0.1 Event Tracing for Windows (ETW)

Figure out if components should be emphasized using emph

ETW is a logging mechanism that is built into the kernel of Windows. It is used by kernel-mode drivers and applications to provide realtime events and tracing features. While ETW is built into most drivers and applications made by Windows, it is also available for developers to use in their own applications. As most privileged applications built into Windows utilize ETW, it is a very good source for telemetry data related to discovering exploit attempts.

In the architecture of ETW events are at the centerpiece where they are created, managed and consumed by different event components[2]. These differentiate between event *providers*, event *consumers*, and event *controllers*. All of these event components handle the workflow of ETW, either by reading or writing, or by controlling the events in some way. This is demonstrated on Figure 1 (ETW model diagram[2]), where *sessions* are at the center of the ETW model. These sessions are controlled by an *controller* and hereafter consumed by a consumer. The following sections will go into detail of how each component works together to provide realtime tracing events.

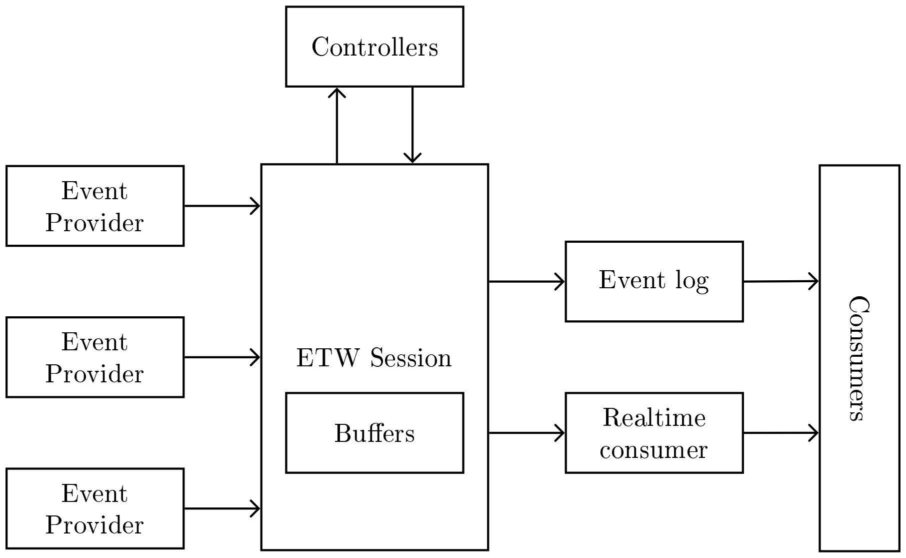

Figure 1: ETW model diagram[2]

0.1.1 Controllers

Controllers are applications, either user-mode or kernel-mode, used to start and manage trace sessions. ETW is special in the way that events are not stored or consumed in any way before a session is started. To start such a session, a *controller* application starts a trace using a Windows Application Programming Interface (API)s such as `StartTrace`. Afterwards specific *providers* can be enabled by using `EnableTraceEx2`. The specific API depend on the type of the provider as explained in the next section, 0.1.2. Controllers also manage buffers and statistics for events consumed in the current session.

0.1.2 Providers

Providers are the system- and userland applications that provide events and data. They do so by registering themselves as a provider, allowing a *controller* to enable or disable events. By having the *controller* control whether events are enabled or not, allows an application to have tracing without generating alerts all the time. This is especially interesting for debugging purposes, which is usually not needed during regular usage of the operating system.

Microsoft define four different types of providers depending on the version of Windows and type of application you are interested in. The reason for having four different types of providers is simply that ETW evolved over time, and as such different providers were added in different versions of Windows[6].

Managed Object Format (MOF) (classic) providers These types of providers are, as the name hints, the original format for specifying ETW providers. MOF providers use MOF classes[3] to define events. MOF classes describe the format of the event registered by the provider to allow the consumer to read the event correctly. As it can be seen on listing 1, a MOF class resemble a struct as known from the C programming language.

```
[EventType{26}, EventTypeName{"SendIPV6"}]
class TcpIp_SendIPV6 : TcpIp
{
    uint32 PID;
    uint32 size;
    object daddr;
    object saddr;
    object dport;
    object sport;
    uint32 startime;
    uint32 endtime;
    uint32 seqnum;
    uint32 connid;
};
```

Listing 1: `TcpIp_SendIPV6 : TcpIp` MOF class

Windows softwarre trace preprocessor (WPP) providers

With WPP providers, Windows moved away from using MOF classes to the Trace Message Format (TMF) format. With TMF the trace format description was moved into the Program Database (PDB) of the binary. For most binaries the PDB can be downloaded from Microsoft symbol servers[5], however not all Windows drivers and applications have public debug symbols, so getting access to the TMF is often a hit or miss.

Manifest-based providers

With manifest-based providers a new format to describe events was implemented. Instead of embedding the format description into the PDB, manifest-based providers embed the manifest directly into the binary as pointed to by the registry keys under `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Publishers`. However, the manifest format is not well documented, making it hard parse and recover the schema needed to understand the events[1]. Manifest-based provider are however the first ETW provider type with the ability to be enabled by more than one trace session simultaneously, which is not possible with MOF and WPP providers.

TraceLogging providers

These types of providers are the newest type of providers in the ETW logging mechanism. Unlike all the previous types of providers, the TraceLogging provider includes event format description into the recorded log data[6] allowing a consumer to easily understand the event data without prior knowledge of the format. As with manifest-based providers, TraceLogging can also be enabled by up to eight trace sessions simultaneously.

0.1.3 Consumers

Consumers are applications that consume events from providers. This is done through event *trace sessions*, where one session is created per provider. Consumers have the ability to both receive events in real time from *trace sessions*, or later on by events stored in log files. Furthermore, events can be filtered by many attributes such as timestamps.

0.1.4 Sessions

ETW sessions are created and managed by controllers to forwards events from one or more providers to a consumer such as the event log or simply a console output. As shown on Figure 1 (ETW model diagram[2]), sessions contain a number of buffers, one for each event provider. The session is responsible for these buffers, ie. the session creates and manages the buffer in its lifetime. Two predefined sessions exists in Windows, that is the *Global Logger Session* and the *NT Kernel Logger Session* handling events occurring early in the system boot process and predefined system events generated by the operating system respectively[4].

Figure 1 (ETW model diagram[2]) shows how the different components of ETW works together in sessions to produce and consume events.