HackInBo®

Spring 2024 Edition

22ª EDIZIONE

We need to talk about ETW

Giulia Q

Giulia Q

Senior Principal Software Engineer
@ [redacted]

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

Agenda

- What is ETW
- The road so far: ETW and security
- Case study: the Security-Auditing provider
- Case study: the Threat-Intelligence provider
- The road not taken
- The road ahead

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

What is ETW

So that we'll know what it is not

Event Tracing for Windows (ETW)

Sometimes, you gotta hand it to Microsoft

Event Tracing for Windows (ETW)

Sometimes, you gotta hand it to Microsoft

- A very high-performance Windows API for **logging events** from applications, services and even drivers, **serializing** them and **consuming** them
  - Efficient enough for kernel profiling (one of its first applications)

Event Tracing for Windows (ETW)

Sometimes, you gotta hand it to Microsoft

- A very high-performance Windows API for **logging events** from applications, services and even drivers, **serializing** them and **consuming** them
  - Efficient enough for kernel profiling (one of its first applications)
- ETW events are **structured**, and have rich **metadata** (process, thread, etc. optionally including even the stack backtrace) and timestamps at the highest possible precision
  - For “free” – automatically included in all events

Event Tracing for Windows (ETW)

Sometimes, you gotta hand it to Microsoft

- A very high-performance Windows API for **logging events** from applications, services and even drivers, **serializing** them and **consuming** them
  - Efficient enough for kernel profiling (one of its first applications)
- ETW events are **structured**, and have rich **metadata** (process, thread, etc. optionally including even the stack backtrace) and timestamps at the highest possible precision
  - For “free” – automatically included in all events
- ETW event *logs* are structured: events can be **grouped**, **correlated** and their **provenance** can be traced

History of ETW

History of ETW

- Introduced in **Windows 2000**
  - Based on Windows Management Instrumentation (WMI) and its Common Information Model (CIM) data model

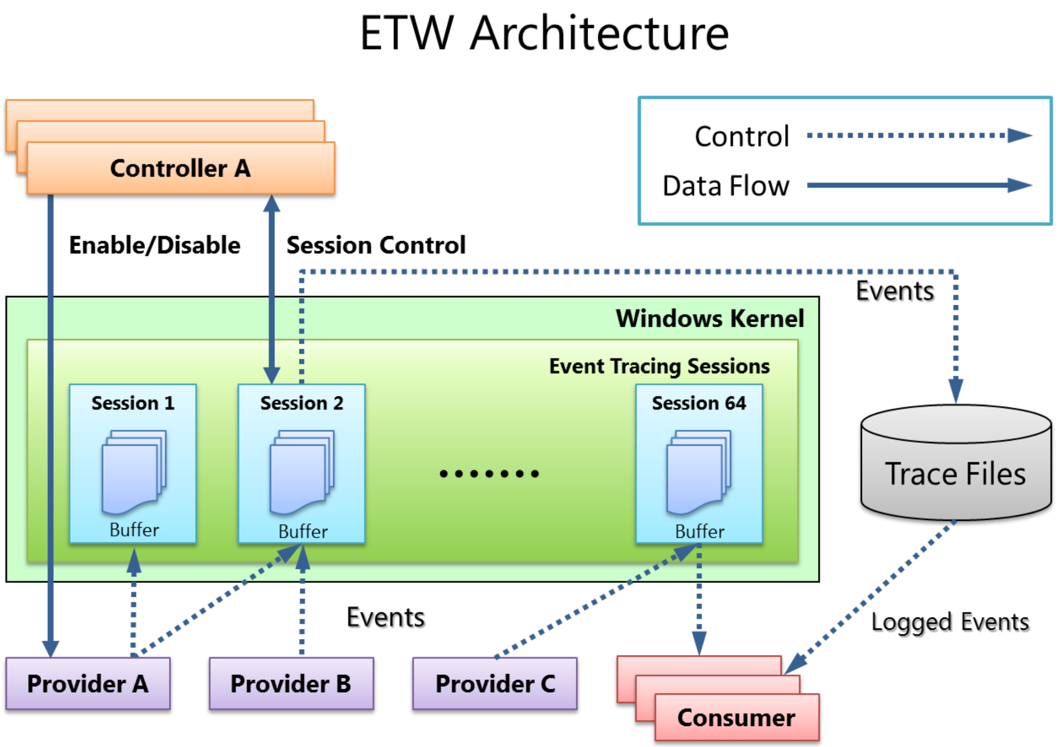
History of ETW

- Introduced in **Windows 2000**
  - Based on Windows Management Instrumentation (WMI) and its Common Information Model (CIM) data model
- Significantly revamped in **Windows Vista/Server 2008**
  - “Manifest-based” providers
    - CIM meets XML Schema
  - Windows Event Log entirely redesigned around ETW

History of ETW

- Introduced in **Windows 2000**
  - Based on Windows Management Instrumentation (WMI) and its Common Information Model (CIM) data model
- Significantly revamped in **Windows Vista/Server 2008**
  - “Manifest-based” providers
    - CIM meets XML Schema
  - Windows Event Log entirely redesigned around ETW
- Hugely improved in **Windows 10/Server 2016**
  - Free-form events (TraceLogging)
  - Extended event metadata (e.g., stack backtraces)
  - Increased buy-in – hundreds of new built-in providers

ETW architecture[1]

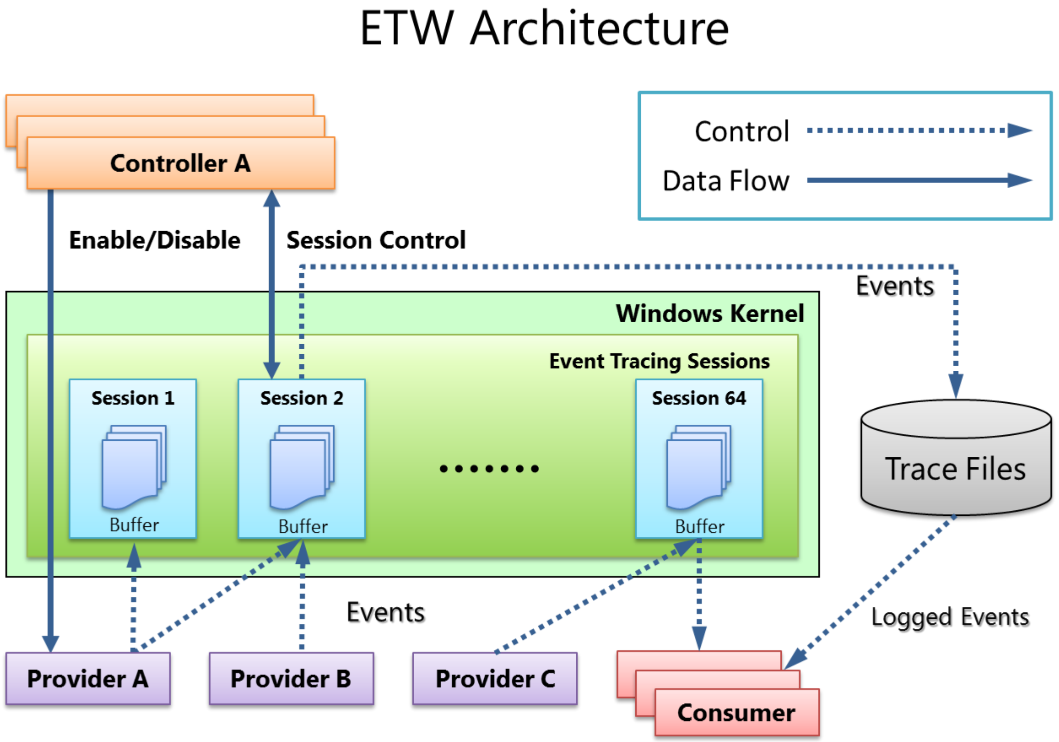


(Microsoft Corporation, 2022)

[1] Microsoft Corporation (2022); section “ETW architecture”

ETW architecture[1]

- **Providers** log events to sessions

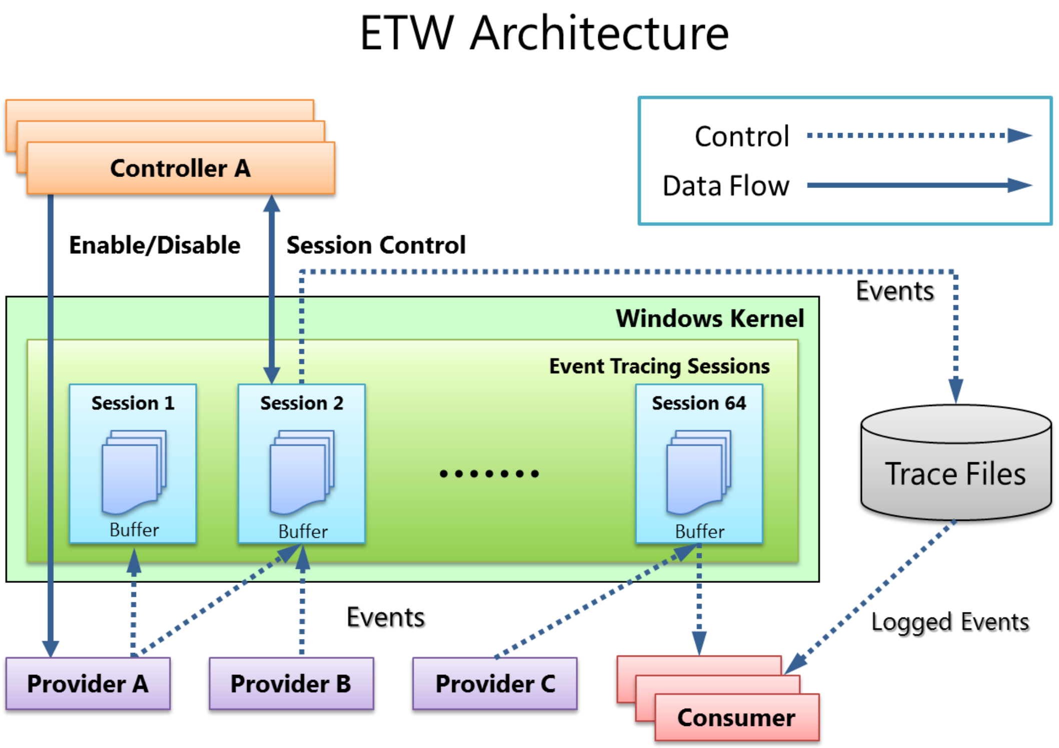


(Microsoft Corporation, 2022)

[1] Microsoft Corporation (2022); section “ETW architecture”

ETW architecture[1]

- **Providers** log events to sessions
- **Sessions** collect events
  - Circular buffer in memory
  - .etl log files

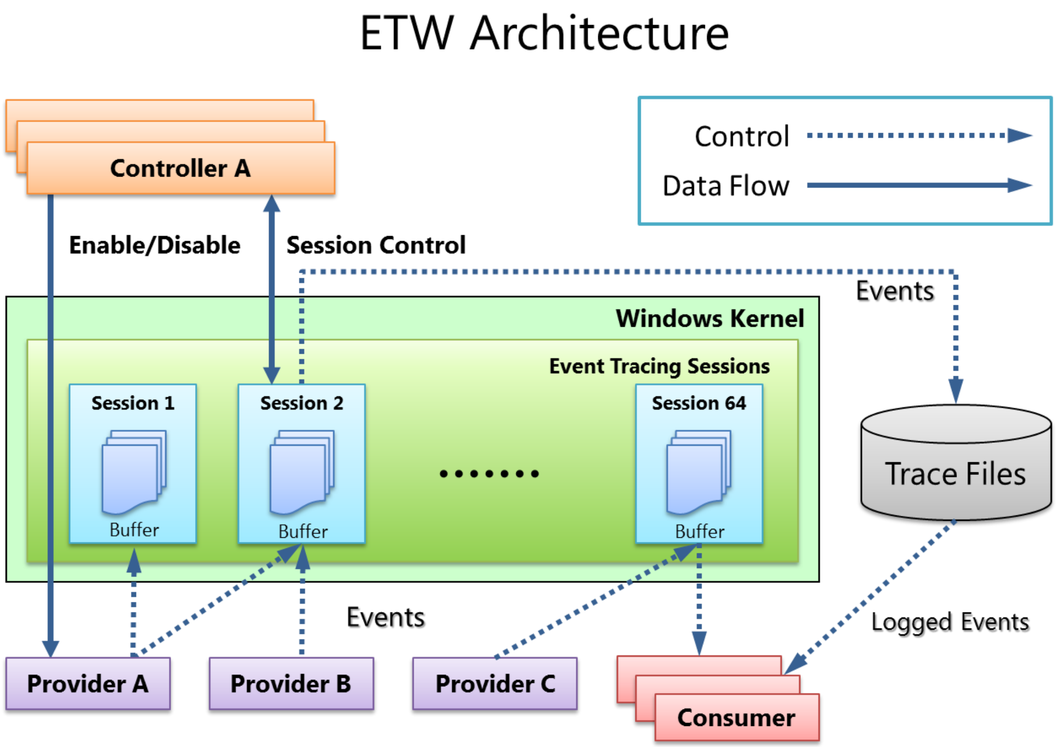


(Microsoft Corporation, 2022)

[1] Microsoft Corporation (2022); section "ETW architecture"

ETW architecture[1]

- **Providers** log events to sessions
- **Sessions** collect events
  - Circular buffer in memory
  - .etl log files
- **Controllers** manage sessions

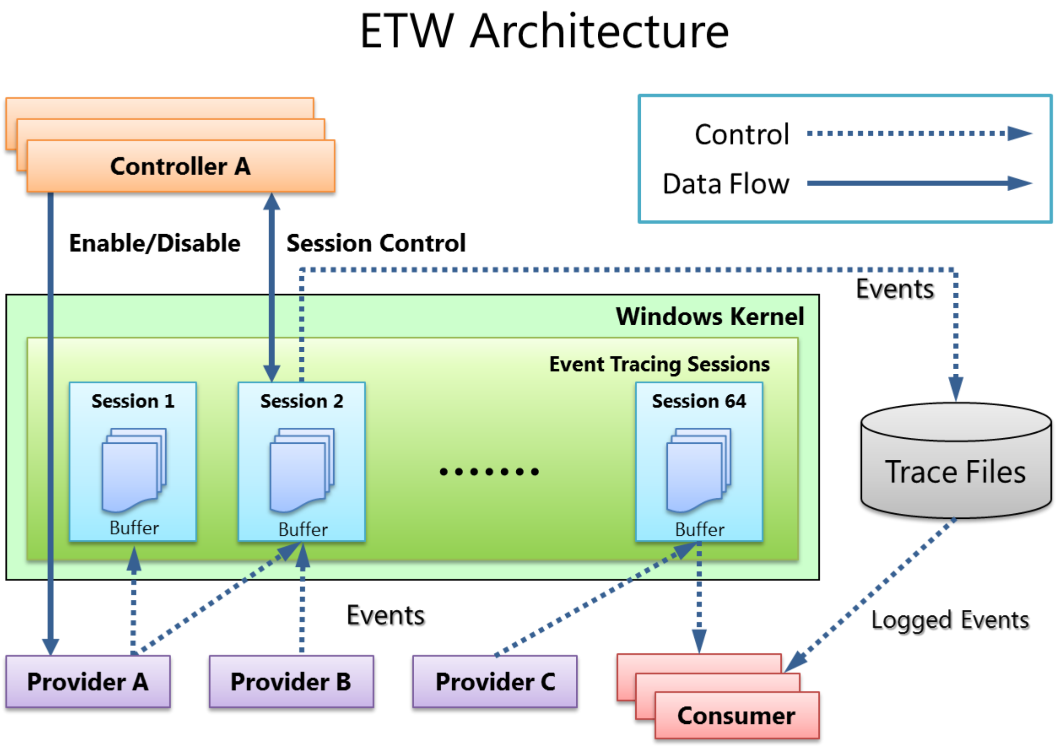


(Microsoft Corporation, 2022)

[1] Microsoft Corporation (2022); section "ETW architecture"

ETW architecture[1]

- **Providers** log events to sessions
- **Sessions** collect events
  - Circular buffer in memory
  - .etl log files
- **Controllers** manage sessions
- **Consumers** read events:
  - Real-time: from sessions
  - File: from .etl log files
  - Windows Event Log is a real-time consumer
    - Has its own log file format (.evtx) that also supports legacy events

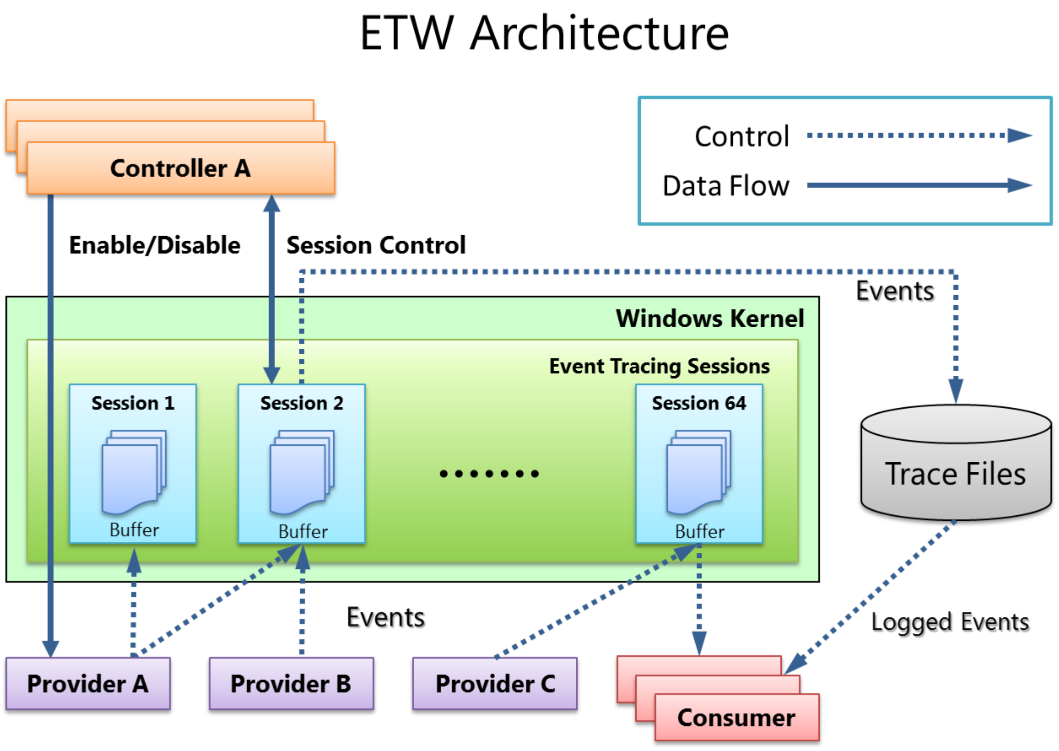


(Microsoft Corporation, 2022)

[1] Microsoft Corporation (2022); section “ETW architecture”

ETW architecture

- We will mostly concentrate on **providers** – two in particular:
  - Security-Auditing
  - Threat-Intelligence



(Microsoft Corporation, 2022)

Why ETW is relevant to me

Why ETW is relevant to me

- Invaluable for diagnosing EDR **performance issues**

Why ETW is relevant to me

- Invaluable for diagnosing EDR **performance issues**
- Irreplaceable as an EDR **data source**
  - “Official” data sources, specifically designed for security
  - Scanning for DFIR artifacts

HackInBo®

Spring 2024 Edition

22ª EDIZIONE

The road so far: ETW and security

Worst log scraping API or best log scraping API?

ETW and security

What’s all the fuss about?

ETW and security

What’s all the fuss about?

- The *de facto* data standard for EDRs and SIEMs
  - Whether they use ETW directly or indirectly (i.e. Windows Event Log)

ETW and security

What’s all the fuss about?

- The *de facto* data standard for EDRs and SIEMs
  - Whether they use ETW directly or indirectly (i.e. Windows Event Log)
- Tools and entire products based on, or enhanced by ETW
  - Microsoft Defender & Defender for Endpoint
  - All state-of-the-art EDRs
  - SysInternals ProcMon & SysMon

ETW and security

What’s all the fuss about?

- The *de facto* data standard for EDRs and SIEMs
  - Whether they use ETW directly or indirectly (i.e. Windows Event Log)
- Tools and entire products based on, or enhanced by ETW
  - Microsoft Defender & Defender for Endpoint
  - All state-of-the-art EDRs
  - SysInternals ProcMon & SysMon
- Exploit write-ups, post-mortems etc. cite specific ETW events[1, 2]

[1] Baril and Itkin (2019)
[2] Rapaport (2019)

ETW and security

ETW and security

The good parts

ETW and security

The good parts

- **Efficient**

ETW and security

The good parts

- **Efficient**
- **Reliable delivery** of events

ETW and security

The good parts

- **Efficient**
- **Reliable delivery** of events
- **Safe** and **convenient**:
  - Consumers (i.e. parsers) are 100% user mode; no installation needed[1]
  - Standard API and data format

[1], [2]. Binarly Team (2021)

ETW and security

The good parts

- **Efficient**
- **Reliable delivery** of events
- **Safe** and **convenient**:
  - Consumers (i.e. parsers) are 100% user mode; no installation needed[1]
  - Standard API and data format
- Data **breadth** and **depth**
  - Breadth: widely used – about **1000 providers**, over **50000 event types**[2]
  - Depth: rich metadata

[1], [2]. Binarly Team (2021)

ETW and security

The good parts

- **Efficient**
- **Reliable delivery** of events
- **Safe** and **convenient**:
  - Consumers (i.e. parsers) are 100% user mode; no installation needed[1]
  - Standard API and data format
- Data **breadth** and **depth**
  - Breadth: widely used – about **1000 providers**, over **50000 event types**[2]
  - Depth: rich metadata

The bad parts

[1], [2]. Binarly Team (2021)

ETW and security

The good parts

- **Efficient**
- **Reliable delivery** of events
- **Safe** and **convenient**:
  - Consumers (i.e. parsers) are 100% user mode; no installation needed[1]
  - Standard API and data format
- Data **breadth** and **depth**
  - Breadth: widely used – about **1000 providers**, over **50000 event types**[2]
  - Depth: rich metadata

The bad parts

- **The worst Windows API ever made**
- **Unreliable timestamps**
- **Unreliable metadata**
- **Unfixed bugs**
- **Asynchronous**
- **Not a security API**
- Based on an **outdated security model**
- **Provider-specific shortcomings**

[1], [2]. Binarly Team (2021)

ETW and security

The good parts

- **Efficient**
- **Reliable delivery** of events
- **Safe** and **convenient**:
  - Consumers (i.e. parsers) are 100% user mode; no installation needed[1]
  - Standard API and data format
- Data **breadth** and **depth**
  - Breadth: widely used – about **1000 providers**, over **50000 event types**[2]
  - Depth: rich metadata

The bad parts

- **The worst Windows API ever made**
- **Unreliable timestamps**
- **Unreliable metadata**
- **Unfixed bugs**
- **Asynchronous**
- **Not a security API**
- Based on an **outdated security model**
- **Provider-specific shortcomings**
- I need a bigger slide

1, 2. Binarly Team (2021)

HackInBo Spring 2024 Edition
22ª EDIZIONE

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, “gotta hand it to Microsoft”

[1] Muratori (2014)

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, “gotta hand it to Microsoft”

- Not even Microsoft employees can stand it

[1] Muratori (2014)

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, "gotta hand it to Microsoft"

- Not even Microsoft employees can stand it
  - The Office365 team has created a wrapper library, KrabsETW

[1] Muratori (2014)

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, “gotta hand it to Microsoft”

- Not even Microsoft employees can stand it
  - The Office365 team has created a wrapper library, KrabsETW
  - In their own words[2]…

[1] Muratori (2014)
[2] Microsoft Corporation (2020b)

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, “gotta hand it to Microsoft”

```
//
//                              /\
//                             ( /   @ @    ()
//                              \  __| |__  /
//                               -/   "   \-
//                              /-|       |-\
//                             / /-\     /-\ \
//                              / /-`---'-\ \
//                               /         \
//
// Summary
// ----------------------------------------------------------------------------
// Krabs is a wrapper around ETW because ETW is the worst API ever made.
```

[1] Muratori (2014)
[2] Microsoft Corporation (2020b)

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, “gotta hand it to Microsoft”

- Not even Microsoft employees can stand it
  - The Office365 team has created a wrapper library, KrabsETW
  - In their own words[2]…

[1] Muratori (2014)
[2] Microsoft Corporation (2020b)

ETW is the worst Windows API ever made[1]

You do not, under any circumstances, “gotta hand it to Microsoft”

- Not even Microsoft employees can stand it
  - The Office365 team has created a wrapper library, KrabsETW
  - In their own words[2]…
- Poorly documented
  - The KrabsETW code cites the work of reverse engineer Geoff Chappell[3]

1. Muratori (2014)
2. Microsoft Corporation (2020b)
3. Microsoft Corporation (2021a)

ETW is neglected

Known issues, aged like fine wine

ETW is neglected

Known issues, aged like fine wine

- **Unreliable timestamps**
  - Occasionally missing
  - Occasionally out of order (“time inversion”)
  - Design flaw: must choose clock between system time and performance counter – can’t have both at the same time[1]

[1] Microsoft Corporation (2021c); table with the possible values for ClientContext

ETW is neglected

Known issues, aged like fine wine

- **Unreliable timestamps**
  - Occasionally missing
  - Occasionally out of order (“time inversion”)
  - Design flaw: must choose clock between system time and performance counter – can’t have both at the same time[1]
- **Unreliable metadata**
  - Stack backtraces are frequently missing

[1] Microsoft Corporation (2021c); table with the possible values for ClientContext

ETW is neglected

Known issues, aged like fine wine

- **Unreliable timestamps**
  - Occasionally missing
  - Occasionally out of order (“time inversion”)
  - Design flaw: must choose clock between system time and performance counter – can’t have both at the same time[1]
- **Unreliable metadata**
  - Stack backtraces are frequently missing
- **Unfixed bugs**[2]

[1] Microsoft Corporation (2021c); table with the possible values for ClientContext
[2] Microsoft Corporation (2023); comment in trace_manager<T>::process_trace()

ETW is neglected

Known issues, aged like fine wine

- **Unreliable timestamps**
  - Occasionally missing
  - Occasionally out of order (“time inversion”)
  - Design flaw: must choose clock between system time and performance counter – can’t have both at the same time[1]
- **Unreliable metadata**
  - Stack backtraces are frequently missing
- **Unfixed bugs**[2]
- **User mode/kernel mode discrepancies**
  - Unsynchronized timestamps
  - Default Activity Id not used by kernel mode providers[3]

[1] Microsoft Corporation (2021c); table with the possible values for ClientContext
[2] Microsoft Corporation (2023); comment in trace_manager<T>::process_trace()
[3] Uhlmann (2023)

ETW is asynchronous

Any later than “right now” may be too late

ETW is asynchronous

Any later than “right now” may be too late

- E**DR**: asynchronous is good for **d**etection, but may be too late for **r**esponse

ETW is asynchronous

Any later than “right now” may be too late

- E**DR**: asynchronous is good for **d**etection, but may be too late for **r**esponse
- Data and metadata are *never* rich enough

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

ETW is asynchronous

Any later than “right now” may be too late

- E**D**R: asynchronous is good for **d**etection, but may be too late for **r**esponse
- Data and metadata are *never* rich enough
  - ETW logs the process/thread and user id **that logged the event**
    - What process/thread *requested* the operation?
    - What about the *channel* on which the operation was requested? (RPC interface, COM class, etc.)
    - What’s the *code identity* of the requestor? (executable signature, hashes, etc.)
    - What about remote requestors?
    - etc.

ETW is asynchronous

Any later than “right now” may be too late

- E**DR**: asynchronous is good for **d**etection, but may be too late for **r**esponse
- Data and metadata are *never* rich enough
  - ETW logs the process/thread and user id **that logged the event**
    - What process/thread *requested* the operation?
    - What about the *channel* on which the operation was requested? (RPC interface, COM class, etc.)
    - What’s the *code identity* of the requestor? (executable signature, hashes, etc.)
    - What about remote requestors?
    - etc.
  - No support for request tracking outside of activity correlation
    - … which doesn’t work in practice

ETW is asynchronous

Any later than “right now” may be too late

- E**D**R: asynchronous is good for **d**etection, but may be too late for **r**esponse
- Data and metadata are *never* rich enough
  - ETW logs the process/thread and user id **that logged the event**
    - What process/thread *requested* the operation?
    - What about the *channel* on which the operation was requested? (RPC interface, COM class, etc.)
    - What’s the *code identity* of the requestor? (executable signature, hashes, etc.)
    - What about remote requestors?
    - etc.
  - No support for request tracking outside of activity correlation
    - … which doesn’t work in practice
  - If the information isn’t logged by the provider or ETW itself, it may be lost forever
    - Asynchronous logging prevents EDRs from adding their own metadata to the event

ETW is not a security API

ETW is not a security API

- ETW is **controllable**, **configurable** and **extensible**. Desirable qualities… but not for a security feature

ETW is not a security API

- ETW is **controllable**, **configurable** and **extensible**. Desirable qualities… but not for a security feature
- ETW access controls are insufficient for security applications
  - Ad-hoc access controls for certain special, security-oriented providers
  - No help for other providers

ETW is not a security API

- ETW is **controllable**, **configurable** and **extensible**. Desirable qualities… but not for a security feature
- ETW access controls are insufficient for security applications
  - Ad-hoc access controls for certain special, security-oriented providers
  - No help for other providers
- ETW **tampering is trivial**[1]:

[1] Teodorescu et al. (2021)

ETW is not a security API

- ETW is **controllable**, **configurable** and **extensible**. Desirable qualities… but not for a security feature
- ETW access controls are insufficient for security applications
  - Ad-hoc access controls for certain special, security-oriented providers
  - No help for other providers
- ETW **tampering is trivial**[1]:
  - Abusing ETW's controllability, configurability and extensibility

[1] Teodorescu et al. (2021)

ETW is not a security API

- ETW is **controllable**, **configurable** and **extensible**. Desirable qualities… but not for a security feature
- ETW access controls are insufficient for security applications
  - Ad-hoc access controls for certain special, security-oriented providers
  - No help for other providers
- ETW **tampering is trivial**[1]:
  - Abusing ETW's controllability, configurability and extensibility
  - Suppressing events
    - Patching logging code or tampering with logger state
    - Provider-specific configuration

[1] Teodorescu et al. (2021)

ETW is not a security API

- ETW is **controllable**, **configurable** and **extensible**. Desirable qualities… but not for a security feature
- ETW access controls are insufficient for security applications
  - Ad-hoc access controls for certain special, security-oriented providers
  - No help for other providers
- ETW **tampering is trivial**[1]:
  - Abusing ETW’s controllability, configurability and extensibility
  - Suppressing events
    - Patching logging code or tampering with logger state
    - Provider-specific configuration
  - Some providers log untrusted data
    - Including widely used providers like Microsoft-Windows-WMI-Activity[2]

[1] Teodorescu et al. (2021)
[2] Uhlmann (2023)

ETW is a living fossil

The perfect security API – for the 90s

ETW is a living fossil

The perfect security API – for the 90s

- Designed back when “**secure OS**” meant “**multi-user OS**”

ETW is a living fossil

The perfect security API – for the 90s

- Designed back when “**secure OS**” meant “**multi-user OS**”
- Events are attributed to **users**, not **code**

ETW is a living fossil

The perfect security API – for the 90s

- Designed back when “**secure OS**” meant “**multi-user OS**”
- Events are attributed to **users**, not **code**
- These are **architectural issues**: ETW can do little about it
  - Can’t log information that the OS does not provide
  - On Windows, **code is mutable**
    - Ironically, code mutation is the classical way to implement an EDR

ETW is log scraping

The best log scraping is still log scraping

- Events are only as good as the provider logging them

ETW is log scraping

The best log scraping is still log scraping

- Events are only as good as the provider logging them
- ***Are* the providers good?**

ETW is log scraping

The best log scraping is still log scraping

- Events are only as good as the provider logging them
- ***Are* the providers good?**
  - Good question

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

Case study: the Security-Auditing provider

Good enough for government work

What it is

And what it does

What it is

And what it does

- An **ETW provider** – full name **Microsoft-Windows-Security-Auditing**
- A special provider with **ad-hoc access controls**[1] and **anti-tampering features**[2, 3]

[1] Microsoft Corporation (2020a)
[2] Chappell (2008a)
[3] Chappell (2008b)

What it is

And what it does

- An **ETW provider** – full name **Microsoft-Windows-Security-Auditing**
- A special provider with **ad-hoc access controls**[1] and **anti-tampering features**[2, 3]
- Logs a hodge-podge of security-related events of all kinds[4]
  - Access control events
  - Authentication events
  - Active Directory events
  - etc.

[1] Microsoft Corporation (2020a)
[2] Chappell (2008a)
[3] Chappell (2008b)
[4] Microsoft Corporation (2021d)

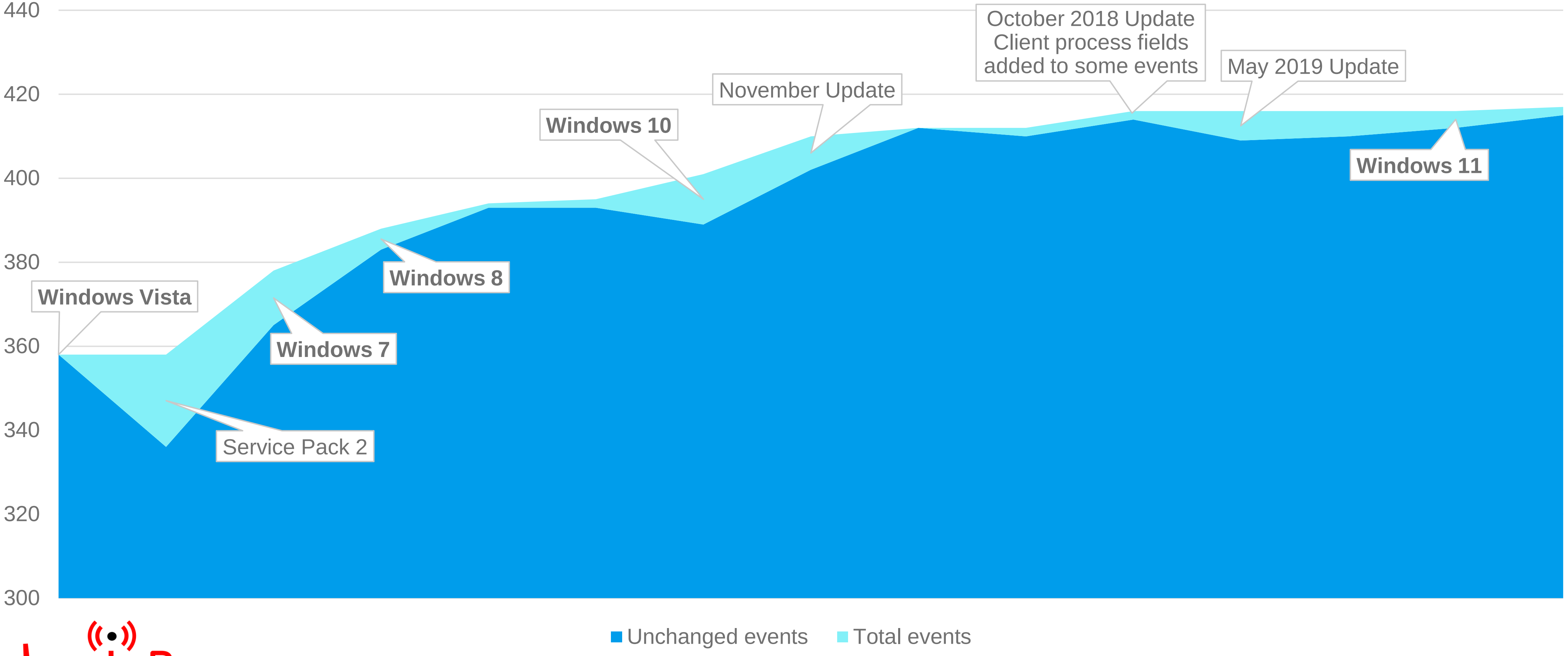
What it is

And what it does

- An **ETW provider** – full name **Microsoft-Windows-Security-Auditing**
- A special provider with **ad-hoc access controls**[1] and **anti-tampering features**[2, 3]
- Logs a hodge-podge of security-related events of all kinds[4]
  - Access control events
  - Authentication events
  - Active Directory events
  - etc.
- Abused as a tamper-proof source of events that aren’t strictly security-related
  - Firewall[5, 6]
  - Device management[7]
  - Task scheduler[8]

1. Microsoft Corporation (2020a)
2. Chappell (2008a)
3. Chappell (2008b)
4. Microsoft Corporation (2021d)
5. Microsoft Corporation (2021e)
6. Microsoft Corporation (2021f)
7. Microsoft Corporation (2021h)
8. Microsoft Corporation (2021g)

Evolution of Security-Auditing



Design of Security-Auditing

Old school security

- Primarily designed to pass compliance

Design of Security-Auditing

Old school security

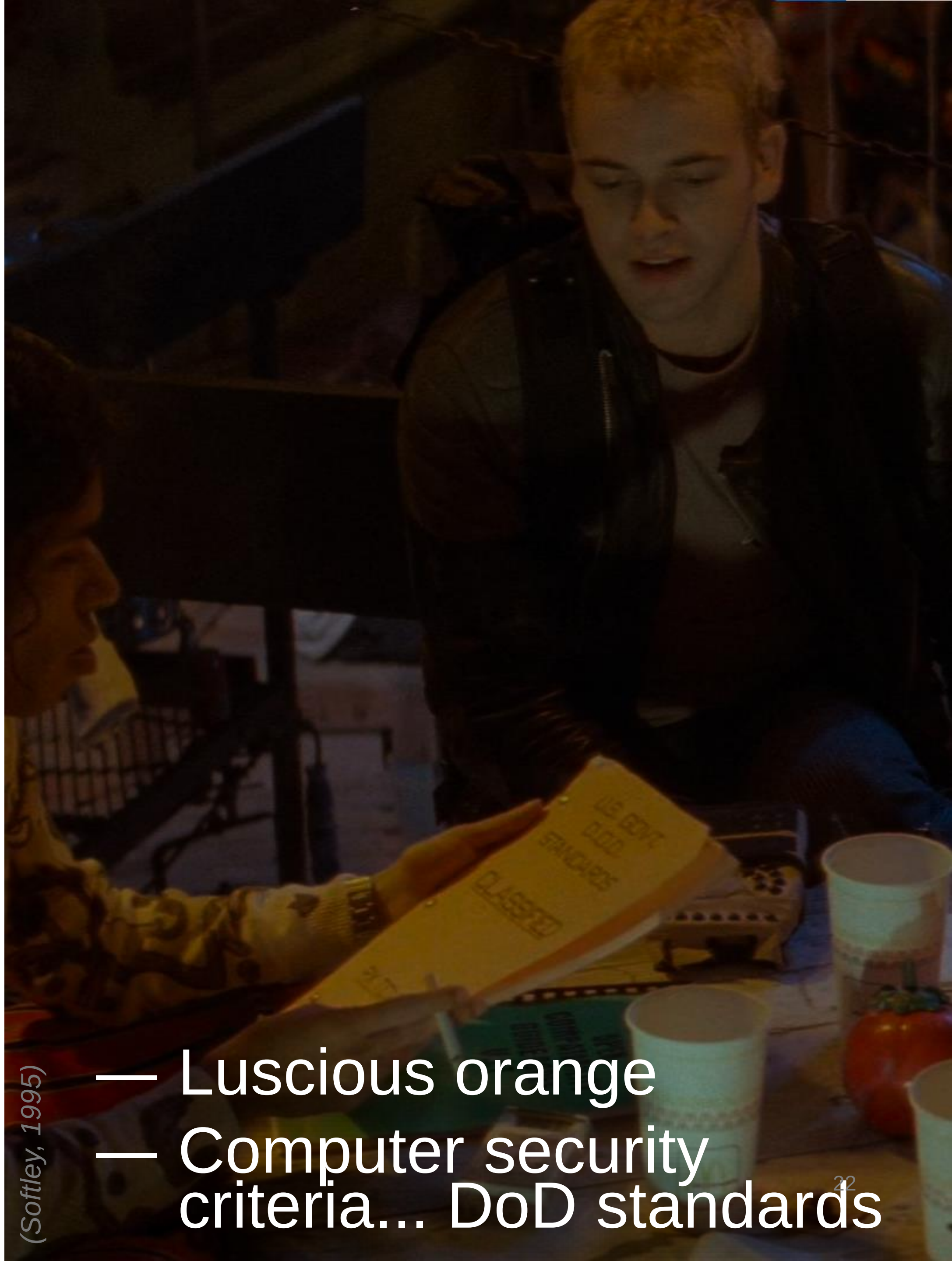
- Primarily designed to pass compliance
  - Originally, **protection class C2** of the DoD **Trusted Computer System Evaluation Criteria (TCSEC)** [1, 2] – best known as the **Orange Book**

[1] Department of Defense (1985); pp. 10, 17–18
[2] Microsoft Corporation (2001)

Design of Security-Auditing

Old school security

- Primarily designed to pass compliance
  - Originally, **protection class C2** of the DoD **Trusted Computer System Evaluation Criteria (TCSEC)** [1, 2] – best known as the **Orange Book**



(Softley, 1995)

— Luscious orange

— Computer security criteria... DoD standards

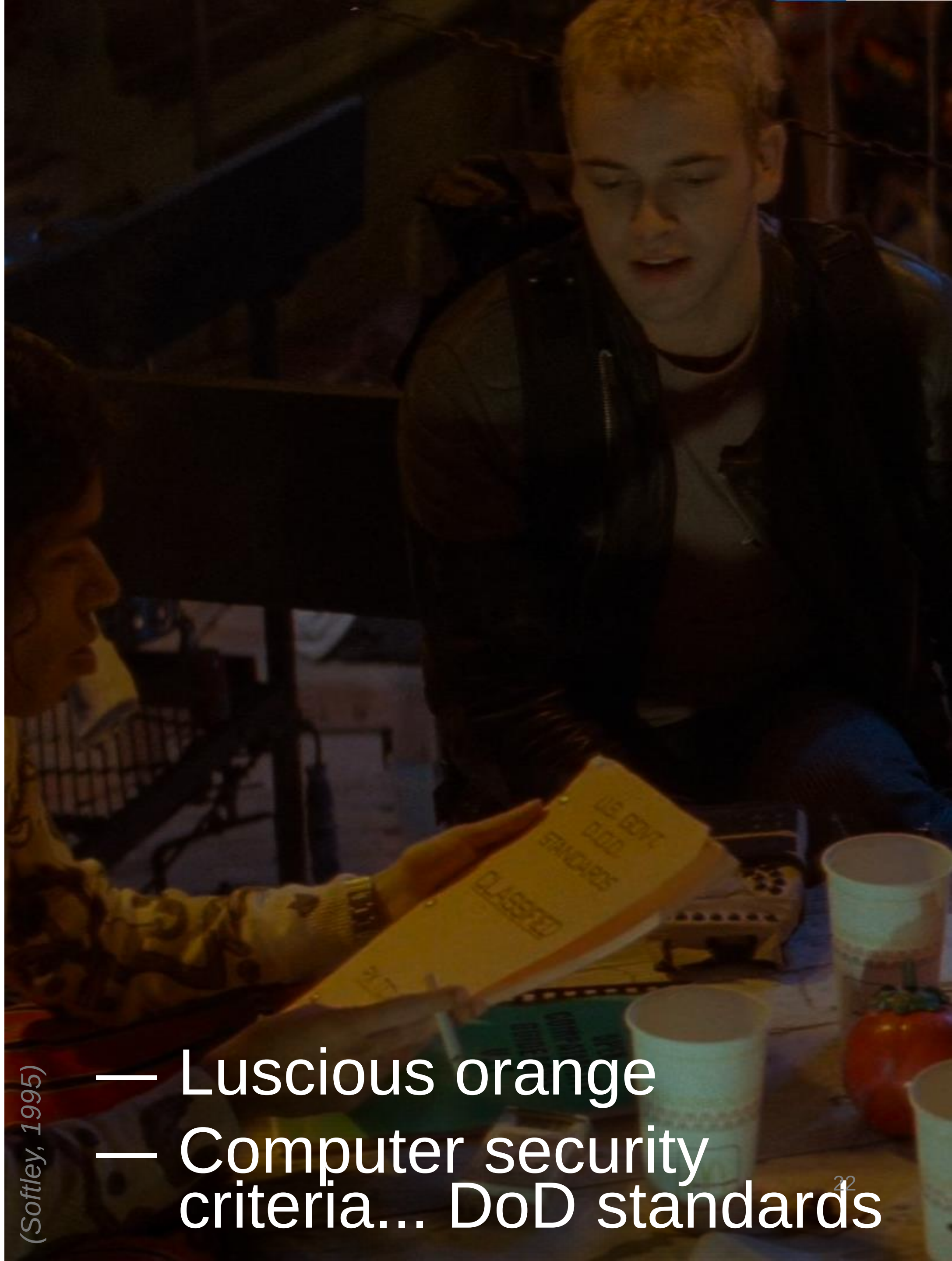
[1] Department of Defense (1985); pp. 10, 17–18

[2] Microsoft Corporation (2001)

Design of Security-Auditing

Old school security

- Primarily designed to pass compliance
  - Originally, **protection class C2** of the DoD **Trusted Computer System Evaluation Criteria (TCSEC)** [1, 2] – best known as the **Orange Book**
- Not otherwise "designed"
  - Fragmented – clearly the work of several uncoordinated teams
  - Inconsistent data schema and data quality
  - Dubious threat modeling



(Softley, 1995)

[1] Department of Defense (1985); pp. 10, 17–18
[2] Microsoft Corporation (2001)

Why Security-Auditing?

Why Security-Auditing?

Pros

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

[1] Chappell (2008b)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

Cons

[1] Chappell (2008b)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

Cons

- **No attribution** for most events
  - Some recent improvements

[1] Chappell (2008b)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

Cons

- **No attribution** for most events
  - Some recent improvements
- Configurable: **events can be disabled**[2]

[1] Chappell (2008b)
[2] Microsoft Corporation (2021i)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

Cons

- **No attribution** for most events
  - Some recent improvements
- Configurable: **events can be disabled**[2]
- **Excessive tamper protection**
  - Can't enable optional metadata like stack backtraces[3]

Spring 2024 Edition

22ª EDIZIONE

[1] Chappell (2008b)
[2] Microsoft Corporation (2021i)
[3] Microsoft Corporation (2020a)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

Cons

- **No attribution** for most events
  - Some recent improvements
- Configurable: **events can be disabled**[2]
- **Excessive tamper protection**
  - Can't enable optional metadata like stack backtraces[3]
- **Inconsistent event structure**

[1] Chappell (2008b)
[2] Microsoft Corporation (2021i)
[3] Microsoft Corporation (2020a)

Why Security-Auditing?

Pros

- **Simply irreplaceable**
  - The only source of extremely important events
  - The only safe way to receive events from certain Windows components (e.g. LSASS)
- **Tamper protection**
  - Only LSASS can write to it[1]
  - Most events cannot be spoofed

Cons

- **No attribution** for most events
  - Some recent improvements
- Configurable: **events can be disabled**[2]
- **Excessive tamper protection**
  - Can’t enable optional metadata like stack backtraces[3]
- **Inconsistent event structure**
- **Worse API and less data** compared to the Security event log
  - Read access more restrictive[4, 5]
  - Some events are not logged to ETW (e.g. event 1102(S))

[1] Chappell (2008b)
[2] Microsoft Corporation (2021i)
[3] Microsoft Corporation (2020a)
[4] Microsoft Corporation (2021b)
[5] Chappell (2008a)

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

Case study: the Threat-Intelligence provider

An attempt was made

What it is

And what it does

What it is

And what it does

- An **ETW provider** – full name **Microsoft-Windows-Threat-Intelligence**, commonly abbreviated **ETW-TI**
- A special provider with **ad-hoc access controls** and **anti-tampering features**[1]
  - Read access is restricted to Defender, and third-party security products by approved Microsoft Virus Initiative (MVI) partners

[1] meekochii (2022)

What it is

And what it does

- An **ETW provider** – full name **Microsoft-Windows-Threat-Intelligence**, commonly abbreviated **ETW-TI**
- A special provider with **ad-hoc access controls** and **anti-tampering features**[1]
  - Read access is restricted to Defender, and third-party security products by approved Microsoft Virus Initiative (MVI) partners
- Logs telemetry about **code injection** and **driver loading**
- A safe alternative to kernel-mode code hooking
  - We would have preferred a synchronous API (e.g. inline hooks[2]), thank you very much

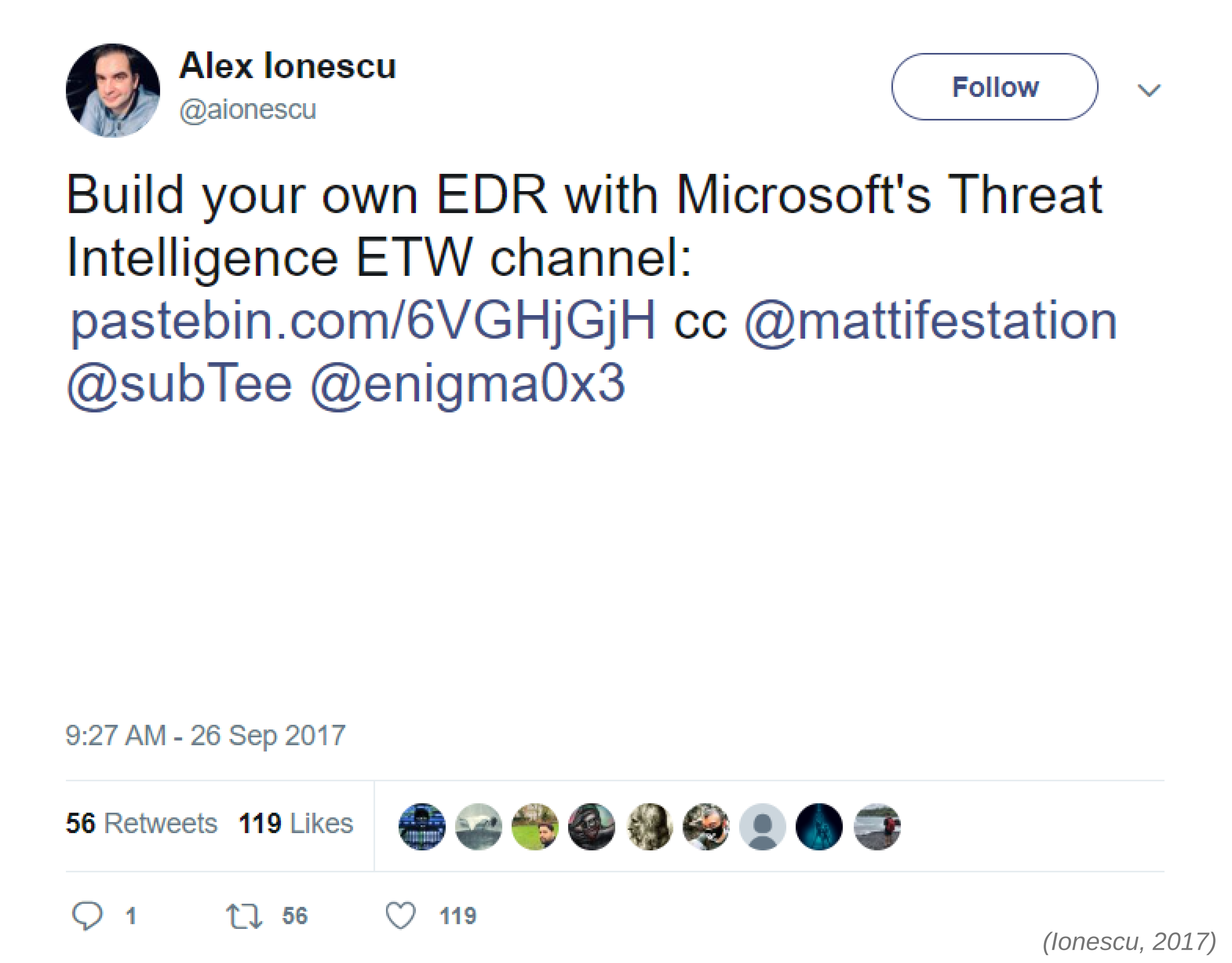
[1] meekochii (2022)
[2] Uhlmann and Bousseaden (2024)

What it is

And what it does

- An **ETW provider** – full name **Microsoft-Windows-Threat-Intelligence**, commonly abbreviated **ETW-TI**
- A special provider with **ad-hoc access controls** and **anti-tampering features**[1]
  - Read access is restricted to Defender, and third-party security products by approved Microsoft Virus Initiative (MVI) partners
- Logs telemetry about **code injection** and **driver loading**
- A safe alternative to kernel-mode code hooking
  - We would have preferred a synchronous API (e.g. inline hooks[2]), thank you very much
- **Greatly overrated**

[1] meekochii (2022)
[2] Uhlmann and Bousseaden (2024)



Alex Ionescu
@aionescu

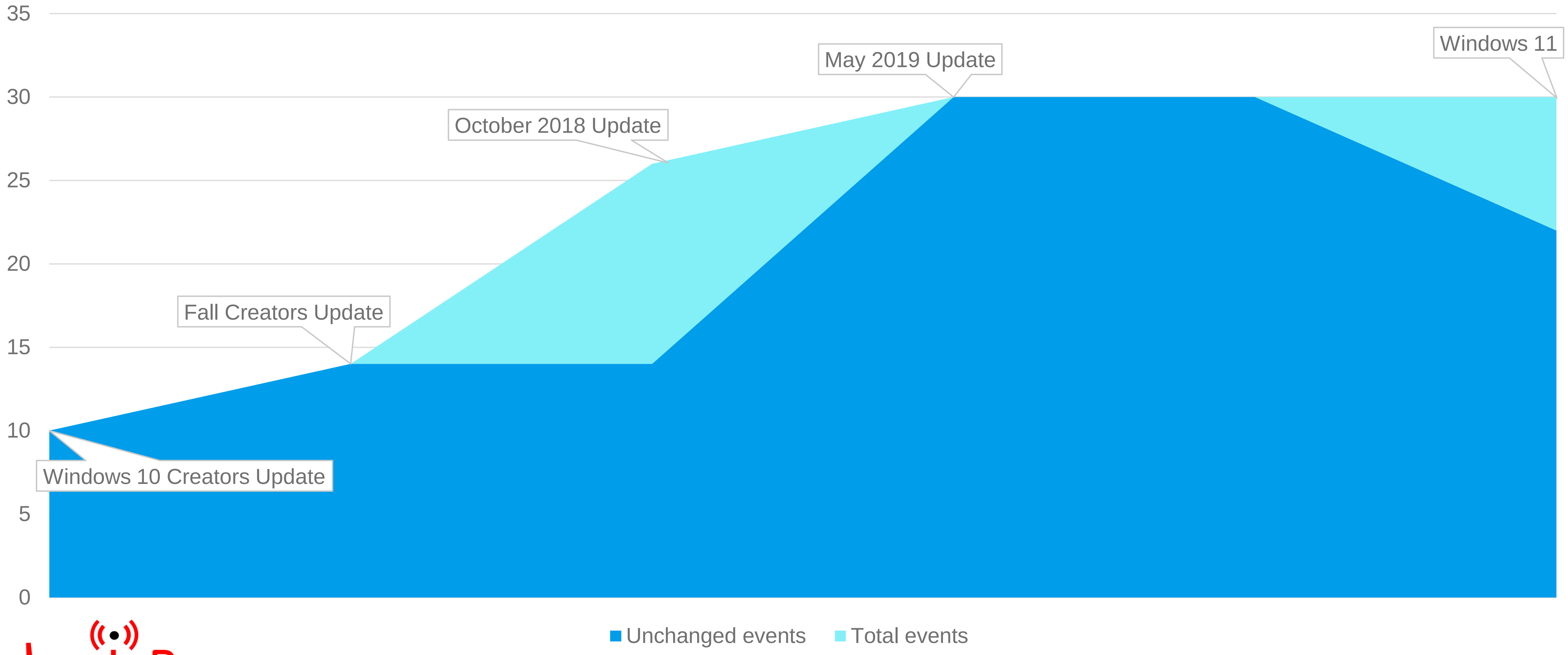
Build your own EDR with Microsoft's Threat Intelligence ETW channel: pastebin.com/6VGHjGjH cc @mattifestation @subTee @enigma0x3

9:27 AM - 26 Sep 2017

56 Retweets 119 Likes

(Ionescu, 2017)

Evolution of ETW-TI



Design of ETW-TI

Compare and contrast with Security-Auditing

Design of ETW-TI

Compare and contrast with Security-Auditing

- Designed *and* implemented by the Defender team[1]
  - Therefore, an ETW provider – Defender is very ETW-centric
    - See also the Microsoft-Antimalware-Scan-Interface provider, specifically designed to be consumed by Defender, *not* Windows Event Log[2]

1, 3. Seifert (2017)
2. Palantir (2019)

Design of ETW-TI

Compare and contrast with Security-Auditing

- Designed *and* implemented by the Defender team[1]
  - Therefore, an ETW provider – Defender is very ETW-centric
    - See also the Microsoft-Antimalware-Scan-Interface provider, specifically designed to be consumed by Defender, *not* Windows Event Log[2]
- Consequences of involving security people:
  - Threat modeling drove the initial design and the evolutions[3, 4]
  - The most complete and consistent logging of **requestor and target process/thread** of any ETW provider

1, 3. Seifert (2017)
2. Palantir (2019)
4. Rapaport (2019)

Why ETW-TI?

Why ETW-TI?

Pros

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse
- A good template for future security-oriented ETW providers
  - Good data
  - Internally consistent
  - Noise pre-filtering

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse
- A good template for future security-oriented ETW providers
  - Good data
  - Internally consistent
  - Noise pre-filtering

Cons

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse
- A good template for future security-oriented ETW providers
  - Good data
  - Internally consistent
  - Noise pre-filtering

Cons

- **Documentation** is almost non-existent[1]
  - Or, “I Joined MVI and All I Got Was This Lousy PDF”

1, 3. Microsoft Corporation (n.d.)

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse
- A good template for future security-oriented ETW providers
  - Good data
  - Internally consistent
  - Noise pre-filtering

Cons

- **Documentation** is almost non-existent[1]
  - Or, “I Joined MVI and All I Got Was This Lousy PDF”
- A victim of OS limitations:
  - **No attribution** for device and driver events
  - Call stacks not always available[2]

1, 3. Microsoft Corporation (n.d.)
2. Uhlmann and Bousseaden (2024)

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse
- A good template for future security-oriented ETW providers
  - Good data
  - Internally consistent
  - Noise pre-filtering

Cons

- **Documentation** is almost non-existent[1]
  - Or, “I Joined MVI and All I Got Was This Lousy PDF”
- A victim of OS limitations:
  - **No attribution** for device and driver events
  - Call stacks not always available[2]
- **Configurable**
  - Some events are opt-in[3]
    - … and can be opted out of[4]

1, 3. Microsoft Corporation (n.d.)
2. Uhlmann and Bousseaden (2024)
4. Meignan (2023)

Why ETW-TI?

Pros

- **Mostly irreplaceable**
  - The existing alternatives are worse
- A good template for future security-oriented ETW providers
  - Good data
  - Internally consistent
  - Noise pre-filtering

Cons

- **Documentation** is almost non-existent[1]
  - Or, “I Joined MVI and All I Got Was This Lousy PDF”
- A victim of OS limitations:
  - **No attribution** for device and driver events
  - Call stacks not always available[2]
- **Configurable**
  - Some events are opt-in[3]
    - … and can be opted out of[4]
- Good, **but not quite there**

[1, 3] Microsoft Corporation (n.d.)
[2] Uhlmann and Bousseaden (2024)
[4] Meignan (2023)

ETW-TI design issues

A little bit of everything

ETW-TI design issues

A little bit of everything

- **Kernel mode events** are easy to suppress
    - Suppress **queue user APC** event by queuing the user APC from a kernel APC[1]
    - Suppressing **process memory read/write** events is trivial

[1] Tsukerman (2019)

ETW-TI design issues

A little bit of everything

- **Kernel mode events** are easy to suppress
    - Suppress **queue user APC** event by queuing the user APC from a kernel APC[1]
    - Suppressing **process memory read/write** events is trivial
- **Suspend/resume** events don't log when a process or thread is *actually* suspended or resumed – they log each *attempt*
    - Not that there is a way to know when a *process* is suspended/resumed – OS limitation
    - **NtAlertResumeThread** is erroneously not considered a thread resume operation
        - Unfixed as of Windows 11 23H2

[1] Tsukerman (2019)

ETW-TI design issues

A little bit of everything

- **Kernel mode events** are easy to suppress
    - Suppress **queue user APC** event by queuing the user APC from a kernel APC[1]
    - Suppressing **process memory read/write** events is trivial
- **Suspend/resume** events don't log when a process or thread is *actually* suspended or resumed – they log each *attempt*
    - Not that there is a way to know when a *process* is suspended/resumed – OS limitation
    - **NtAlertResumeThread** is erroneously not considered a thread resume operation
        - Unfixed as of Windows 11 23H2
- Several classes of events can be suppressed on a per-process basis by anyone with the Debug or Tcb privileges (e.g. Administrators or LocalSystem)[2]
    - Fixed in Windows 11: only certified antimalware software can change the flag[3]

[1] Tsukerman (2019)
[2], [3] Meignan (2023)

ETW-TI design issues

A little bit of everything

- **Kernel mode events** are easy to suppress
    - Suppress **queue user APC** event by queuing the user APC from a kernel APC[1]
    - Suppressing **process memory read/write** events is trivial
- **Suspend/resume** events don’t log when a process or thread is *actually* suspended or resumed – they log each *attempt*
    - Not that there is a way to know when a *process* is suspended/resumed – OS limitation
    - **NtAlertResumeThread** is erroneously not considered a thread resume operation
        - Unfixed as of Windows 11 23H2
- Several classes of events can be suppressed on a per-process basis by anyone with the Debug or Tcb privileges (e.g. Administrators or LocalSystem)[2]
    - Fixed in Windows 11: only certified antimalware software can change the flag[3]
- **Driver and device** events log so little data that they are virtually useless

HackInBo® Spring 2024 Edition

22ª EDIZIONE

1. Tsukerman (2019)
2, 3. Meignan (2023)

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

The road not taken

How literally everyone but Microsoft does it

Policy modules

The perfect security API does not exi–

Policy modules

The perfect security API does not exi–

- Policy modules are a **design pattern** and not a single API
- Kernel-mode API for **modular**, **synchronous** hooking of *all* security-sensitive operations
  - Designed for **access control**, but perfectly good for **auditing** too

Policy modules

The perfect security API does not exi–

- Policy modules are a **design pattern** and not a single API
- Kernel-mode API for **modular**, **synchronous** hooking of *all* security-sensitive operations
  - Designed for **access control**, but perfectly good for **auditing** too
- **FreeBSD** (since 5.0 – Jan 2003) [1]: TrustedBSD Mandatory Access Control (MAC) Framework
  - “MAC” is a misnomer
    - Historical reasons – we’ll see later

[1], [3]. The TrustedBSD Project (2017b)

Policy modules

The perfect security API does not exi–

- Policy modules are a **design pattern** and not a single API
- Kernel-mode API for **modular**, **synchronous** hooking of *all* security-sensitive operations
  - Designed for **access control**, but perfectly good for **auditing** too
- **FreeBSD** (since 5.0 – Jan 2003) [1]: TrustedBSD Mandatory Access Control (MAC) Framework
  - “MAC” is a misnomer
    - Historical reasons – we’ll see later
- **Linux** (since 2.6 – Dec 2003) [2]: Linux Security Modules (LSM)

[1], [3]. The TrustedBSD Project (2017b)
[2]. The TrustedBSD Project (2017a)

Policy modules

The perfect security API does not exi–

- Policy modules are a **design pattern** and not a single API
- Kernel-mode API for **modular**, **synchronous** hooking of *all* security-sensitive operations
  - Designed for **access control**, but perfectly good for **auditing** too
- **FreeBSD** (since 5.0 – Jan 2003) [1]: TrustedBSD Mandatory Access Control (MAC) Framework
  - “MAC” is a misnomer
    - Historical reasons – we’ll see later
- **Linux** (since 2.6 – Dec 2003) [2]: Linux Security Modules (LSM)
- **macOS** (since 10.5 “Leopard” – Oct 2007) [3]: MAC policy modules
  - macOS 11 “Big Sur” (Nov 2020) shipped with *seven* policy modules[4]

Spring 2024 Edition

22ª EDIZIONE

[1, 3] The TrustedBSD Project (2017b)
[2] The TrustedBSD Project (2017a)
[4] Student (2021)

Policy modules: genealogy

Microsoft and the “Not Invented Here (NIH) syndrome”

- What do FreeBSD, Linux and macOS have in common?

Policy modules: genealogy

Microsoft and the “Not Invented Here (NIH) syndrome”

- What do FreeBSD, Linux and macOS have in common?
- **All of them include a port of TrustedBSD**, the first implementation of policy modules[1]
  - An uninterrupted legacy dating back to 1992, when the NSA starts working on **Distributed Trusted Mach (DTMach)** [2]
  - Often called “MAC” policy modules because the first module ever (**SEBSD** – a BSD port of **SELinux**) implemented a **Mandatory Access Control (MAC)** policy[3]

[1], [3]. The TrustedBSD Project (2017a)
[2]. Smalley (2000)

Policy modules: genealogy

Microsoft and the “Not Invented Here (NIH) syndrome”

- What do FreeBSD, Linux and macOS have in common?
- **All of them include a port of TrustedBSD**, the first implementation of policy modules[1]
  - An uninterrupted legacy dating back to 1992, when the NSA starts working on **Distributed Trusted Mach (DTMach)** [2]
  - Often called “MAC” policy modules because the first module ever (**SEBSD** – a BSD port of **SELinux**) implemented a **Mandatory Access Control (MAC)** policy[3]
- Microsoft missed multiple chances to adopt TrustedBSD, or its predecessors, or any of their concepts

[1], [3]. The TrustedBSD Project (2017a)
[2]. Smalley (2000)

Endpoint Security framework

The natural evolution of policy modules

Endpoint Security framework

The natural evolution of policy modules

- Introduced in **macOS 10.15 "Catalina"**[1] as a user-mode projection of the policy module API[2]
  - macOS 10.15 bans third-party kernel-mode code[3]
  - The policy module API was never officially documented or supported[4]

1, 3. White (2020)
2. Levin (2019)
4. Apple Inc. (2008)

Endpoint Security framework

The natural evolution of policy modules

- Introduced in **macOS 10.15 "Catalina"**[1] as a user-mode projection of the policy module API[2]
  - macOS 10.15 bans third-party kernel-mode code[3]
  - The policy module API was never officially documented or supported[4]
- A true security API: **not controllable, nor configurable, nor extensible**
- Both **synchronous** ("authorization") and **asynchronous** ("notification") modes

1, 3. White (2020)
2. Levin (2019)
4. Apple Inc. (2008)

Endpoint Security framework

The natural evolution of policy modules

- Introduced in **macOS 10.15 “Catalina”**[1] as a user-mode projection of the policy module API[2]
  - macOS 10.15 bans third-party kernel-mode code[3]
  - The policy module API was never officially documented or supported[4]
- A true security API: **not controllable, nor configurable, nor extensible**
- Both **synchronous** (“authorization”) and **asynchronous** (“notification”) modes
- For each event:
  - **System time** *and* **monotonic clock** timestamps[5]
  - **User** *and* **code identity**[6]

1, 3. White (2020)
2. Levin (2019)
4. Apple Inc. (2008)
5. Apple Inc. (2024a)
6. Apple Inc. (2024b)

Endpoint Security framework

The natural evolution of policy modules

- Introduced in **macOS 10.15 “Catalina”**[1] as a user-mode projection of the policy module API[2]
  - macOS 10.15 bans third-party kernel-mode code[3]
  - The policy module API was never officially documented or supported[4]
- A true security API: **not controllable, nor configurable, nor extensible**
- Both **synchronous** (“authorization”) and **asynchronous** (“notification”) modes
- For each event:
  - **System time** *and* **monotonic clock** timestamps[5]
  - **User** *and* **code identity**[6]
- Nobody’s perfect: **no grouping or correlation of events, no request tracking**

1, 3. White (2020)
2. Levin (2019)
4. Apple Inc. (2008)
5. Apple Inc. (2024a)
6. Apple Inc. (2024b)

HackInBo®

Spring 2024 Edition

22ª EDIZIONE

The road ahead

Temper your expectations

Secure Future Initiative (SFI) [1]

The future of Windows security

[1] Weston (2024b)

Secure Future Initiative (SFI) [1]

The future of Windows security

- Secure hardware platform: **Pluton** and **TPM 2.0**

[1] Weston (2024b)

Secure Future Initiative (SFI)[1]

The future of Windows security

- Secure hardware platform: **Pluton** and **TPM 2.0**
- Expansion of **Virtualization Based Security (VBS)**

[1] Weston (2024b)

Secure Future Initiative (SFI) [1]

The future of Windows security

- Secure hardware platform: **Pluton** and **TPM 2.0**
- Expansion of **Virtualization Based Security (VBS)**
- Sandboxing with **AppContainers**

[1] Weston (2024b)

Secure Future Initiative (SFI) [1]

The future of Windows security

- Secure hardware platform: **Pluton** and **TPM 2.0**
- Expansion of **Virtualization Based Security (VBS)**
- Sandboxing with **AppContainers**
- Removal of **unfixable legacy features**
  - NTLM, printer drivers, weak RSA keys…

[1] Weston (2024b)

Secure Future Initiative (SFI) [1]

The future of Windows security

- Secure hardware platform: **Pluton** and **TPM 2.0**
- Expansion of **Virtualization Based Security (VBS)**
- Sandboxing with **AppContainers**
- Removal of **unfixable legacy features**
  - NTLM, printer drivers, weak RSA keys…
- **Rewrite it in Rust** (applause)

Rust?
(applause)

(Weston, 2024a)

[1] Weston (2024b)

HackInBo® Spring 2024 Edition
22ª EDIZIONE

The future of the Windows security architecture

Same as it ever was

The future of the Windows security architecture

Same as it ever was

- **No architecture changes**
  - New features built on top of old ones[1]
  - TPM, VBS, AppContainers, etc. already exist – their scope will simply be expanded

[1] Weston (2024b)

The future of the Windows security architecture

Same as it ever was

- **No architecture changes**
  - New features built on top of old ones[1]
  - TPM, VBS, AppContainers, etc. already exist – their scope will simply be expanded
- Stricter hardware requirements
  - Windows may scale down, but Windows *security* won’t

[1] Weston (2024b)

The future of the Windows security architecture

Same as it ever was

- **No architecture changes**
  - New features built on top of old ones[1]
  - TPM, VBS, AppContainers, etc. already exist – their scope will simply be expanded
- Stricter hardware requirements
  - Windows may scale down, but Windows *security* won’t
- Safe to assume that **ETW won’t be fixed, expanded *or* supplanted**

[1] Weston (2024b)

HackInBo®

Spring 2024 Edition

22ª EDIZIONE

Thank you!

In memoriam

Geoff Chappell

? – September 3, 2023[1]

[1] Goodspeed (2023)

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

Appendix: bibliography

Standing on the shoulders of giants

References

Apple Inc. (2008). Kernel’s MAC framework. *Technical Q&A*, *QA1574*. Apple Developer Documentation Archive. https://developer.apple.com/library/archive/qa/qa1574/_index.html

Apple Inc. (2024a). *es_message_t*. Apple Developer Documentation; Apple Inc. https://developer.apple.com/documentation/endpointsecurity/es_message_t

Apple Inc. (2024b). *es_process_t*. Apple Developer Documentation; Apple Inc. https://developer.apple.com/documentation/endpointsecurity/es_process_t

Baril, D., & Itkin, E. (2019, August 7). *He said, she said: Poisoned RDP offense and defense* [YouTube]. Black Hat USA 2019. https://www.youtube.com/watch?v=3wncyS-QOBk

Binarly Team. (2021, November 15). Modern EDR design issues: Bypassing ETW-based solutions. *Binarly*. https://www.binarly.io/blog/design-issues-of-modern-edrs-bypassing-etw-based-solutions

Chappell, G. (2008a, November 21). *EtwRegisterSecurityProvider*. Geoff Chappell, Software Analyst; Geoff Chappell. https://www.geoffchappell.com/studies/windows/win32/ntdll/api/etw/registersecurityprovider.htm

Chappell, G. (2008b, November 21). *EtwWriteUMSecurityEvent*. Geoff Chappell, Software Analyst; Geoff Chappell. https://www.geoffchappell.com/studies/windows/win32/ntdll/api/etw/writeumsecurityevent.htm

References

Department of Defense. (1985). *Trusted Computer System Evaluation Criteria* (5200.28-STD). United States Department of Defense.

Goodspeed, T. (2023, September 4). *@Travisgoodspeed*. https://twitter.com/travisgoodspeed/status/1698715249593958560

Ionescu, A. (2017, September 26). Build your own EDR with Microsoft’s Threat Intelligence ETW channel. *@Aionescu*. https://twitter.com/aionescu/status/912715300931555329

Levin, J. (2019, November 6). *Endpoint Security*. *OS Internals. https://newosxbook.com/articles/eps.html

meekochii. (2022, September 19). Introduction into Microsoft Threat Intelligence drivers (ETW-TI). *Meeko Labs*. https://research.meekolab.com/introduction-into-microsoft-threat-intelligence-drivers-etw-ti

Meignan, M. (2023, October 9). A universal EDR bypass built in Windows 10. *RiskInsight*. https://www.riskinsight-wavestone.com/en/2023/10/a-universal-edr-bypass-built-in-windows-10/

Microsoft Corporation. (n.d.). *Microsoft Virus Initiative (MVI) Documentation*. Microsoft Partner Center. Confidential.

Microsoft Corporation. (2001). C2 evaluation and certification for Windows NT. *Microsoft Knowledge Base, Q93362*. KnowledgeBase Archive. https://jeffpar.github.io/kbarchive/kb/093/Q93362/

References

Microsoft Corporation. (2020a, February 9). *examples/NativeExamples/user_trace_005.cpp*. KrabsETW; GitHub. https://github.com/microsoft/krabsetw/blob/master/examples/NativeExamples/user_trace_005.cpp

Microsoft Corporation. (2020b, February 9). *krabs/krabs.hpp*. KrabsETW; GitHub. https://github.com/microsoft/krabsetw/blob/master/krabs/krabs.hpp

Microsoft Corporation. (2021a, January 8). *krabs/krabs/perfinfo_groupmask.hpp*. KrabsETW; GitHub. https://github.com/microsoft/krabsetw/blob/master/krabs/krabs/perfinfo_groupmask.hpp

Microsoft Corporation. (2021b, July 1). *Event logging security*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/windows/win32/eventlog/event-logging-security

Microsoft Corporation. (2021c, August 19). *WNODE_HEADER structure (Wmistr.h)*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/windows/win32/etw/wnode-header

Microsoft Corporation. (2021d, September 6). *Advanced security audit policy settings*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/advanced-security-audit-policy-settings

References

Microsoft Corporation. (2021e, September 6). *Audit Filtering Platform Connection*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/audit-filtering-platform-connection

Microsoft Corporation. (2021f, September 6). *Audit Filtering Platform Policy Change*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/audit-filtering-platform-policy-change

Microsoft Corporation. (2021g, September 6). *Audit Other Object Access Events*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/audit-other-object-access-events

Microsoft Corporation. (2021h, September 6). *Audit PNP Activity*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/audit-pnp-activity

Microsoft Corporation. (2021i, October 13). *AuditSetSystemPolicy function*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/windows/win32/api/ntsecapi/nf-ntsecapi-auditsetsystempolicy

References

Microsoft Corporation. (2022, May 16). *Instrumenting your code with ETW*. Learn; Microsoft Corporation. https://learn.microsoft.com/en-us/windows-hardware/test/weg/instrumenting-your-code-with-etw

Microsoft Corporation. (2023, April 4). *krabsetw/krabs/krabs/etw.hpp at master · microsoft/krabsetw*. KrabsETW; GitHub. https://github.com/microsoft/krabsetw/blob/master/krabs/krabs/etw.hpp

Microsoft Corporation. (2024). *Group Policy: Audit Configuration Extension* (MS-GPAC). Microsoft Corporation. https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-gpac/10d91136-2d82-46b9-9677-cf4d47ba2261

Muratori, C. (2014, August 6). The worst API ever made. *Casey's Tech Stuff*. https://caseymuratori.com/blog_0025

Palantir. (2019, February 28). Tampering with Windows Event Tracing: Background, offense, and defense. Palantir Blog. https://blog.palantir.com/tampering-with-windows-event-tracing-background-offense-and-defense-4be7ac62ac63

Rapaport, A. (2019, March 25). *From alert to driver vulnerability: Microsoft Defender ATP investigation unearths privilege escalation flaw*. Microsoft Security Blog; Microsoft Corporation. https://www.microsoft.com/en-us/security/blog/2019/03/25/from-alert-to-driver-vulnerability-microsoft-defender-atp-investigation-unearths-privilege-escalation-flaw/

References

Seifert, C. (2017, November 13). Detecting reflective DLL loading with Windows Defender ATP. Microsoft Security Blog; Microsoft Corporation. https://www.microsoft.com/en-us/security/blog/2017/11/13/detecting-reflective-dll-loading-with-windows-defender-atp/

Smalley, S. (2000, December 26). *Flask: Flux Advanced Security Kernel*. Flux Research Group; The University of Utah. https://www-old.cs.utah.edu/flux/fluke/html/flask.html

Softley, I. (Director). (1995, September 15). *Hackers*. MGM/UA Distribution Co.

Student, T. (2021, August 17). macOS 11’s hidden security improvements. *Malwarebytes Labs*. https://www.malwarebytes.com/blog/news/2021/08/macos-11s-hidden-security-improvements

Teodorescu, C., Korkin, I., & Golchikov, A. (2021, November 10). *Veni, no vidi, no vici: Attacks on ETW blind EDR sensors* [Virtual]. Black Hat Europe. https://www.youtube.com/watch?v=wZG0h1q7fMg

The TrustedBSD Project. (2017a, March 18). *SEBSD: Port of SELinux FLASK and Type Enforcement to TrustedBSD*. The TrustedBSD Project. http://www.trustedbsd.org/sebsd.html

The TrustedBSD Project. (2017b, March 18). *TrustedBSD Mandatory Access Control (MAC) framework*. The TrustedBSD Project. http://www.trustedbsd.org/mac.html

References

Tsukerman, P. (2019, September 27). *Bypassing the Microsoft-Windows-Threat-Intelligence kernel APC injection sensor*. https://medium.com/@philiptsukerman/bypassing-the-microsoft-windows-threat-intelligence-kernel-apc-injection-sensor-92266433e0b0

Uhlmann, J. (2023, March 29). Effective parenting - detecting LRPC-based parent PID spoofing. *Elastic Security Labs*. https://www.elastic.co/security-labs/effective-parenting-detecting-lrpc-based-parent-pid-spoofing

Uhlmann, J., & Bousseaden, S. (2024, January 9). Doubling down: Detecting in-memory threats with kernel ETW call stacks. *Elastic Security Labs*. https://www.elastic.co/security-labs/doubling-down-etw-callstacks

Weston, D. (2024a, April 19). *Windows 11: The journey to security by default* [In-Person]. BlueHat IL 2023. https://www.youtube.com/watch?v=8T6ClX-y2AE

Weston, D. (2024b, May 20). *New Windows 11 features strengthen security to address evolving cyberthreat landscape*. Microsoft Security Blog; Microsoft Corporation. https://www.microsoft.com/en-us/security/blog/2024/05/20/new-windows-11-features-strengthen-security-to-address-evolving-cyberthreat-landscape/

White, M. (2020, June 24). *Build an Endpoint Security app* [Video]. WWDC20. https://developer.apple.com/videos/play/wwdc2020/10159/

HackInBo®

Spring 2024 Edition

22ª EDIZIONE

Appendix: resources

Further reading and useful tools

Resources

Microsoft Learn

The official documentation for all Microsoft products, services, open protocols, file formats, etc. Comprehensive, and high-quality.

https://learn.microsoft.com/en-us/

Pavel Yosifovich, Mark Russinovich, Alex Ionescu, David Solomon and Andrea Allievi, *Windows Internals*; 7th Edition. 2017/2021, Pearson Education

The Windows internals classic.

ISBNs 978-0133986464 (part 1), 978-0135462447 (part 2)

Gary Nebbett, *Windows NT/2000 Native API Reference*. 2000, Macmillan Technical Publishing

The *other* Windows internals classic. Dated, partly obsolete, but still useful.

ISBN 978-1578701995

Geoff Chappel, Software Analyst. 1997–2003

Geoff Chappel’s Windows internals goldmine and immortal legacy. Invaluable.

https://www.geoffchappell.com/

Resources

Winsider Seminars & Solutions, *phnt*

C library of definitions of undocumented Windows structures and functions – the best of its kind.

https://github.com/winsiderss/phnt

Microsoft, *Message Analyzer*

The best ETW event analyzer ever made; doubles as a network sniffer. Discontinued in 2019 and no longer offered for download. Still works perfectly, although it doesn't support the latest ETW features.

The last version was archived by Rafael Rivera at https://github.com/riverar/messageanalyzer-archive

Pavel Yosifovich, *ETW Explorer*

Viewer for ETW provider metadata: events, keywords, strings and a reconstruction of the instrumentation manifest. A must-have.

https://github.com/zodiacon/EtwExplorer

Microsoft, *KrabsETW*

"Krabs is a wrapper around ETW because ETW is the worst API ever made." Libraries for C++ and .NET.

https://github.com/microsoft/krabsetw/

Resources

Jackson T., *Telemetry Sourcerer*

Open source tool for experimenting with ETW tampering

https://github.com/jthuraisamy/TelemetrySourcerer

Bruce Dawson, *Random ASCII*

Blog on Windows software performance. Includes invaluable information and tools for working with ETW for performance applications.

https://randomascii.wordpress.com/

HackInBo®
Spring 2024 Edition
22ª EDIZIONE

Appendix: acknowledgements

It takes a village

Acknowledgements

Elia Florio

Technical advisor

… and many humble people who declined a credit

Technical advisors

Copy editors

Preview attendees

HackInBo®
Spring 2024 Edition
22ª EDIZIONE