# Tampering with Windows Event Tracing: Background, Offense, and Defense

Palantir · Follow
Published in Palantir Blog · 13 min read · Dec 24, 2018

208 1

goal of this blog post is to share our knowledge with the community by
co
te

## Introduction to ETW and event logging

The ETW architecture differentiates between event *providers*, event *consumers*, and event *tracing sessions*. Tracing sessions are responsible for collecting events from providers and for relaying them to log files and consumers. Sessions are created and configured by *controllers* like the built-in `logman.exe` command line utility. Here are some useful commands for exploring existing trace sessions and their respective ETW providers; note that these must usually be executed from an elevated context.

### List all running trace sessions

```
> logman query -ets

Data Collector Set                  Type    Status
-------------------------------------------------
Circular Kernel Context Logger      Trace   Running
AppModel                            Trace   Running
ScreenOnPowerStudyTraceSession      Trace   Running
DiagLog                             Trace   Running
EventLog-Application                Trace   Running
EventLog-System                     Trace   Running
LwtNetLog                           Trace   Running
NtfsLog                             Trace   Running
TileStore                           Trace   Running
UBPM                                Trace   Running
WdiContextLog                       Trace   Running
WiFiSession                         Trace   Running
UserNotPresentTraceSession          Trace   Running
Diagtrack-Listener                  Trace   Running
MSDTC_TRACE_SESSION                 Trace   Running
WindowsUpdate_trace_log             Trace   Running
```

```
Level:                255
Properties:           65
Filter Type:          0

Provider:
Name:                 Microsoft-Windows-WDAG-Service
Provider Guid:        {728B02D9-BF21-49F6-BE3F-91BC06F7467E}
Level:                255
KeywordsAll:          0x0
KeywordsAny:          0x8000000000000000
Properties:           65
Filter Type:          0

...

Provider:
Name:                 Microsoft-Windows-PowerShell
Provider Guid:        {A0C1853B-5C40-4B15-8766-3CF1C58F985A}
Level:                255
KeywordsAll:          0x0
KeywordsAny:          0x9000000000000000 (Microsoft-Windows-
PowerShell/Operational,Microsoft-Windows-PowerShell/Admin)
Properties:           65
Filter Type:          0
```

This command details the configuration of the trace session itself, followed by the configuration of each provider that the session is subscribed to, including the following parameters:

- **Name**: The name of the provider. A provider only has a name if it has a registered manifest, but it always has a unique GUID.
- **Provider GUID**: The unique GUID for the provider. The GUID and/or name of a provider is useful when performing research or operations on a specific provider.
- **Level**: The logging level specified. Standard logging levels are: 0 — Log

in event log-related trace sessions, you will see the high byte

```
0x01 - Admin channel
0x02 - Debug channel
0x04 - Analytic channel
0x08 - Operational channel
```

- **Properties:** This refers to optional ETW properties that can be specified when writing the event. The following values are currently supported (more information here):

```
0x001 - EVENT_ENABLE_PROPERTY_SID
0x002 - EVENT_ENABLE_PROPERTY_TS_ID
0x004 - EVENT_ENABLE_PROPERTY_STACK_TRACE
0x008 - EVENT_ENABLE_PROPERTY_PSM_KEY
0x010 - EVENT_ENABLE_PROPERTY_IGNORE_KEYWORD_0
0x020 - EVENT_ENABLE_PROPERTY_PROVIDER_GROUP
0x040 - EVENT_ENABLE_PROPERTY_ENABLE_KEYWORD_0
0x080 - EVENT_ENABLE_PROPERTY_PROCESS_START_KEY
0x100 - EVENT_ENABLE_PROPERTY_EVENT_KEY
0x200 - EVENT_ENABLE_PROPERTY_EXCLUDE_INPRIVATE
```

From a detection perspective, EVENT_ENABLE_PROPERTY_SID, EVENT_ENABLE_PROPERTY_TS_ID, EVENT_ENABLE_PROPERTY_PROCESS_START_KEY are valuable fields to collect. For example, EVENT_ENABLE_PROPERTY_PROCESS_START_KEY generates a value that uniquely identifies a process. Note that Process IDs are not unique identifiers for a process instance.

### Enumerating all registered ETW providers

Th [illegible] supplying their name and GUID. An ETW provider is registered if it has a binary manifest stored in the `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers\{PROVIDER_GUID}` registry key. For example, the `Microsoft-Windows-PowerShell` provider has the following registry values:

ETW and the event log know how to properly parse and display event information to a user based on binary-serialized information in the `WEVT_TEMPLATE` resource present in the binaries listed in the `ResourceFileName` registry value. This resource is a binary representation of an instrumentation manifest (i.e., the schema for an ETW provider). The binary structure of `WEVT_TEMPLATE` is under-documented, but there are at least two tools available to assist in parsing and recovering event schema, WEPExplorer and Perfview.

### Viewing an individual provider

The `logman` tool prints basic information about a provider. For example:

The listings shows supported keywords and logging values, as well as all processes that are registered to emit events via this provider. This output is useful for understanding how existing trace sessions filter on providers. It is also useful for initial discovery of potentially interesting information that could be gathered from via an ETW trace.

Entries listed with GUID are providers lacking a manifest. They will typically be related to WPP or TraceLogging, both of which are beyond the scope of this blog post. It is possible to retrieve provider names and event metadata for these providers types. For example, here are some of the resolved provider names from the unnamed providers above:

One of the great security features of PowerShell version 5 is scriptblock au

Mi

(warning level) if the scriptblock contains any suspicious terms. The following C# code is executed to generate the event log:

From PowerShell

From PowerShell

The `WriteEvent` method is implemented as follows:

From PowerShell

Finally, the event information is marshaled and EventWriteTransfer is called, supplying the Microsoft-Windows-PowerShell provider with event data.

The relevant data supplied to `EventWriteTransfer` is as follows:

Normally, this event would not be logged but because the Application

rs which will log the event despite a keyword value not being specified.

- Event data: the scriptblock contents and event fields

Upon receiving the event from the PowerShell ETW provider, the event log service parses the binary `WEVT_TEMPLATE` schema (original XML schema) and presents human-readable, parsed event properties/fields:

### amsi.dll event tracing

You may have observed that Windows 10 has an AMSI/Operational event log that is typically empty. To understand why events are not logged to this event log, you would first have to inspect how data is fed to the AMSI ETW provider (`Microsoft-Antimalware-Scan-Interface - {2A576B87-09A7-520E-C21A-4942F0271D67}`) and then observe how the Application event log trace session (`EventLog-Application`) subscribes to the AMSI ETW provider. Let's start by looking at the provider registration in the Application event log. The

- Events should be captured even if an event keyword value is zero as

This information on its own does not explain why AMSI events are not logged, but it supplies needed context upon inspecting how `amsi.dll` writes events to ETW. By loading `amsi.dl` into IDA, we can see that there was a single call to the `EventWrite` function within the internal `CAmsiAntimalware::GenerateEtwEvent` function:

The relevant portion of the call to `EventWrite` is the `EventDescriptor`

- Keyword: `0x8000000000000001` (AMSI/Operational OR Event1). These

We now understand that 1101 events not logged to the Application event log because it only considers events where the keyword value matches 0x8000000000000000. In order to fix this issue and get events pumping into the event log, either the Application event log trace session would need to be modified (not recommended and requires SYSTEM privileges) or you could create your own persistent trace session (e.g., an autologger) to capture AMSI events in the event log. The following PowerShell script creates such a trace session:

```
$AutoLoggerGuid = "{$((New-Guid).Guid)}"
New-AutologgerConfig -Name MyCustomAutoLogger -Guid
$AutoLoggerGuid -Start Enabled
Add-EtwTraceProvider -AutologgerName MyCustomAutoLogger -Guid
'{2A576B87-09A7-520E-C21A-4942F0271D67}' -Level 0xff -
MatchAnyKeyword 0x80000000000001 -Property 0x41
```

After running the above command, reboot, and the AMSI event log will begin to populate.

fed into the event log. One would have to resort to speculation as to how the

AN

mi

## ETW tampering techniques

If the goal of an attacker is to subvert event logging, ETW provides a stealthy mechanism to affect logging without itself generating an event log trail. Below is a non-exhaustive list of tampering techniques that an attacker can use to cut off the supply of events to a specific event log.

Tampering techniques can generally be broken down into two categories:

1. Persistent, requiring reboot — i.e., a reboot must occur before the attack takes effect. Changes can be reverted, but would require another reboot. These attacks involve altering autologger settings — persistent ETW trace sessions with settings in the registry. There are more types of persistent attacks than ephemeral attacks, and they are usually more straightforward to detect.
2. Ephemeral — i.e., where the attack can take place without a reboot.

### Autologger provider removal

**Tampering category**: Persistent, requiring reboot
**Minimum permissions required**: Administrator
**Detection artifacts:** Registry key deletion:
`HKLM\SYSTEM\CurrentControlSet\Control\WMI\Autologger\AUTOLOGGER_NAME\{PROVIDER_GUID}`

**Description:** This technique involves the removal of a provider entry from a

**Tampering category:** Persistent, requiring reboot

**Mi**

**De**

`HKLM\SYSTEM\CurrentControlSet\Control\WMI\Autologger\AUTOLOGGER_NAME\{PROVIDER_GUID} - EnableProperty (REG_DWORD)`

**Description:** This technique involves alerting the `Enable` keyword of an autologger session. For example, by default, all ETW provider entries in the `EventLog-Application` autologger session are set to `0x41` which translates to `EVENT_ENABLE_PROPERTY_SID` and `EVENT_ENABLE_PROPERTY_ENABLE_KEYWORD_0`. `EVENT_ENABLE_PROPERTY_ENABLE_KEYWORD_0` is not documented; it specifies that any events generated for a provider should be logged even if the keyword value is set to 0. An attacker could swap out `EVENT_ENABLE_PROPERTY_ENABLE_KEYWORD_0` for `EVENT_ENABLE_PROPERTY_IGNORE_KEYWORD_0`, resulting in a value of `0x11`, which would result in all events where the keyword is 0 to not be logged. For example, PowerShell eventing supplies a 0 keyword value with its events, resulting in no logging to the PowerShell event log.

**Example:** The following PowerShell code disables Microsoft-Windows-PowerShell event logging:

```
Set-EtwTraceProvider -Guid '{A0C1853B-5C40-4B15-8766-
3CF1C58F985A}' -AutologgerName 'EventLog-Application' -Property
0x11
```

In the above example, `A0C1853B-5C40-4B15-8766-3CF1C58F985A` refers to the Microsoft-Windows-PowerShell ETW provider. This command will end up

**Tampering category:** Ephemeral

**Mi**

**De** e associated with this event. While the technique example below indicates that `logman.exe` was used to perform the attack, an attacker can obfuscate their techniques by using Win32 APIs directly, WMI, DCOM, PowerShell, etc.

**Description:** This technique involves removing an ETW provider from a trace session, cutting off its ability to supply a targeted event log with events until a reboot occurs, or until the attacker restores the provider. While an attacker must have SYSTEM privileges to perform this attack, it is unlikely that defenders will notice such an attack if they rely on event logs for threat detection.

**Example:** The following PowerShell code immediately disables Microsoft-Windows-PowerShell event logging until a reboot occurs or the attacker restores the ETW provider:

```
logman update trace EventLog-Application --p Microsoft-Windows-
PowerShell -ets
```

**Alternative detection artifacts/ideas:**

- Event ID 12 within the Microsoft-Windows-Kernel-EventTracing/Analytic log indicates when a trace session is modified, but it doesn't supply the provider name or GUID that was removed, so it would be difficult to confidently determine whether or not something suspicious occurred

- The frequency at which providers are removed from Application event logs in large environments is not currently known. As as fallback, it is still advised to log the execution of `logman.exe`, `wpr.exe`, and PowerShell in your environment.

## Conclusion

Identifying blind spots and assumptions in Alerting and Detection Strategies is a crucial step in ensuring the resilience of detections. Since ETW is at the core of the event logging infrastructure, gaining an in-depth understanding of ETW tampering attacks is a valuable way to increase the integrity of security-related data sources.

## Further Reading

- ETW — Overview
- Instrumenting Your Code with ETW
- Event Tracing for Windows: Reducing Everest to Pike's Peak

## Written by Palantir

12.1K Followers · Editor for Palantir Blog

Follow

### More from Palantir and Palantir Blog

Palantir in Palantir Blog

**Dev versus Delta: Demystifying engineering roles at Palantir**

Palantirians talk about what it's like to work at a company that does engineering differently.

Apr 8, 2019 1.2K 4

Palantir in Palantir Blog

**The Future of Drone Navigation**

Introducing Palantir's Visual Navigation (VNav)

Oct 14 71 2

## Recommended from Medium

Kostas

### Telemetry on Linux vs. Windows: A Comparative Analysis

A look at how Windows and Linux manage telemetry to support incident response...

✦ Sep 3 👏 177

RED TEAM

### Malware Development Part 8 : Reverse Shell Via Dll Hijacking

“From DLL to Shell: A Step-by-Step Guide to Reverse Shell via DLL Hijacking”

✦ Jun 22 👏 147

Lists

**Tech & Tools**
21 stories · 332 saves

**Stories to Help You Grow as a Software Developer**
19 stories · 1452 saves

**Medium's Huge List of Publications Accepting...**
378 stories · 3815 saves

**Staff Picks**
755 stories · 1416 saves

Tamir Suliman

**Comparing Popular SIEM Data Pipeline Designs: Elastic , ArcSig...**

We will continue with what we discussed on Part 2. In this article we will provide breif...

✦ Sep 29 9 1

Chicken0248

**[LetsDefend Write-up] Log Analysis With Sysmon**

Our company has experienced a breach on one of its endpoints. Your task is to investiga...

Sep 2 3

See more recommendations

Help Status About Careers Press Blog Privacy Terms Text to speech Teams