Common Event Format: Event Interoperability Standard

Sponsored by:
ArcSight, Inc.
5 Results Way, Cupertino, CA 95014, USA
Email: CEF@arcsight.com

Event Interoperability Standard

This paper proposes a standard for the interoperability of event- or log-generating devices. Especially in the security world, a myriad of formats are used for event reporting, which greatly complicates integration.

Each vendor has its own format for reporting event information, but beyond this, these event formats often lack key information necessary to integrate the events from their devices.

Motivation

The mission of this standard is to improve the interoperability of infrastructure devices by better aligning the logging output from participating technology vendors.

This standard guides event producers to log in a format that is both useful, and more importantly, parsable by ArcSight or any vendor following the standard. Further, this standard assures that an event and its semantics contain all necessary information.

Common Exchange Format

This proposal defines a simple event format that can be readily adopted by vendors of both security and non-security devices. This format is intended to contain the most relevant information and make it easy for event consumers to parse and use events.

To simplify integration, we use syslog as a transport mechanism. This applies a common prefix to each message, containing the date and hostname:

```
Jan 18 11:07:53 zurich message
```

If an event producer is unable to write syslog messages, it is still possible to write the events to a file. In this case, omit the syslog header and start the message with the format defined below.

It is important to note that this part of the message need not be explicitly generated by the event producer. The remainder of the message is formatted using a common prefix composed of fields delimited by a bar ("|") character. The prefix is mandatory and all specified fields need to be present. Additional fields are specified in the *Extension*. The format is:

```
CEF:Version|Device Vendor|Device Product|Device Version|Signature
ID|Name|Severity|Extension
```

The *Extension* part of the message is a placeholder for additional fields. Those fields are documented in the **Event Dictionary** below and are logged as key-value pairs.

Here are definitions for the prefix fields:

Version is an integer and identifies the version of the CEF format. Event consumers use this information to determine what the following fields represent. Currently only version 0 (zero) is established in the above format. Experience may show that other fields need to be added to the “prefix” and therefore require a version number change. Adding new formats is handled through the standards body.

Device Vendor, Device Product and ***Device Version*** are strings that uniquely identify the type of sending device. No two products may use the same device-vendor and device-product pair. There is no central authority managing these pairs. Event producers have to ensure that they assign unique name pairs.

Signature ID is a unique identifier per event-type. This can be a string or an integer. Signature ID identifies the type of event reported. In the intrusion detection system (IDS) world, each signature or rule that detects certain activity has a unique signature ID assigned. This is a requirement for other types of devices as well, and helps correlation engines deal with the events.

Name is a string representing a human-readable and understandable description of the event. The event name should not contain information that is specifically mentioned in other fields. For example: “Port scan from 10.0.0.1 targeting 20.1.1.1” is not a good event name. It should be: “Port scan”. The other information is redundant and can be picked up from the other fields.

Severity is an integer and reflects the importance of the event. Only numbers from 0 to 10 are allowed, where 10 indicates the most important event.

Extension is a collection of key-value pairs. The keys are part of a predefined set. The standard allows for including additional keys as outlined later. An event can contain any number of key-value pairs in any order, separated by spaces (“ “). If a field contains a space, such as a file name, this is okay and can be logged in exactly that manner. For example: `fileName=c:\Program Files\ArcSight` is a valid token.

Here is a sample message to illustrate appearance:

```
Sep 19 08:26:10 zurich CEF:0|security|threatmanager|1.0|100|worm
successfully stopped|10|src=10.0.0.1 dst=2.1.2.2 spt=1232
```

Here are further details about character encoding:

- The entire message has to be **UTF-8** encoded.

- If a **pipe** (**|**) is used in the prefix, it has to be escaped with a backslash (\). But note that pipes in the extension do not need escaping. Here is an example message:

```
Sep 19 08:26:10 zurich CEF:0|security|threatmanager|1.0|100|detected a
\| in message|10|src=10.0.0.1 act=blocked a | dst=1.1.1.1
```

- If a **backslash** (\) is used in the prefix, it has to be escaped with another backslash (\). Again, note that backslashes in the extension do not need escaping. Here is an example:

```
Sep 19 08:26:10 zurich CEF:0|security|threatmanager|1.0|100|detected a
\\ in packet|10|src=10.0.0.1 action=blocked a \ dst=1.1.1.1
```

- If an **equal sign** (=) is used in the extensions, it has to be escaped with a backslash (\). Equal signs in the prefix need no escaping. For example:

```
Sep 19 08:26:10 zurich CEF:0|security|threatmanager|1.0|100|detected a =
in message|10|src=10.0.0.1 action=blocked a \= dst=1.1.1.1
```

- **Multi-line** fields can be sent by Common Event Format (CEF) by encoding the newline character as **\n** or **\r**. Note that multiple lines are only allowed in the value part of the extensions. See this example:

```
Sep 19 08:26:10 zurich CEF:0|security|threatmanager|1.0|100|Detected a
threat. No action needed.|10|src=10.0.0.1 message=Detected a threat.\nNo
action needed.
```

Common Extension Dictionary

The following table contains predefined keys that establish usages for both event producers and consumers. The standard allows for defining additional keys, with the understanding that those fields may not be interpreted by other event consumers.

The table below contains **key names** as well as the **full name** for each key. The **key name** is the one that is required in events.

Key Name	Full Name	Data Type	Meaning
`act`	deviceAction	String	Action mentioned in the event.

Key Name	Full Name	Data Type	Meaning
`app`	application Protocol	String	Application level protocol, example values are: HTTP, HTTPS, SSHv2, Telnet, POP, IMAP, IMAPS, etc.
`in`	bytesIn	Integer	Number of bytes transferred inbound. Inbound relative to the source to destination relationship, meaning that data was flowing from source to destination.
`out`	bytesOut	Integer	Number of bytes transferred outbound. Outbound relative to the source to destination relationship, meaning that data was flowing from destination to source.
`dst`	destination Address	IPv4 Address	Identifies destination that the event refers to in an IP network. The format is an IPv4 address. Example: “192.168.10.1”
`dhost`	destination HostName	FQDN	Identifies the destination that the event refers to in an IP network. The format is a fully qualified domain name associated with the destination node. Example: “zurich.domain.com”
`dmac`	destination MacAddress	String	Six colon-separated hexadecimal numbers. Example: “00:0D:60:AF:1B:61”
`dntdom`	destination NtDomain	String	The Windows domain name of the destination address.
`dpt`	destination Port	Integer	The valid port numbers are between 0 and 65535.
`dproc`	destination	String	The name of the process

Key Name	Full Name	Data Type	Meaning
	ProcessName		which is the event's destination. For example: "telnetd", or "sshd".
`duid`	destination UserId	String	Identifies the destination user by ID. For example, in UNIX, the root user is generally associated with user ID 0.
`dpriv`	destination UserPrivileges	String	The allowed values are: “Administrator”, “User”, and “Guest”. This identifies the destination user's privileges. In UNIX, for example, activity executed on the root user would be identified with destinationUserPrivileges of “Administrator”. This is an idealized and simplified view on privileges and can be extended in the future.
`duser`	destination UserName	String	Identifies the destination user by name. This is the user associated with the event's destination. E-mail addresses are also mapped into the UserName fields. The recipient is a candidate to put into destinationUserName.
`end`	endTime	TimeStamp	The time at which the activity related to the event ended. The format is `MMM dd yyyy HH:mm:ss` or milliseconds since epoch (Jan 1st 1970). An example would be reporting the end of a session.
`fname`	fileName	String	Name of the file.
`fsize`	fileSize	Integer	Size of the file.

Key Name	Full Name	Data Type	Meaning
`msg`	message	String	An arbitrary message giving more details about the event. Multi-line entries can be produced by using \n as the new-line separator.
`rt`	receiptTime	TimeStamp	The time at which the event related to the activity was received. The format is `MMM dd yyyy HH:mm:ss` or milliseconds since epoch (Jan 1st 1970).
`request`	requestURL	String	In the case of an HTTP request, this field contains the URL accessed. The URL should contain the protocol as well, e.g., “http://www.security.com”
`src`	SourceAddress	IPv4 Address	Identifies the source that an event refers to in an IP network. The format is an IPv4 address. Example: “192.168.10.1”
`shost`	sourceHostName	FQDN	Identifies the source that an event refers to in an IP network. The format is a fully qualified domain name associated with the source node. Example: “zurich.domain.com”
`smac`	sourceMacAddress	MAC Address	Six colon-separated hexadecimal numbers. Example: “00:0D:60:AF:1B:61”
`sntdom`	sourceNtDomain	String	The Windows domain name for the source address.
`spt`	sourcePort	Integer	The valid port numbers are 0 to 65535.

Key Name	Full Name	Data Type	Meaning
`spriv`	sourceUser Privileges		The allowed values are: “Administrator”, “User”, and “Guest”. It identifies the source user's privileges. In UNIX, for example, activity executed by the root user would be identified with sourceUserPrivileges of “Administrator”. This is an idealized and simplified view on privileges and can be extended in the future.
`suid`	sourceUserId	String	Identifies the source user by ID. This is the user associated with the source of the event. For example, in UNIX, the root user is generally associated with user ID 0.
`suser`	sourceUserName	String	Identifies the source user by name. E-mail addresses are also mapped into the UserName fields. The sender is a candidate to put into sourceUserName.
`start`	startTime	TimeStamp	The time when the activity the event referred to started. The format is `MMM dd yyyy HH:mm:ss` or milliseconds since epoch (Jan 1st 1970).
`proto`	transport Protocol	String	Identifies the Layer-4 protocol used. The possible values are protocol names such as TCP or UDP.