CybOX Release Notes

This document provides a high-level summary of the changes between CybOX 1.0 and 2.0.

Cross-Cutting Changes

This section describes changes that have broad impact across all schemas in the CybOX Language.

- Many elements, attributes, and types were renamed to try to employ more uniform naming conventions across the schema. In most places, the new names are only slightly different from the old names (removing hyphens and using underscores and/or changing capitalization). A few exceptions:
    - It was felt calling the fields characterizing some detail of an object "attributes" (e.g., BaseObjectAttributeType) might lead to confusion with the use of XML attributes when describing CybOX fields. As such, Object fields are now called "properties" and all types used for characterizing Object fields that included "attribute" in their name replaced that with "property". (E.g., BaseObjectAttributeType became BaseObjectPropertyType)
    - DefinedObjectType and the element Defined_Object that used it were felt to be unclear names. These changed to ObjectPropertiesType and Properties, respectively, to reflect the fact that they are used to contain a list of properties associated with a specific Object.
- Significant changes to BaseObjectPropertyType and its derived types. BaseObjectPropertyType is the base type for all Object property fields. As all Objects use extensions of this type to define their properties, this change has cross cutting impact.
    - Derived types of BaseObjectPropertyType no longer enforce type conformance but instead allow any string to appear. For example, previously IntegerObjectPropertyType, a derived type from BaseObjectPropertyType, restricted its contents to being XML integers. However, this precluded the use of some conditions such as regular expressions. Now IntegerObjectPropertyType allows any string for its value. The expectation is that the value of a field using this type would be an integer, but the XML schema does not enforce this. However, now authors can use patterns in this field without causing validation errors.
    - Previously, the body of a BaseObjectPropertyType (or one of its derived types) was expected to be a singleton. If authors wished to specify a list of values, they needed to use the @value_set attribute and leave the body of the property field blank. Now @value_set has been removed and all values associated with a property, be they a list or a singleton, appear within the body of the property. Lists are separated by commas. Commas are now a reserved character in property bodies, so if authors wish to include a comma that is not meant to be a list separator then it needs to be escaped by replacing it with ",".
- Significant structuring to the ObjectPropertyGroup. This attribute group defined attributes used by Object property fields. Specifically, it included information on characterizing an observed object property, identification of the property (id/idref), and for describing patterns on that property. As all Objects utilize this attribute group in their properties (indirectly) changes to this attribute group have broad impact.
    - The attribute group was broken into two attribute groups: PatternFieldGroup, which includes attributes for describing patterns (condition, pattern_type, etc.), and

BaseObjectPropertyGroup, which contains all other attributes specifically associated with Object properties (id, idref, appears_random, etc.)
    - Added new attributes to characterize obfuscation and defanging of an Object's property (to BaseObjectPropertyGroup)
    - Added a mask field to support bitwise operators for describing patterns on Object properties (to PatternFieldGroup)
    - Removed the value_set, start_range, and end_range attributes. Depending on the condition, an Object property's values might appear in these attributes instead of in the body of the Object property. Object properties have been refactored and now values are always in the body regardless of the pattern condition
    - Previously the use of regular expressions involved picking a regex syntax from a fairly exhaustive enumeration and then using that particular format in defining one's regular expression. While flexible, this approach made compatibility extremely hard as one could never know which syntax one might see. Instead, CybOX now defines a special subset of regular expression syntax which is compatible with several widely uses regular expression syntaxes. It is expected that authors should use this common regular syntax. If authors do so, they do not need to identify a syntax using the @regex_syntax attribute. Authors are still free to use other syntaxes if they need special capabilities not found in the common subset and in this case they should provide a value in the @regex_syntax attribute. @regex_syntax no longer uses an enumeration of syntax names but is simply a string field - this reflects that use of a regex syntax other than the defined common syntax should be viewed as a customization and may not be understood by all users
- Significant redesign of the use of Custom Properties in Objects. Previously, the expectation had been that one would either use a defined CybOX object or one would define an Object using only custom properties. In the new design, custom properties are now an integral part of the schema for any defined Object, allowing defined Objects to be easily extended. (A new Custom Object has was defined for Objects that are entirely defined using custom properties.) This involved moving the Custom_Properties element from the ObjectType to the ObjectPropertiesType, from which all defined CybOX Objects are derived. The @type attribute, which was previously in ObjectType and was there to characterize custom properties, has been removed; this functionality has been subsumed and expanded on by the new @name and @description attributes for individual custom property instances (i.e., PropertyType) and the <Description> and @custom_name fields of the new Custom Object.
- It was felt that many of the enumerations in CybOX needed to be easily extensible. To support this, a separate CybOX Vocabularies schema was defined (cybox_default_vocabularies.xsd). This defines several controlled vocabularies all of which extend a base ControlledVocabularyStringType defined in cybox_common.xsd. Authors can define their own extension from this base type and use their own custom vocabulary instead of or in addition to the default vocabularies provided with the CybOX release. Authors can also just provide string values in these fields without formally defining an associated vocabulary.
- Created the concept of "Extension Schemas" for CybOX. These schemas utilize externally defined schemas (to avoid having CybOX reinvent existing schemas that fill a particular need). To avoid directly importing all such schemas, these schemas are utilized indirectly through well-defined extension points using the xsi:Type XML capability. This allows authors to use a simple default (without any need to import an external schema), use the provided extension schema for more detailed characterization, or define their own custom structure to characterize some concept. Currently, the only use of Extension Schemas in CybOX is limited to the specification of

software and hardware platform identity, where a default extension has been created pointing to NIST's Common Platform Enumeration (CPE) Applicability Language schema. Future releases of CybOX may make further use of Extension Schemas if beneficial.
- Removed all instances of xs:any and xs:anyAttribute in favor of more well-defined extension mechanisms
- Annotations added and refined across all schemas

Core and Common Changes

This section describes changes to the core structures of CybOX as expressed in the cybox_core.xsd and cybox_common.xsd schemas.

- Changed the namespaces of both schemas
- Renamed cybox_common_types.xsd to cybox_common.xsd
- Removed circular imports from the common and core schemas by moving some type/element definitions. In addition, in some places common entities made use of structures defined in specific Object schemas. For example, MeasuredSourceType in the common schema includes the ability to characterize a system and previously imported the System Object for this purpose, creating a circular reference of imports since the System Object imports the common schema. To avoid the circular import while still allowing use of structures defined in CybOX objects, an abstract type is utilized that can be instantiated through the use of an xsi:Type attribute in CybOX content. This allows properties defined in CybOX Objects (such as the System Object) to be used by fields in the core and common schemas without creating circular imports, which can be problematic for some XML parsers.
- Replaced the earlier custom platform identification structures with a simple platform specification type. This type is designed to be extensible to allow more expressive platform characterization
- Removed the Stateful_Measure element from ObjectType. It was felt this was not adding any benefit
- Replaced the earlier custom structured text structures with a simple string. Use of structuring is now supported by letting authors use CDATA blocks to hide markup from validation engines and having the markup language used identified using the @structuring_format attribute.
- Added a new XOR bitwise operator for describing patterns on Object properties
- The @type attribute in ActionType is now optional
- Add a field to characterize the type of a tool and to provide references to additional information to the ToolInformationType
- Added new default object relationships (usable in RelatedObjectsType)
- When composing observables, removed the NOT operator from the enumeration (as its meaning could be ambiguous) and instead added a boolean @negate attribute
- Removed the StateType as, after removal of redundant components, it was just an alias for ObjectType. All previous uses of StateType now use ObjectType
- Within the Observable type, renamed Ease_of_Obfuscation and Obfuscation_Techniques to Ease_of_Evasion and Evasion_Techniques, respectively, in order to avoid possible confusion with the new Obfuscation attributes associated with Object properties
- Moved Ease_of_Evasion, Evasion_techniques, and Noisiness (in ObservableType) within a common parent element called Pattern_Fidelity to more clearly indicate their use
- Removed ActionType/@network_protocol as it was now redundant.

- Broke out all anonymous types and turned them into named types to simplify binding generation.
- Significant refactoring of CybOX Action capabilities (in ActionType and other types that it uses)
- Removed extraneous references to other standards (SAFES, CWE, CAPEC, etc.) These are not used directly by CybOX and so references were unnecessary
- Created a new PatternableFieldType (using the newly broken out PatternFieldGroup attribute group) for using in characterizing fields that are not Object properties, but which could nonetheless hold pattern information.

CybOX Object Changes

This section describes specific structural changes to certain CybOX Objects. Note that some of the cross-cutting issues described above affect CybOX Objects. As such, all previously existing CybOX objects have changed to some degree. This section lists CybOX Object that had structural changes went beyond the previously mentioned cross-cutting changes.

- Address - Restructured properties
- Code - New properties
- DNS Query - New properties
- File - New properties
- Email Message - New properties and regrouping of existing properties
- HTTP Session - New properties
- Library - New properties
- Memory - New properties
- Network Connection - New properties
- Network Flow - Restructured properties
- Network Packet - Restructured properties
- Process - New properties. Some properties moved to the Network Connection Object
- Socket - Restructured properties
- System - Restructured properties
- URI Object - New properties
- WHOIS - New properties
- Win Executable File - Multiple fixes
- Win Network Route Entry Object - Added an enumeration for protocols
- Win Service - Multiple fixes
- Win Waitable Timer - Clarified fields
- X509 Certificate - Multiple fixes to properties

The following objects were renamed:

- Renamed the Windows Registry Object to "Windows Registry Key Object"
- Renamed the Network Route Object to "Network Route Entry Object"

In addition, the following new Objects were defined:

- PDF File Object - for characterizing properties specific to PDF files
- Custom Object - for describing an object using only custom properties
- Link Object - for characterizing filesystem links
- Socket Address - for characterizing address-port combinations

In addition, for all Objects:

- Removed the version of the Object from the filename. Object versions now appear within the @version attribute and annotations within each schema file