and redirects it to the fake memory page
- ✕ – MemoryRanger switches between enclaves to allow authorized access

MemoryRanger prevents patching ETW structures



The online version is here – <https://www.binarly.io/posts>

Conclusion

- ETW was created for performance monitoring and telemetry.
- ETW is widely used by various EDRs and cybersecurity solutions.
- ETW architecture weaknesses allow bypassing ETW via various attack vectors.
- EtwCheck provides trustworthy runtime checking to detect ETW attacks.
- MemoryRanger can prevent patching ETW data in kernel memory.
- Conclusion to conclusion: attack vectors on ETW originating in the **firmware** 

Thank you

binarly.io

github.com/binarly-io