Hidden Telemetry

# Uncovering TraceLogging ETW Providers You're Not Using (Yet)

Asuka Nakajima (@AsuNa_jp),
Senior Security Research Engineer, Elastic

elastic

# Asuka Nakajima

Senior Security Research Engineer, Elastic

## Endpoint Security R&D

- ✓ Especially developing new detection features for EDR (Elastic Defend)
- ✓ 10+ years of experience in cyber security R&D

## Founder of CTF for GIRLS (est. 2014)

- ✓ First female infosec community in Japan

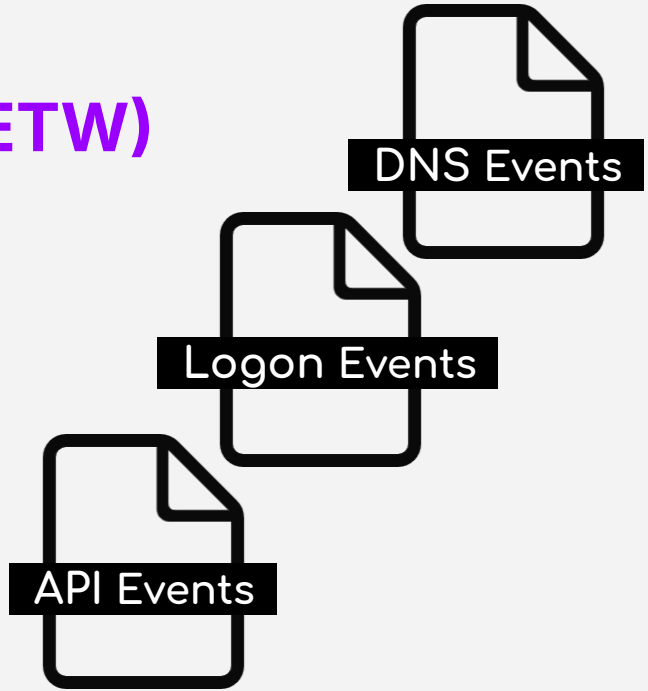
## Review Board Member

- ✓ BlackHatUSA / BlackHatAsia / CODE BLUE

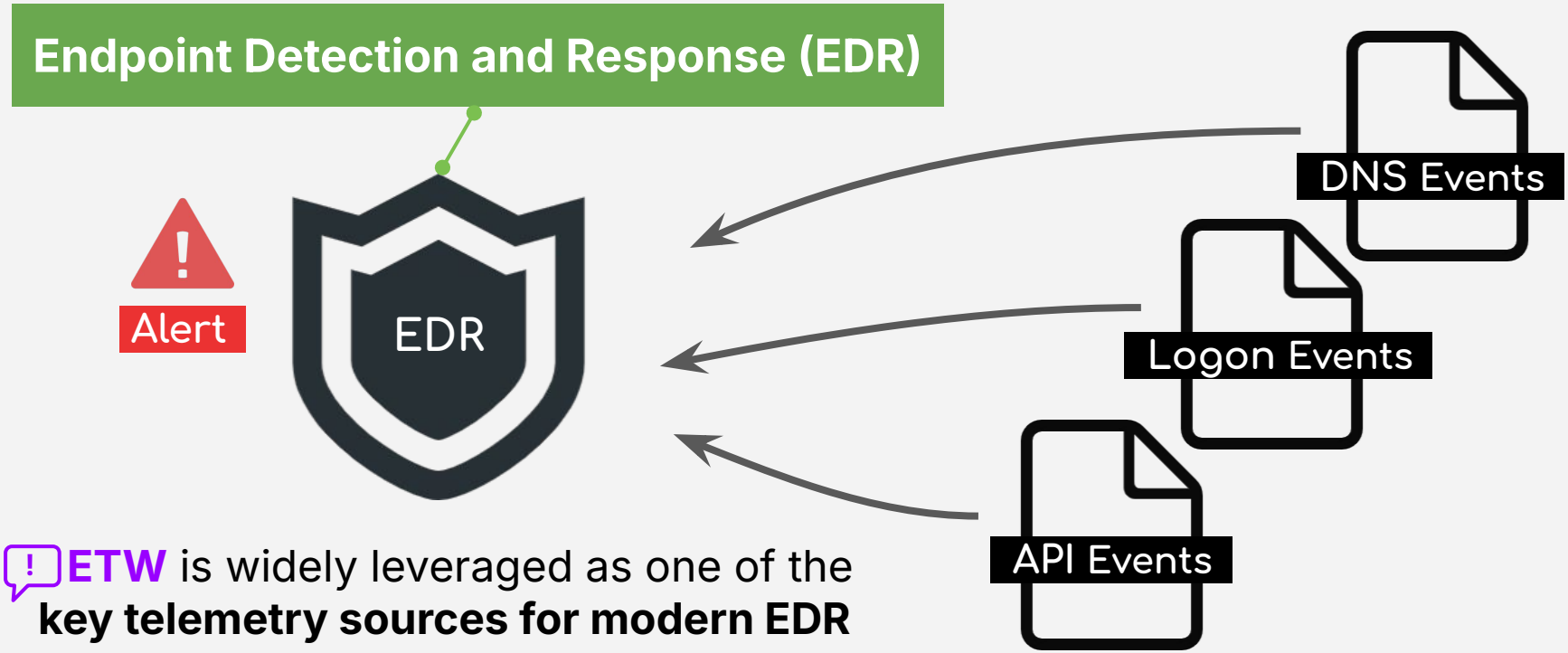
# What is Event Tracing for Windows (ETW)? 🤔

## Event Tracing for Windows (ETW)

A built-in Windows logging and tracing framework that provides detailed visibility into what's happening on a machine by collecting system and application events.

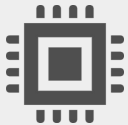


# What is Event Tracing for Windows (ETW)? 🤔




**ETW** is widely leveraged as one of the **key telemetry sources for modern EDR products**

# Key Components of ETW Architecture [1/2]



## Provider

Applications and drivers that emit events

## Consumer

Applications that receive events

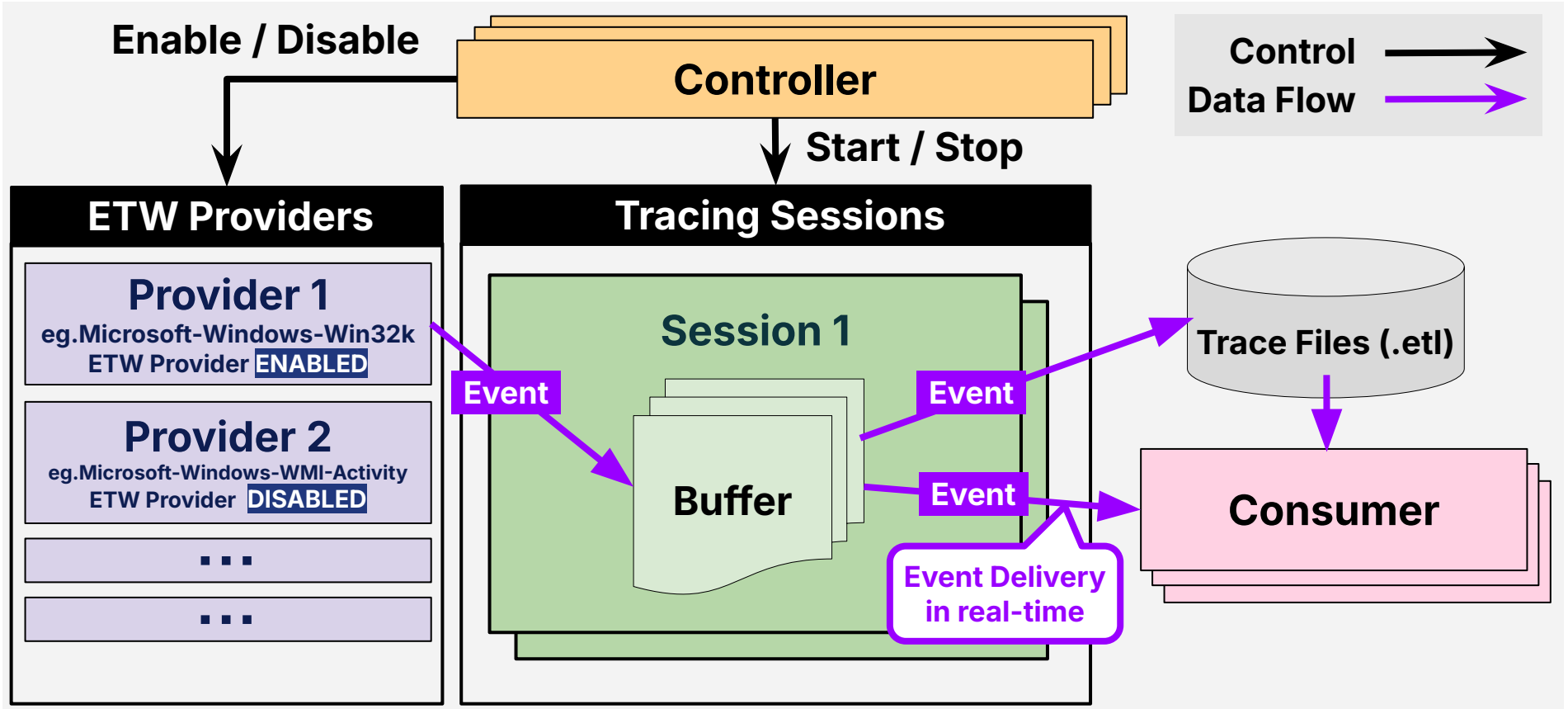
## Session

Relays events sent from provider, storing them in a buffer

## Controller

Manages trace sessions and providers

# Key Components of ETW Architecture [2/2]



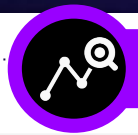
# Types of ETW Providers

| | Legacy Providers | Modern Providers ✨ |
|---|---|---|
| ***Event Providers***<br>Administrative, operational ,analytical and debugging purposes (for admin/users) | **MOF* based providers**<br>(*Managed Object Format) | **Manifest-based providers** |
| ***Trace Providers***<br>Tracing, diagnostics and debugging purposes (for original developers) | **WPP* providers**<br>(*Windows software trace preprocessor) | **TraceLogging providers** |

https://learn.microsoft.com/en-us/windows/win32/etw/about-event-tracing#types-of-providers

# Manifest-based vs. TraceLogging: Key Differences

## Manifest-based

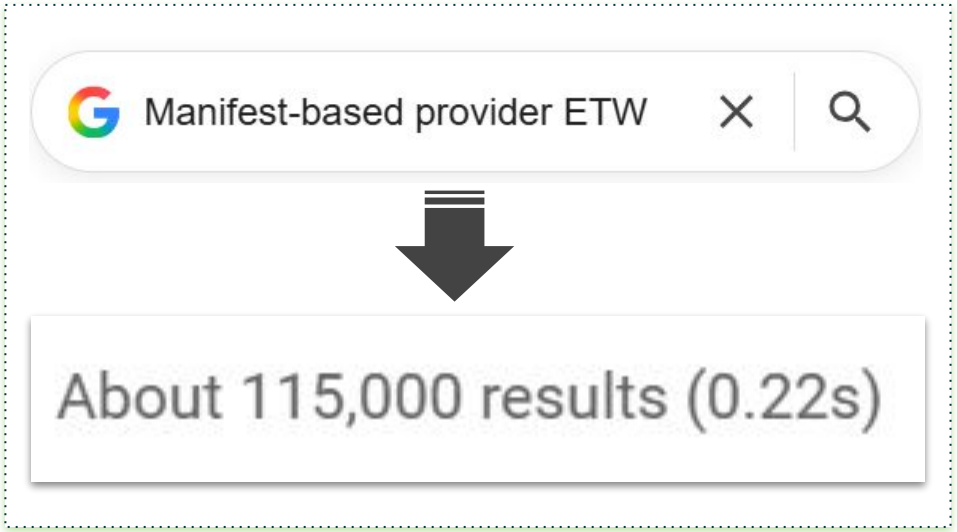


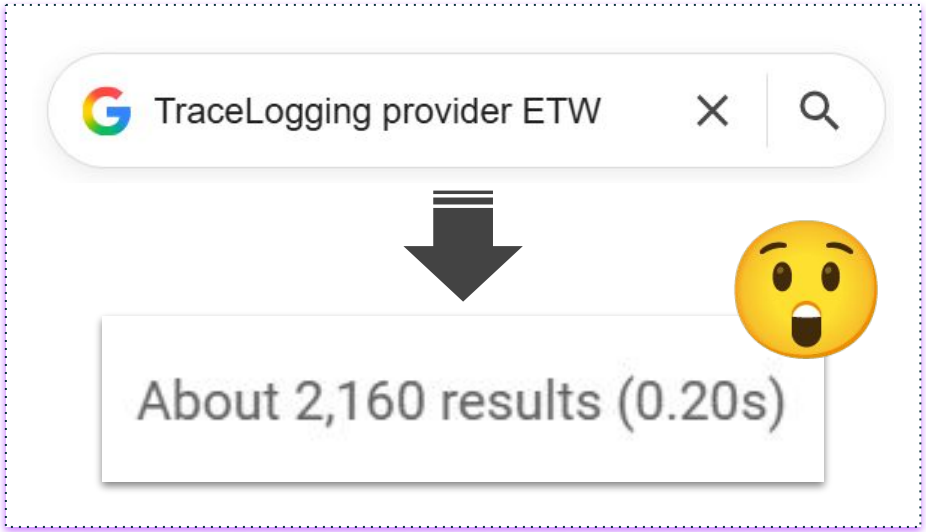
## TraceLogging

| | | |
|---|---|---|
| Providers typically named with hyphens (e.g. ***Microsoft-Windows-Firewall***). Events have stable numeric IDs. | 1 | Providers typically named with dots (e.g. ***Microsoft.Windows.Firewall***). Events identified by name, not ID. |
| **Metadata** (provider, event, etc) defined in an XML manifest file. Embedded in the binary's *.rsrc* section. | 2 | Self-describing events (no manifest file) **Metadata** is typically embedded in the binary's *.rdata* section. |
| **Registered & Discoverable:** A registry entry is created at install time. Easy to enumerate providers and events via standard tools and APIs. | 3 | **No registration & Not discoverable:** No registry entry. No standard tool can list providers / events. |

# TraceLogging: Current State of Research

TraceLogging providers have significantly less publicly available information/research compared to Manifest-based providers.




Note: Search results were retrieved in April 2026.

# TraceLogging: Current State of Research

TraceLogging providers have significantly less publicly available information/research compared to Manifest-based providers.

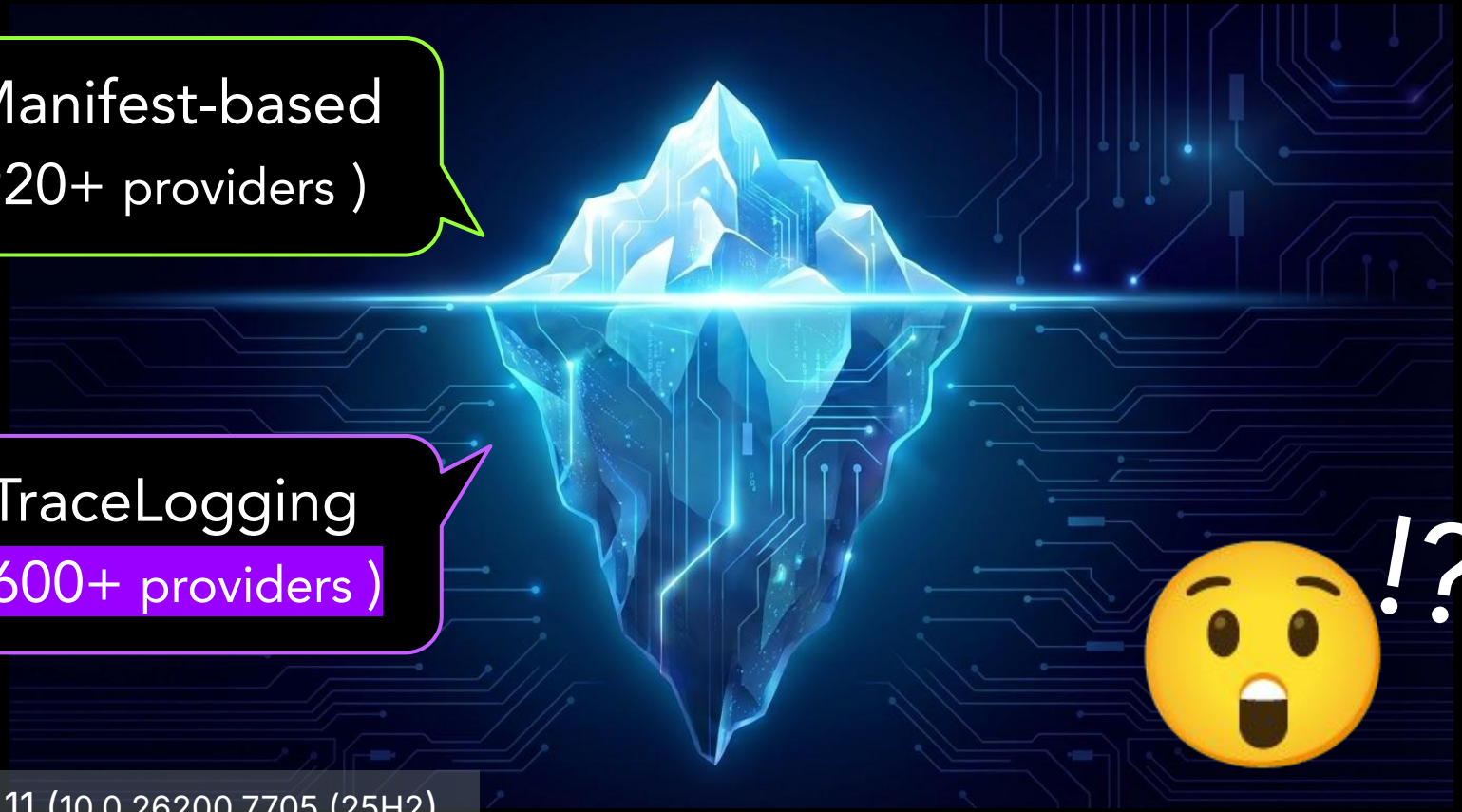
https://www.youtube.com/watch?v=KDn5XiKODtQ


REcon 2019 (Matt Graeber)

Released TraceLogging Metadata Parser *(TLGMetadataParser.psm1)*

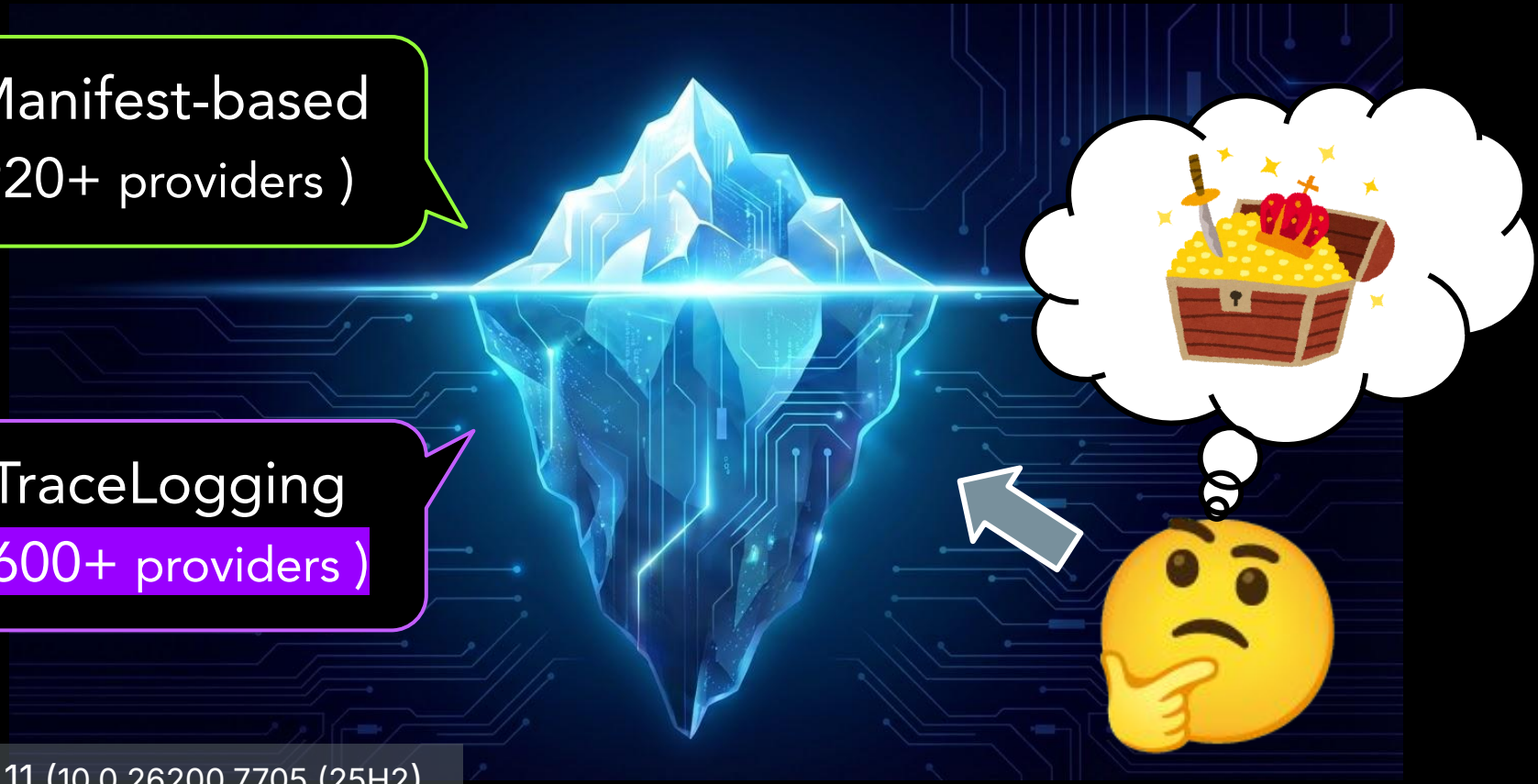
TLGMetadataParser.psm1: https://gist.github.com/mattifestation/edbac1614694886c8ef4583149f53658

# TraceLogging: An Unexplored Goldmine ?




Windows 11 (10.0.26200.7705 (25H2)

# TraceLogging: An Unexplored Goldmine ?




Windows 11 (10.0.26200.7705 (25H2)

# TraceLogging Provider: Example Code

**TlgExample.exe**

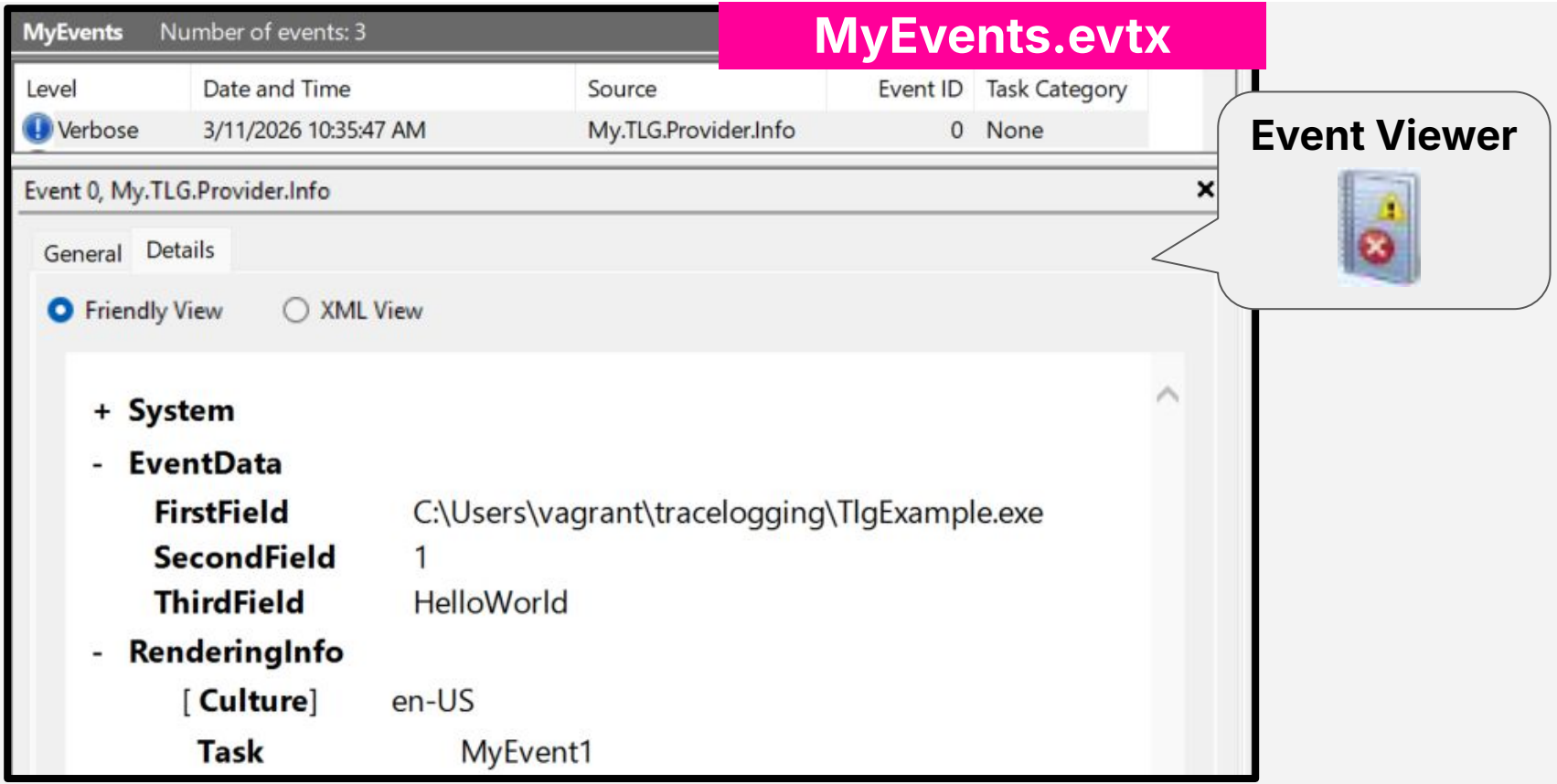
```
#include <windows.h>
#include <TraceLoggingProvider.h>

TRACELOGGING_DEFINE_PROVIDER(g_hProviderA,
    "My.TLG.ProviderA.Info", // ProviderName
    (0xb3864c38,0x4273,0x58c5,0x54,0x5b,0x8b,0x36,0x08,0x34,0x34,0x71)); // GUID

int main(int argc, char* argv[])
{
    TraceLoggingRegister(g_hProviderA);

    TraceLoggingWrite(g_hProviderA,"MyEvent1", // EventName
                TraceLoggingString(argv[0],"FirstField"),        // Field #1
                TraceLoggingInt32(argc, "SecondField"),          // Field #2
                TraceLoggingString("Helloworld","ThirdField")); // Field #3

    TraceLoggingUnregister(g_hProviderA);
    return 0;
}
```

# TraceLogging Provider: How to Capture Events

## Capture Events

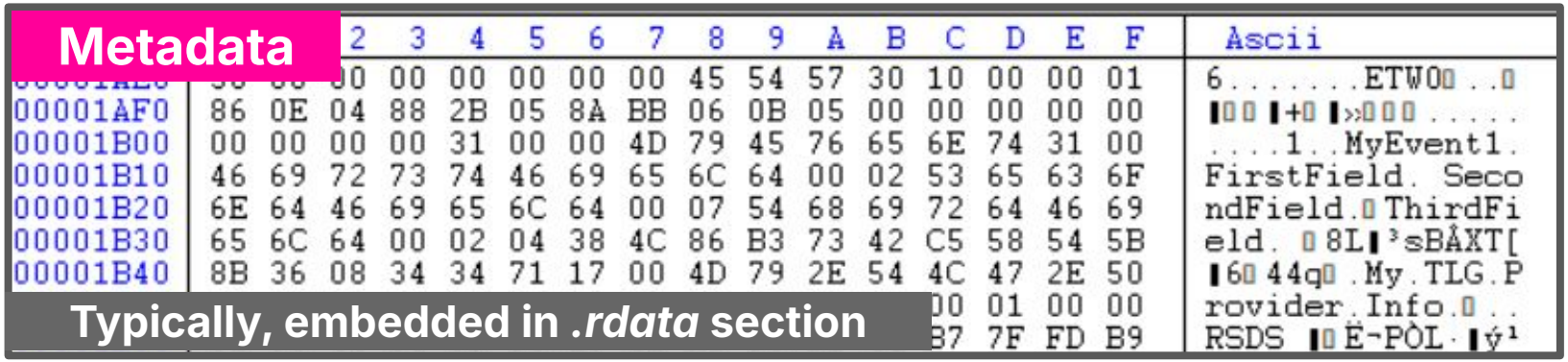
```
PS C:\> New-EtwTraceSession -Name MyProviderATrace -LocalFilePath
C:\temp\MyEvents.etl

PS C:\> Add-EtwTraceProvider -SessionName MyProviderATrace -Guid
'{b3864c38-4273-58c5-545b-8b3608343471}' -MatchAnyKeyword 0xFFFFFFFFFFFF
-Level 0xFF

PS C:\> .\TlgExample.exe

PS C:\> Remove-EtwTraceSession -Name MyProviderATrace

PS C:\> tracerpt C:\temp\MyEvents.etl -o C:\temp\MyEvents.evtx -of EVTX
```

# TraceLogging Provider: How to Capture Events



# TraceLogging Provider: Metadata Overview

**Metadata**



| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | Ascii |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 00 | 00 | 00 | 00 | 00 | 00 | 45 | 54 | 57 | 30 | 10 | 00 | 00 | 01 | 6.....ETW0□..□ |
| 00001AF0 | 86 | 0E | 04 | 88 | 2B | 05 | 8A | BB | 06 | 0B | 05 | 00 | 00 | 00 | 00 | 00 | ▌□□▌+□▌»□□□..... |
| 00001B00 | 00 | 00 | 00 | 00 | 31 | 00 | 00 | 4D | 79 | 45 | 76 | 65 | 6E | 74 | 31 | 00 | ....1..MyEvent1. |
| 00001B10 | 46 | 69 | 72 | 73 | 74 | 46 | 69 | 65 | 6C | 64 | 00 | 02 | 53 | 65 | 63 | 6F | FirstField. Seco |
| 00001B20 | 6E | 64 | 46 | 69 | 65 | 6C | 64 | 00 | 07 | 54 | 68 | 69 | 72 | 64 | 46 | 69 | ndField.□ThirdFi |
| 00001B30 | 65 | 6C | 64 | 00 | 02 | 04 | 38 | 4C | 86 | B3 | 73 | 42 | C5 | 58 | 54 | 5B | eld. □8L▌³sBÅXT[ |
| 00001B40 | 8B | 36 | 08 | 34 | 34 | 71 | 17 | 00 | 4D | 79 | 2E | 54 | 4C | 47 | 2E | 50 | ▌6□44q□.My.TLG.P |
| | | | | | | | | | | | 00 | 01 | 00 | 00 | rovider.Info.□.. |
| | | | | | | | | | | | B7 | 7F | FD | B9 | RSDS ▌□Ë¬PÒL·▌ý¹ |

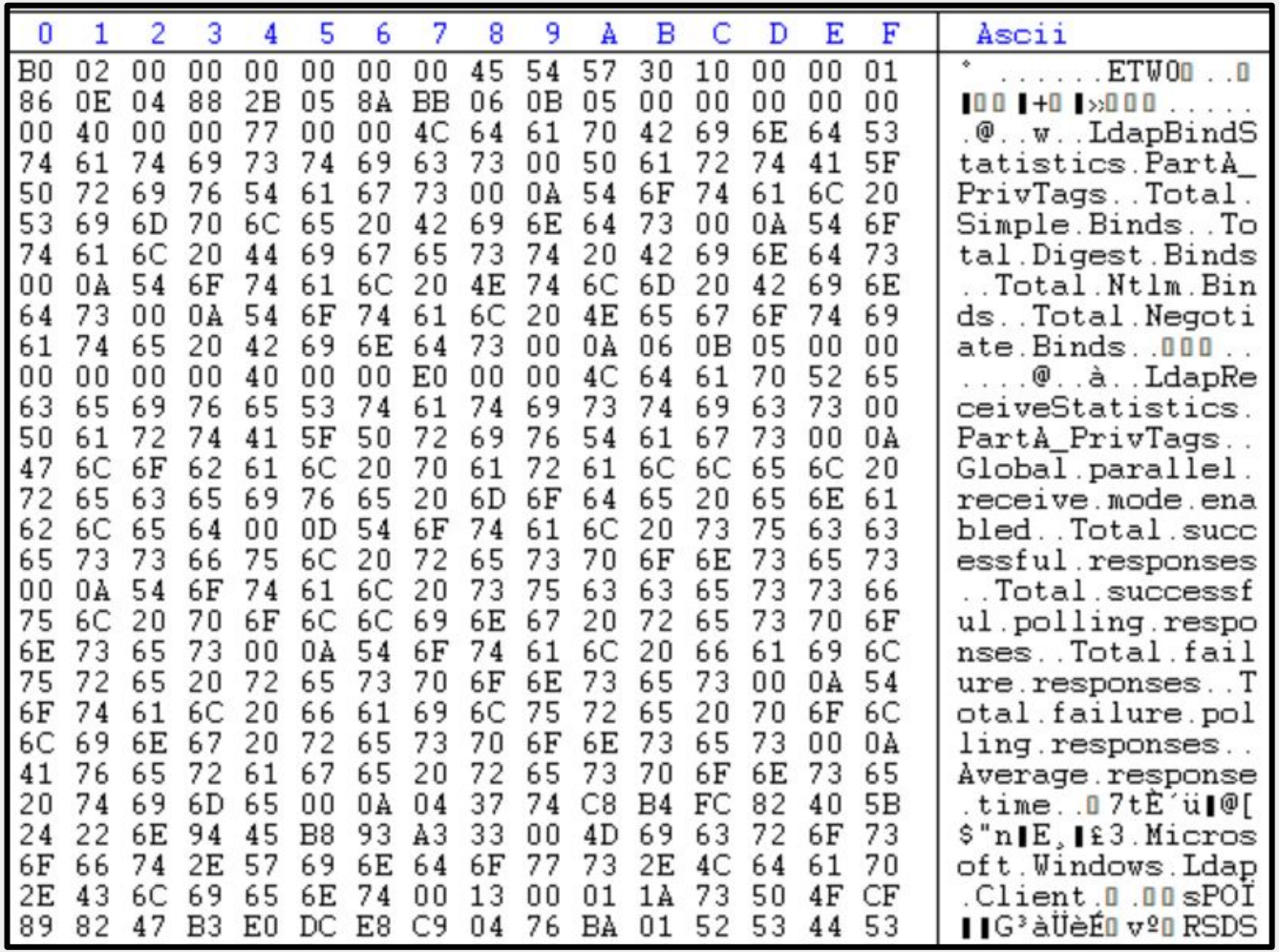
**Typically, embedded in *.rdata* section**

## What the metadata typically contains

- ✓ **Provider name** / **Provider GUID** / Provider Group GUID
- ✓ **Event name** / Level / Keyword / Opcode / Channel
  - **Field names and types**
- ✓ Header / BlobType Information (provider/event/end of metadata)

Note: Details of the metadata blob structure are documented in TraceLoggingProvider.h and TLGMetadataParser.psm1

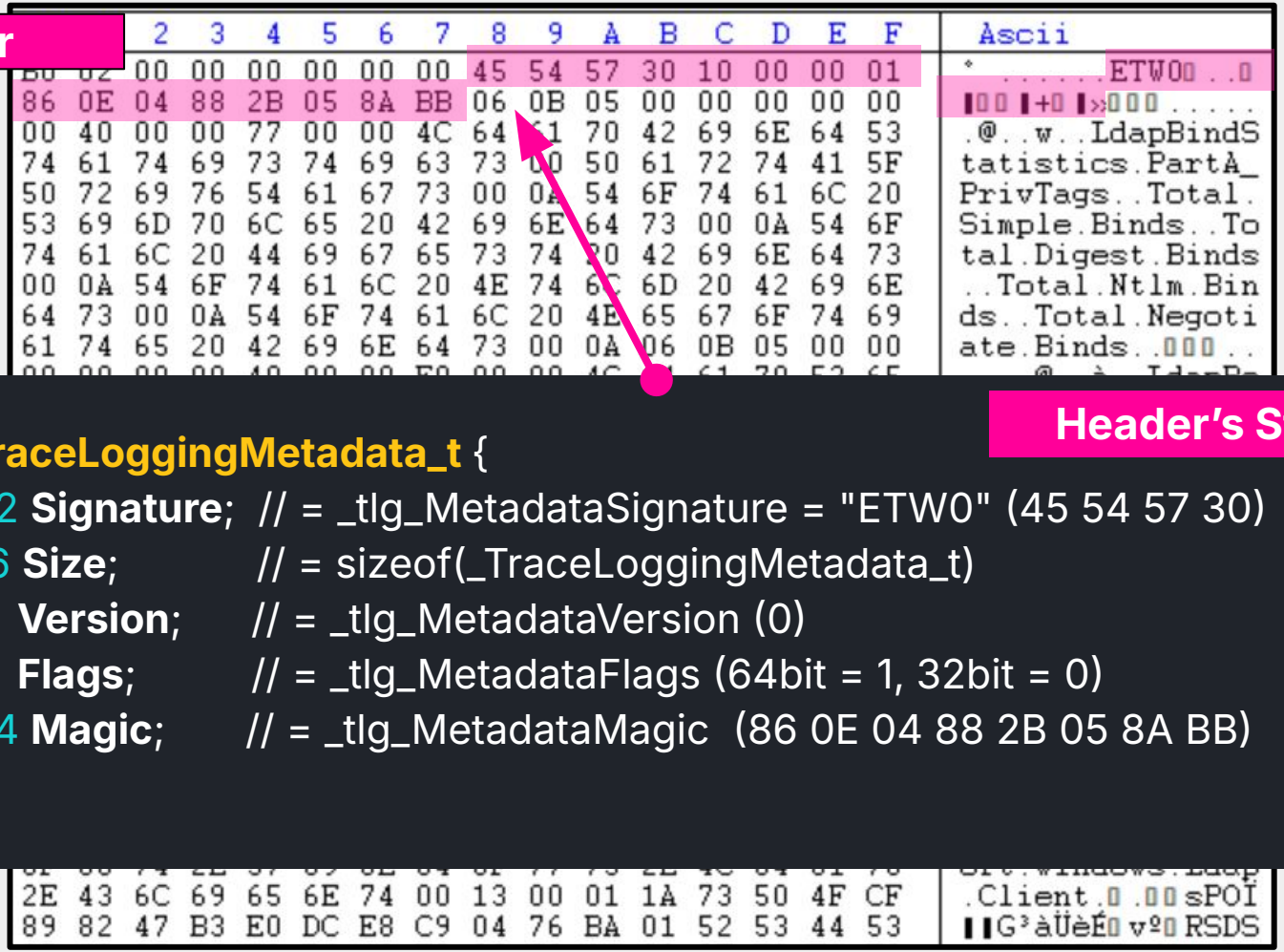
# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)



| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | Ascii |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0 | 02 | 00 | 00 | 00 | 00 | 00 | 00 | 45 | 54 | 57 | 30 | 10 | 00 | 00 | 01 | °.....ETW0□..□ |
| 86 | 0E | 04 | 88 | 2B | 05 | 8A | BB | 06 | 0B | 05 | 00 | 00 | 00 | 00 | 00 | ▮□□▮+□▮»□□□..... |
| 00 | 40 | 00 | 00 | 77 | 00 | 00 | 4C | 64 | 61 | 70 | 42 | 69 | 6E | 64 | 53 | .@..w..LdapBindS |
| 74 | 61 | 74 | 69 | 73 | 74 | 69 | 63 | 73 | 00 | 50 | 61 | 72 | 74 | 41 | 5F | tatistics.PartA_ |
| 50 | 72 | 69 | 76 | 54 | 61 | 67 | 73 | 00 | 0A | 54 | 6F | 74 | 61 | 6C | 20 | PrivTags..Total. |
| 53 | 69 | 6D | 70 | 6C | 65 | 20 | 42 | 69 | 6E | 64 | 73 | 00 | 0A | 54 | 6F | Simple.Binds..To |
| 74 | 61 | 6C | 20 | 44 | 69 | 67 | 65 | 73 | 74 | 20 | 42 | 69 | 6E | 64 | 73 | tal.Digest.Binds |
| 00 | 0A | 54 | 6F | 74 | 61 | 6C | 20 | 4E | 74 | 6C | 6D | 20 | 42 | 69 | 6E | ..Total.Ntlm.Bin |
| 64 | 73 | 00 | 0A | 54 | 6F | 74 | 61 | 6C | 20 | 4E | 65 | 67 | 6F | 74 | 69 | ds..Total.Negoti |
| 61 | 74 | 65 | 20 | 42 | 69 | 6E | 64 | 73 | 00 | 0A | 06 | 0B | 05 | 00 | 00 | ate.Binds..□□□.. |
| 00 | 00 | 00 | 00 | 40 | 00 | 00 | E0 | 00 | 00 | 4C | 64 | 61 | 70 | 52 | 65 | ....@..à..LdapRe |
| 63 | 65 | 69 | 76 | 65 | 53 | 74 | 61 | 74 | 69 | 73 | 74 | 69 | 63 | 73 | 00 | ceiveStatistics. |
| 50 | 61 | 72 | 74 | 41 | 5F | 50 | 72 | 69 | 76 | 54 | 61 | 67 | 73 | 00 | 0A | PartA_PrivTags.. |
| 47 | 6C | 6F | 62 | 61 | 6C | 20 | 70 | 61 | 72 | 61 | 6C | 6C | 65 | 6C | 20 | Global.parallel. |
| 72 | 65 | 63 | 65 | 69 | 76 | 65 | 20 | 6D | 6F | 64 | 65 | 20 | 65 | 6E | 61 | receive.mode.ena |
| 62 | 6C | 65 | 64 | 00 | 0D | 54 | 6F | 74 | 61 | 6C | 20 | 73 | 75 | 63 | 63 | bled..Total.succ |
| 65 | 73 | 73 | 66 | 75 | 6C | 20 | 72 | 65 | 73 | 70 | 6F | 6E | 73 | 65 | 73 | essful.responses |
| 00 | 0A | 54 | 6F | 74 | 61 | 6C | 20 | 73 | 75 | 63 | 63 | 65 | 73 | 73 | 66 | ..Total.successf |
| 75 | 6C | 20 | 70 | 6F | 6C | 6C | 69 | 6E | 67 | 20 | 72 | 65 | 73 | 70 | 6F | ul.polling.respo |
| 6E | 73 | 65 | 73 | 00 | 0A | 54 | 6F | 74 | 61 | 6C | 20 | 66 | 61 | 69 | 6C | nses..Total.fail |
| 75 | 72 | 65 | 20 | 72 | 65 | 73 | 70 | 6F | 6E | 73 | 65 | 73 | 00 | 0A | 54 | ure.responses..T |
| 6F | 74 | 61 | 6C | 20 | 66 | 61 | 69 | 6C | 75 | 72 | 65 | 20 | 70 | 6F | 6C | otal.failure.pol |
| 6C | 69 | 6E | 67 | 20 | 72 | 65 | 73 | 70 | 6F | 6E | 73 | 65 | 73 | 00 | 0A | ling.responses.. |
| 41 | 76 | 65 | 72 | 61 | 67 | 65 | 20 | 72 | 65 | 73 | 70 | 6F | 6E | 73 | 65 | Average.response |
| 20 | 74 | 69 | 6D | 65 | 00 | 0A | 04 | 37 | 74 | C8 | B4 | FC | 82 | 40 | 5B | .time..□7tÈ´ü▮@[ |
| 24 | 22 | 6E | 94 | 45 | B8 | 93 | A3 | 33 | 00 | 4D | 69 | 63 | 72 | 6F | 73 | $"n▮E¸▮£3.Micros |
| 6F | 66 | 74 | 2E | 57 | 69 | 6E | 64 | 6F | 77 | 73 | 2E | 4C | 64 | 61 | 70 | oft.Windows.Ldap |
| 2E | 43 | 6C | 69 | 65 | 6E | 74 | 00 | 13 | 00 | 01 | 1A | 73 | 50 | 4F | CF | .Client.□.□□sPOÏ |
| 89 | 82 | 47 | B3 | E0 | DC | E8 | C9 | 04 | 76 | BA | 01 | 52 | 53 | 44 | 53 | ▮▮G³àÜèÉ□v º□RSDS |

# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)

**Header**



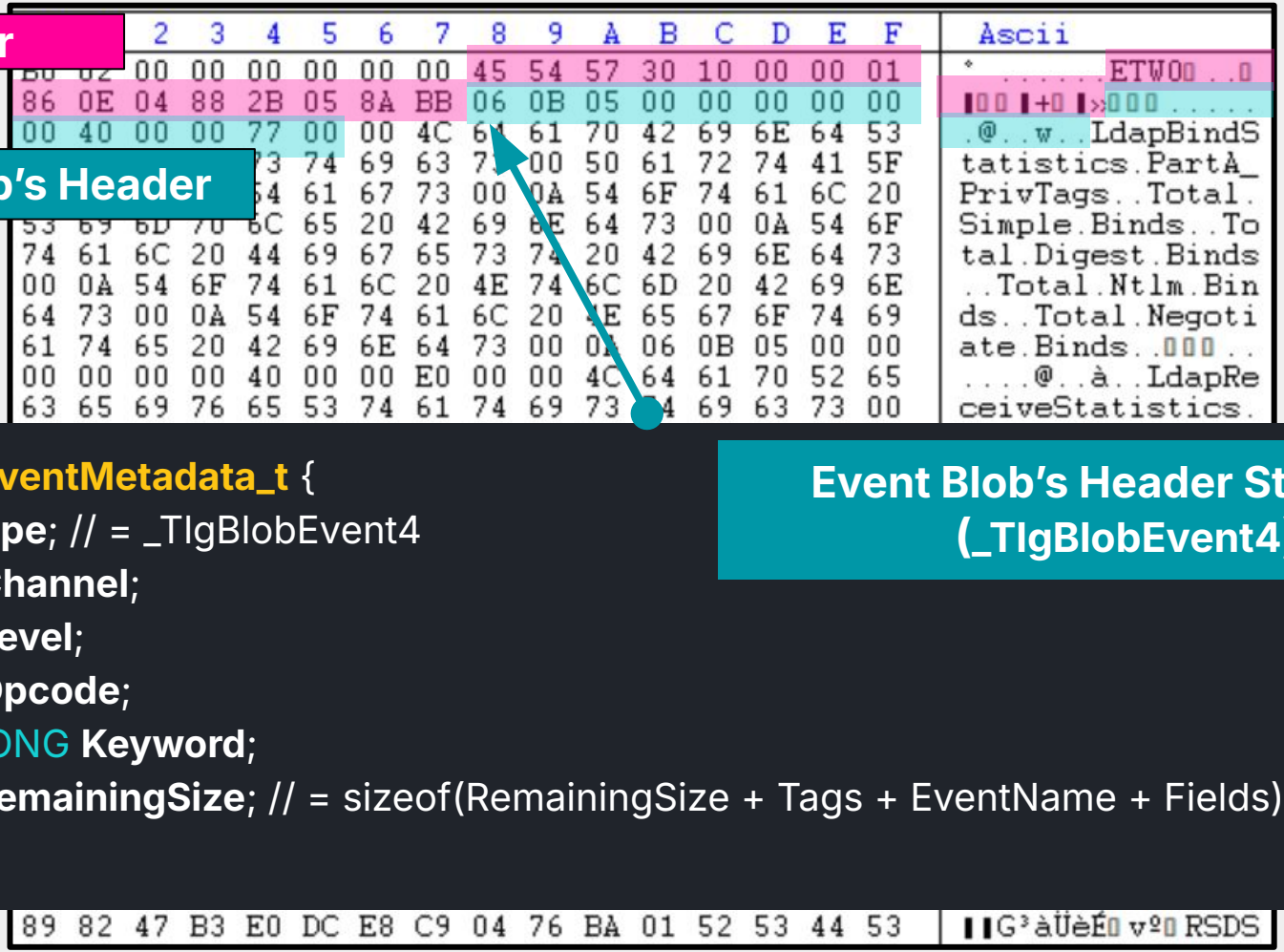

**Header's Structure**

```
struct _TraceLoggingMetadata_t {
   UINT32 Signature;  // = _tlg_MetadataSignature = "ETW0" (45 54 57 30)
   UINT16 Size;       // = sizeof(_TraceLoggingMetadata_t)
   UINT8  Version;    // = _tlg_MetadataVersion (0)
   UINT8  Flags;      // = _tlg_MetadataFlags (64bit = 1, 32bit = 0)
   UINT64 Magic;      // = _tlg_MetadataMagic  (86 0E 04 88 2B 05 8A BB)
};
```

# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)

**Header**

**Event Blob's Header**




**Event Blob's Header Structure (_TlgBlobEvent4)**

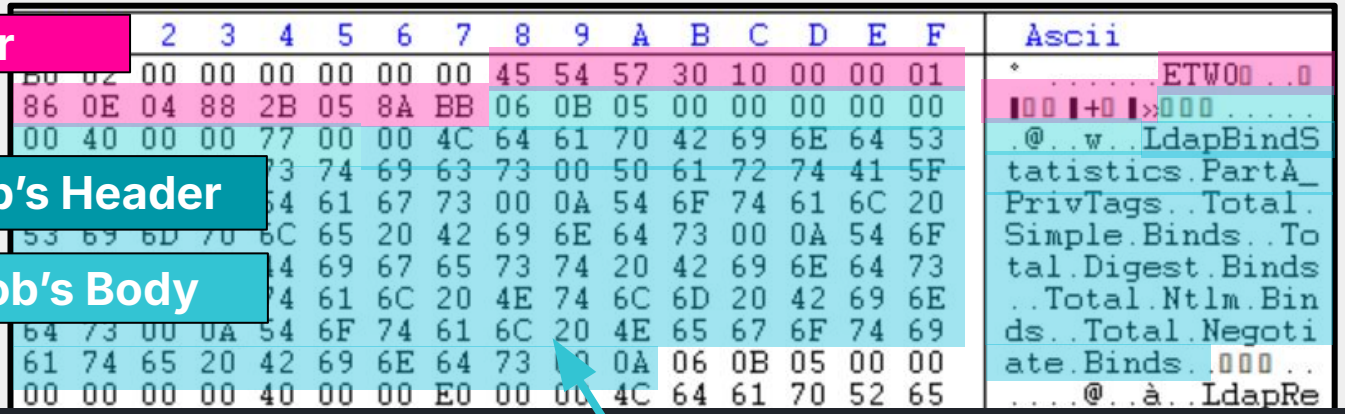
```
struct _tlgEventMetadata_t {
    UINT8 Type; // = _TlgBlobEvent4
    UCHAR Channel;
    UCHAR Level;
    UCHAR Opcode;
    ULONGLONG Keyword;
    UINT16 RemainingSize; // = sizeof(RemainingSize + Tags + EventName + Fields)
};
```

# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)

**Header**

**Event Blob's Header**

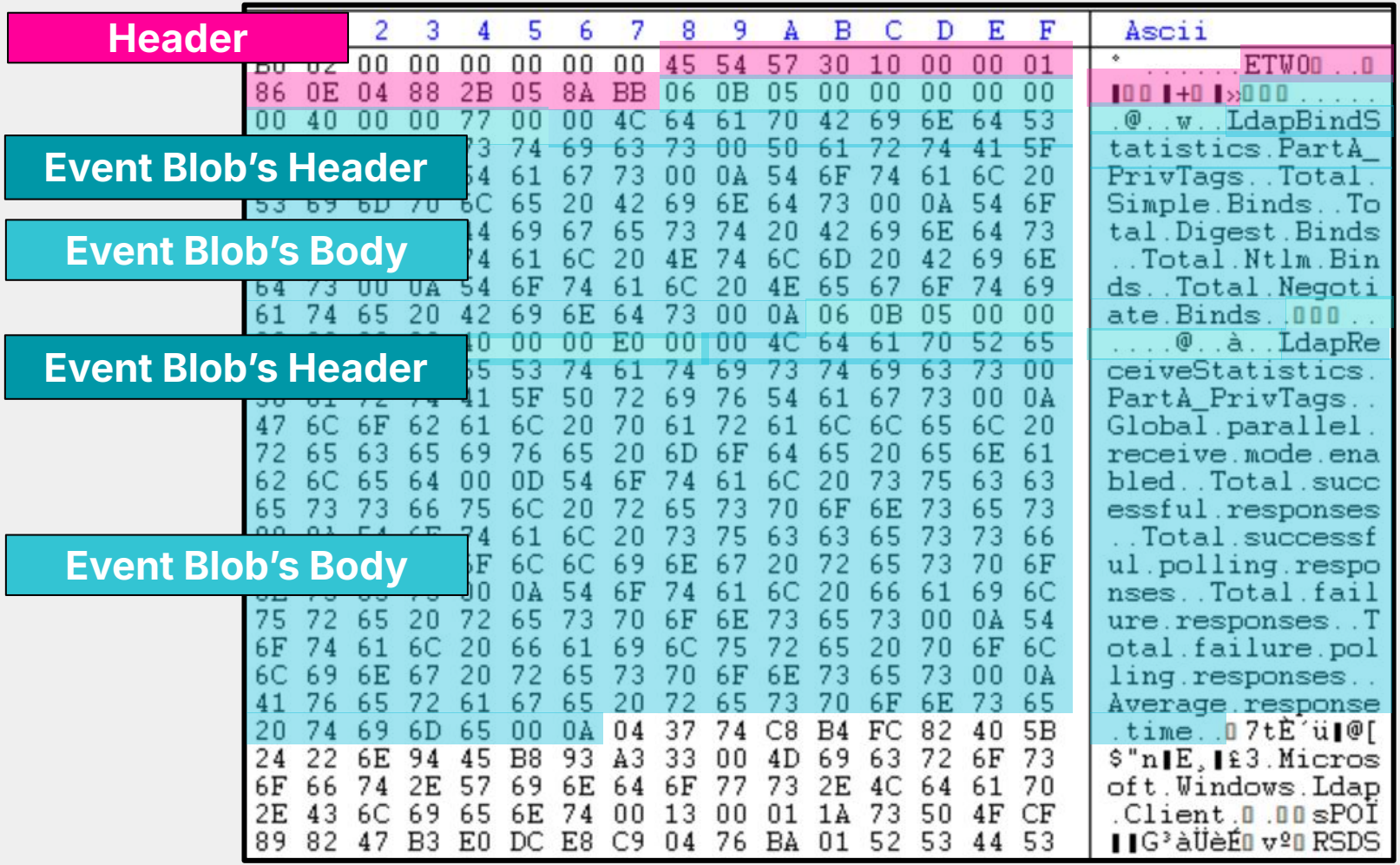
**Event Blob's Body**




**Event Blob's Body (_TlgBlobEvent4)**

```
UINT8 Tags[];
char EventName[sizeof("eventName")]; // UTF-8 nul-terminated event name
for each field {
    char FieldName[sizeof("fieldName")]; // UTF-8 nul-terminated field name
    UINT8 InType; // TlgIn_t
    UINT8 OutType; // TlgOut_t
    UINT8 Tags[];
    UINT16 ValueCount;
    UINT16 TypeInfoSize;
    char TypeInfo[TypeInfoSize];
}
```

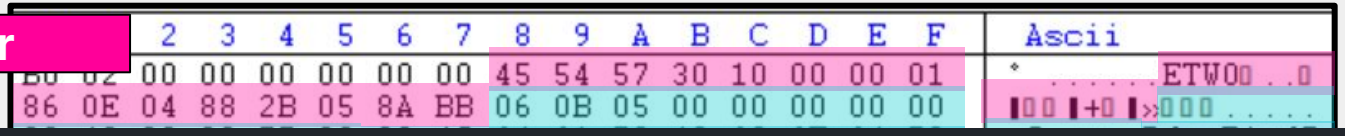
# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)




```
   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  Ascii
00 00 00 00 00 00 45 54 57 30 10 00 00 01  ......ETW0..
86 0E 04 88 2B 05 8A BB 06 0B 05 00 00 00 00 00  ....+.»..... 
00 40 00 00 77 00 00 4C 64 61 70 42 69 6E 64 53  .@..w..LdapBindS
   73 74 69 63 73 00 50 61 72 74 41 5F           tatistics.PartA_
   54 61 67 73 00 0A 54 6F 74 61 6C 20           PrivTags..Total.
   65 20 42 69 6E 64 73 00 0A 54 6F              Simple.Binds..To
   44 69 67 65 73 74 20 42 69 6E 64 73           tal.Digest.Binds
   74 61 6C 20 4E 74 6C 6D 20 42 69 6E           ..Total.Ntlm.Bin
64 73 00 0A 54 6F 74 61 6C 20 4E 65 67 6F 74 69  ds..Total.Negoti
61 74 65 20 42 69 6E 64 73 00 0A 06 0B 05 00 00  ate.Binds.....
   00 00 00 E0 00 00 4C 64 61 70 52 65           ....@..à..LdapRe
   65 53 74 61 74 69 73 74 69 63 73 00           ceiveStatistics.
   41 5F 50 72 69 76 54 61 67 73 00 0A           PartA_PrivTags..
47 6C 6F 62 61 6C 20 70 61 72 61 6C 6C 65 6C 20  Global.parallel.
72 65 63 65 69 76 65 20 6D 6F 64 65 20 65 6E 61  receive.mode.ena
62 6C 65 64 00 0D 54 6F 74 61 6C 20 73 75 63 63  bled..Total.succ
65 73 73 66 75 6C 20 72 65 73 70 6F 6E 73 65 73  essful.responses
   74 61 6C 20 73 75 63 63 65 73 73 66           ..Total.successf
   6F 6C 6C 69 6E 67 20 72 65 73 70 6F           ul.polling.respo
   00 0A 54 6F 74 61 6C 20 66 61 69 6C           nses..Total.fail
75 72 65 20 72 65 73 70 6F 6E 73 65 73 00 0A 54  ure.responses..T
6F 74 61 6C 20 66 61 69 6C 75 72 65 20 70 6F 6C  otal.failure.pol
6C 69 6E 67 20 72 65 73 70 6F 6E 73 65 73 00 0A  ling.responses..
41 76 65 72 61 67 65 20 72 65 73 70 6F 6E 73 65  Average.response
20 74 69 6D 65 00 0A 04 37 74 C8 B4 FC 82 40 5B  .time...7tÈ´ü.@[
24 22 6E 94 45 B8 93 A3 33 00 4D 69 63 72 6F 73  $"n.E¸.£3.Micros
6F 66 74 2E 57 69 6E 64 6F 77 73 2E 4C 64 61 70  oft.Windows.Ldap
2E 43 6C 69 65 6E 74 00 13 00 01 1A 73 50 4F CF  .Client.....sPOÏ
89 82 47 B3 E0 DC E8 C9 04 76 BA 01 52 53 44 53  ..G³àÜèÉ.vº.RSDS
```

# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)

**Header**

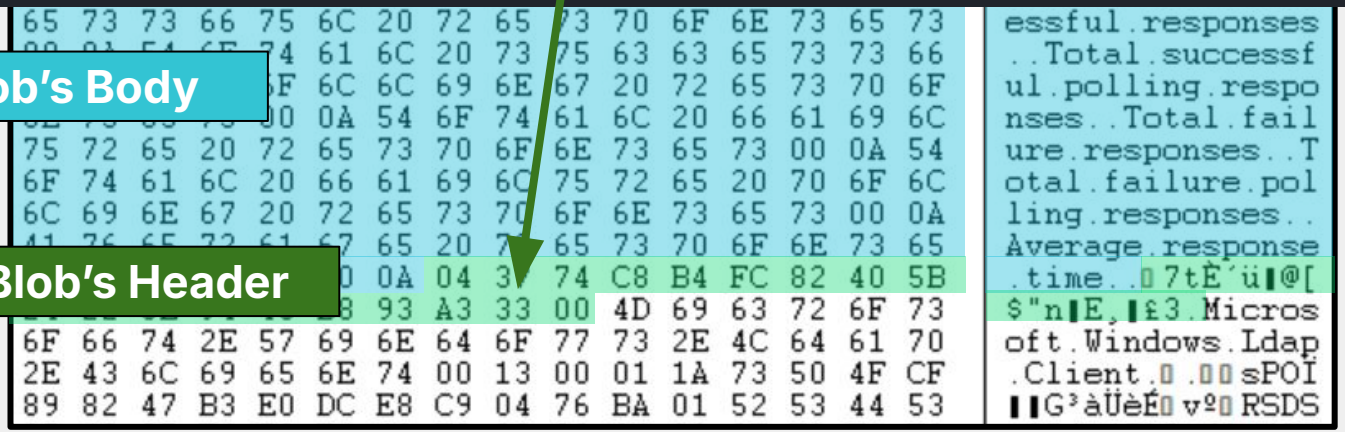



```
struct _tlgProviderMetadata_t {
    UINT8 Type; // = _TlgBlobProvider3
    GUID ProviderId;
    UINT16 RemainingSize; // = sizeof(RemainingSize + ProviderName)
};
```

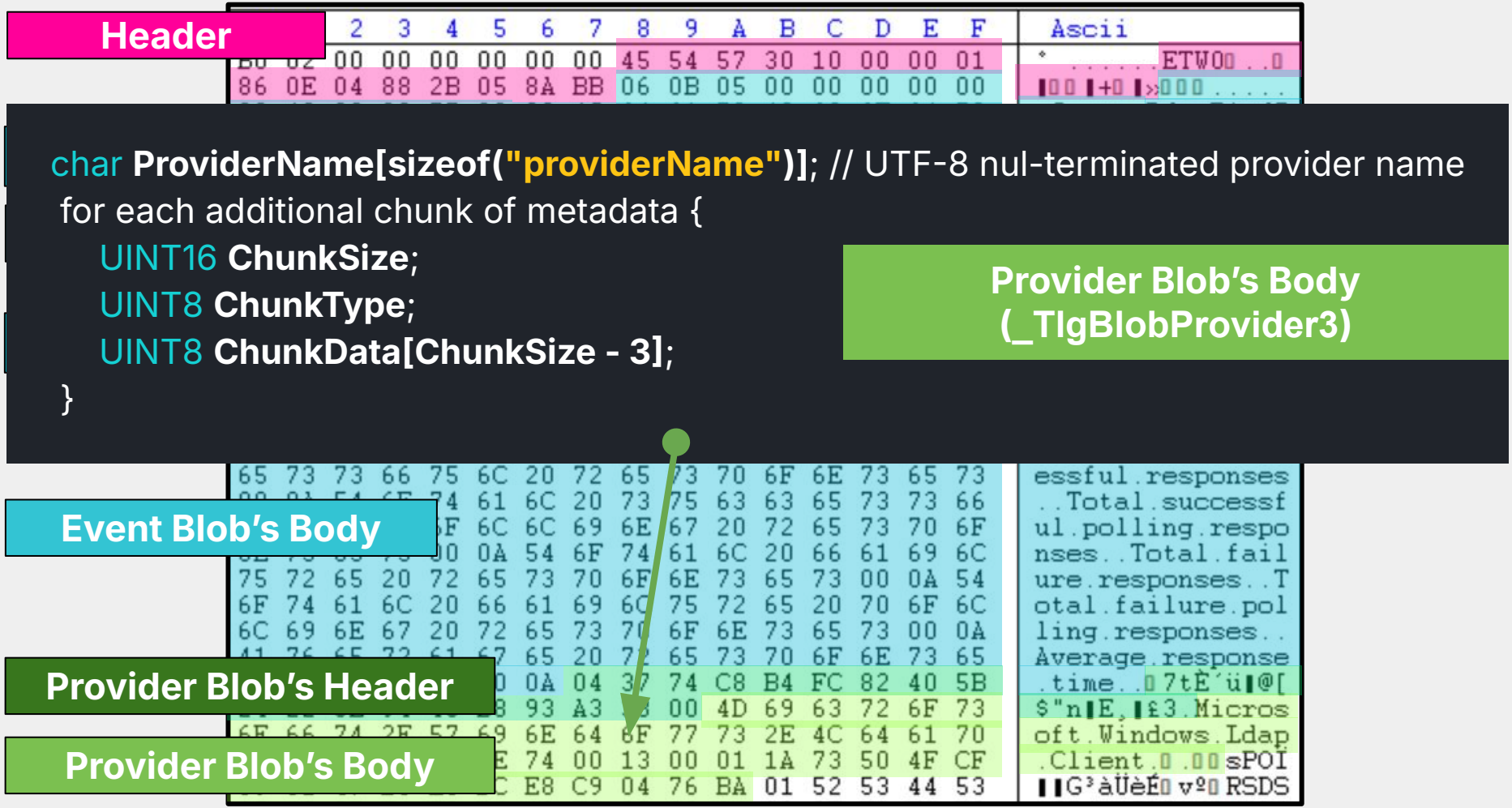
**Provider Blob's Header (_TlgBlobProvider3)**

**Event Blob's Body**

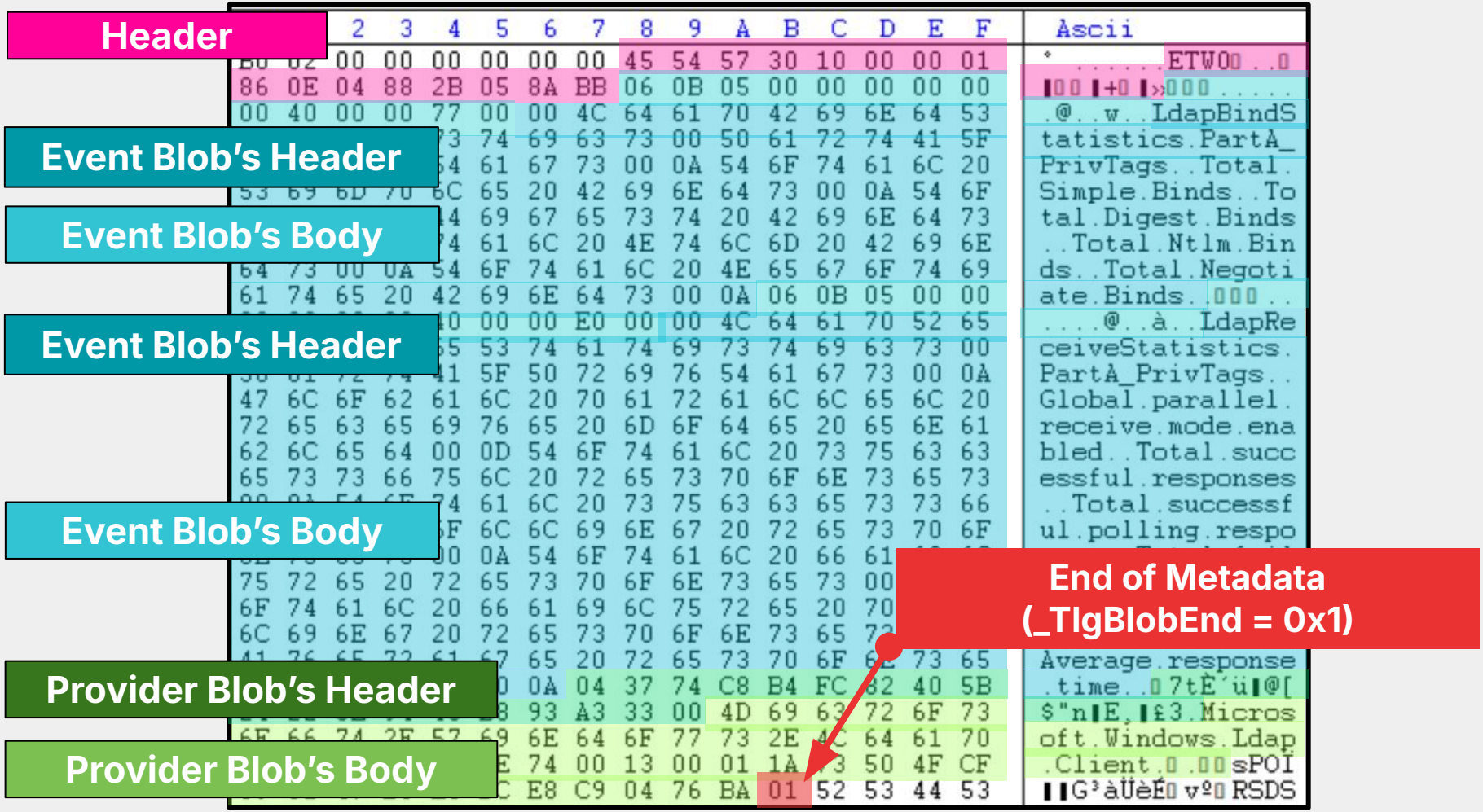
**Provider Blob's Header**



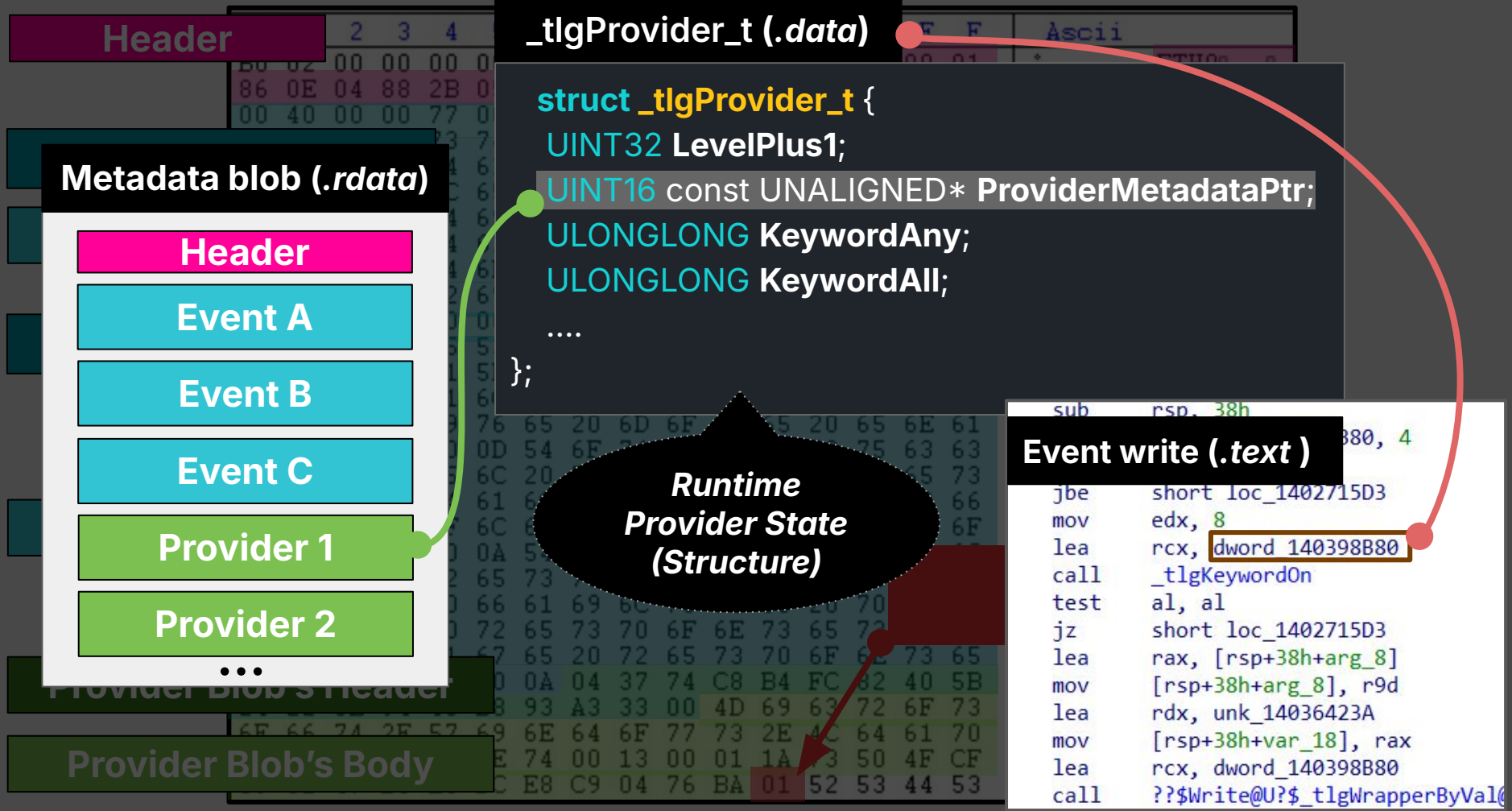
# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)



# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)

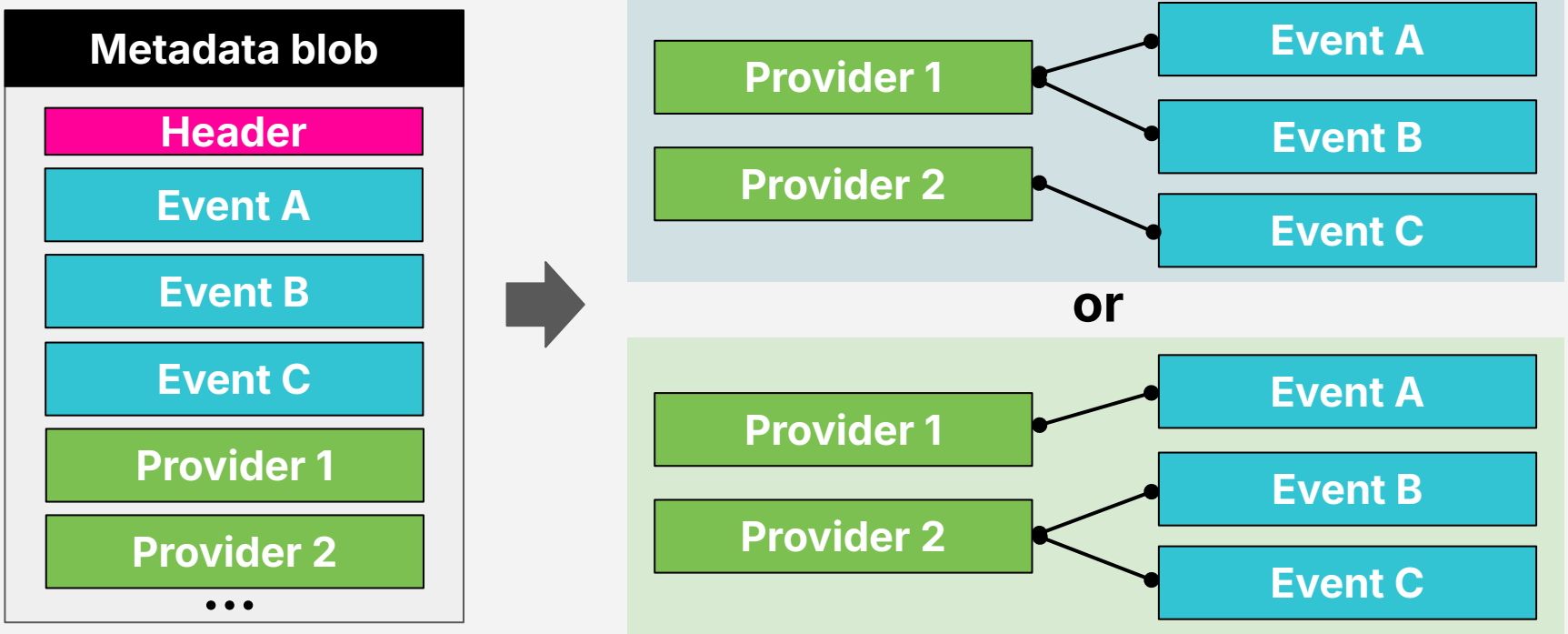


# TraceLogging Provider Metadata (Microsoft.Windows.Ldap.Client, Wldap32.dll)



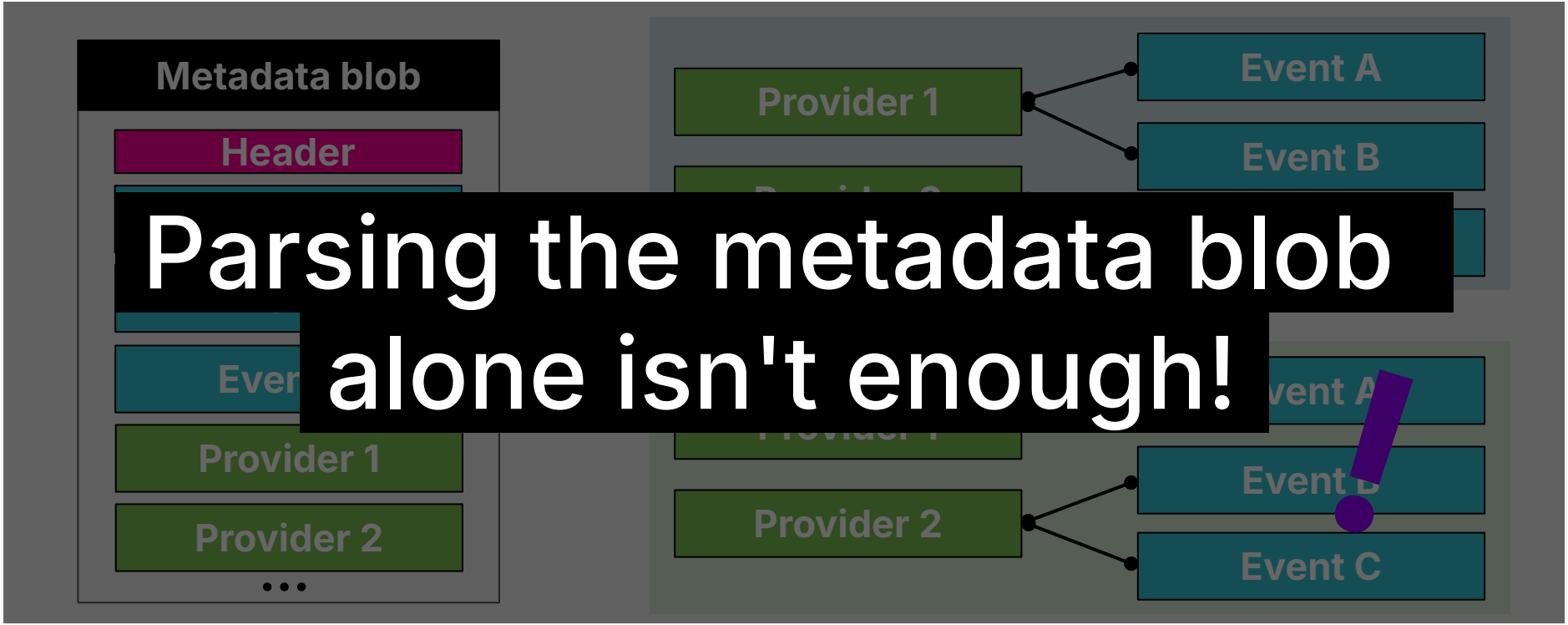
# The Missing Link: Event-to-Provider Ownership

**Metadata blob** has **no ownership markers between events and providers**



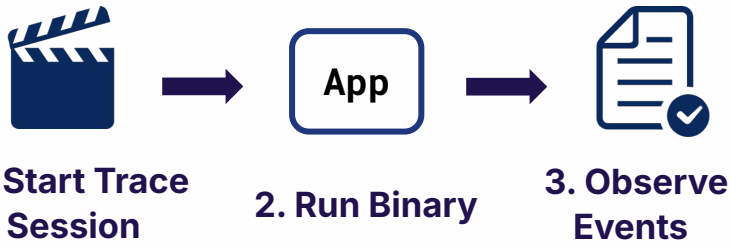
# The Missing Link: Event-to-Provider Ownership

**Metadata blob** has **no ownership markers between events and providers**



# Mapping Events to Providers: Two Approaches

## Capture at Runtime




✅ Easy to set up

❌ Events that don't fire during execution are invisible

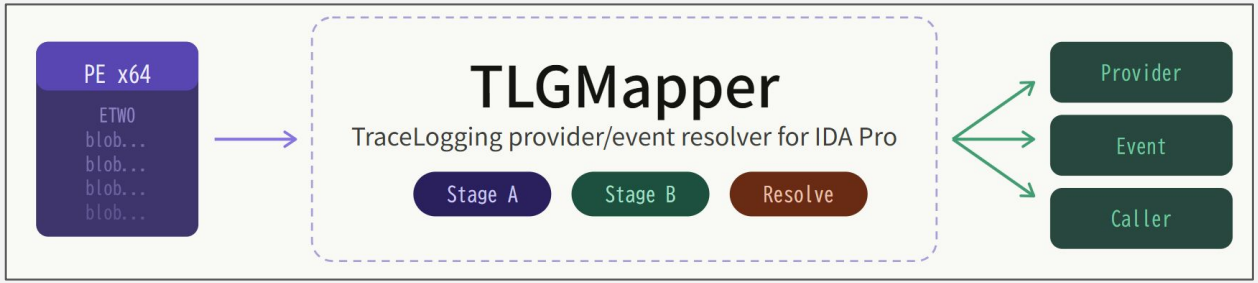
## Static Binary Analysis

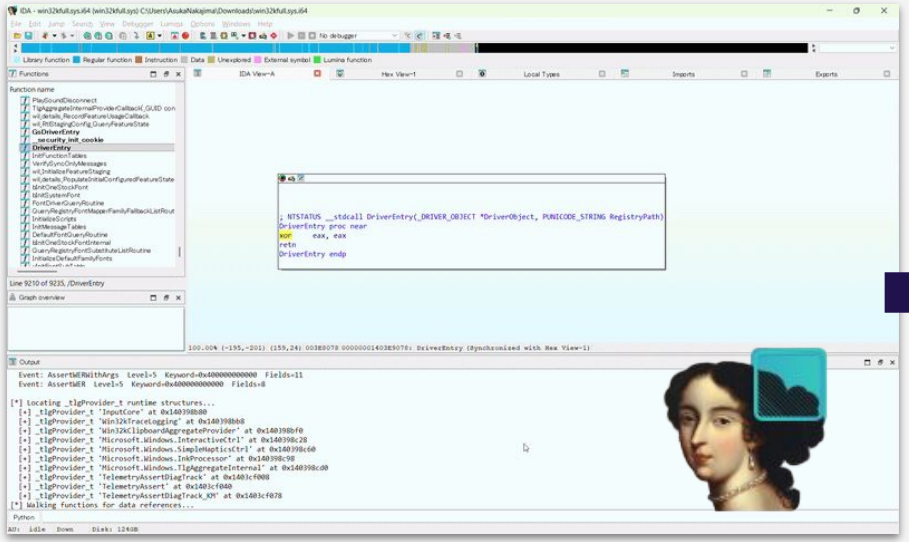

✅ No execution needed

✅ Theoretically covers all events

❌ Manual analysis is time consuming and labor-intensive

# TLGMapper: Automating the Static Analysis



| Provider | Event Name | |
|---|---|---|
| InputCore | ApplyWindowAction::ApplyEnd | ?Ap |
| InputCore | ApplyWindowAction::ApplyFailed | ?Ap |
| InputCore | ApplyWindowAction::ApplyStart | ?Ap |
| InputCore | Cursor::Api::ChangeCursorApiMode | ?Ch |
| InputCore | Cursor::Api::ForceSetCurrentCursorShape | ?Fo |
| InputCore | Cursor::Api::HidePointer | ?Hi |
| InputCore | Cursor::Api::MovePointer | ?Mo |
| InputCore | Cursor::Api::MovePointerAsync | ?Mo |
| InputCore | Cursor::Api::RequestCursorSynchronizatio… | ?Re |
| InputCore | Cursor::Api::SetCursorSynchronizationMode | ?Se |
| InputCore | Cursor::Api::SetPointerShape | ?Se |

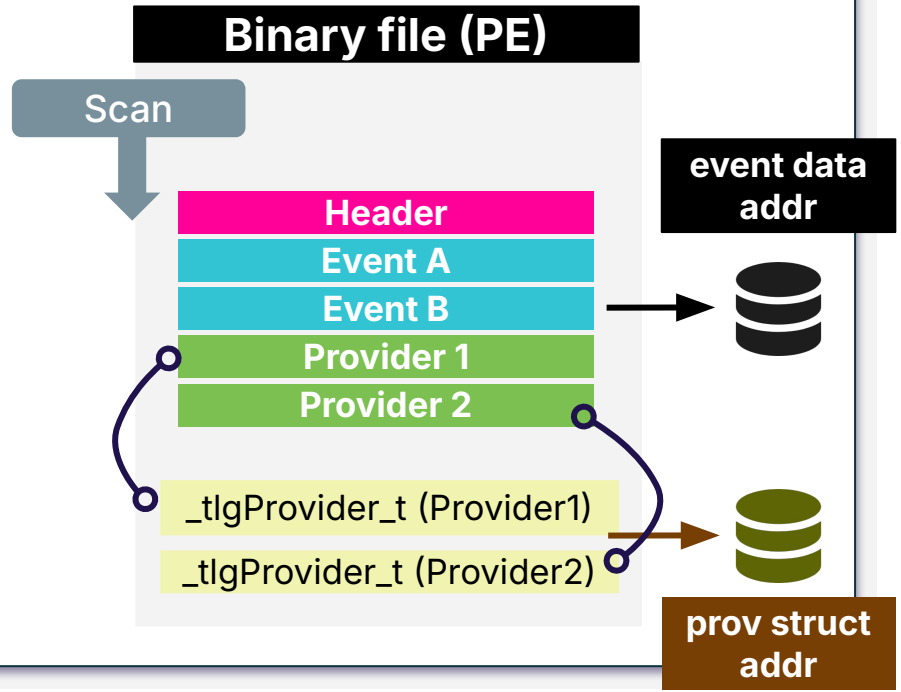
# TLGMapper: How it Works (Stage A)

**Stage A : Metadata Extraction** | **Stage B: Provider-Event Resolution**

**A1**: Header signature/magic scanning

**A2**: Metadata blob parsing (mainly to collect event data addr and provider info)

**A3**: Resolve provider struct (_tlgProvider_t) locations and store their addresses



# TLGMapper: How it Works (Stage B)

**Stage A : Metadata Extraction** → **Stage B: Provider-Event Resolution**

## B1: Sweep all functions for data refs (+call refs)

**Function**

```
        rsp, 38h
        cs:dword_140398B80, 4
mov     r9d, ecx
jbe     short loc_1402715D3
mov     edx, 8
lea     rcx, dword_140398B80
call    _tlgKeywordOn
test    al, al
jz      short loc_1402715D3
lea     rax, [rsp+38h+arg_8]
mov     [rsp+38h+arg_8], r9d
lea     rdx, unk_14036423A
mov     [rsp+38h+var_18], rax
lea     rcx, dword_140398B80
call    ??$Write@U?$_tlgWrapperByVal(
```

**event data addr**

0x14036423A…(EventA)
0x14036419A…(EventB)

**prov struct addr**

0x140398b80… (Provider1)
0x140398cc0… (Provider2)

# TLGMapper: How it Works (Stage B)

**Stage A : Metadata Extraction** | **Stage B: Provider-Event Resolution**

## B1: Sweep all functions for data refs (+call refs)

**Function**

```
        rsp, 38h
        cs:dword_140398B80, 4
mov     r9d, ecx
jbe     short loc_1402715D3
mov     edx, 8
lea     rcx, dword_140398B80
call    _tlgKeywordOn
test    al, al
jz      short loc_1402715D3
lea     rax, [rsp+38h+arg_8]
mov     [rsp+38h+arg_8], r9d
lea     rdx, unk_14036423A
mov     [rsp+38h+var_18], rax
lea     rcx, dword_140398B80
call    ??$Write@U?$_tlgWrapperByVal(
```

*Found event refs !*

**event data addr**

0x14036423A…(EventA)
0x14036419A…(EventB)

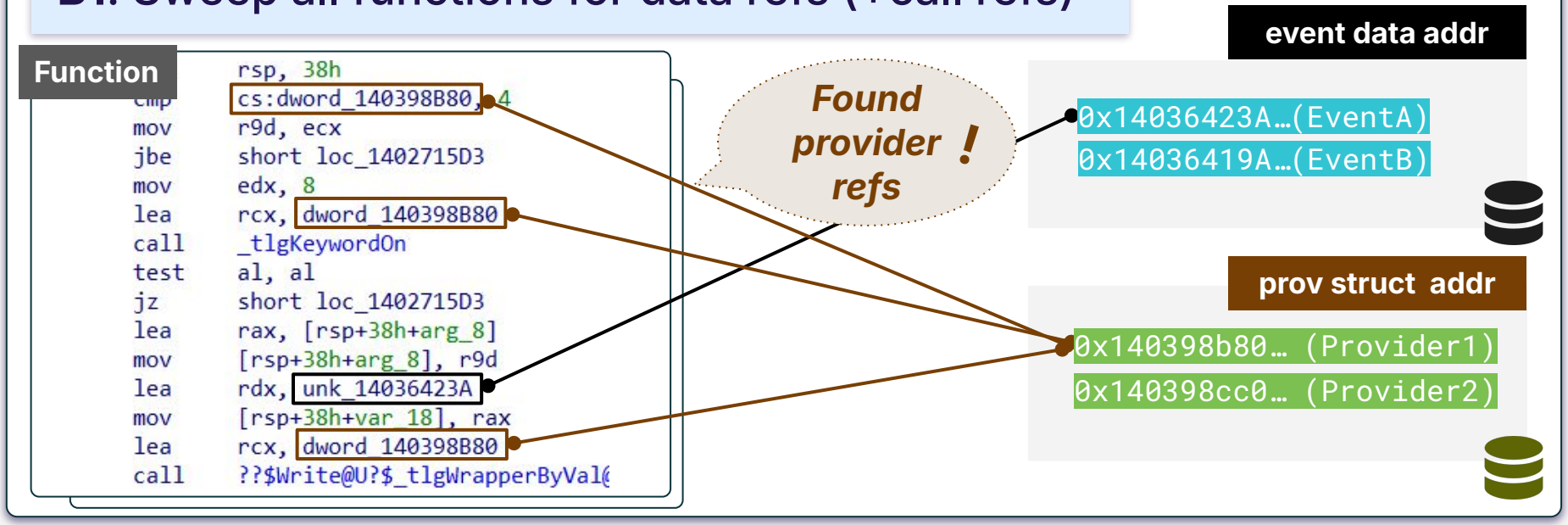
**prov struct addr**

0x140398b80… (Provider1)
0x140398cc0… (Provider2)

# TLGMapper: How it Works (Stage B)

**Stage A : Metadata Extraction** | **Stage B: Provider-Event Resolution**

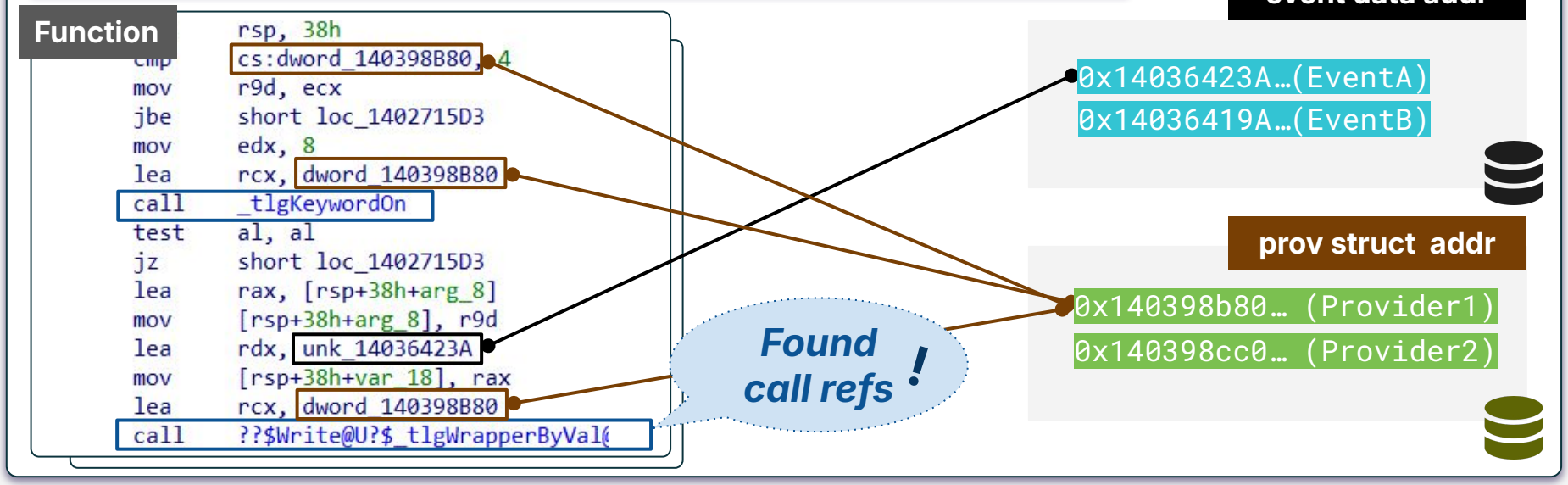
## B1: Sweep all functions for data refs (+call refs)



# TLGMapper: How it Works (Stage B)

**Stage A : Metadata Extraction** | **Stage B: Provider-Event Resolution**

## B1: Sweep all functions for data refs (+call refs)



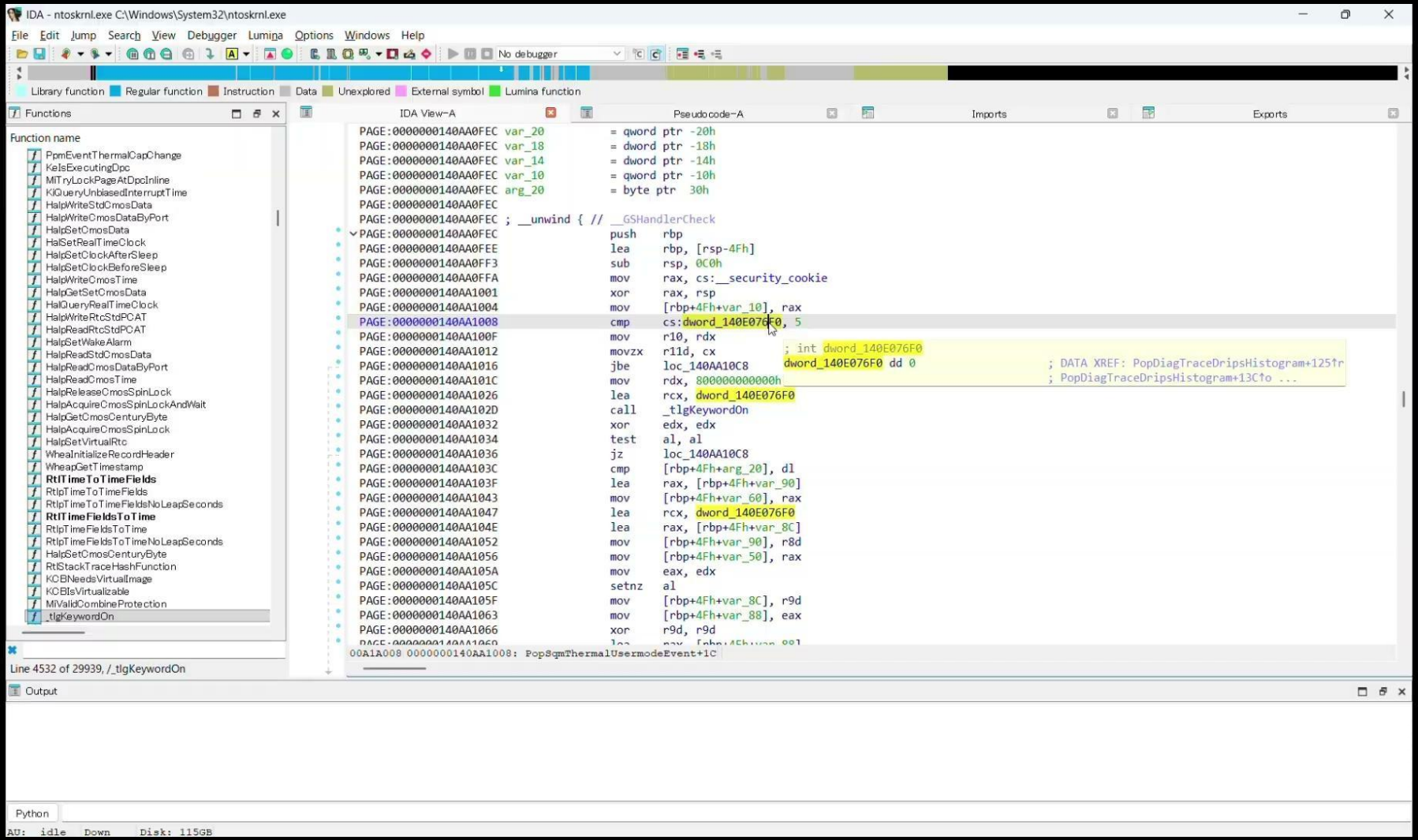
# TLGMapper: How it Works (Stage B)

Stage A : Metadata Extraction → **Stage B: Provider-Event Resolution**

## B2: Resolution priority chain (per event) — first match wins, then stop

| | |
|---|---|
| 🥇 **1** *Single Provider* | Binary contains only one provider |
| ↓ | |
| 🥈 **2** *Direct-Preceding* | Provider ref appears before the event ref in the same function |
| ↓ | |
| 🥉 **3** *Direct-Nearest* | Closest provider ref by address in the same function |
| ↓ | |
| 🔗 **4** *CallGraph-dN* | DFS through callees (max depth: 3), searching backward first |
| ↓ | |
| ❓ **5** *Unknown* | No provider could be resolved |

# DEMO Time! ✨


IDA - ntoskrnl.exe C:\Windows\System32\ntoskrnl.exe
File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
Function name
PpmEventThermalCapChange
KeIsExecutingDpc
MiTryLockPageAtDpcInline
KiQueryUnbiasedInterruptTime
HalpWriteStdCmosData
HalpWriteCmosDataByPort
HalpSetCmosData
HalSetRealTimeClock
HalpSetClockAfterSleep
HalpSetClockBeforeSleep
HalpWriteCmosTime
HalpGetSetCmosData
HalQueryRealTimeClock
HalpWriteRtcStdPCAT
HalpReadRtcStdPCAT
HalpSetWakeAlarm
HalpReadStdCmosData
HalpReadCmosDataByPort
HalpReadCmosTime
HalpReleaseCmosSpinLock
HalpAcquireCmosSpinLockAndWait
HalpGetCmosCenturyByte
HalpAcquireCmosSpinLock
HalpSetVirtualRtc
WheaInitializeRecordHeader
WheapGetTimestamp
RtlTimeToTimeFields
RtlpTimeToTimeFields
RtlpTimeToTimeFieldsNoLeapSeconds
RtlTimeFieldsToTime
RtlpTimeFieldsToTime
RtlpTimeFieldsToTimeNoLeapSeconds
HalpSetCmosCenturyByte
RtlStackTraceHashFunction
KCBNeedsVirtualImage
KCBIsVirtualizable
MiValidCombineProtection
_tlgKeywordOn
Line 4532 of 29939, /_tlgKeywordOn
IDA View-A
Pseudocode-A
Imports
Exports
PAGE:0000000140AA0FEC var_20 = qword ptr -20h
PAGE:0000000140AA0FEC var_18 = dword ptr -18h
PAGE:0000000140AA0FEC var_14 = dword ptr -14h
PAGE:0000000140AA0FEC var_10 = qword ptr -10h
PAGE:0000000140AA0FEC arg_20 = byte ptr 30h
PAGE:0000000140AA0FEC
PAGE:0000000140AA0FEC ; __unwind { // __GSHandlerCheck
PAGE:0000000140AA0FEC push rbp
PAGE:0000000140AA0FEE lea rbp, [rsp-4Fh]
PAGE:0000000140AA0FF3 sub rsp, 0C0h
PAGE:0000000140AA0FFA mov rax, cs:__security_cookie
PAGE:0000000140AA1001 xor rax, rsp
PAGE:0000000140AA1004 mov [rbp+4Fh+var_10], rax
PAGE:0000000140AA1008 cmp cs:dword_140E076F0, 5
PAGE:0000000140AA100F mov r10, rdx
PAGE:0000000140AA1012 movzx r11d, cx
PAGE:0000000140AA1016 jbe loc_140AA10C8
PAGE:0000000140AA101C mov rdx, 800000000000h
PAGE:0000000140AA1026 lea rcx, dword_140E076F0
PAGE:0000000140AA102D call _tlgKeywordOn
PAGE:0000000140AA1032 xor edx, edx
PAGE:0000000140AA1034 test al, al
PAGE:0000000140AA1036 jz loc_140AA10C8
PAGE:0000000140AA103C cmp [rbp+4Fh+arg_20], dl
PAGE:0000000140AA103F lea rax, [rbp+4Fh+var_90]
PAGE:0000000140AA1043 mov [rbp+4Fh+var_60], rax
PAGE:0000000140AA1047 lea rcx, dword_140E076F0
PAGE:0000000140AA104E lea rax, [rbp+4Fh+var_8C]
PAGE:0000000140AA1052 mov [rbp+4Fh+var_90], r8d
PAGE:0000000140AA1056 mov [rbp+4Fh+var_50], rax
PAGE:0000000140AA105A mov eax, edx
PAGE:0000000140AA105C setnz al
PAGE:0000000140AA105F mov [rbp+4Fh+var_8C], r9d
PAGE:0000000140AA1063 mov [rbp+4Fh+var_88], eax
PAGE:0000000140AA1066 xor r9d, r9d
; int dword_140E076F0
dword_140E076F0 dd 0 ; DATA XREF: PopDiagTraceDripsHistogram+125↑r
; PopDiagTraceDripsHistogram+13C↑o ...
00A1A008 0000000140AA1008: PopSqmThermalUsermodeEvent+1C
Output
Python
AU: idle Down Disk: 115GB

# Surveying TraceLogging Providers Across Windows

# TraceLogging Providers Survey: Target Binary Files

- **Target OSs**
  - Windows 11, (10.0.26200.7705 (25H2))
  - WindowsServer 2025, (10.0.26100.32230)
- **Target Directories**
  - `C:\Windows\*`
  - `C:\Program Files\*`
  - Excluded “WinSxS” directory
- **Target Files**
  - .exe/.dll/.sys (64-bit only) *
  - Win11: Total 2564 files / WinSvr2025: Total 2342 files

Identified binaries containing TraceLogging providers using *TLGMetadataParser.psm1*

Performed Provider-Event mapping with *TLGMapper*

* Note: A small number of TraceLogging provider binaries were excluded due to file permission issues.

# Result: Overall TraceLogging Providers / Events

## Windows 11

**ver. 10.0.26200.7705 (25H2)**

2672 providers

73000+ unique events

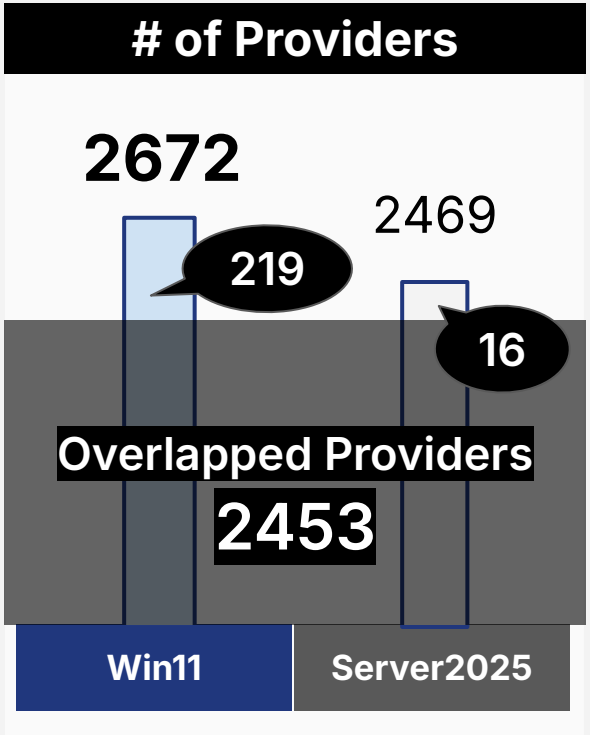
## Windows Server 2025

**ver. 10.0.26100.32230**

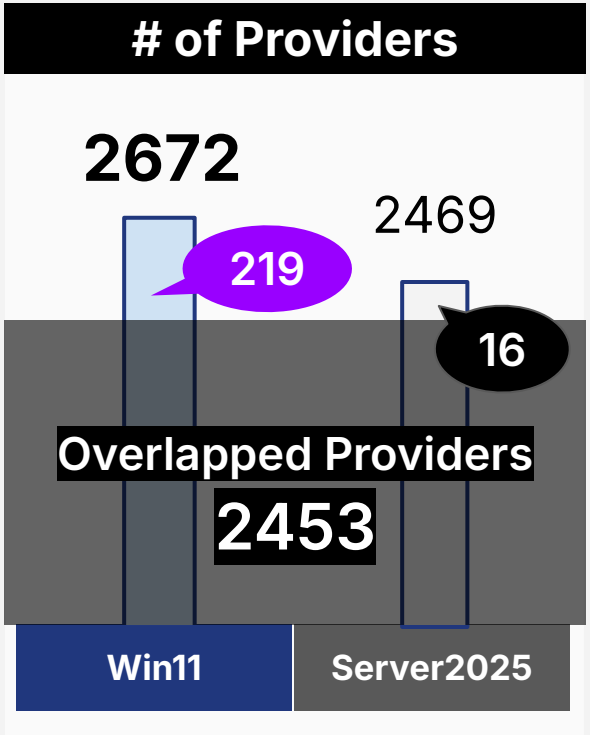
2469 providers

64000+ unique events

# Windows 11 vs. Windows Server 2025



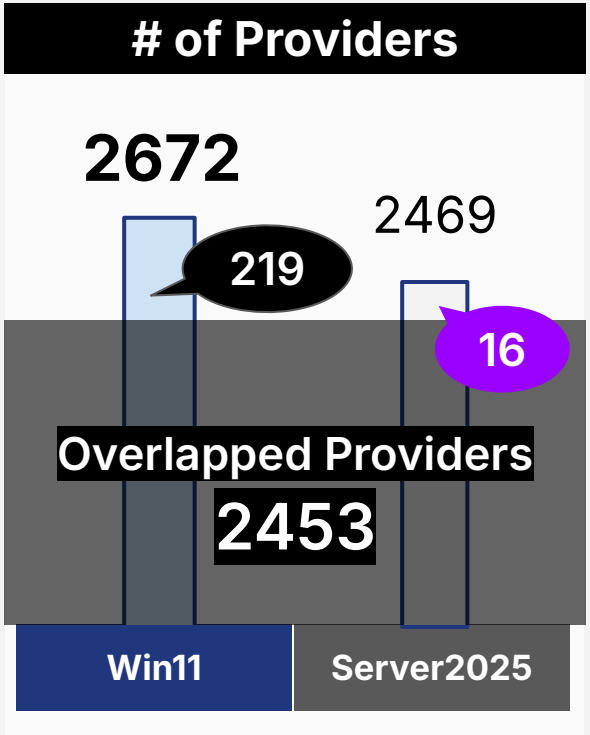
# Windows 11 vs. Windows Server 2025




Windows 11: Richer consumer/client-oriented features

- Microsoft.FamilySafety.*
- Microsoft.Windows.FamilySafety.*
- Microsoft.Xbox.*
- Microsoft.Windows.Xbox.*
- Microsoft.Windows.Onecore.WebThreatDefence.*
- Microsoft.Windows.Security.Biometrics.*

# Windows 11 vs. Windows Server 2025




## Windows 11: Richer consumer/client-oriented features

- Microsoft.FamilySafety.*
- Microsoft.Windows.FamilySafety.*
- Microsoft.Xbox.*
- Microsoft.Windows.Xbox.*
- Microsoft.Windows.Onecore.WebThreatDefence.*
- Microsoft.Windows.Security.Biometrics.*

## Windows Server 2025: Server-oriented features

- Microsoft.Windows.AzureArcSetup
- Microsoft.Windows.SrvInitConfigProvider

# What Drives the 920+ vs. 2600+ Gap?

# What Drives the 920+ vs. 2600+ Gap?

| *Application* | *Manifest-based* | *TraceLogging* |
|---|---|---|
| **notepad.exe** | ×None | Microsoft.Notepad |
| **calc.exe** | ×None | MicrosoftCalculator |
| **wextract.exe** | ×None | Microsoft.Windows.Wextract |

**Recap**

**Manifest-based**: User/Admin-facing telemetry

**TraceLogging**: Developer-focused telemetry

# What Drives the 920+ vs. 2600+ Gap?

| *Application* | *Manifest-based* | *TraceLogging* |
|---|---|---|
| **notepad.exe** | ×None | Microsoft.Notepad |
| **calc.exe** | ×None | MicrosoftCalculator |
| **wextract.exe** | ×None | Microsoft.Windows.Wextract |

My guess is that Microsoft does not create Manifest-based providers for logs that are not considered useful to users/admins.

**AND / OR**

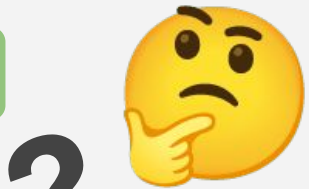
First creates a TraceLogging provider, and then adds a Manifest-based provider if the logs also appear useful to users/admins?

# Doppelgänger Providers

**Microsoft-Windows-Ldap-Client**

**Microsoft.Windows.Ldap.Client**

Manifest-based



TraceLogging

# Doppelgänger Providers (Examples)

## Manifest-based

Microsoft-Antimalware-Scan-Interface
Microsoft-Windows-Firewall
Microsoft-Windows-SmartScreen
Microsoft-Windows-Ldap-Client
Microsoft-Antimalware-Scan-Interface
Microsoft-Windows-Security-Kerberos
Microsoft-Windows-Security-Netlogon
Microsoft-Windows-Kernel-Registry
Microsoft-Windows-Kernel-Dump
Microsoft-Windows-TaskScheduler

## TraceLogging

Microsoft.Antimalware.Scan.Interface
Microsoft.Windows.Firewall
Microsoft.Windows.SmartScreen
Microsoft.Windows.Ldap.Client
Microsoft.Antimalware.Scan.Interface
Microsoft.Windows.Security.Kerberos
Microsoft.Windows.Security.Netlogon
Microsoft.Windows.Kernel.Registry
Microsoft.Windows.Kernel.Dump
Microsoft.Windows.TaskScheduler

# Ldap Client Provider's Events (Manifest vs. TraceLogging)

## Microsoft-Windows-Ldap-Client

- ✔ **Event ID 1 ~ 29, 31**
  - Ldap client trace events
- ✔ **Event ID 30**
  - Ldap search API event
    - ScopeOfSearch
    - SearchFilter
    - DistinguishedName
    - AttributeList
    - ProcessId

## Microsoft.Windows.Ldap.Client

- ✔ **LdapBindStatistics**
- ✔ **LdapReceiveStatistics**

# AMSI Provider's Events (Manifest vs. TraceLogging)

## Microsoft-Antimalware-Scan-Interface

- ✓ **Event ID 1101**
  - ■ AmsiScanBuffer event
    - session
    - scanStatus
    - scanResult
    - appname
    - contentname
    - contentsize
    - original size
    - **content**
    - hash
    - contentFiltered
    - hashoriginalcontent

## Microsoft.Antimalware.Scan.Interface

- ✓ **AmsiProvider**
- ✓ **AmsiProviderFailTrust**
- ✓ **AmsiScan**
  - PartA_PrivTags
  - ProviderIndex
  - Guid
  - Duration
  - Hr
  - Result
- ✓ **AmsiUacScan**

# AMSI Provider's Events (Manifest vs. TraceLogging)

## Microsoft-Antimalware-Scan-Interface

- ✓ **Event ID 1101**
  - ■ AmsiScanBuffer event
    - session
    - scanStatus
    - scanResult
    - appname
    - contentname
    - contentsize
    - original size
    - **content**
    - hash
    - contentFiltered
    - hashoriginalcontent

## Microsoft.Antimalware.Scan.Interface

- ✓ **AmsiProvider**
- ✓ **AmsiProviderFailTrust**
- ✓ **AmsiScan**
  - PartA_PrivTags
  - ProviderIndex
  - Result
- ✓ **AmsiUacScan**

**Microsoft-Antimalware-UacScan** provider has a **UacScan** event, which contains richer fields than **AmsiUacScan**

# Advantages of TraceLogging Providers [1/4]

Manifest-based

>

LOSE..

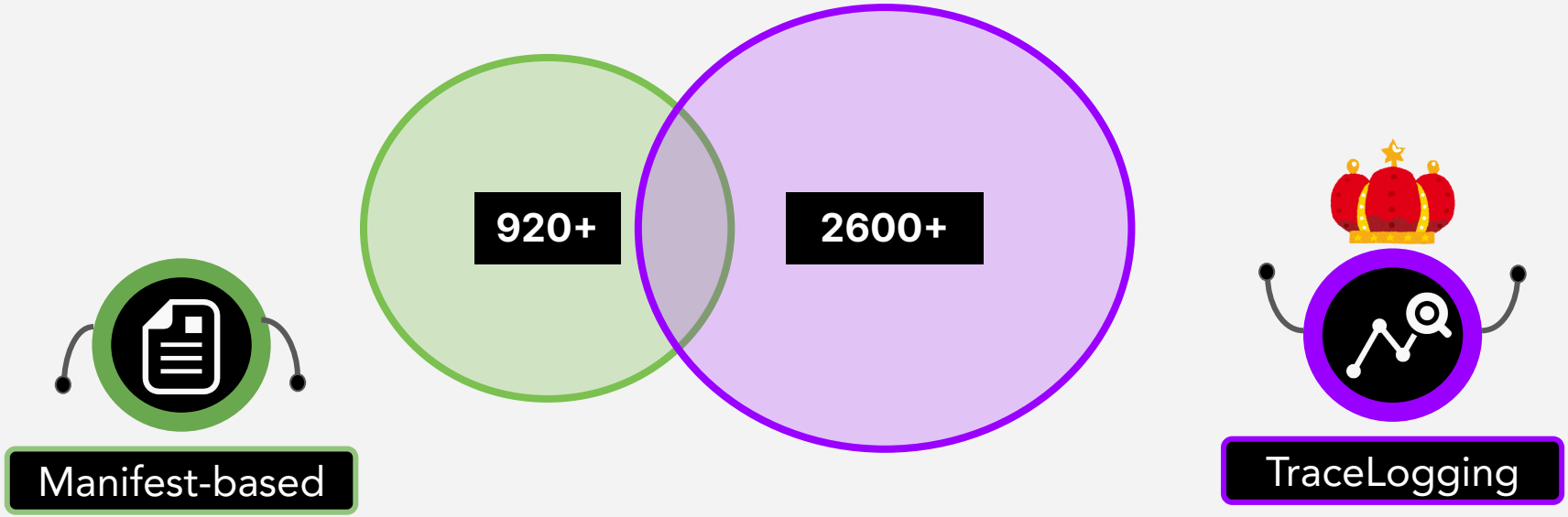
TraceLogging

?

🤔

# Advantages of TraceLogging Providers [2/4]

## One of the key strengths is their broader coverage




Note: Results of the analysis of providers with the same or similar names in Manifest-based and TraceLogging:
https://github.com/AsuNa-jp/TLGMapper/blob/main/misc/OverlappedProviders/OverlappedProviders.md

# Advantages of TraceLogging Providers [3/4]

## Copilot / AI-related providers

- Microsoft.Windows.UDK.WindowsCopilot
- Microsoft.Windows.Shell.WindowsCopilot
- Microsoft.Windows.Desktop.CopilotHotkeyManager
- Microsoft.Windows.Desktop.AIAssistance
- Microsoft.Windows.AI.Platform
- Microsoft.Windows.AI.MachineLearning
- Microsoft.Windows.AI.WindowsMLRuntime
- Microsoft.Windows.AIFabric

# Advantages of TraceLogging Providers [3/4]

## Copilot / AI-related providers

- **Microsoft.Windows.UDK.WindowsCopilot**

| *EventNames* | *Fields* |
|---|---|
| CopilotLaunch | • FormCode<br>• PartA_PrivTags |
| SendFileWithConfirmationBehavior | • PartA_PrivTags |
| SendPromptWithConfirmationBehavior | • PartA_PrivTags |

- **Microsoft.Windows.AIFabric**

# Advantages of TraceLogging Providers [3/4]

## Copilot / AI-related providers

- **Microsoft.Windows.UDK.WindowsCopilot**
- **Microsoft.Windows.Shell.WindowsCopilot**

| *EventNames* | *Fields* | |
|---|---|---|
| FeatureEnablement | • featureName<br>• userEnabled<br>• FeatureEnabledReason | • windowsEnabled<br>• FeatureEnabledReason |
| WindowsCopilotSessionSummaryEvent | • PartA_PrivTags<br>• componentName<br>• endApplication<br>• actionNamesCompleted<br>• actionCompleted<br>• resultKeptNeedsEvaluation | • featureName<br>• startApplication<br>• featureIdentifiersSeen<br>• featureViewed<br>• durationMs<br>• resultKept |

# Advantages of TraceLogging Providers [4/4]

| Providers related to relatively new features (Examples) | |
|---|---|
| User Choice Protection Driver | • Microsoft.Windows.Kernel.UCPD |
| Win32 App Isolation | • Microsoft.Windows.Security.IsolationApi |

# Advantages of TraceLogging Providers [4/4]

## Providers related to relatively new features (Examples)

| | |
|---|---|
| User Choice Protection Driver | • Microsoft.Windows.Kernel.UCPD |
| Win32 App Isolation | • Microsoft.Window |

| *EventNames* (an excerpt) |
|---|
| DefBrowserModify |
| CopilotModify |
| FakeMSApp |
| OfficeModify |
| UIAAttack |

# Advantages of TraceLogging Providers [4/4]

## Providers related to relatively new features (Examples)

| | |
|---|---|
| User Choice Protection Driver | • Microsoft.Windows.Kernel.UCPD |
| Win32 App Isolation | • Microsoft.Windows.Security.IsolationApi |

Might be useful for detecting attacks against new features?

Keep in mind that, because these are intended for developers, the event contents may change.

# Examples of Useful Providers From a Security Perspective

# AttackSurfaceMonitor: Provider Overview

**Provider GUID**
`c4e507b1-7224-4737-bde0-ced9284e7073`

## Provider Binary File:

- ✓ `C:\Windows\System32\ntoskrnl.exe` (Windows Kernel)

## Useful Event(s):

- ✓ **Ast.DeviceCreated (IoCreateDevice API)**
  - Important Fields - DeviceName/DeviceType/DeviceCharacteristics/DeviceObject
- ✓ Ast.DeviceSDDLChanged (IoCreateDeviceSecure API)
- ✓ **Ast.IoctlCalled (DeviceIoControl API)**
  - Important Fields - DeviceObject/IoControlCode/DeviceNameTailHash

## Use Cases:

- ✓ **Suspicious IOCTL-based device driver abuse detection (BYOVD and legitimate feature misuse)**

# AttackSurfaceMonitor: Event Details [1/4]

```
C:\Users\asp\Desktop\Release\Release>kdu.exe -dse 0 -prv 59
[#] Kernel Driver Utility v1.4.8 (build 2603) started, (c) 2020 - 2026 KDU Proje
[#] Built at Fri Apr 10 13:05:46 2026, header checksum 0x6A424
[#] Supported x64 OS : Windows 7 and above
[*] CPU vendor string: GenuineIntel
[*] Windows version: 10.0 build 26200
[*] SecureBoot is disabled on this machine
[+] MSFT Driver block list is enabled
[+] Selected provider: 59
[+] Hypervisor present: "VMware" (VMwareVMware)
[+] Environment diagnostics: VMware VMCI, VMware balloon, VMware HGFS, Hyper-V inte ation, Hy
[+] Drivers database "drv64.dll" loaded at 0x00007FFF5F680000
[+] Database KDUEXT version is OK
[+] Firmware type (FirmwareTypeUefi)
[+] Provider: "IPCType Device Driver for 64bit", Name "IPCType"
[+] Superfetch prerequisites validated, SE_PROF_SINGLE_PROCESS_PRIVILEGE adjusted
[+] Extracting vulnerable driver as "C:\Users\asp\Desktop\Release\Release\IPCType.sys"
[+] Vulnerable driver "IPCType" loaded
[+] Driver device "IPCType" has been opened successfully
[+] Executing post-open callback for given provider
[+] Driver device security descriptor set successfully
[+] Module "CI.dll" loaded for pattern search
[+] Superfetch memory map built: 0x13155D entries from 0x7 rang
[+] DSE flags (0xFFFFF80510D70004) value: 6, new value to be wr
[+] Kernel memory write complete, verifying data
[+] Write result verification succeeded
[+] Vulnerable driver "IPCType" unloaded
[+] Vulnerable driver file removed
```

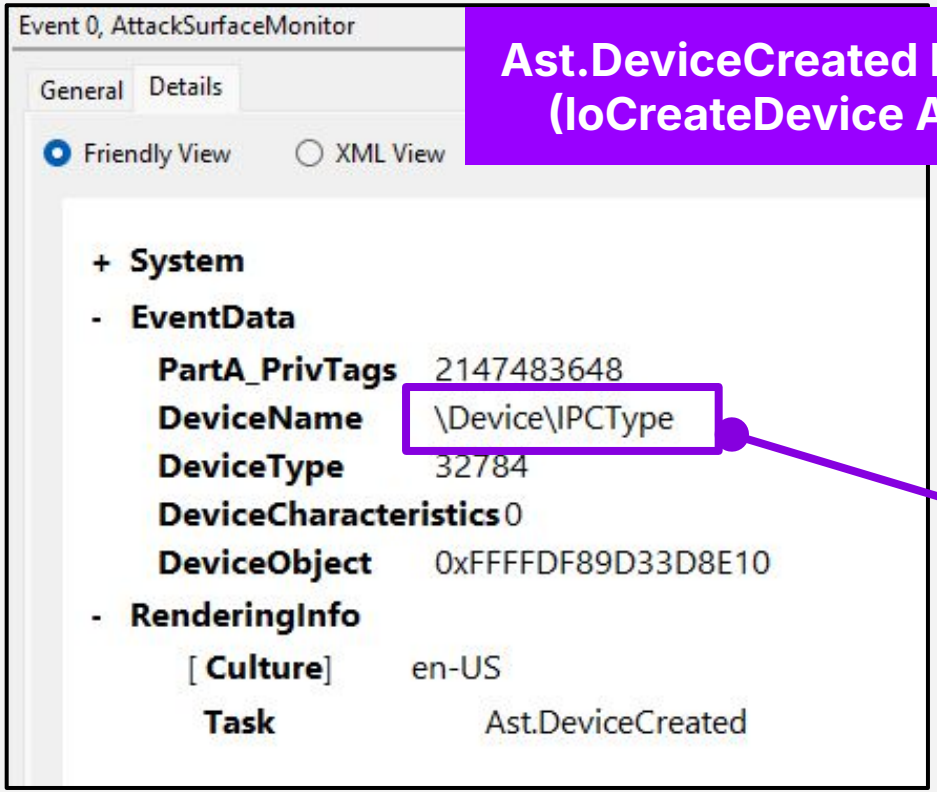
KDU by hfiref0x

https://github.com/hfiref0x/KDU

> kdu.exe -dse 0 -prv 59

**Load and abuse a vulnerable driver (IPCType.sys)**

# AttackSurfaceMonitor: Event Details [2/4]



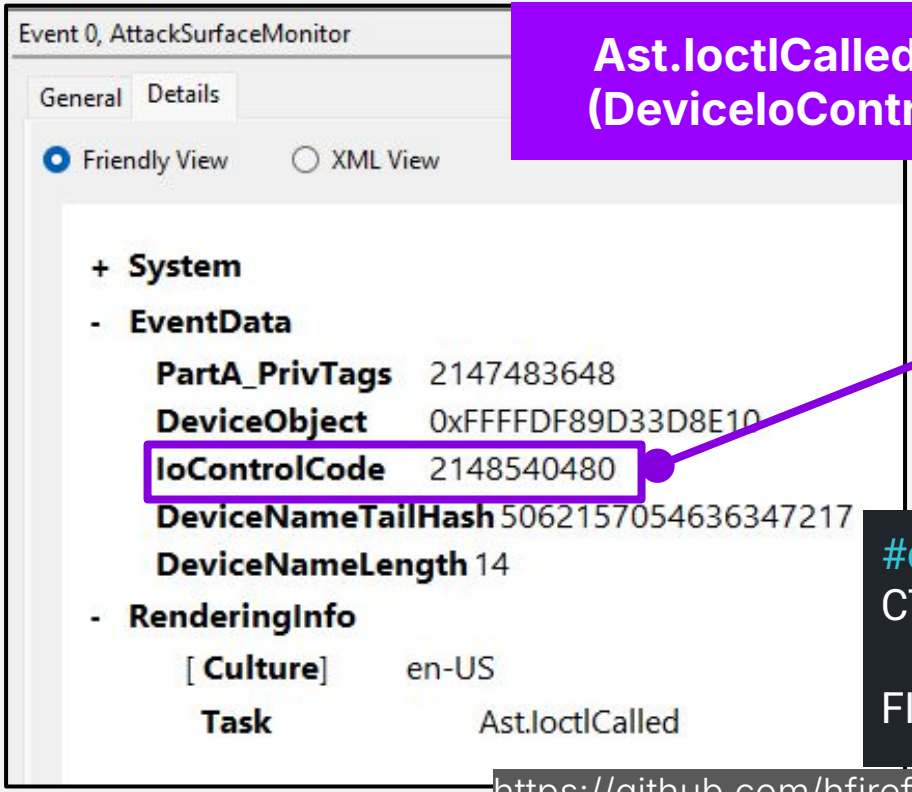

**Ast.DeviceCreated Event (IoCreateDevice API)**

DeviceName:
**\Device\IPCType**

# AttackSurfaceMonitor: Event Details [3/4]


**Ast.DeviceSDDLChanged Event (IoCreateDeviceSecure API)**

# AttackSurfaceMonitor: Event Details [4/4]




**Ast.IoctlCalled Event (DeviceIoControl API)**

**Map buffer IOCTL code**

IoControlCode:
**2148540480 (0x80102040)**

```
#define IOCTL_IPCTYPE_MAPBUFFER
CTL_CODE(IPCTYPE_DEVICE_TYPE,     \
    IPCTYPE_MAP_FUNCID, METHOD_BUFFERED,
FILE_ANY_ACCESS) // 0x80102040
```

https://github.com/hfiref0x/KDU/blob/master/Source/Hamakaze/idrv/ipcdec.h

# AttackSurfaceMonitor: Behavior Detection Rule

## Rule Name: Process Suspended via TTD Monitor Driver

```
api where
  process.Ext.api.name == "DeviceIoControl" and
  process.Ext.api.parameters.device : "\\Device\\com_microsoft_idna_ProcLaunchMon" and

 /* IOCTL 0x228034 to suspend a PID */
  process.Ext.api.parameters.io_control_code == 2261044 and
  not _arraysearch(process.thread.Ext.call_stack_final_user_module.code_signature, $caller,
  $caller.trusted == true)
```

https://github.com/elastic/protections-artifacts/blob/fc8633fd1aca367bd7e986a2e62f6851b150ad8f/behavior/rules/windows/defense_evasion_process_suspended_via_ttd_monitor_driver.toml

# Microsoft.Windows.Kernel.SysEnv: Provider Overview

**Provider GUID**
`a9fdf37b-d72d-4051-a3cd-d422103ce079`

## Provider Binary File:

- ✓ `C:\Windows\System32\ntoskrnl.exe` (Windows Kernel)

## Useful Event(s):

- ✓ **GetVariable** and **SetVariable**
  - Important Fields - variableName/vendorGuid/attributes/status

## Use Cases:

- ✓ **BootKit/SecureBoot Bypass Detection**
  - Detect reconnaissance reads of SecureBoot status via GetVariable
  - Monitor SetVariable calls targeting UEFI variables for anomalous write patterns

# Microsoft.Windows.Kernel.SysEnv: Event Details

## Example of SecureBoot Status Recon

```
PS C:\> Get-SecureBootUEFI -Name SecureBoot
```


* Note: This event was triggered on Windows 11 build 10.0.26200.8037.

# Microsoft.Windows.ShellExecute: Provider Overview

**Provider GUID**
382b5e24-181e-417f-a8d6-2155f749e724

## Provider Binary File:

- ✓ `C:\Windows\System32\windows.storage.dll`

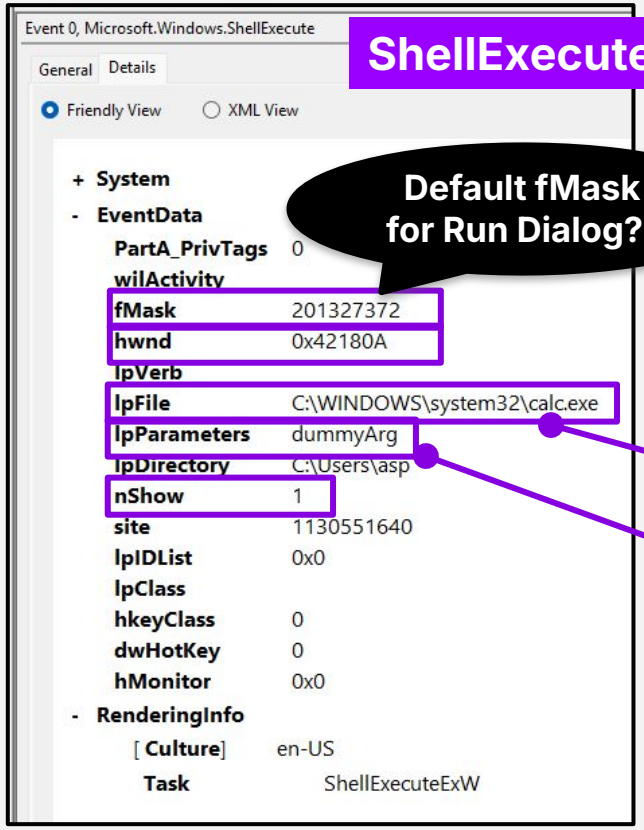
## Useful Event(s):

- ✓ **ShellExecuteExW**
  - Important Fields - lpVerb/lpFile/lpParameters/lpDirectory/fMask/nShow/hwnd/lpClass

## Use Cases:

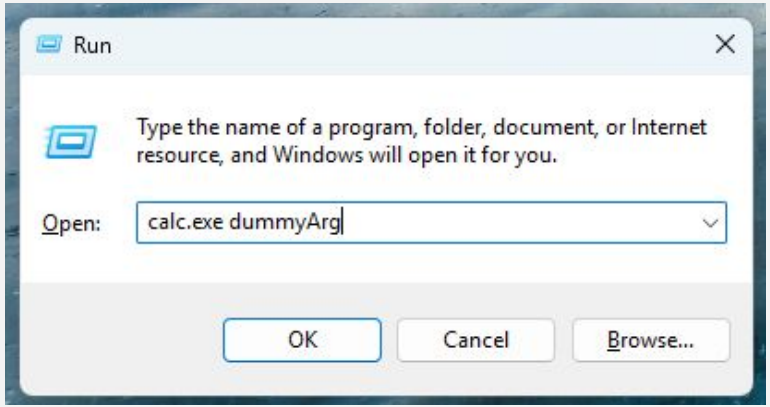
- ✓ Detect suspicious, hidden, or indirect process launches via *ShellExecuteExW*

# Microsoft.Windows.ShellExecute: Event Details




**ShellExecuteExW Event**

**Default fMask for Run Dialog?***




lpFile: **C:\WINDOWS\system32\calc.exe**

lpParameters: **dummyArg**

Might be useful for detecting ClickFix-style attacks?

🤔

*Note: On both Windows 11 and Server 2025, this mask was consistently observed when executing via the Run dialog.

# Interesting Providers I Haven’t Explored Yet

| | |
|---|---|
| Microsoft.Windows.Security.BCrypt | Microsoft.Windows.Firewall.API |
| Microsoft.Windows.Security.Ncrypt | Microsoft.Windows.Kernel.Security |
| Microsoft.Windows.Security.TokenBroker | Microsoft.Windows.Kernel.Registry |
| Microsoft.Windows.Security.PlatformCryptoProvider | Microsoft.Web.Platform |
| Microsoft.Windows.Security.Certificates | Microsoft.Windows.App.Browser |
| Microsoft.Windows.Security.CAPI | Microsoft.Windows.ComOleAut32 |
| Microsoft.Windows.Security.WIdp | Microsoft.AAD.Runtime |
| Microsoft.Windows.Security.Shutinit | Microsoft.Windows.Networking.DNS |
| Microsoft.Windows.Security.Kerberos | Microsoft.Windows.Networking.WFP.User |
| Microsoft.Windows.Security.CodeIntegrity.Wintrust | CombaseTraceLoggingProvider |

New Release!

https://github.com/AsuNa-jp/TLGMapper

- TLGMapper.py — IDA python script for TraceLogging event-to-provider resolution (tested with IDA pro 8.x+)
- Per-provider event catalog — Event details organized by provider
- Raw extracted data for each binary (CSV files)


Thank you!
✉ : asuka.nakajima@elastic.co
X : @AsuNa_jp