Open Cybersecurity Schema Framework Taxonomy

Author: Paul Agbabian
Date: April 2022
Status: RFC

Introduction to the Framework

This document describes the taxonomy scheme used in the Open Cybersecurity Schema Framework (OCSF). A taxonomy is a classification mechanism with conformation rules. The OCSF is a framework for creating schemas and also delivers a security schema using the framework's taxonomy.

The framework is made up of a set of data types, an attribute dictionary, a set of objects, and the taxonomy. Individual properties, or fields are termed attributes and each has a specific data type.

The scalar data types are defined on top of primitive data types such as strings, integers, floating point numbers and booleans. Object data type is a collection of attributes of any data type including other objects. Arrays support any of the data types. Examples of scalar data types are Timestamp, IP Address, MAC Address, Pathname, and User Name. Examples of Object data types are Process, Device, User, Certificate and File.

Some data types have constraints on their valid values or ranges, for example Enum integer types are constrained to a specific set of integer values. Enum integer typed attributes are an important part of the framework constructs and used in place of strings where possible to ensure consistency.

A standard dictionary of all available attributes and their types and constraints as well as a set of standard objects are the building blocks of the framework. Appendix A and B describe the OCSF Guidelines and data types respectively.

Building a schema from the framework building blocks requires a taxonomy. The framework is not restricted to cybersecurity nor to events, however the initial focus of the framework has been a standard schema for cybersecurity events.

Taxonomy Constructs

There are four fundamental dimensions of the OCSF taxonomy:

- Category

- Event Class
- Profile
- Extension

Each is defined with a unique identifier.

An Event Class will have different activities and dispositions at runtime, via the activity_id and disposition_id Enum values. A derived identifier capturing the actual event instance type (i.e. including the activity and disposition, along with the class) is the unique Event ID. A snippet of an Endpoint File Activity event example is shown below.

```
{
   "category_uid": 1,
   "class_uid": 1004,
   "activity_id": 2,
   "event_name": "Endpoint File Activity:Read",
   "event_time": "1970-01-20T02:09:34.676997Z",
   "event_uid": 100400,
   "message": "File foobar.json opened",
   "severity_id": 1,
   "time": 1649374676992
}
```

Comparison with MITRE ATT&CK[1] Framework

The MITRE ATT&CK Framework is widely used in the cybersecurity domain. While the purpose and content type of the two frameworks are different, there are some similarities with OCSF's taxonomy that may be instructive to those with familiarity with ATT&CK.

Categories are similar to Tactics, which have unique IDs. Event Classes are similar to Techniques, which have unique IDs. Profiles are similar to Matrices[2], which have unique names. Event IDs are similar to Procedures which have unique IDs. Profiles can filter the Event Classes and Categories similar to how Matrices filter Techniques and Tactics.

Differences from MITRE ATT&CK are that in OCSF, Event Classes are in only one Category, while MITRE ATT&CK Techniques can be part of multiple Tactics. Similarly MITRE ATT&CK Procedures can be used in multiple Techniques. MITRE ATT&CK has Sub-techniques while OCSF does not have Sub-Event Classes.

---

[1] MITRE ATT&CK: https://attack.mitre.org/

[2] MITRE ATT&CK Matrix: https://attack.mitre.org/matrices/enterprise/

OCSF is open and extensible by vendors, and end customers while the content within MITRE ATT&CK is released by MITRE.

Event Class

Events are represented by Event Classes, which are a particular set of attributes and objects representing a log line or telemetry submission at a point in time. Event classes have semantics that describe what happened, either a particular activity, disposition or both. Each event class has a unique class_uid attribute value which is the event class identifier. Event class_uid friendly names are descriptive of some type of activity, such as Email Activity or Process Activity. The semantics of the class are defined by the specific activity, via the activity_id attribute, such a File Opened or Email Received. Other attributes of the class indicate the details such as the file name, or the email sender. The unique combination of a class_uid and activity_id is represented by the event_uid attribute.

Category

A Category organizes event classes that represent a particular domain. For example, a category can include event classes for different types of events that may be found in an access log, or audit log, or network and system events. Each category has a unique category_uid attribute value which is the category identifier. Category IDs also have friendly names, such as System Activity, Network Activity, Audit, etc.

An example of categories with some of their event classes is shown in the below table.

System Activity	**Network Activity**	**Audit Activity**	**Windows Logs**	**Email Activity**
File Activity	DNS Activity	Account Change	Registry Key	Email Delivery
Folder Activity	HTTP Activity	Authentication	Registry Value	Email File
Kernel Activity	Flow Activity	Authorization	Resource	Email URL
Memory		Entity	Windows Event	
Module				
Peripheral				
Process				
Scheduled Job				

An alternate organization for categories proposed by AWS[3] is shown below.

Category
Access Logs
Cloud Operations
Container Operations
Custom Application Logs
Database Activity
DNS Activity
System Activity
Audit Activity
Management and Governance
Network Flow Activity
Security Alerts and Findings
Web Application Firewalls

Note that this organization narrows down network activity into more specific DNS Activity and Flow Activity rather than relying on classes within a broader category shown in the first table. Finding the right granularity of categories is an important topic of discussion. Categorization is not structural while event classes are structural (i.e. they define the particular attributes and specific Enum values for the event type).

Many events in the cloud can be classified as a network activity. But the key question to ask is, do the logs from these diverse systems provide the same context or information? Would there be a family of event classes that make sense in a single category? For example, does the NLB Access log provide context/info similar to a Flow log? Are they structured in the same fashion? Would we obscure the meaning of these logs if we normalize them under the same category? Would the resultant category make sense on its own or will it lose its contextual meaning all together?

[3] Refer to the document OCSF Schema Collaboration: Initial Decisions

Profile

Profiles are overlays on event classes, effectively a dynamic mix-in class of attributes and objects with their requirements and constraints.[4] While event classes specialize their category domain, a profile can augment event classes with a set of attributes independent of category. Multiple profiles can be added to an event class via an array of profile_uid values. This composition approach allows for reuse of event classes vs. creating new classes one by one that include the same attributes. Event classes and instances of events that support the profile can be filtered via the profile_uid across all categories.

For example, a Malware profile that adds MITRE ATT&CK and Malware objects to system activity classes avoids having to recreate a new event class, or many classes, with all of the same attributes as the system activity classes. A query for events of the class will return all the events, with or without the security information, while a query for just the profile will return events across all event classes that support the security profile. A Host profile and a User profile can add Device, Process and User objects to network activity event classes when the network activity log source is a user's computer.

Proposals for three built-in profiles for Malware, Host and User are shown in the below table with their attributes.

Malware Profile	**Host Profile**	**User Profile**
disposition_id / disposition		user
attacks	device	is_user_present
cvssv2	actor_process	user_entities
malware / other_malware	connection_info	accounts
quarantine_uid		user_result

Other profiles can be product oriented, such as Firewall, IDS, VA, DLP etc. if they need to add attributes to existing classes. They can also be more general, platform oriented, such as for Cloud or Windows environments.

For example, AWS services log events with an ARN (AWS Resource Name) and an AWS IAM Account. An AWS specific profile can be added to any event class or category of classes that includes arn and IAM account attributes.

[4] Refer to Proposal 3: Profiles in the document OCSF Schema Collaboration: Initial Decisions

Profiles can be used for enrichment of a specific product: e.g. Splunk Behavioral Analytics needs source and destination enrichment. A vendor specific profile can add enrichment to the relevant event classes.

Extension

Extensions are additional categories, event classes, attributes, objects or profiles. A schema is the aggregation of core schema entities and extensions. Extensions allow a particular vendor or customer to create a new schema or augment an existing schema. Extensions can also be used to factor out non-essential schema domains keeping a schema small. Extensions use the framework in the same way as a new schema, optionally creating categories, profiles or event classes from the dictionary. Extensions can add new attributes to the dictionary, including new objects. As with categories, event classes and profiles, extensions have unique IDs within the framework as well as versioning.

Splunk has extended the core schema draft with new experimental categories, new event classes and some new attributes and objects. Examples of Splunk draft extensions are shown in the table below.

Finding	**Policy**	**Remediation**	**Diagnostic**
Finding Report	Clipboard Content Protection	File Remediation	CPU Usage
Incident Associate	Compliance	Folder Remediation	Memory Usage
Incident Closure	Compliance Scan	Unsuccessful Remediation	Status
Incident Creation	Content Protection	Startup Application Remediation	Throughput
Incident Update	Information Protection	User Session Remediation	

Profile Application Examples

Using example categories and event classes from a preceding section, examples of how profiles might be applied to event classes are shown below.

System Activity

The following **would** all include the Host profile and **may** include the Malware profile:
File Activity
Folder Activity
Kernel Activity
Memory Activity
Module Activity
Peripheral Activity
Process Activity
Resource Activity
Scheduled Job Activity

The following (alerts) **would** all include the Host profile and **would** include the Malware profile:
Authentication
Boot Record Activity

Windows Activity

The following **would** include the Host profile and **may** include the Malware profile:
Registry Key Activity
Registry Value Activity

The following **would** include the Host profile:

Windows Event

Network Activity

The following **may** include the Host profile and **may** include the Malware profile:
DNS Activity
HTTP Activity
Network Activity

Audit Activity

The following **would** include the User profile, **may** include the Host profile and **would not** include the Malware profile:
Account Change
Authentication
Authorization
Entity Activity

Personas

There are three personas that are users of the framework and/or the schema built with the framework. The *author* persona is who creates or extends the schema. The *mapper* persona is who translates or creates events from a source to the schema. The *analyst* persona is who searches the data, writes rules or analytics against the schema, or creates reports from the schema. For example, a vendor may write a translation from a native source format into the schema but also extend the schema to accommodate vendor specific attributes or operations. The vendor is operating as both the mapper and author personas. A SOC analyst that collects the data in a SIEM writes rules against the events and searches events during investigation. The SOC analyst is operating as the analyst persona.

Personae: author, mapper, analyst. All can consider the profile from a different perspective.

Authors define profiles, and the profiles are applicable to specific classes or categories.

End users, e.g. analysts, can use the browser to select applicable profiles at the class level.

Mappers can add the profile ID and associated attributes to specific events mapped to logs.

Appendix A

Guidelines and Conventions

The Open Cybersecurity Schema Framework (OCSF) guidelines and conventions.

Attribute Levels

The event schema defines *Core*, *Optional*, and *Reserved* attributes.

Core Attributes

Attributes that are most common across all use cases are defined as core attributes. The core attributes are marked as Required or Recommended.

Optional Attributes

Optional attributes may apply to more narrow use cases, or may be more open to interpretation depending on the use case. The optional attributes are marked as Optional.

Reserved Attributes

Reserved attributes are set by the logging system and must not be used in the event data. The reserved attributes are marked as Reserved.

Guidelines for attribute names

- Attribute names must be a valid UTF-8 sequence.
- Attribute names must be all lower case.
- Combine words using underscore.
- No special characters except underscore.
- Use present tense unless the attribute describes historical information.
- Use singular and plural names properly to reflect the field content.
  For example, use `events_per_sec` rather than `event_per_sec`.
- When an attribute represents multiple entities, the attribute name should be pluralized and the value type should be an array.
  Example: `process.loaded_modules` includes multiple values -- a loaded module names list.
- Avoid repetition of words.
  Example: `host.host_ip` should be `host.ip`.
- Avoid abbreviations when possible.
  Some exceptions can be made for well-accepted abbreviations. Example: `ip`, or names such as `os`, `geo`.

Extending the Schema

The Open Cybersecurity Schema Framework can be extended by adding new attributes, objects, and event classes.

To extend the schema create a new directory in the `schema/extensions` directory. The directory structure is the same as the top level schema directory and it may contain the following files and subdirectories:

categories.json	Create it to define a new event category to reserve a range of class IDs.
dictionary.json	Create it to define new attributes.
events/	Create it to define new event classes.
objects/	Create it to define new objects.

Appendix B

Data Types

The predefined data types. The data type of a value specifies what kind of data that value can have.

Attribute	Base Type	Constraints	Description
boolean_t		false, true	Boolean data type.
email_t	String	^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$	Email address type. For example: john_doe@example.com.
hash_t	String	Max length: 64	File Hash value. A unique value that corresponds to the content of the (le.
file_name_t	String	^[a-zA-Z0-9._ -]+$	File name. For example: /text-file.txt.

float_t			Real Floating-point data type.
hostname_t	String	^(([a-zA-Z0-9]|[a-zA-Z0-9] [a-zA-Z0-9\-]*[a-zA-Z0-9])\.)*([A-Za-z0-9][A-Za-z0- 9][A-Za-z0-9\-]*[A-Za-z0- 9])$	A unique name assigned to a device that is connected to a speci(c computer network. A domain name in general is an Internet address that can be resolved through the Domain Name System (DNS). For example: r2- d2.example.com.
ip_t	String	Max length: 40 /^(?>(?>([a-f0-9]{1,4})(?>:(?1)){7}|(?!(?:.*[a-f0-9](?>:|$)){8,})((?1)(?>:(?1)){0,6})?::(?2)?)\|(?>(?>(?1)(?>:(?1)){5}:\|(?!(?:.*[a-f0-9]:){6,})(?3)?::(? >((?1)(?>:(?1)){0,4}):)?)?(25[0-5]\|2[0-4][0-9]\|1[0-9] {2}\|[1-9]?[0-9])(?>\.(?4)){3}))$/iD	Internet Protocol address (IP address), in either IPv4 or IPv6 format.
port_t	Integer	0-65,535	IP TCP/UDP port number. For example: 80 or 22.
integer_t			Basic signed integer data type.
json_t			Embedded JSON value. A value can be a string, or a number, or true or false or null, or an object or an array. These structures can be nested. See www.json.org.
long_t			8-byte long, signed integer data type.

mac_t	String	Max length: 32 ^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$	Media Access Control (MAC) address. For example: 18:36:F3:98:4F:9A.
object_t			Object is an unordered set of name/value pairs. For