CybOX 3.0 Playground

This document is a generic container for content and normative text that is a work in progress. This content has NOT yet been agreed upon nor fully discussed on this list.

As content in this document begins to be discussed it will be MOVED OUT of this document and into the actual pre-draft specification documents.

Additionally, any content that is not required right now in the pre-draft specification documents but will be required at a later date should be MOVED from the pre-draft specification documents INTO this document for temporary storage.

Extension Namespaces

It may be useful to consider introducing the concept of "namespaces" as a means of describing where an Object Extension fits into a logical hierarchy. For example, consider a File Object with the following extensions:

```
"extended_properties": {
       "cybox.file.metadata": {"mime_type":    "vnd.microsoft.portable-executable"},
       "cybox.filesystem.metadata.ext3": {"inode": "34483923"},
       "cybox.os.win.executable.metadata.pebinary": {"exports":[{"name":"foo_app"}]},
       "x_lookingglass.file.ubermetadata": {"stuff": [{"foo": "bar"}]}
   }
```

Besides allowing for a more precise statement of what an Extension characterizes, namespaces would make it much easier to build useful expressions around mutual exclusivity with regards to extensions. For instance, `cybox.filesystem.metadata.*` could be used to state that a particular extension is mutually exclusive with all other extensions associated with file system metadata. Such expressions could also be used as a means of expressing compatibility/interoperability with regards to the particular extensions that are supported by a product or CTI data exchange.

Network Connection Object

- We'll have a simplified Network Connection Object based on the existing proposal
- We'll have a simplified Network Flow model/extension based on some of the work IBM has been doing
- Other extensions/additions (I.e., for the rest of the OSI model) will follow in later minor CybOX revisions

- Default extensions will be: IP, connection state, flow, IP, TCP, UDP, ICMP, HTTP, DNS
- Source and Destination types will be based on specific relationships

Properties

Inherits From	Inherited Properties		
`cybox-object`	**all**		
Property Name	**Type**	**Description**	
type (inherited from `cybox-object`)	`string`	Indicates that this object is a CybOX Network Connection Object. The value of this field **MUST** be `network-connection-object`	
start_time	`timestamp`	Specifies the date/time the network connection was initiated.	
end_time	`timestamp`	Specifies the date/time the network connection was closed.	
protocols	`dictionary`	Specifies the protocols used in each layer (OSI model) of the network connection. The key for each dictionary item MUST be an entry from the `layer-enum` and its corresponding value MUST be an entry from the `protocol-name-enum`.	

OSI Layer Enum (`layer-enum`)

Value	Description
layer1	Specifies layer 1 of the OSI model - the physical layer.
layer2	Specifies layer 2 of the OSI model - the data link layer.
layer3	Specifies layer 3 of the OSI model - the network layer.

layer4	Specifies layer 4 of the OSI model - the transport layer.
layer5	Specifies layer 5 of the OSI model - the session layer.
layer6	Specifies layer 6 of the OSI model - the presentation layer.
layer7	Specifies layer 7 of the OSI model - the application layer.

Protocol Name Enum (`protocol-name-enum`)

Value	Description
tcp	
udp	
http	
ftp	

HTTP Extension (`http-extension`)

The HTTP extension specifies a default extension for capturing network connection properties specific to HTTP. The key for this extension when used in the **`extended_properties`** dictionary MUST be *http*.

Properties

Property Name	Type	Description
http_request_line	`http-request-line`	Specifies the HTTP request line of the HTTP client request.
http_request_header	`http-request-header`	Specifies all of the HTTP header fields that may be found in the HTTP client request.
http_message_body_length	`integer`	Specifies the length of the HTTP message body (if included), in bytes.
http_message_body_data	`string`	Specifies the data contained in the HTTP message body (if included).

HTTP Request Line (`http-request-line`)

Properties

Property Name	Type	Description
http_method	`http-method-enum`	Specifies the HTTP method portion of the HTTP request line.
value	`string`	Specifies the value (typically a resource path) portion of the HTTP request line.
version	`string`	Specifies the HTTP version portion of the HTTP request line.

HTTP Method Enum (`http-method-enum`)

Values

Value	Description
get	`Specifies the ‘get’ HTTP method.`
post	`Specifies the ‘post’ HTTP method.`
head	`Specifies the ‘head’ HTTP method.`
put	`Specifies the ‘put’ HTTP method.`

HTTP Request Header (`http-request-header`)

Properties

Property Name	Type	Description
accept	`string`	Specifies the HTTP Request Accept header field, which defines the Content-Types that are acceptable.
accept_charset	`string`	Specifies the HTTP Request Accept-Charset header field, which defines the character sets that are acceptable.
accept_language	`string`	Specifies the HTTP Request Accept-Language header field, which defines the acceptable languages for

		response.
accept_datetime	`string`	Specifies the HTTP Request Accept-Datetime header field, which defines the acceptable version time.
accept_encoding	`string`	Specifies the HTTP Request Accept-Encoding header field, which defines the acceptable encodings.
authorization	`string`	Specifies the HTTP Request Authorization header field, which defines the authentication credentials for use in HTTP authentication.
cache_control	`string`	Specifies the HTTP Request Cache-Control header field, which defines the directives that MUST be obeyed by all caching mechanisms along the request/response chain.
connection	`string`	Specifies the HTTP Request Connection header field, which defines the type of connection that the user-agent would prefer.
cookie	`string`	Specifies the HTTP Request Cookie header field, which defines the HTTP cookie previously sent by the server.
content_length	`integer`	Specifies the HTTP Request Content-Length header field, which defines the length of the request body in octets.
content_md5	`string`	Specifies the HTTP Request Content-MD5 header field, which defines a Base64 encoded binary MD5 sum of the content of the request body.
content_type	`string`	Specifies the HTTP Request Content-Type header field, which defines a the MIME type of the body of the request (used with POST and PUT requests).
date	`timestamp`	Specifies the HTTP Request Date header field, which defines the date and time that the message was sent.

expect	`string`	Specifies the HTTP Request Expect header field, which defines the particular server behaviors that are required by the client.
from	`email-addr-object`	Specifies the HTTP Request From header field, which defines the email address of the user making the request.
host	`string`	Specifies the HTTP Request Host header field, which the domain name of the server and the TCP port number on which the server is listening.
if_match	`string`	Specifies the HTTP Request If-Match header field, which allows the action to be performed if the client supplied entity matches the same entity on the server.
if_modified_since	`string`	Specifies the HTTP Request If-Modified-Since header field, which allows a 304 Not Modified response to be returned if content is unchanged since the input date/time.
if_none_match	`string`	Specifies the HTTP Request If-Modified-Since header field, which allows the action to be performed only if the client supplied entity does not match the same entity on the server.
if_range	`string`	Specifies the HTTP Request If-Range header field, which allows the client to request the part(s) of the entity that they are missing, or otherwise the new entity.
if_unmodified_since	`string`	Specifies the HTTP Request If-Unmodified-Since header field, which allows a response to be sent only if the entity has not been modified since a specific date/time.
max_forwards	`integer`	Specifies the HTTP Request Max-Forwards header field, which defines the maximum number of times the message can be forwarded through proxies or gateways.

pragma	`string`	Specifies the HTTP Request Pragma header field, which defines any implementation-specific values that may have various anywhere along the request-response chain.
proxy_authorization	`string`	Specifies the HTTP Request Proxy-Authorization header field, which defines the authorization credentials for connecting to a proxy.
range	`string`	Specifies the HTTP Request Range header field, which defines the range, in bytes, for requesting only part of an entity (bytes are numbered from 0).
referer	`uri-object`	Specifies the HTTP Request Range Referer field, which defines the address of the previous web page from which a link to the currently requested page was followed.
te	`string`	Specifies the HTTP Request TE field, which defines the transfer encodings the user agent is willing to accept.
user_agent	`string`	Specifies the HTTP Request User-Agent field, which defines the user agent string of the user agent.
via	`string`	Specifies the HTTP Request Via field, which defines any proxies through which the request was sent.
warning	`string`	Specifies the HTTP Request Warning field, which defines any general warnings about possible problems with the entity body.
dnt	`string`	Specifies the non-standard HTTP Request DNT field, which is typically used to request that a web application disable their tracking of a user.

IP Extension (`ip-extension`)

The IP extension specifies a default extension for capturing network connection properties specific to IP. The key for this extension when used in the **`extended_properties`** dictionary MUST be *ip*.

Properties

Property Name	Type	Description
tos	`string`	

TCP Extension (`tcp-extension`)

The HTTP extension specifies a default extension for capturing network connection properties specific to TCP. The key for this extension when used in the **`extended_properties`** dictionary MUST be *tcp*.

Properties

Property Name	Type	Description
source_port	`integer`	A number from 0 - 65535
destination_port	`integer`	A number from 0 - 65535
source_flags	`string`	Source TCP flags in hexadecimal format
destination_flags	`string`	Destination TCP flags in hexadecimal format

UDP Extension (`udp-extension`)

The UDP extension specifies a default extension for capturing network connection properties specific to UDP. The key for this extension when used in the **`extended_properties`** dictionary MUST be *udp*.

Properties

Property Name	Type	Description

source_port	`integer`	A number from 0 - 65535
destination_port	`integer`	A number from 0 - 65535

ICMP Extension (`icmp-extension`)

The ICMP extension specifies a default extension for capturing network connection properties specific to ICMP. The key for this extension when used in the **`extended_properties`** dictionary MUST be *icmp*.

Properties

Property Name	Type	Description
type	`string`	The ICMP type byte in hexadecimal format
code	`string`	The ICMP code byte in hexadecimal format

Connection State Extension (`connection-state-extension`)

The Connection State extension specifies a default extension for capturing network connection properties associated with connection state. The key for this extension when used in the **`extended_properties`** dictionary MUST be *connection-state*.

Properties

Property Name	Type	Description
connection_state	`string`	The connection state with regards to the service/protocol specified in the service field.
overall_state	`string`	The overall state of the connection.

Flow Extension (`flow-extension`)

The Connection State extension specifies a default extension for capturing network connection properties associated with network flow. The key for this extension when used in the **`extended_properties`** dictionary MUST be *flow*.

Properties

Property Name	Type	Description
source_bytes	`integer`	The number of bytes sent from the source to the destination
destination_bytes	`integer`	The number of bytes sent from the destination to the source
source_packets	`integer`	The number of packets sent from the source to the destination
destination_packets	`integer`	The number of packets sent destination to the source
source_payload	`string`	Base64 encoded array of bytes sent from the source to the destination. This field can be NULL in the case of a unidirectional flow.
destination_payload	`string`	Base64 encoded array of bytes sent from the destination to the source. This field can be NULL in the case of a unidirectional flow

Workflows

```
{
  "type": "indicator",
  ...
  "pattern": {

  }
}
```

```
{
  "type": "Observation",
  "created_by_ref": "identity--1",
  "created_time": "2016-03-23T01:01:01Z",
  "observed_time": "2016-03-23T01:01:01Z",
  "observable" : {
    "type": "email",
    "subject": "Please click this!",
    "attachments": [
      {
        "type": "file",
        "file_name": "malicious.doc.exe",
        "hashes": {
          "md5": "3773a88f65a5e780c8dff9cdc3a056f3",
          "x_foo_hash": "aaaabbbbccccddddeeeeffff0123457890"
      }
    ]
  }
}

{
  "type": "network-connection",
  "id": "network-connection--1",
  "spec_version": "cybox-3.0",
  "802.3": {
    "ethernet-src": "aa:bb:cc:dd:ee:ff",
    "ethernet-dst": "11:22:33:44:55:66",
    "ipv4": {
      "ip-src": "10.1.2.3",
      "ip-dst": "123.123.123.123"
    }
  }
} // the one reason I did it this way was to allow for things like ATM,
FIDDI, Zigbee, 802.11, etc.  And it is just meant to show how it could be,
not that this is the right way…  Basically I do not want 20 small objects
for a network session to each be linked together via a different
relationship object. When you want to tell a device or tell someone about
something on the network, the it is by definition all related anyway, so we
should just embed that information.  Today this would be done by taking a
```

screen shot from a wireshark interface, for example, or from one of the many network forensics tools.

```
{
  “type”: “network-frame”,
  “spec_version”: “cybox-3.0”,
  “802.3”: {
    “ethernet-src”: “aa:bb:cc:dd:ee:ff",
    "ethernet-dst": "11:22:33:44:55:66",
    “ip”: {
      “version”: 4,
      "ip-src": "10.1.2.3",
      "ip-dst": "123.123.123.123"
      “tcp”: {
        “port-src”: 38453,
        “port-dst”: 22,
        “data”: “jweoifjsdoijef”
     }
   },
  "802.11": {
    "bssid": "foo",
    ...
  }
}
```

```
{
  "type": "ip-addr"
}
```

1. Share IP addresses in a watchlist
    a. Sends indicator with list of IPs
    b. One of the recipients sends back a sighting of one IP in that watchlist.
2. Represent domains in a watchlist

Object ←→ Object Relationships

The Relationship object is used to link together CybOX Objects.

Supported Relationship Classes

We still need to define the classes of relationships that we want to support the expression of; this will be partly based on the design of our CybOX Object data models, as well as on the existing set of Object relationship from CybOX 2.x.

Notionally, it's likely that we'll support the following classes of relationships:

- Container
    - E.g., Object **A** *contains* Object **B**
- Contextual
    - E.g., Object **A** *resolves to* Object **B**

Other classes of relationships, especially those around Objects that are *used by* other Objects (i.e., inside of an Object field), will instead be represented by embedded Objects.

Implications

There are a few import implications associated with the use/support of Object → Object relationships in CybOX:

- This will have a direct impact on the CybOX Object data model and how Objects are designed
- Objects will require IDs so that they can be referenced in relationships
- Relationships will need to be clearly documented and/or constrained, so that there aren't multiple ways of expressing the same "fact"

Open Questions

- Should CybOX Object IDs be local IDs or GUIDs?
    - *Current thinking*: they should be local IDs, so that they are unique only to the container in which they are encapsulated. Therefore, they are not necessarily TLOs, and do not need to be versioned, etc.
- How should CybOX Object IDs be defined?
    - Should we permit CybOX Object IDs to be overloaded?
    - *Current thinking*: they should be defined as simple integers, so that they are not mistaken for UUIDs.
- How should CybOX Object relationships be scoped?
    - Locally?
        - They are applicable only in the context of the container in which they are encapsulated

            - E.g., only in the scope of a particular STIX Observation
        - Globally?
            - They are applicable in the context of the entire document in which they are encapsulated
                - E.g., in the scope of an entire STIX Package
        - *Current thinking*: this is something that we need to think through, though it likely makes more sense to support only locally scoped relationships.
            - Are there specific use cases where global relationships are necessary?
                - What if we want to relate two CybOX Objects that are different Observations?
- How should CybOX Object relationships reference their source/targets?
    - By ID?
    - By position in an array?

Properties

Property Name	Type	Description
type (required)	`string`	The value of this field **MUST** be `object-relationship`
kind_of_relationship (required)	`string`	The descriptor for this relationship.
kind_of_relationship_ext (optional)	`vocab-ext`	The descriptor for this relationship, using a non-standard vocabulary.
source_ref (required)	`string`	The ID of the source (from) object.
target_ref (required)	`string`	The ID of the target (to) object.
is_directional (required)	`boolean`	Indicates whether the relationship is directional. For values in the standard vocabulary, this attribute may be hardcoded to a particular value.

Examples

Description	GitHub Link
Domain Redirection/Resolution	JSON Example

Actions

CybOX Actions are intended to represent effects on CybOX Objects, as perpetuated by another CybOX Object. Examples include the creation of a file, the reading of a Windows registry key value, and the initiation of a network connection.

Properties

Inherits From	Inherited Properties	
`cybox-base`	**`type, id, spec_version, created_time`**	
Property Name	**Type**	**Description**
`type` (required)	`string`	Indicates that this object is a CybOX Action. The value of this field **MUST** be `cybox-action`
`name`	`string`	The name of the Action.
`name_ext`	`vocab-ext`	The name of the Action, using a non-standard vocabulary.
`associated_objects`	`array` of type `associated-object`	The CybOX Objects associated with, affected by, or used in, the action.

Associated Object (`associated-object`)

The Associated Object type represents a reference to a CybOX Object along with the nature of how it was associated with, affected by, or used in a CybOX Action.

Property Name	Type	Description
`type` (required)	`string`	Indicates that this object is a CybOX Associated Object. The value of this field **MUST** be `associated-object`
`object_ref`	`string`	A reference to a CybOX Object that was associated with a CybOX Action.
`association`	`string`	The nature of the association

		between the Object and an Action.
association_ext	`vocab-ext`	The nature of the association between the Object and an Action, using a non-standard vocabulary.

Examples

Description	GitHub Link
Create File	JSON Example
Read Registry Key Value	JSON Example