

THE MITRE CORPORATION

# **STIX™ 1.1.1**

## **EXPLOIT TARGET SPECIFICATION (v1.1.1)**

---

APRIL 20, 2015

*The Structured Threat Information eXpression (STIX™) framework defines eight core constructs and the relationships between them for the purposes of modeling cyber threat information and enabling cyber threat information analysis and sharing. This specification document defines the Exploit Target construct, which encompasses one or more potential Courses of Action.*

## **Acknowledgements**

The authors would like to thank the STIX Community for its input and help in reviewing this document.

## **Trademark Information**

STIX, the STIX logo, and CybOX are trademarks of The MITRE Corporation. All other trademarks are the property of their respective owners.

## **Warnings**

MITRE PROVIDES STIX "AS IS" AND MAKES NO WARRANTY, EXPRESS OR IMPLIED, AS TO THE ACCURACY, CAPABILITY, EFFICIENCY, MERCHANTABILITY, OR FUNCTIONING OF STIX. IN NO EVENT WILL MITRE BE LIABLE FOR ANY GENERAL, CONSEQUENTIAL, INDIRECT, INCIDENTAL, EXEMPLARY, OR SPECIAL DAMAGES, RELATED TO STIX OR ANY DERIVATIVE THEREOF, WHETHER SUCH CLAIM IS BASED ON WARRANTY, CONTRACT, OR TORT, EVEN IF MITRE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.<sup>1</sup>

## **Feedback**

The STIX development team welcomes any feedback regarding the STIX Exploit Target Specification. Please send any comments, questions, or suggestions to [stix@mitre.org](mailto:stix@mitre.org).<sup>2</sup>

---

<sup>1</sup> For detailed information see [TOU].

<sup>2</sup> For more information about the STIX Language, please visit [STIX].

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	STIX Specification Documents .....	4
1.2	Document Conventions.....	5
1.2.1	Keywords.....	5
1.2.2	Fonts.....	5
1.2.3	UML Package References.....	6
1.2.4	UML Diagrams.....	6
1.2.4.1	Class Properties .....	6
1.2.4.2	Diagram Icons and Arrow Types.....	7
1.2.4.3	Color Coding .....	7
1.2.5	Property Table Notation .....	8
1.2.6	Property and Class Descriptions .....	8
<b>2</b>	<b>Background Information .....</b>	<b>10</b>
2.1	Exploit Target-Related Component Data Models .....	10
<b>3</b>	<b>STIX Exploit Target Data Model .....</b>	<b>12</b>
3.1	ExploitTargetVersionType Enumeration .....	15
3.2	VulnerabilityType Class .....	15
3.2.1	CVSSVectorType Class.....	18
3.2.1.1	CVSSScoreType Data Type .....	19
3.2.1.2	CVSSBaseVectorType Data Type.....	19
3.2.1.3	CVSSTemporalVectorType Data Type.....	19
3.2.1.4	CVSSEnvironmentalVectorType Data Type .....	19
3.2.2	AffectedSoftwareType Class .....	20
3.3	WeaknessType Class .....	20
3.4	ConfigurationType Class.....	21
3.5	PotentialCOAsType Class.....	21
3.6	RelatedExploitTargetsType Class .....	23
	<b>Appendix – XML Implementation.....</b>	<b>25</b>
	<b>References .....</b>	<b>26</b>

## 1 Introduction

The Structured Threat Information eXpression (STIX™) framework defines eight component data models: Observable, Indicator, Incident, TTP, ExploitTarget, CourseOfAction, Campaign, and ThreatActor. This document serves as the specification for the STIX Exploit Target Version 1.1.1 data model.

As defined within the STIX language, an Exploit Target is a vulnerability or weakness in software, systems, networks or configurations that is targeted for exploitation by the TTP of a Threat Actor. Recognizing a lack of current standardized approaches for generalized characterizations, STIX leverages community knowledge and best practices to define a new Exploit Target structure for representing exploit target information. Portions of the Exploit Target structure use existing standardized approaches to characterize vulnerabilities, weaknesses, and configurations.

More explicitly, the identifier constructs from the Common Vulnerabilities and Exposures (CVE®) and the Open Source Vulnerability Database (OSVDB) are used to identify publicly disclosed vulnerabilities. The Common Vulnerability Reporting Framework (CVRF) format is used to capture a detailed, structured characterization of vulnerabilities not identified in CVE or OSVDB (this allows for the characterization of zero-day vulnerabilities). The identifier construct from the Common Weakness Enumeration (CWE™) is used to identify weaknesses, and the identifier construct from the Common Configuration Enumeration (CCE™) is used to identify configuration issues.

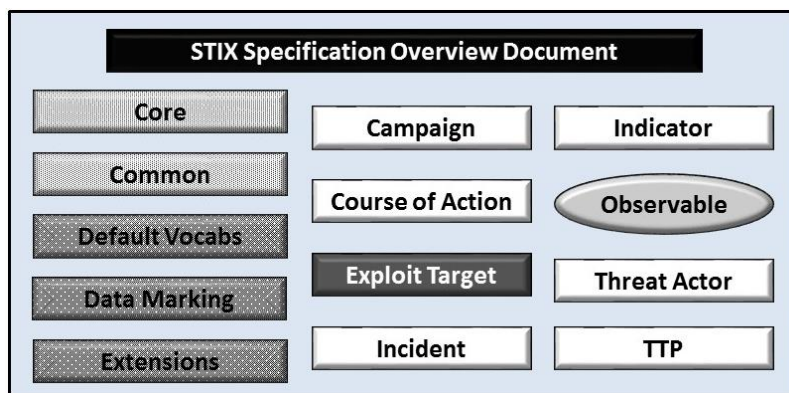
In Section 1.1 we discuss STIX specification documents, and in Section 1.2 we give document conventions. In Section 2, we give background information necessary to fully understand the Exploit Target data model, and we present the Exploit Target data model specification details in Section 3. The appendix gives information about corresponding XML implementations. References are provided in the final section.

### 1.1 STIX Specification Documents

The STIX specification corresponds to a formal UML model and a set of textual specification documents that explain the UML model. Specification documents have been written for each of the individual data models that compose the full STIX UML model.

The STIX specification overview document provides a comprehensive overview of the full set of STIX data models [STIX<sub>O</sub>], which in addition to the eight top-level component data models mentioned in the Introduction, includes a core data model, a common data model, a cross-cutting data marking data model, and a set of default controlled vocabularies. [STIX<sub>O</sub>] also summarizes the relationship of STIX to other languages, and outlines general STIX data model conventions.

Figure 1-1 illustrates the set of specification documents that are available. The color black is used to indicate the specification overview document, altered shading differentiates the overarching Core and Common data models from the supporting data models (default vocabularies, data marking, and extensions), and the color white indicates the component data models. The Observable component data model is shown as an oval shape to indicate that it is defined as a CybOX specification (see [STIX<sub>o</sub>] for details). This Exploit Target specification document is highlighted in its associated color (see Section 1.2.4.3). For a list of all STIX documents and related information sources, please see [STIX<sub>o</sub>].



**Figure 1-1.** STIX Language v1.1.1 specification documents

All specification documents can be found on this STIX Website [STIX-SPECS].

## 1.2 Document Conventions

The following conventions are used in this document.

### 1.2.1 Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in *RFC 2119* [RFC2119].

### 1.2.2 Fonts

The following font and font style conventions are used in the document:

- Capitalization is used for STIX high level concepts, which are defined in the STIX Specification Overview [STIX<sub>o</sub>].

Examples: Indicator, Course of Action, Threat Actor

- The Courier New font is used for writing UML objects.

Examples: RelatedIndicatorsType, stixCommon:StatementType

Note that all high level concepts have a corresponding UML object. For example, the Course of Action high level concept is associated with a UML class named, `CourseOfActionType`.

- The '*italic*' font (with single quotes) is used for noting actual, explicit values for STIX Language properties. The *italic* font (without quotes) is used for noting example values.

Example: '*PackageIntentVocab-1.0*,' *high*, *medium*, *low*

### 1.2.3 UML Package References

Each STIX data model is captured in a different UML package (e.g., Core package, Campaign package, etc.) where the packages together compose the full STIX UML model. To refer to a particular class of a specific package, we use the format `package_prefix:class`, where `package_prefix` corresponds to the appropriate UML package. The STIX™ 1.1.1 Specification Overview document [STIX<sub>0</sub>] contains a list of the packages used by the Exploit Target data model, along with the associated prefix notation, a description, and an example.

Note that in this specification document, we do not explicitly specify the package prefix for any classes that originate from the Indicator data model.

### 1.2.4 UML Diagrams

This specification makes use of UML diagrams to visually depict relationships between STIX Language constructs. Note that the diagrams have been extracted directly from the full UML model for STIX; they have not been constructed purely for inclusion in the specification documents. Typically, diagrams are included for the primary class of a data model, and for any other class where the visualization of its relationships between other classes would be useful. This implies that there will be very few diagrams for classes whose only properties are either a data type or a class from the STIX Common data model. Other diagrams that are included would be for classes that specialize a superclass, and for abstract or generalized classes that are extended by one or more subclasses.

In UML diagrams, classes are often presented with their attributes elided, to avoid clutter. The fully described class can usually be found in a related diagram. A class presented with an empty section at the bottom of the icon indicates that there were no other attributes than the ones that are visualized using associations.

#### 1.2.4.1 Class Properties








Generally, a class property can be shown in a UML diagram as either an attribute or an association (i.e., the distinction between attributes and associations is somewhat subjective). In order to make the size of UML diagrams in the specifications manageable, we have chosen to capture most properties as attributes and to capture only higher level

properties as associations, especially in the main top-level component diagrams. In particular, we will always capture properties of UML data types as attributes. For example, properties of a class that are identifiers, titles, and timestamps will be represented as attributes.

#### 1.2.4.2 Diagram Icons and Arrow Types

Diagram icons are used in a UML diagram to indicate whether a shape is a class, enumeration or data type, and decorative icons are used to indicate whether an element is an attribute of a class or an enumeration literal. In addition, two different arrow styles indicate either a directed association relationship (regular arrowhead) or a generalization relationship (triangle-shaped arrowhead). The icons and arrow styles we use are shown and described in Table 1-1.

**Table 1-1.** UML diagram icons

Icon	Description
	This diagram icon indicates a class. If the name is in italics, it is an abstract class.
	This diagram icon indicates an enumeration.
	This diagram icon indicates a data type.
	This decorator icon indicates an attribute of a class. The green circle means its visibility is public. If the circle is red or yellow, it means its visibility is private or protected.
	This decorator icon indicates an enumeration literal.
	This arrow type indicates a directed association relationship.
	This arrow type indicates a generalization relationship.

#### 1.2.4.3 Color Coding

The shapes of the UML diagrams are color coded to indicate the data model associated with a class. The colors used in the Indicator specification are illustrated in



Figure 1-2.



**Figure 1-2.** Data model color coding

### 1.2.5 Property Table Notation

Throughout Section 3, tables are used to describe the properties of each data model class. Each property table consists of a column of names to identify the property, a type column to reflect the datatype of the property, a multiplicity column to reflect the allowed number of occurrences of the property, and a description column that describes the property. Package prefixes are provided for classes outside of the Exploit Target data model (see Section 1.2.3).

Note that if a class is a specialization of a superclass, only the properties that constitute the specialization are shown in the property table (i.e., properties of the superclass will not be shown). However, details of the superclass may be shown in the UML diagram.

In addition, properties that are part of a “choice” relationship (e.g., Prop1 OR Prop2 is used but not both) will be denoted by a unique letter subscript (e.g., API\_Call<sub>A</sub>, Code<sub>B</sub>) and single logic expression in the Multiplicity column. For example, if there is a choice of property API\_Call<sub>A</sub> and Code<sub>B</sub>, the expression “A(1)|B(0..1)” will indicate that the API\_Call property can be chosen with multiplicity 1 or the Code property can be chosen with multiplicity 0 or 1.

### 1.2.6 Property and Class Descriptions

Each class and property defined in STIX is described using the format, “The X property verb Y.” For example, in the specification for the STIX Indicator, we write, “The id property specifies a globally unique identifier for the kill chain instance.” In fact, the verb “specifies” could have been replaced by any number of alternatives: “defines,” “describes,” “contains,” “references,” etc.

However, we thought that using a wide variety of verb phrases might confuse a reader of a specification document because the meaning of each verb could be interpreted slightly differently. On the other hand, we didn’t want to use a single, generic verb, such as “describes,” because although the different verb choices may or may not be meaningful from an implementation standpoint, a distinction could be useful to those interested in the modeling aspect of STIX.

Consequently, we have chosen to use the three verbs, defined as follows, in class and property descriptions:

Verb	STIX Definition
------	-----------------



<u>captures</u>	Used to record and preserve information without implying anything about the structure of a class or property. Often used for properties that encompass general content. This is the least precise of the three verbs.
	<p><i>Examples:</i></p> <p>The <code>Source</code> property characterizes the source of the sighting information. Examples of details <u>captured</u> include identifying characteristics, time-related attributes, and a list of the tools used to collect the information.</p> <p>The <code>Description</code> property <u>captures</u> a textual description of the Indicator.</p>
<u>characterizes</u>	Describes the distinctive nature or features of a class or property. Often used to describe classes and properties that themselves comprise one or more other properties.
	<p><i>Examples:</i></p> <p>The <code>Confidence</code> property <u>characterizes</u> the level of confidence in the accuracy of the overall content captured in the Incident.</p> <p>The <code>ActivityType</code> class <u>characterizes</u> basic information about an activity a defender might use in response to a Campaign.</p>
<u>specifies</u>	Used to clearly and precisely identify particular instances or values associated with a property. Often used for properties that are defined by a controlled vocabulary or enumeration; typically used for properties that take on only a single value.
	<p><i>Example:</i></p> <p>The <code>version</code> property <u>specifies</u> the version identifier of the STIX Campaign data model used to capture the information associated with the Campaign.</p>

## 2 Background Information

In this section, we provide high level information about the Exploit Target data model that is necessary to fully understand the Exploit Target data model specification details given in Section 3.

### 2.1 Exploit Target-Related Component Data Models

As will be explicitly detailed in Section 3, a STIX Exploit Target leverages the Course of Action data model (as indicated by the outward-oriented arrow). Figure 2-1 illustrates the relationship between the Exploit Target and the other core constructs. As stated in Section 1.1, each of these components is defined in a separate specification document.



**Figure 2-1.** High level view of the Exploit Target data model

In this section, we give a high level summary of the relationship between the Exploit Target data model and the Course of Action to which an Exploit Target may refer. We also make note of the fact that the Exploit Target data model can be self-referential. Other relationships are defined in the specification of the component from which they originate.

- **Course of Action**

A STIX Course of Action (COA) component is used to convey information about courses of action that may be taken either in response to an attack or as a preventative measure prior to an attack. A Course of Action component captures a variety of information such as the Course of Action's objective, likely impact, efficacy, and cost. Please see the STIX Course of Action data model specification [STIX<sub>COA</sub>] for details.

The Exploit Target data model references the Course of Action data model as a means to identify Courses of Actions that may be relevant in the mitigation of the Exploit Target.

- **Exploit Target**

The Exploit Target data model is self-referential, enabling one Exploit Target to reference other Exploit Targets that are asserted to be related. Self-referential

relationships between Exploit Targets may indicate general associativity or can be used to indicate relationships between different versions of the same Exploit Target.

### 3 STIX Exploit Target Data Model

The primary class of the STIX Exploit Target package is the `ExploitTargetType` class, which characterizes potential targets for exploitation by capturing characteristics of targeted victims that may make them vulnerable to attack. Similar to the primary classes of all the component data models in STIX, the `ExploitTargetType` class extends a base class defined in the STIX Common data model; more specifically, it specializes the `ExploitTargetBaseType` base class, which provides the essential identifier (`id`) and identifier reference (`idref`) properties.

The relationship between the `ExploitTargetType` class and the `ExploitTargetBaseType` base class, as well as the properties of the `ExploitTargetType` class, are illustrated in the UML diagram given in Figure 3-1.

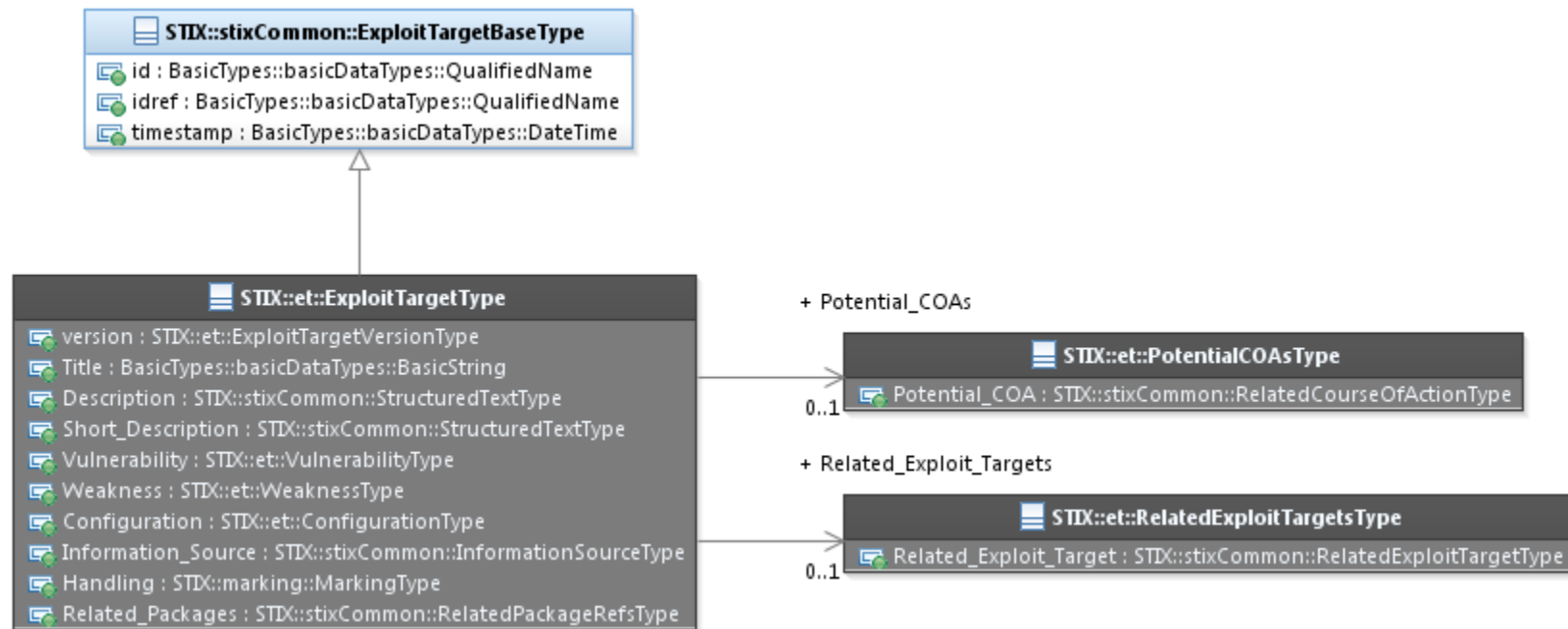


Figure 3-1. UML diagram of the `ExploitTargetType` class

The property table, which includes property descriptions and corresponds to the UML diagram given in Figure 3-1, is provided in Table 3-1.

**Table 3-1.** Properties of the `ExploitTargetType` class

Name	Type	Multiplicity	Description
<b>version</b>	<code>ExploitTargetVersionType</code>	0..1	The <code>version</code> property specifies the version identifier of the STIX Exploit Target data model used to capture the information associated with the Exploit Target.
<b>Title</b>	<code>basicDataTypes:BasicString</code>	0..1	The <code>Title</code> property captures a title for the Exploit Target and reflects what the content producer thinks the Exploit Target as a whole should be called. The <code>Title</code> property is typically used by humans to reference a particular Exploit Target; however, it is not suggested for correlation.
<b>Description</b>	<code>stixCommon:StructuredTextType</code>	0..1	The <code>Description</code> property captures a textual description of the Exploit Target. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
<b>Short_Description</b>	<code>stixCommon:StructuredTextType</code>	0..1	The <code>Short_Description</code> property captures a short textual description of the objective of this <code>CourseOfAction</code> . This property is secondary and should only be used if the <code>Description</code> property is already populated and another, shorter description is available.
<b>Vulnerability</b>	<code>VulnerabilityType</code>	0..*	The <code>Vulnerability</code> property characterizes a vulnerability that is a potential target for exploitation. Examples of information captured include a description of the vulnerability (in a structured or unstructured fashion), a CVE identifier, an OSVDB identifier, and CVSS

			information.
<b>Weakness</b>	WeaknessType	0..*	The <code>Weakness</code> property characterizes a weakness that is a potential target for exploitation. Examples of information captured include a description of the weakness and a CWE identifier.
<b>Configuration</b>	ConfigurationType	0..*	The <code>Configuration</code> property characterizes a configuration that is a potential target for exploitation. Examples of information captured include a description of the configuration issue and a CCE identifier.
<b>Potential_COAs</b>	PotentialCOAsType	0..1	The <code>Potential_COAs</code> property specifies a set of one or more Course of Actions that may be relevant for the remediation or mitigation of this Exploit Target.
<b>Information_Source</b>	stixCommon: InformationSourceType	0..1	The <code>Information_Source</code> property characterizes the source of the Exploit Target information. Examples of details captured include identifying characteristics, time-related attributes, and a list of tools used to collect the information.
<b>Handling</b>	marking:MarkingType	0..1	The <code>Handling</code> property specifies the appropriate data handling markings for the properties of this Exploit Target. The marking scope is limited to the Exploit Target and the content it contains. Note that data handling markings can also be specified at a higher level.
<b>Related_Exploit_Targets</b>	RelatedExploitTargetsType	0..1	The <code>Related_Exploit_Targets</code> property specifies a set of one or more other Exploit Targets related to this Exploit Target.
<b>Related_Packages</b>	stixCommon: RelatedPackagesRefsType	0..1	The <code>Related_Packages</code> property specifies a set of one or more STIX Packages that are related to the Exploit Target.

### 3.1 ExploitTargetVersionType Enumeration

The `ExploitTargetVersionType` enumeration is an inventory of all versions of the Exploit Target data model that are valid in STIX Version 1.1.1. The enumeration literals are given in Table 3-2.

**Table 3-2.** Literals of the `ExploitTargetVersionType` enumeration

Enumeration Literal	Description
<b>1.0</b>	Exploit Target data model Version 1.0
<b>1.0.1</b>	Exploit Target data model Version 1.0.1
<b>1.1</b>	Exploit Target data model Version 1.1
<b>1.1.1</b>	Exploit Target data model Version 1.1.1

### 3.2 VulnerabilityType Class

The `VulnerabilityType` class characterizes an individual vulnerability. In addition to capturing basic information and references to vulnerability registries, this class is extensible to enable the structured description of a vulnerability. STIX v1.1.1 defines a default subclass to leverage the Common Vulnerability Reporting Format (CVRF) schema<sup>3</sup>: the `CVRF1.1InstanceType` class (see [STIX<sub>EXT</sub>]).

---

<sup>3</sup> There is no UML model defined for the CVRF; it is outside the scope of the STIX 1.1.1 specification.

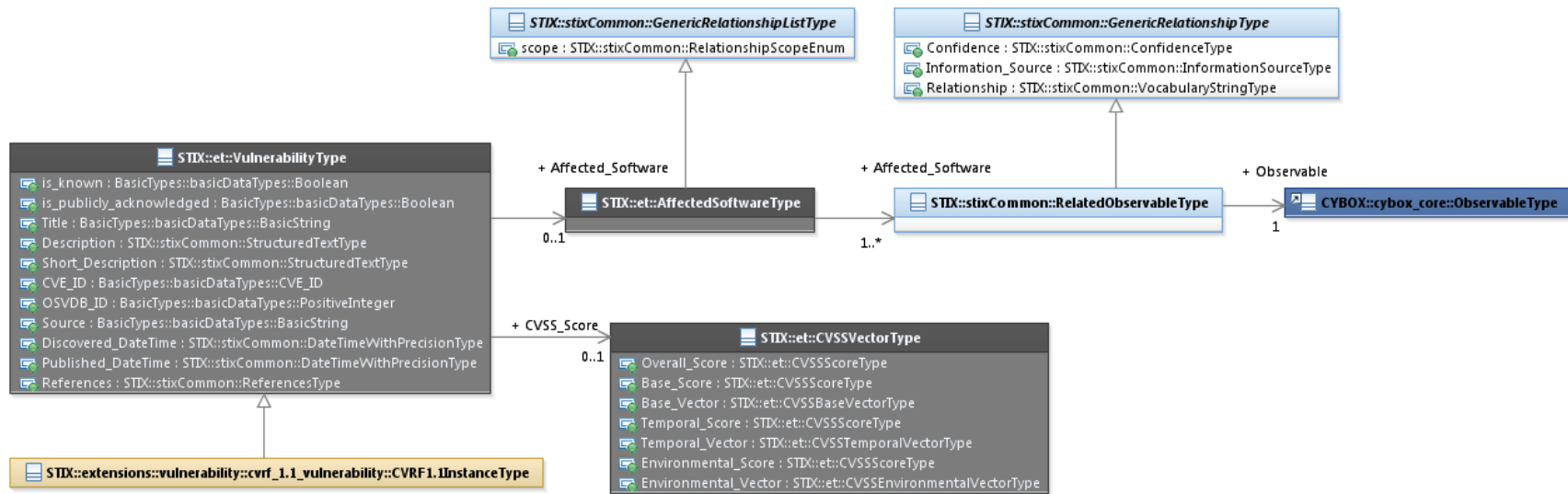


Figure 3-2. UML diagram of the VulnerabilityType class

Table 3-3. Properties of the VulnerabilityType class

Name	Type	Multiplicity	Description
<b>is_known</b>	<code>basicDataTypes:Boolean</code>	0..1	The <code>is_known</code> property specifies whether or not the vulnerability is known (i.e., not a 0-day) or unknown (i.e., a 0-day) at the time it is characterized.
<b>is_publicly_acknowledged</b>	<code>basicDataTypes:Boolean</code>	0..1	The <code>is_publicly_acknowledged</code> property specifies whether or not the vulnerability is publicly acknowledged by the vendor at the time it is characterized.
<b>Title</b>	<code>basicDataTypes:String</code>	0..1	The <code>Title</code> property captures a title for the vulnerability and reflects what the content producer thinks the vulnerability as a whole should be called. The <code>Title</code> property is typically used by humans to reference a particular vulnerability;



			however, it is not suggested for correlation.
<b>Description</b>	stixCommon: StructuredTextType	0..1	The <code>Description</code> property captures a textual description of the vulnerability. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
<b>Short_Description</b>	stixCommon: StructuredTextType	0..1	The <code>Short_Description</code> property captures a short textual description of the vulnerability. This property is secondary and should only be used if the <code>Description</code> property is already populated and another, shorter description is available.
<b>CVE_ID</b>	basicDataTypes:CVE_ID	0..1	The <code>CVE_ID</code> property specifies a Common Vulnerability and Exposures (CVE) identifier for the vulnerability.
<b>OSVDB_ID</b>	basicDataTypes: PositiveInteger	0..1	The <code>OSVDB_ID</code> property specifies an Open Source Vulnerability Database (OSVDB) identifier for the vulnerability.
<b>Source</b>	basicDataTypes:String	0..1	The <code>Source</code> property captures a textual description or a URL of the original source of the vulnerability information.
<b>CVSS_Score</b>	CVSSVectorType	0..1	The <code>CVSS_Score</code> property captures the full Common Vulnerability Scoring System (CVSS) v2.0 base, temporal, and environmental vectors.
<b>Discovered_DateTime</b>	stixCommon: DateTimeWithPrecisionType	0..1	The <code>Discovered_DateTime</code> property specifies the date and time at which the vulnerability was discovered. To avoid ambiguity, all timestamps SHOULD include a specification of the time zone. In addition to specifying a date and time, the <code>Date_Time</code> property may also capture a <code>precision</code> property to specify the granularity with which the time should be considered, as specified by the <code>DateTimePrecisionEnum</code> enumeration (e.g., <code>'hour,'</code> <code>'minute'</code> ). If omitted, the default precision is <code>'second.'</code> Digits in a timestamp that are beyond the specified precision SHOULD be zeroed out.

<b>Published_DateTime</b>	<code>stixCommon:DateTimeWithPrecisionType</code>	0..1	The <code>Published_DateTime</code> property specifies the date and time at which information about the vulnerability was published. To avoid ambiguity, all timestamps SHOULD include a specification of the time zone. In addition to specifying a date and time, the <code>Date_Time</code> property may also capture a <code>precision</code> property to specify the granularity with which the time should be considered, as specified by the <code>DateTypePrecisionEnum</code> enumeration (e.g., <code>'hour'</code> , <code>'minute'</code> ). If omitted, the default precision is <code>'second.'</code> Digits in a timestamp that are beyond the specified precision SHOULD be zeroed out.
<b>Affected_Software</b>	<code>AffectedSoftwareType</code>	0..1	The <code>Affected_Software</code> property specifies a set of one or more software products that is affected by this vulnerability. It leverages the CybOX <code>ObservableType</code> class.
<b>References</b>	<code>stixCommon:ReferencesType</code>	0..1	The <code>References</code> property specifies a set of one or more related references associated with the vulnerability.

### 3.2.1 CVSSVectorType Class

The `CVSSVectorType` class characterizes Common Vulnerability Scoring System (CVSS) data associated with the vulnerability.

The property table for the `CVSSVectorType` class is given in Table 3-4.

**Table 3-4.** Properties of the `CVSSVectorType` class

Name	Type	Multiplicity	Description
<b>Overall_Score</b>	<code>CVSSScoreType</code>	0..1	The <code>Overall_Score</code> property specifies the CVSS 2.0 overall score. Note that this is not the same as the unadjusted CVSS base score, which should be specified in the <code>Base_Score</code> property.

<b>Base_Score</b>	CVSSScoreType	0..1	The <code>Base_Score</code> property specifies the unadjusted CVSS 2.0 base score.
<b>Base_Vector</b>	CVSSBaseVectorType	0..1	The <code>Base_Vector</code> property specifies the CVSS 2.0 base vector.
<b>Temporal_Score</b>	CVSSScoreType	0..1	The <code>Temporal_Score</code> property specifies the CVSS 2.0 temporal score.
<b>Temporal_Vector</b>	CVSSTemporalVectorType	0..1	The <code>Temporal_Vector</code> property specifies the CVSS 2.0 temporal vector.
<b>Environmental_Score</b>	CVSSScoreType	0..1	The <code>Environmental_Score</code> property specifies the CVSS 2.0 environmental score.
<b>Environmental_Vector</b>	CVSSEnvironmentalVectorType	0..1	The <code>Environmental_Vector</code> property specifies the CVSS 2.0 environmental vector.

### 3.2.1.1 CVSSScoreType Data Type

The `CVSSScoreType` data type specializes the `basicDataTypes:BasicString` data type by restricting it to the pattern: `((10)|[0-9])\.[0-9]`

### 3.2.1.2 CVSSBaseVectorType Data Type

The `CVSSBaseVectorType` data type specializes the `basicDataTypes:BasicString` data type by restricting it to the pattern: `AV:[LAN]/AC:[HML]/Au:[MSN]/C:[NPC]/I:[NPC]/A:[NPC]`

### 3.2.1.3 CVSSTemporalVectorType Data Type

The `CVSSTemporalVectorType` data type specializes the `basicDataTypes:BasicString` data type by restricting it to the pattern: `E:([UFH]|(POC)|(ND))/RL:([WU]|(OF)|(TF)|(ND))/RC:([C]|(UC)|(UR)|(ND))`

### 3.2.1.4 CVSSEnvironmentalVectorType Data Type

The `CVSSEnvironmentalVectorType` data type specializes the `basicDataTypes:BasicString` data type by restricting it to the pattern: `CDP:([NLH]|(LM)|(MH)|(ND))/TD:([NLMH]|(ND))/CR:([LMH]|(ND))/IR:([LMH]|(ND))/AR:([LMH]|(ND))`

### 3.2.2 AffectedSoftwareType Class

The `AffectedSoftwareType` class specifies a set of platforms and software that are affected by a vulnerability. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The property table for the `AffectedSoftwareType` class is given in Table 3-5.

**Table 3-5.** Properties of the `AffectedSoftwareType` class

Name	Type	Multiplicity	Description
<b>Affected_Software</b>	<code>stixCommon:RelatedObservableType</code>	1..*	The <code>Affected_Software</code> property characterizes a single software product or platform affected by this vulnerability.

### 3.3 WeaknessType Class

The `WeaknessType` class characterizes a weakness as a potential Exploit Target.

The property table for the `WeaknessType` class is given in Table 3-6.

**Table 3-6.** Properties of the `WeaknessType` class

Name	Type	Multiplicity	Description
<b>Description</b>	<code>stixCommon:StructuredTextType</code>	0..1	The <code>Description</code> property captures a textual description of the weakness. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
<b>CWE_ID</b>	<code>basicDataTypes:CWE_ID</code>	0..1	The <code>CWE_ID</code> property specifies a Common Weakness Enumeration (CWE) identifier for a particular weakness.

### 3.4 ConfigurationType Class

The `ConfigurationType` class characterizes a software or hardware configuration as a potential Exploit Target.

The property table for the `WeaknessType` class is given in Table 3-7.

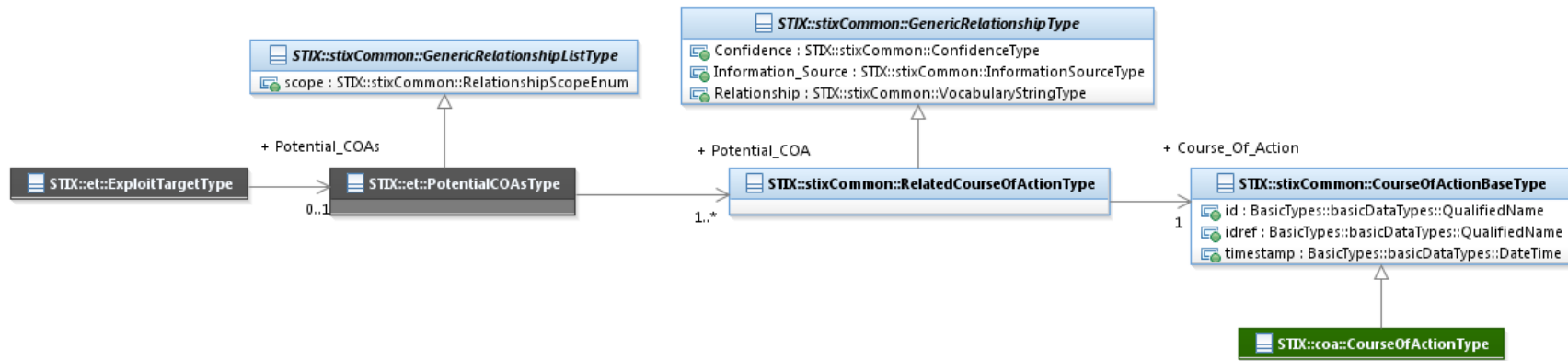
**Table 3-7.** Properties of the `ConfigurationType` class

Name	Type	Multiplicity	Description
<b>Description</b>	<code>stixCommon:StructuredTextType</code>	0..1	The <code>Description</code> property captures a textual description of the configuration. Any length is permitted. Optional formatting is supported via the <code>structuring_format</code> property of the <code>StructuredTextType</code> class.
<b>Short_Description</b>	<code>stixCommon:StructuredTextType</code>	0..1	The <code>Short_Description</code> property captures a short textual description of the configuration. This property is secondary and should only be used if the <code>Description</code> property is already populated and another, shorter description is available.
<b>CCE_ID</b>	<code>basicDataTypes:CCE_ID</code>	0..1	The <code>CCE_ID</code> property specifies a Common Configuration Enumeration (CCE) identifier for a particular configuration item.

### 3.5 PotentialCOAsType Class

The `PotentialCOAsType` class specifies a set of one or more potential Course of Actions (COAs) for the Exploit Target. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The UML diagram corresponding to the `PotentialCOAsType` class is shown in Figure 3-3.



**Figure 3-3.** UML diagram of the PotentialCOAsType class

The property table given in Table 3-8 corresponds to the UML diagram shown in Figure 3-3.

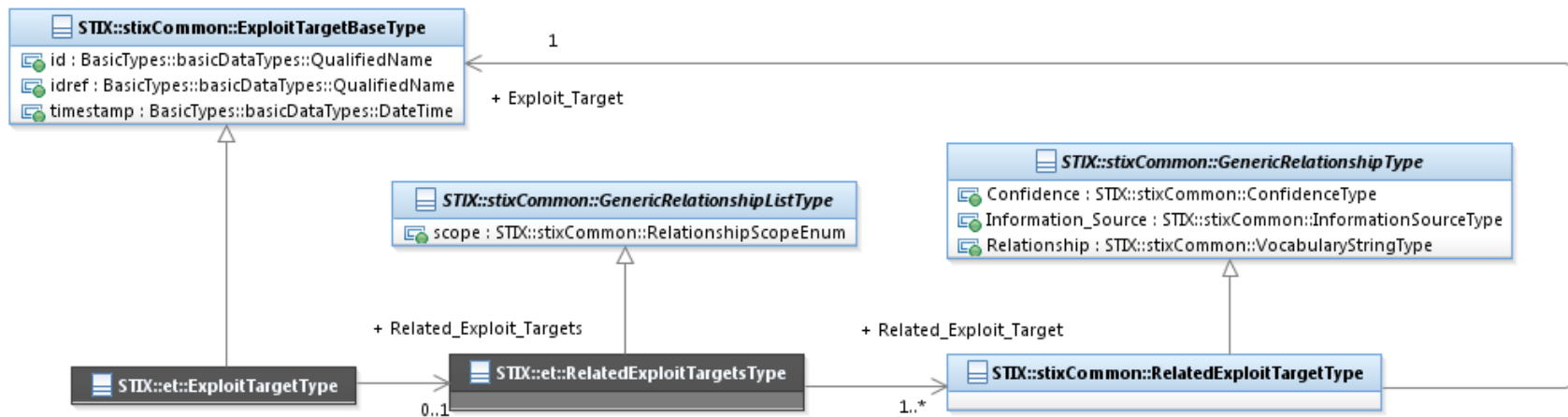
**Table 3-8.** Properties of the PotentialCOAsType class

Name	Type	Multiplicity	Description
<b>Potential_COA</b>	stixCommon:RelatedCourseOfActionType	1..*	The Potential_COA property specifies a Course of Action potentially relevant for the remediation or mitigation of this Exploit Target and characterizes this relevance relationship by capturing information such as the level of confidence that the Course of Action and the Exploit Target are related, the source of the relationship information, and the type of relationship.

### 3.6 RelatedExploitTargetsType Class

The `RelatedExploitTargetsType` class specifies a set of one or more other Exploit Targets asserted as related to this Exploit Target and therefore is a self-referential relationship. It extends the `GenericRelationshipListType` superclass defined in the STIX Common data model, which specifies the scope (whether the elements of the set are related individually or as a group).

The UML diagram corresponding to the `RelatedExploitTargetType` class is shown in Figure 3-4.



**Figure 3-4.** UML diagram of the `RelatedExploitTargetsType` class

The property table given in Table 3-9 corresponds to the UML diagram shown in Figure 3-4.

**Table 3-9.** Properties of the `RelatedExploitTargetsType` class

Name	Type	Multiplicity	Description
<b>Related_Exploit_Target</b>	<code>stixCommon:RelatedExploitTargetType</code>	1..*	The <code>Related_Exploit_Target</code> property specifies another Exploit Target associated with this Exploit Target and characterizes the

			relationship between the Exploit Targets by capturing information such as the level of confidence that the Exploit Targets are related, the source of the relationship information, and type of the relationship. A relationship between Exploit Targets may represent assertions of general associativity or different versions of the same Exploit Target.
--	--	--	--



## **Appendix – XML Implementation**

The initial implementation for STIX v1.1.1 uses XML schema as a structured mechanism for detailed discussion, collaboration and refinement among the communities involved. The complete listing of XML representation resources can be found on the STIX website [REL].

## References

References made in this document are listed below.

- [CybOX<sub>COR</sub>] CybOX™ Core Specification (*not yet available*).
- [REL] STIX™ Exploit Target Model as implement in XSD  
[https://stix.mitre.org/language/version4.1/xxx\\_schema.xsd](https://stix.mitre.org/language/version4.1/xxx_schema.xsd)
- [RFC2119] RFC 2119 – Key words for use in RFCs to Indicate Requirement Levels  
<http://www.ietf.org/rfc/rfc2119.txt>
- [STIX] STIX™ Web Site  
<https://stix.mitre.org>
- [STIX-SPECS] STIX™ Project Github Site  
<http://github.com/STIXProject/specifications>
- [STIX<sub>COA</sub>] STIX™ 1.1.1 Course of Action (COA) Specification Version 1.1.1  
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX<sub>EXT</sub>] STIX™ 1.1.1 Extension Specification Version 1.1.1  
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX<sub>O</sub>] STIX™ 1.1.1 Specification Overview  
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [STIX<sub>TTP</sub>] STIX™ 1.1.1 TTP Specification Version 1.1.1  
<http://stix.mitre.org/about/documents/XXXX.pdf>
- [TOU] Terms of Use  
<http://stix.mitre.org/about/termsfuse.html>