TTA Standard

정보통신단체표준(영문표준)

TTAE.OT-xx.xxxx-Part1 제정일: 2018년 xx월 xx일

구조화된 위협 정보 표현 규격(STIX^{TM}) 버전 2.0 - 제1부: STIX 핵심 개념

Structured Threat Information eXpression(STIX^{TM}) Version 2.0 - Part1: STIX Core Concepts

한국정보통신기술협회
Telecommunications Technology Association

표준초안 검토 위원회　사이버보안 프로젝트그룹(PG503)

표준안 심의 위원회　정보보호 기술위원회(TC5)

	성명	소 속	직위	위원회 및 직위	표준번호
표준(과제) 제안	김종현	한국전자통신연구원	책임연구원	사이버보안 프로젝트 그룹 위원	미정
	박성민	한국인터넷진흥원	선임연구원	위원	미정
	김낙현	한국인터넷진흥원	선임연구원	위원	미정
	이철호	국가보안기술 연구소	책임연구원	위원	미정
표준 초안 작성자	김종현	한국전자통신연구원	책임연구원	사이버보안 프로젝트 그룹 위원	미정
	박성민	한국인터넷진흥원	선임연구원	위원	미정
	김낙현	한국인터넷진흥원	선임연구원	위원	미정
	이철호	국가보안기술 연구소	책임연구원	위원	미정
	염흥열	순천향대학교	교수	위원	미정
	김익균	한국전자통신연구원	책임연구원	사이버보안 프로젝트 그룹 위원	미정
사무국 담당	박수정	TTA	책임연구원	-	

본 표준 발간 이전에 접수된 지식재산권 확약서 정보는 본 표준의 '부록(지식재산권 확약서 정보)'에 명시하고 있으며, 이후 접수된 지식재산권 확약서는 TTA 웹사이트에서 확인할 수 있습니다.
본 표준과 관련하여 접수된 확약서 외의 지식재산권이 존재할 수 있습니다.

발행인 : 한국정보통신기술협회 회장
발행처 : 한국정보통신기술협회
13591, 경기도 성남시 분당구 분당로 47
Tel : 031-724-0114, Fax : 031-724-0109
발행일 : 2018.xx

서 문

1 표준의 목적

이 표준은 STIX(Structured Threat Information Expression) 2.0에서 사용하는 핵심 개념을 정의한다. STIX 핵심 개념을 정의하여 사이버 위협 인텔리전스(CTI, Cyber Threat Intelligence)를 효율적이고 일관성 있게 공유하고, 미래 가능성 있는 사이버 공격들을 효과적으로 대응할 수 있도록 한다.

2 주요 내용 요약

이 표준은 CTI 정보를 교환하는데 사용되는 언어인 STIX 2.0의 핵심 개념들(공통 데이터 형식, STIX 객체, 데이터 표시, 번들, 어휘, 사용자지정 등)을 정의한다. STIX 2.0 시리즈는 이전 버전의 활용 사례를 바탕으로 대폭 재설계되었으며, 따라서 STIX 1.2.1에서 정의했던 몇몇 객체와 속성을 생략했다. 특히, STIX 2.0은 공동 위협 분석, 위협 공유 자동화, 탐지 및 대응 자동화와 같은 다양한 기능을 제공하고 기존의 문제점을 개선하도록 설계되었다. 또한 STIX 2.0 시리즈에 포함된 객체는 CTI 공유를 위한 소비자와 생산자의 기본적인 요구사항을 충족한다.

3 인용 표준과의 비교

3.1 인용 표준과의 관련성

이 표준은 인용표준(STIX™ Version 2.0. Part1: STIX Core Concepts)을 영문 그대로 완전 수용하는 표준이다.

3.2 인용 표준과 본 표준의 비교표

TTAE.xx-xx.xxxx	STIX™ Version 2.0. Part1: STIX Core Concepts	비고
1. 소개	1. Introduction	동일
2. 공통 데이터 형식	2. Common Data Types	동일
3. STIX 객체	3. STIX Objects	동일
4. 데이터 표시	4. Data Markings	동일
5. 번들	5. Bundle	동일
6. 어휘	6. Vocabularies	동일
7. STIX 사용자 지정	7. Customizing STIX	동일
8. 적합성	8. Conformance	동일
부속서 A. 용어 사전	Appendix A. Glossary	동일
부속서 B. 감사의 글	Appendix B. Acknowledgements	동일
부속서 C. 개정이력	Appendix C. Revision History	동일

Preface

1 Purpose

This standard defines key concepts used in STIX(Structured Threat Information Expression) 2.0. Define the key concepts for STIX to share CTI efficiently and consistently respond to potential cyber attacks.

2 Summary

Structured Threat Information Expression (STIX™) is a language and serialization format used to exchange cyber threat intelligence (CTI). STIX enables organizations to share CTI with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively. STIX is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more.

3 Relationship to Reference Standards

3.1 The relationship of international standards

The standard is fully equivalent to STIX^{TM} Version 2.0. Part1: STIX Core Concepts.

3.2 Differences between International standards(recommendation) and this standard

TTAE.xx-xx.xxxx	STIX^{TM} Version 2.0. Part1: STIX Concepts	Remarks
1. Introduction	1. Introduction	Equals
2. Common Data Types	2. Common Data Types	Equals
3. STIX Objects	3. STIX Objects	Equals
4. Data Markings	4. Data Markings	Equals
5. Bundle	5. Bundle	Equals
6. Vocabularies	6. Vocabularies	Equals
7. Customizing STIX	7. Customizing STIX	Equals
8. Conformance	8. Conformance	Equals
Appendix A. Glossary	Appendix A. Glossary	Equals
Appendix B. Acknowledgements	Appendix B. Acknowledgements	Equals
Appendix C. Revision History	Appendix C. Revision History	Equals

목 차

STIX™ Version 2.0. Part 1: STIX Core Concepts

Committee Specification 01

19 July 2017

Specification URIs

This version:

http://docs.oasis-open.org/cti/stix/v2.0/cs01/part1-stix-core/stix-v2.0-cs01-part1-stix-core.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v2.0/cs01/part1-stix-core/stix-v2.0-cs01-part1-stix-core.html
http://docs.oasis-open.org/cti/stix/v2.0/cs01/part1-stix-core/stix-v2.0-cs01-part1-stix-core.pdf

Previous version:

http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.html
http://docs.oasis-open.org/cti/stix/v2.0/csprd02/part1-stix-core/stix-v2.0-csprd02-part1-stix-core.pdf

Latest version:

http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.docx (Authoritative)
http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.html
http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.pdf

Technical Committee:

OASIS Cyber Threat Intelligence (CTI) TC

Chair:

Richard Struse (Richard.Struse@HQ.DHS.GOV), DHS Office of Cybersecurity and Communications (CS&C)

Editors:

Bret Jordan (bret_jordan@symantec.com), Symantec Corp.
Rich Piazza (rpiazza@mitre.org), MITRE Corporation
John Wunder (jwunder@mitre.org), MITRE Corporation

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- (this document) *STIX™ Version 2.0. Part 1: STIX Core Concepts.* http://docs.oasis-open.org/cti/stix/v2.0/cs01/part1-stix-core/stix-v2.0-cs01-part1-stix-core.html.
- *STIX™ Version 2.0. Part 2: STIX Objects.* http://docs.oasis-open.org/cti/stix/v2.0/cs01/part2-stix-objects/stix-v2.0-cs01-part2-stix-objects.html.
- *STIX™ Version 2.0. Part 3: Cyber Observable Core Concepts.* http://docs.oasis-open.org/cti/stix/v2.0/cs01/part3-cyber-observable-core/stix-v2.0-cs01-part3-cyber-observable-core.html.
- *STIX™ Version 2.0. Part 4: Cyber Observable Objects.* http://docs.oasis-open.org/cti/stix/v2.0/cs01/part4-cyber-observable-objects/stix-v2.0-cs01-part4-cyber-observable-objects.html.
- *STIX™ Version 2.0. Part 5: STIX Patterning.* http://docs.oasis-open.org/cti/stix/v2.0/cs01/part5-stix-patterning/stix-v2.0-cs01-part5-stix-patterning.html.

Related work:

This specification replaces or supersedes:

- *STIX™ Version 1.2.1. Part 1: Overview.* Edited by Sean Barnum, Desiree Beck, Aharon Chernin, and Rich Piazza. Latest version: http://docs.oasis-open.org/cti/stix/v1.2.1/stix-v1.2.1-part1-overview.html.
- *CybOX™ Version 2.1.1. Part 01: Overview.* Edited by Trey Darley, Ivan Kirillov, Rich Piazza, and Desiree Beck. Latest version: http://docs.oasis-open.org/cti/cybox/v2.1.1/cybox-v2.1.1-part01-overview.html.

This specification is related to:

- *TAXII™ Version 2.0.* Edited by John Wunder, Mark Davidson, and Bret Jordan. Latest version: http://docs.oasis-open.org/cti/taxii/v2.0/taxii-v2.0.html.

Abstract:

Structured Threat Information Expression (STIX™) is a language for expressing cyber threat and observable information. This document defines concepts that apply across all of STIX and defines the overall structure of the STIX language.

Status:

This document was last revised or approved by the OASIS Cyber Threat Intelligence (CTI) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=cti#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/cti/.

This Committee Specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/cti/ipr.php).

Note that any machine-readable content (Computer Language Definitions) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:

[STIX-v2.0-Pt1-Core]

STIX™ Version 2.0. Part 1: STIX Core Concepts. Edited by Rich Piazza, John Wunder, and Bret Jordan. 19 July 2017. OASIS Committee Specification 01. http://docs.oasis-open.org/cti/stix/v2.0/cs01/part1-stix-core/stix-v2.0-cs01-part1-stix-core.html. Latest version: http://docs.oasis-open.org/cti/stix/v2.0/stix-v2.0-part1-stix-core.html.

Notices

CONNECTED WITH THESE STANDARDS OR THEIR COMPONENT PARTS OR ANY PROVIDED DOCUMENTATION, WHETHER OR NOT BASED UPON WARRANTY, CONTRACT, TORT, OR OTHERWISE, WHETHER OR NOT INJURY WAS SUSTAINED BY PERSONS OR PROPERTY OR OTHERWISE, AND WHETHER OR NOT LOSS WAS SUSTAINED FROM, OR AROSE OUT OF THE RESULTS OF, OR USE OF, THE STANDARDS, THEIR COMPONENT PARTS, AND ANY PROVIDED DOCUMENTATION. THE UNITED STATES GOVERNMENT DISCLAIMS ALL WARRANTIES AND LIABILITIES REGARDING THE STANDARDS OR THEIR COMPONENT PARTS ATTRIBUTABLE TO ANY THIRD PARTY, IF PRESENT IN THE STANDARDS OR THEIR COMPONENT PARTS AND DISTRIBUTES IT OR THEM "AS IS."

Table of Contents

1. 1 Introduction

Structured Threat Information Expression (STIX™) is a language and serialization format used to exchange cyber threat intelligence (CTI). STIX enables organizations to share CTI with one another in a consistent and machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively. STIX is designed to improve many different capabilities, such as collaborative threat analysis, automated threat exchange, automated detection and response, and more.

In response to lessons learned in implementing previous versions, STIX has been significantly redesigned and, as a result, omits some of the objects and properties defined in STIX 1.2.1 (see *STIX™ Version 1.2.1 Part 1: Overview*). The objects chosen for inclusion in STIX 2.0 represent a minimally viable product (MVP) that fulfills basic consumer and producer requirements for CTI sharing. Objects and properties not included in STIX 2.0, but deemed necessary by the community, will be included in future releases.

1.0 IPR Policy

This Committee Specification is provided under the Non-Assertion Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC’s web page (https://www.oasis-open.org/committees/cti/ipr.php).

1.1 Terminology

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 [RFC2119].

All text is normative except for examples, the overview (section 1.4), and any text marked non-normative.

1.2 Normative References

[IEEE 754-2008] “IEEE Standard for Floating-Point Arithmetic”, IEEE 754-2008, August 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4610935/

[ISO10646] “ISO/IEC 10646:2014 Information technology -- Universal Coded Character Set (UCS)”, 2014. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c063182_ISO_IEC_10646_2014.zip

[RFC2119] Bradner, S., “"Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, http://www.rfc-editor.org/info/rfc3339.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, http://www.rfc-editor.org/info/rfc3986.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, http://www.rfc-editor.org/info/rfc4122.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014. http://www.rfc-editor.org/info/rfc7159.txt**.**

[TLP] Traffic Light Protocol, Version 1.0 (TLP). (2016, Aug. 25). FIRST. [Online]. Available: https://first.org/tlp

1.3 Non-Normative References

[CAPEC] Common Attack Pattern Enumeration and Classification (CAPEC). (2014, Nov. 7). The MITRE Corporation. [Online]. Available: http://capec.mitre.org.

[Casey 2007] Casey, T., Threat Agent Library Helps Identify Information Security Risks

September 2007. [Online]. Available: https://communities.intel.com/servlet/JiveServlet/downloadBody/1151-102-1-1111/Threat_Agent_Library_07-2202w.pdf.

[Casey 2015] Casey, T., “Understanding Cyberthreat Motivations to Improve Defense”, Intel, February 2015. [Online]. Available: https://communities.intel.com/servlet/JiveServlet/previewBody/23856-102-1-28290/understanding-cyberthreat-motivations-to-improve-defense-paper-l.pdf.

[Goessner 2007] Goessner, S., “JSONPath - XPath for JSON”, February 2007. [Online]. Available:

http://goessner.net/articles/JsonPath/.

[JSON Schema] OASIS Cyber Threat Intelligence (CTI) TC, “cti-stix2-json-schemas”, OASIS. [Online]. Available: https://github.com/oasis-open/cti-stix2-json-schemas.

[Mell 2005] Mell, P., Kent, K. and Nusbaum, J., “Guide to Malware Incident Prevention and Handling”, NIST Special Publication 800-83, November 2005. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-83.pdf.

[VERIS] VERIS Community Database. (n.d.). [Online]. Available: http://vcdb.org/

1.4 Overview

1.4.1 Graph-Based Model

STIX 2.0 is a connected graph of nodes and edges. STIX Domain Objects define the graph nodes and STIX relationships (including STIX Relationship Objects and embedded relationships) define the edges. The full set of STIX Domain Objects and STIX Relationship Objects are known as STIX Objects. This graph-based language conforms to common analysis approaches and allows for flexible, modular, structured, and consistent representations of CTI.

1.4.2 STIX™ Domain Objects

STIX 2.0 defines a set of STIX Domain Objects (SDOs): Attack Pattern, Campaign, Course of Action, Identity, Indicator, Intrusion Set, Malware, Observed Data, Report, Threat Actor, Tool, and Vulnerability. Each of these objects corresponds to a concept commonly used in CTI. Using the building blocks of SDOs alongside STIX relationships, entities can create and share broad and comprehensive CTI.

STIX Domain Objects all share a common set of properties. These common properties provide standard capabilities such as versioning, data marking (representing how data can be shared and used), and extensibility.

STIX Domain Objects are defined in *STIX™ Version 2.0. Part 2: STIX Objects.*

1.4.3 STIX™ Relationships

A relationship is a link between STIX Objects that describes the way in which the objects are related. Most relationships are represented using STIX Relationship Objects (SROs), while other special embedded relationships are represented as ID references.

The generic Relationship object is one of two SROs and is used for most relationships in STIX. This generic Relationship object contains a property called **relationship_type** to describe more specifically what the relationship represents. This specification defines a set of known terms to use for the **relationship_type** property between SDOs of specific types. For example, the Indicator SDO defines a relationship from itself to Malware with a **relationship_type** of indicates to describe how the Indicator can be used to detect the presence of that Malware. In addition to the terms defined in the specification, STIX also allows for custom terms to be used as the relationship type.

Currently the only other SRO (besides a generic Relationship) is the Sighting relationship object. The Sighting object is used to capture cases where an entity has "seen" an SDO, such as sighting an indicator. Sighting is a separate SRO because it contains additional properties such as **count** that are only applicable to Sighting relationships. Other SROs may be defined in future versions of STIX if new relationships are identified that also require additional properties not present on the generic Relationship object.

In addition to relationships created using the SROs (Relationship and Sighting), STIX also uses ID references to represent embedded relationships. Embedded relationships are simply ID reference properties on STIX Objects that contain the ID of a different STIX Object. Embedded relationships are used when the property is an inherent part of the object and not something that a third party might add or something that might require a confidence. Because they represent a simply inherent linkage and have no other properties, an SRO is not needed to represent them. An embedded relationship can only be asserted by the creator of the object ("object creator") it is contained in.

For example, the entity that created a STIX Object is an inherent, factual part of that object and therefore that information is captured in an embedded relationship contained in the **created_by_ref** property rather than through the use of an SRO.

Embedded relationships (ID references) are described in section 3.2 and STIX Relationship Objects (SROs) are defined in section 3 of *STIX™ Version 2.0. Part 2: STIX Objects*.

1.4.4 Cyber Observables

Some parts of the STIX language require describing structured representation of observed objects and their properties in the cyber domain. These capabilities differ from the parts of STIX used to describe higher-level concepts in many ways and are therefore contained in a separate section of the specification. The Cyber Observable sections describe one or more observed data points, for example, information about a file that existed, a process that was observed running, or that network traffic occurred between two IPs. It describes the facts concerning **what** happened, but not necessarily the who or when, and never the why.

Cyber Observables are defined by two documents in this specification. *STIX™ Version 2.0. Part 3: Cyber Observable Core Concepts* describes and defines Cyber Observable Core Concepts, which are the parts of STIX that are specific to representation of cyber observables. *STIX™ Version 2.0. Part 4: Cyber Observable Objects* contains a library of Cyber Observable Objects: definitions for the types of things that can be observed.

1.4.5 STIX™ Patterning

In order to enhance detection of possibly malicious activity on networks and endpoints, a standard language is needed to describe what to look for in a cyber environment. The STIX Patterning language allows matching against timestamped Cyber Observable data (such as STIX Observed Data Objects) collected by a threat intelligence platform or other similar system so that other analytical tools and systems can be configured to react and handle incidents that might arise. STIX Patterning is a general concept that can be used anywhere, but in STIX it is currently used by the Indicator object.

STIX Patterning is defined in *STIX™ Version 2.0. Part 5: STIX Patterning*.

1.4.6 Vocabularies

Many STIX Objects contain properties whose values can be selected from a defined set of values. These sets of values are called vocabularies and are defined in STIX in order to enhance interoperability by increasing the likelihood that different entities use the same exact string to represent the same concept. If used consistently, vocabularies make it less likely that one entity refers to the energy sector as “Energy” and another as “Energy Sector”, thereby making comparison and correlation easier.

While using predefined values from STIX vocabularies is encouraged, in some cases this is not possible or desirable. STIX supports this by defining vocabularies as “open”, where entities are permitted to use values outside of the suggested vocabulary.

STIX vocabularies are defined in section 6. Properties that are defined as open vocabularies identify a suggested vocabulary from that section. For example, the Indicator **`labels`** property, as defined in section 2.5 of *STIX™ Version 2.0. Part 2: STIX Objects*, uses the Indicator Label vocabulary as defined in section 6.5.

1.4.7 Serialization

STIX is defined independent of any specific storage or serialization. However, the mandatory-to-implement (MTI) serialization for STIX 2.0 is JSON [RFC7159]. In other words, all STIX-conformant tools have to implement support for JSON and can implement support for other serializations.

JSON schemas have been developed by members of the Cyber Threat Intelligence Technical Committee and are available in the cti-stix2-json-schemas OASIS Open Repository [JSON Schema]. The JSON schemas are informative and serve as a best effort attempt to validate that STIX 2.0 content meets the structural requirements identified in this specification. This specification is the normative description of STIX 2.0.

As JSON is the MTI serialization, all examples in this document are expressed in JSON.

1.4.8 Transporting STIX™

STIX 2.0 is transport-agnostic, i.e., the structures and serializations do not rely on any specific transport mechanism. A companion CTI specification, TAXII™, is designed specifically to transport STIX Objects. STIX provides a Bundle (see section 5) as a container for STIX Objects to allow for transportation of bulk STIX data, especially over non-TAXII communication mechanisms.

1.5 Naming Requirements

1.5.1 Property Names and String Literals

In the JSON serialization all property names and string literals **MUST** be exactly the same, including case, as the names listed in the property tables in this specification. For example, the SDO common property **`created_by_ref`** must result in the JSON key name "created_by_ref". Properties marked required in the property tables **MUST** be present in the JSON serialization.

1.5.2 Reserved Names

Reserved property names are marked with a type called `RESERVED` and a description text of “RESERVED FOR FUTURE USE”. Any property name that is marked as `RESERVED` **MUST NOT** be present in STIX content conforming to this version of the specification.

1.6 Document Conventions

1.6.1 Naming Conventions

All type names, property names and literals are in lowercase, except when referencing canonical names defined in another standard (e.g. literal values from an IANA registry). Words in property names are separated with an underscore (_), while words in type names and string enumerations are separated with a hyphen (-). All type names, property names, object names, and vocabulary terms are between three and 250 characters long.

1.6.2 Font Colors and Style

The following color, font and font style conventions are used in this document:

The `Consolas` font is used for all type names, property names and literals.

type names are in red with a light red background – `threat-actor`

property names are in bold style – **`created_at`**

literals (values) are in blue with a blue background – `malicious-activity`

All relationship types are string literals, therefore they will also appear in blue with a blue background – `related-to`

In an object's property table, if a common property is being redefined in some way, then the background is dark grey.

All examples in this document are expressed in JSON. They are in `Consolas` 9-point font, with straight quotes, black text and a `light grey background`, and 2-space indentation.

Parts of the example may be omitted for conciseness and clarity. These omitted parts are denoted with the ellipses (...).

The term “hyphen” is used throughout this document to refer to the ASCII hyphen or minus character, which in Unicode is “hyphen-minus”, U+002D.

2. 2 Common Data Types

This section defines the common types used throughout STIX. These types will be referenced by the “Type” column in other sections. This section defines the names and permitted values of common types that are used in the STIX information model; it does not, however, define the meaning of any properties using these types. These types may be further restricted elsewhere in the document.

Type	Description
`boolean`	A value of `true` or `false`.
`external-reference`	A non-STIX identifier or reference to other related external content.
`float`	An IEEE 754 [IEEE 754-2008] double-precision number.
`hashes`	One or more cryptographic hashes.
`identifier`	An identifier (ID) for a STIX Domain Object, STIX Relationship Object, Bundle, or Marking Definition.
`integer`	A whole number.
`kill-chain-phase`	A name of a kill chain phase.
`list`	A sequence of values ordered based on how they appear in the list. The phrasing “`list` of type `<type>`” is used to indicate that all values within the list **MUST** conform to the specified type.
`open-vocab`	A value from a STIX open (`open-vocab`) or suggested vocabulary.
`string`	A series of Unicode characters.
`timestamp`	A time value (date and time).

2.1 Boolean

Type Name: `boolean`

A `boolean` is a value of either true or false. Properties with this type **MUST** have a value of `true` or `false`.

The JSON MTI serialization uses the JSON boolean type [RFC7159], which is a literal (unquoted) `true` or `false`.

Examples

```
{
  ...
  "summary": true,
  ...
}
```

2.2 External Reference

Type Name: `external-reference`

External references are used to describe pointers to information represented outside of STIX. For example, a Malware object could use an external reference to indicate an ID for that malware in an external database or a report could use references to represent source material.

The JSON MTI serialization uses the JSON object type [RFC7159] when representing `external-reference`.

2.2.1 Properties

Property Name	Type	Description
`source_name` (required)	`string`	The source within which the `external-reference` is defined (system, registry, organization, etc.).
`description` (optional)	`string`	A human readable description.
`url` (optional)	`string`	A URL reference to an external resource [RFC3986].
`hashes` (optional)	`hashes`	Specifies a dictionary of hashes for the contents of the **`url`**. This **SHOULD** be provided when the **url** property is present.
`external_id` (optional)	`string`	An identifier for the external reference content.

2.2.2 Requirements

In addition to the **`source_name`** property, at least one of the **`description`**, **`url`**, or **`external_id`** properties **MUST** be present.

Examples

An `external-reference` to a VERIS Community Database (VCDB) [VERIS] entry

```
{
  ...
  "external_references": [
    {
      "source_name": "veris",
      "external_id": "0001AA7F-C601-424A-B2B8-BE6C9F5164E7",
      "url": "https://github.com/vz-risk/VCDB/blob/master/data/json/0001AA7F-C601-424A-B2B8-
             BE6C9F5164E7.json",
      "hashes": {
        "SHA-256": "6db12788c37247f2316052e142f42f4b259d6561751e5f401a1ae2a6df9c674b"
      }
    }
  ],
  ...
}
```

An `external-reference` from the CAPEC™ [CAPEC] repository

```
{
  ...
  "external_references": [
    {
      "source_name": "capec",
      "external_id": "CAPEC-550"
    }
  ],
  ...
}
```

An `external-reference` from the CAPEC repository with URL

```
{
  ...
  "external_references": [
    {
      "source_name": "capec",
      "external_id": "CAPEC-550",
      "url": "http://capec.mitre.org/data/definitions/550.html"
    }
  ],
  ...
}
```

An `external-reference` to ACME Threat Intel's report document

```
{
  ...
  "external_references": [
    {
      "source_name": "ACME Threat Intel",
      "description": "Threat report",
      "url": "http://www.example.com/threat-report.pdf"
    }
  ],
  ...
}
```

An `external-reference` to a Bugzilla item

```
{
  ...
  "external_references": [
    {
      "source_name": "ACME Bugzilla",
      "external_id": "1370",
      "url": "https://www.example.com/bugs/1370"
    }
  ],
  ...
}
```

An `external-reference` to an offline threat report (i.e., e-mailed, offline, etc.)

```
{
  ...
  "external_references": [
    {
      "source_name": "ACME Threat Intel",
      "description": "Threat report"
    }
  ],
  ...
}
```

2.3 Float

Type Name: `float`

The float data type represents an IEEE 754 [IEEE 754-2008] double-precision number (e.g., a number with a fractional part). However, because the values ±Infinity and NaN are not representable in JSON, they are not valid values in STIX.

In the JSON MTI serialization, floating point values are represented by the JSON number type [RFC7159].

Examples

```
{
  ...
  "distance": 8.321,
  ...
}
```

2.4 Hashes

Type Name: `hashes`

The Hashes type represents 1 or more cryptographic hashes, as a special set of key/value pairs. Accordingly, the name of each hashing algorithm **MUST** be specified as a key in the dictionary and **MUST** identify the name of the hashing algorithm used to generate the corresponding value. This name **SHOULD** either be one of the values defined in the `hash-algorithm-ov` OR a custom value prepended with “x_” (e.g., “x_custom_hash”).

Keys **MUST** be unique in each `hashes` property, **MUST** be in ASCII, and are limited to the characters a-z (lowercase ASCII), A-Z (uppercase ASCII), numerals 0-9, hyphen (-), and underscore (_). Keys **SHOULD** be no longer than 30 ASCII characters in length, **MUST** have a minimum length of 3 ASCII characters, **MUST** be no longer than 256 ASCII characters in length.

Examples

SHA-256 and Custom Hash

```
{
  "SHA-256": "6db12788c37247f2316052e142f42f4b259d6561751e5f401a1ae2a6df9c674b",
  "x_foo_hash": "aaaabbbbccccddddeeeeffff0123457890"
}
```

2.5 Identifier

Type Name: `identifier`

An `identifier` universally and uniquely identifies a SDO, SRO, Bundle, or Marking Definition. Identifiers **MUST** follow the form *`object-type--UUIDv4`*, where *`object-type`* is the exact value (all type names are lowercase strings, by definition) from the `type` property of the object being identified or referenced and where the *`UUIDv4`* is an RFC 4122-compliant Version 4 UUID. The UUID **MUST** be generated according to the algorithm(s) defined in RFC 4122, section 4.4 (Version 4 UUID) [RFC4122].

The JSON MTI serialization uses the JSON string type [RFC7159] when representing `identifier`.

Examples

```
{
  ...
  "type": "indicator",
  "id": "indicator--e2e1a340-4415-4ba8-9671-f7343fbf0836",
  ...
}
```

```
{
  ...
  "type": "threat-actor",
  "id": "threat-actor--5ee9db36-4a1e-4dd4-bb32-2551eda97f4a",
  ...
}
```

2.6 Integer

Type Name: `integer`

The integer data type represents a whole number. Unless otherwise specified, all integers **MUST** be capable of being represented as a signed 64-bit value ([-(2**63)+1, (2**63)-1]). Additional restrictions **MAY** be placed on the type as described where it is used.

In the JSON MTI serialization, integers are represented by the JSON number type [RFC7159].

Examples

```
{
  ...
  "count": 8,
  ...
}
```

2.7 Kill Chain Phase

Type Name: `kill-chain-phase`

The `kill-chain-phase` represents a phase in a kill chain, which describes the various phases an attacker may undertake in order to achieve their objectives.

The JSON MTI serialization uses the JSON object type [RFC7159] when representing `kill-chain-phase`.

Property Name	Type	Description
`kill_chain_name` (required)	`string`	The name of the kill chain. The value of this property **SHOULD** be all lowercase (where lowercase is defined by the locality conventions) and **SHOULD** use hyphens instead of spaces or underscores as word separators.
`phase_name` (required)	`string`	The name of the phase in the kill chain. The value of this property **SHOULD** be all lowercase (where lowercase is defined by the locality conventions) and **SHOULD** use hyphens instead of spaces or underscores as word separators.

When referencing the Lockheed Martin Cyber Kill Chain™, the **`kill_chain_name`** **MUST** be `lockheed-martin-cyber-kill-chain`.

Examples

Example specifying the “reconnaissance” phase from the Lockheed Martin Cyber Kill Chain

```
{
  ...
  "kill_chain_phases": [
    {
      "kill_chain_name": "lockheed-martin-cyber-kill-chain",
      "phase_name": "reconnaissance"
    }
  ],
  ...
}
```

Example specifying the "pre-attack" phase from the "foo" kill-chain

```
{
  ...
  "kill_chain_phases": [
    {
      "kill_chain_name": "foo",
      "phase_name": "pre-attack"
    }
  ],
  ...
}
```

2.8 List

Type Name: `list`

The `list` type defines a sequence of values ordered based on how they appear in the list. The phrasing "`list` of type `<type>`" is used to indicate that all values within the list **MUST** conform to the specified type. For instance, `list` of type `integer` means that all values of the list must be of the `integer` type. This specification does not specify the maximum number of allowed values in a `list`, however every instance of a `list` **MUST** have at least one value. Specific STIX object properties may define more restrictive upper and/or lower bounds for the length of the list.

Empty lists are prohibited in STIX and **MUST NOT** be used as a substitute for omitting the property if it is optional. If the property is required, the list **MUST** be present and **MUST** have at least one value.

The JSON MTI serialization uses the JSON array type [RFC7159], which is an ordered list of zero or more values.

Examples

```
{
  ...
  "observed_data_refs": [
    "observed-data--b67d30ff-02ac-498a-92f9-32f845f448cf",
    "observed-data--c96f4120-2b4b-47c3-b61f-eceaa54bd9c6",
    "observed-data--787710c9-1988-4a1b-9761-a2de5e19c62f"
  ],
  ...
}
```

2.9 Open Vocabulary

Type Name: `open-vocab`

The `open-vocab` type is represented as a `string`. For properties that use this type there will be a list of suggested values, known as the suggested vocabulary, that is identified in the definition for that property. The suggested vocabularies are defined in section 6. The value of the property **SHOULD** be chosen from the suggested vocabulary but **MAY** be any other `string` value. Values that are not from the suggested vocabulary **SHOULD** be all lowercase (where lowercase is defined by the locality conventions) and **SHOULD** use hyphens instead of spaces or underscores as word separators.

A consumer that receives STIX content with one or more `open-vocab` terms not defined in the suggested vocabulary **MAY** ignore those values.

The JSON MTI serialization uses the JSON string type [RFC7159] when representing `open-vocab`.

Examples

Example using value from the suggested vocabulary. In this example the Indicator **`labels`** property is an open vocabulary and we are using one of the suggested vocabulary values.

```
{
  ...,
  "labels": ["malicious-activity"],
  ...
}
```

Example using a custom value. In this example, for the same Indicator **`labels`** property, we are not using a value in the suggested vocabulary.

```
{
  ...,
  "labels": ["pbx-fraud-activity"],
  ...
}
```

2.10 String

Type Name: `string`

The `string` data type represents a finite-length string of valid characters from the Unicode coded character set [ISO10646]. Unicode incorporates ASCII and the characters of many other international character sets.

The JSON MTI serialization uses the JSON string type [RFC7159], which mandates the UTF-8 encoding for supporting Unicode.

Examples

```
{
  ...
  "name": "The Black Vine Cyberespionage Group",
  ...
}
```

2.11 Timestamp

Type Name: `timestamp`

The `timestamp` type defines how dates and times are represented in STIX.

The JSON MTI serialization uses the JSON string type [RFC7159] when representing `timestamp`.

2.11.1 Requirements

The `timestamp` property **MUST** be a valid RFC 3339-formatted timestamp [RFC3339] using the format `YYYY-MM-DDTHH:mm:ss[.s+]Z` where the “s+” represents 1 or more sub-second values. The brackets denote that sub-second precision is optional, and that if no digits are provided, the decimal place **MUST NOT** be present.

The timestamp **MUST** be represented in the UTC timezone and **MUST** use the “Z” designation to indicate this.

Examples

```
{
  ...
  "created": "2016-01-20T12:31:12.123Z",
  ...
}
```

3. 3 STIX™ Objects

This section outlines the common properties and behavior across all SDOs and SROs.

The JSON MTI serialization uses the JSON object type [RFC7159] when representing all STIX Objects.

3.1 Common Properties

Property Name	Type	Description
type (required)	string	The **type** property identifies the type of STIX Object. The value of the **type** property **MUST** be the name of one of the types of STIX Object defined in sections 2 and 3 of *STIX™ Version 2.0. Part 2: STIX Objects* (e.g., indicator) or the name of a custom object as defined by section 7.2.
id (required)	identifier	The **id** property universally and uniquely identifies this object. All objects with the same **id** are considered different versions of the same object. Because the object type is part of the identifier, it is invalid for objects of different types to share the same **id**.
created_by_ref (optional)	identifier	The **created_by_ref** property specifies the ID of the Identity object that describes the entity that created this object. If this attribute is omitted, the source of this information is undefined. This may be used by object creators who wish to remain anonymous.
created (required)	timestamp	The **created** property represents the time at which the first version of this object was created. The object creator can use the time it deems most appropriate as the time the object was created. The **created** property **MUST NOT** be changed when creating a new version of the object. The **created** timestamp **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). See section 3.4 for further definition of versioning.

modified (required)	timestamp	The **modified** property represents the time that this particular version of the object was created. The object creator can use the time it deems most appropriate as the time this version of the object was modified. The value of the **modified** property for a given object version **MUST** be later than or equal to the value of the **created** property. Object creators **MUST** set the **modified** property when creating a new version of an object. The **modified** timestamp **MUST** be precise to the nearest millisecond (exactly three digits after the decimal place in seconds). See section 3.4 for further definition of versioning.
revoked (optional)	boolean	The **revoked** property indicates whether the object has been revoked. Revoked objects are no longer considered valid by the object creator. Revoking an object is permanent; future versions of the object with this **id** **MUST NOT** be created. The default value of this property is false. See section 3.4 for further definition of versioning.
labels (optional)	list of type string	The **labels** property specifies a set of classifications. Each STIX Object can define a suggested vocabulary for the **labels** property. For example, the Indicator object, as defined in section 2.5 of *STIX™ Version 2.0. Part 2: STIX Objects*, uses the Indicator Label vocabulary as defined in section 6.5. In some cases (generally, when a suggested vocabulary is defined) the **labels** property is then required for that specific SDO. If a vocabulary is defined, items in this list **SHOULD** come from the vocabulary. Additional labels **MAY** be added beyond what is in the suggested vocabulary.
external_references	list of type	The **external_references** property

(optional)	external-reference	specifies a list of external references which refers to non-STIX information. This property is used to provide one or more URLs, descriptions, or IDs to records in other systems.
object_marking_refs (optional)	list of type identifier	The **object_marking_refs** property specifies a list of IDs of marking-definition objects that apply to this object. See section 4 for further definition of data markings.
granular_markings (optional)	list of type granular-marking	The **granular_markings** property specifies a list of granular markings applied to this object. See section 4 for further definition of data markings.

3.2 IDs and References

The **id** property universally and uniquely identifies an SDO, SRO, Bundle, or Marking Definition. It **MUST** meet the requirements of the identifier type (see section 2.5).

All STIX Objects (as well as Bundle and Marking Definition) use identifiers as defined by the identifier type. The identifier type is also used to define properties that are *ID references* to other constructs (such as the **created_by_ref** property in all STIX Objects). *Resolving* an ID reference is the process of identifying and obtaining the actual object referred to by the ID reference property. ID references resolve to an object when the value of the ID reference property (e.g., **created_by_ref**) is an exact match with the **id** property of another object. If a consumer has access to multiple versions of an object, the consumer **SHOULD** interpret any references to that object as referring to the latest version as defined in section 3.4. ID references can refer to objects to which the consumer/producer may not currently have. This specification does not address the implementation of ID reference resolution.

3.3 Object Creator

The object creator is the entity (e.g., system, organization, instance of a tool) that generates the **id** property for a given object. Object creators are represented as Identity objects. An embedded relationship to the Identity object representing the object creator **MAY** be captured in the **created_by_ref** property (or that property can be left blank, meaning the object creator is anonymous).

Entities that re-publish an object from another entity without making any changes to the object, and thus maintaining the original **id**, are not considered the object creator and **MUST NOT** change the **created_by_ref** property. An entity that accepts objects and republishes them with modifications, additions, or omissions **MUST** create a new **id** for the object. They are considered the object creator of the new object for purposes of versioning.

3.4 Versioning

Versioning is the mechanism that object creators use to update and revoke the STIX Objects that they create. This section describes the versioning process and normative rules for performing versioning and revocation. STIX Objects are versioned using the **revoked**, **created**, and **modified** properties. See the properties table in section 3.1 for full definitions and normative usage of those properties.

STIX Objects **MAY** be versioned in order to update, add, or remove information. A version of a STIX Object is identified uniquely by the combination of its **id** and **modified** properties. The first version of the object **MUST** have the same timestamp for the **created** and **modified** properties. More recent values of the **modified** property indicate later versions of the object. Implementations **MUST** consider the version of the STIX Object with the most recent **modified** value to be the most recent state of the object. For every new version of an object, the **modified** property **MUST** be updated to represent the time that the new version was created. If a consumer receives two objects that are different, but have the same **id** and **modified** timestamp, it is not defined how the consumer handles the objects. This specification does not address how implementations should handle versions of the object that are not current.

STIX Objects have a single *object creator*, the entity that generates the **id** for the object and creates the first version. The object creator may (but not necessarily will) be identified in the **created_by_ref** property of the object. Only the object creator is permitted to create new versions of a STIX Object. Producers other than the object creator **MUST NOT** create new versions of that object. If a producer other than the object creator wishes to create a new version, they **MUST** instead create a new object with a new **id**. They **SHOULD** additionally create a `derived-from` Relationship object to relate their new object to the original object that it was derived from.

Every representation (each time the object version is serialized and shared) of a version of an object (identified by the object's **id** and **modified** properties) **MUST** always have the same set of properties and the same values for each property. In order to change the value of any property, or to add or remove properties, the **modified** property **MUST** be updated with the time of the change to indicate a new version.

Objects can also be revoked, which means that they are no longer considered valid by the object creator. As with issuing a new version, only the object creator is permitted to revoke a STIX Object. A value of `true` in the **revoked** property indicates that an object (including the current version and all past versions) has been revoked. Revocation is permanent: once an object is marked as revoked, later versions of that object **MUST NOT** be created. Changing the **revoked** property to indicate that an object is revoked is an update to the object, and therefore its **modified** property **MUST** be updated at the same time. This specification does not address how implementations should handle revoked data.

3.4.1 Versioning Timestamps

There are two timestamp properties used to indicate when STIX Objects were created and modified: **created** and **modified**. The **created** property indicates the time the first version of the object was created. The **modified** property indicates the time the specific version of the object was created. The **modified** time **MUST NOT** be earlier than the **created** time. This specification does not address the specifics of how implementations should determine the value of the creation and modification times for use in the **created** and **modified** properties (e.g. one system might use when the object is first added to the local database as the creation time, while another might use the time when the object is first distributed as STIX).

3.4.2 New Version or New Object?

Eventually an implementation will encounter a case where a decision must be made regarding whether a change is a new version of an existing object or is different enough that it is a new object. This is generally considered a data quality problem and therefore this specification does not provide any normative text.

However, to assist implementers and promote consistency across implementations, some rules of thumb are provided. Any time a change indicates a *material change* to the meaning of the object, a new object with a different **id** should be used. A material change is any change that the object creator believes substantively changes the meaning of the object. As an example, an object creator might consider changing a Threat Actor from one country to another is a material change. These decisions are always made by the object creator. The object creator should also think about relationships to the object when deciding if a change is material. If the change would invalidate the usefulness of relationships to the object, then the change is considered material and a new object **id** should be used.

Examples

Example of a new version

One object creator has decided that the previous name they used for a SDO is incorrect. They consider that change as an update to the object.

Note: the IDs in the example below use a simplified format to help illustrate the changing IDs more clearly.

Step #	STIX Object	Object Creator Action

1	`{`
	`"type": "example",`
	`"id": "example--1",`
	`"created": "2016-05-01T06:13:14.000Z",`
	`"modified": "2016-05-01T06:13:14.000Z",`
	`"name": "attention",`
	`"description": "this is the description"`
	`}`
		Original version of an object is created.

2	N/A, STIX is not involved in this step	Object creator changes the name in their internal database.

3	`{`
	`"type": "example",`
	`"id": "example--1",`
	`"created": "2016-05-01T06:13:14.000Z",`
	`"modified": "2016-05-08T03:43:44.000Z",`
	`"name": "Attention!",`
	`"description": "this is the description"`
	`}`
		Object creator updates the **`modified`** property.

Example of derived object

One object creator has decided that the previous name they used for a SDO is incorrect. They consider that change fundamental to the meaning of the object and therefore revoke the object and issue a new one.

Step #	STIX Object	Object Creator Action

1	`{`
	`"type": "example",`
	`"id": "example--1",`
	`"created": "2016-05-01T06:13:14.000Z",`
	`"modified": "2016-05-01T06:13:14.000Z",`
	`"name": "attention",`
	`"description": "this is the description"`
	`}`
		Original object created (via new id and setting **`created`** and **`modified`** to the same value).

2	N/A, STIX is not involved in this step	Object creator changes the name in their internal database.

3	`{`
	`"type": "example",`
	`"id": "example--1",`
	`"created": "2016-05-01T06:13:14.000Z",`
	`"modified": "2016-05-08T03:43:44.000Z",`
	`"name": "attention",`
	`"description": "this is the description",`
	`"revoked": true`
	`}`
		Object creator revokes the existing object by setting **`revoked`** to `true`. The **`modified`** property is updated.

4	`{`
	`"type": "example",`
	`"id": "example--2",`
	`"created": "2016-05-08T03:43:44.000Z",`
	`"modified": "2016-05-08T03:43:44.000Z",`
	`"name": "Something completely different",`
	`"description": "this is the description"`
	`}`
		Object creator creates a new object (with a new **`id`** and setting **`created`** and **`modified`** to the same value).

5	`{`
	`"type": "relationship",`
	`"id": "relationship--3",`
	`"created": "2016-05-08T03:43:44.000Z",`
	`"modified": "2016-05-08T03:43:44.000Z",`
	`"relationship_type": "derived-from",`
	`"source_ref": "example--1",`
	`"target_ref": "example--2"`
	`}`
		(Optional) Object creator creates a new Relationship indicating that the new object is derived from the old object.

Example consumer workflow

This section describes an example workflow where a consumer receives multiple updates to a particular object. (In this example, the STIX Objects have been truncated for brevity.)

Step #	STIX Object	Recipient Action

1	`{`
	`"type": "example",`
	`"id": "example--1",`
	`"created": "2016-05-01T06:13:14.000Z",`
	`"modified": "2016-05-01T06:13:14.000Z"`
	`}`
		Consumer stores example object because this is the first time they have seen the object.

2	`{`
	`  "type": "example",`
	`  "id": "example--1",`
	`  "created": "2016-05-01T06:13:14.000Z",`
	`  "modified": "2016-05-08T03:43:44.000Z"`
	`}`
		Consumer updates example object because the received **`modified`** property is later than the object that is currently stored.

3	`{`
	`  "type": "example",`
	`  "id": "example--1",`
	`  "created": "2016-05-01T06:13:14.000Z",`
	`  "modified": "2016-05-06T06:23:45.000Z"`
	`}`
		Consumer ignores this object because they already have a newer version of the object.
		Note: consumer might choose to store meta-information about received objects, including versions that were received out-of-order. The consumer also may choose to store a copy for reference.

4	`{`
	`  "type": "example",`
	`  "id": "example--1",`
	`  "created": "2016-05-01T06:13:14.000Z",`
	`  "modified": "2016-05-11T06:41:21.000Z",`
	`  "revoked": true`
	`}`
		Consumer decides to delete example object, but keeps some metadata regarding the object.

5	`{`
	`  "type": "example",`
	`  "id": "example--1",`
	`  "created": "2016-05-01T06:13:14.000Z",`
	`  "modified": "2016-05-10T17:28:54.000Z"`
	`}`
		Consumer ignores this object because they already have a newer version of the object (the revoked version).

Example object creator workflow

This section describes an example workflow where a object creator publishes multiple updates to a particular object. This scenario assumes a human using a STIX implementation. (In this example, the STIX Objects have been truncated for brevity.)

Step #	STIX Object	User Action
1	N/A – STIX is not involved in this scenario. (Tools *could* choose to create and track STIX versions for internal changes, but it is not required by the specification.)	User clicks a create button in the user interface, creates a SDO, then clicks save. This action causes information to be stored in the product's database.
2	`{`	The user clicks the "share" button, delivering the intelligence to sharing

	"type": "example", "id": "example--2", "created": "2016-05-01T06:13:14.000Z", "modified": "2016-05-01T06:13:14.000Z" }	partners.
3	N/A – STIX is not involved in this scenario. (Tools *could* choose to create and track STIX versions for internal changes, but it is not required by the specification.)	The user performs additional analysis within the STIX implementation, performing multiple modifications and saving their work multiple times.

4	{
	"type": "example",
	"id": "example--2",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-03T16:33:51.000Z"
	}
		The user, happy with the status of their work, decides to provide an update to some properties of the previously published object (not shown).

5	{
	"type": "example",
	"id": "example--2",
	"created": "2016-05-01T06:13:14.000Z",
	"modified": "2016-05-08T13:35:12.000Z",
	"revoked": true
	}
		The user receives lots of negative feedback regarding the quality of their work and decides to retract the object by pressing the “revoke” button.

3.5 Common Relationships

Each SDO has its own set of relationship types that are specified in the definition of that SDO. The following common relationship types are defined for all SDOs. See section 1.4.3 for more information about relationships.

Relationship Type	Source	Target	Description
derived-from	*<STIX Domain Object>*	*<STIX Domain Object of same type>*	The information in the target object is based on information from the source object. derived-from is an explicit relationship between two separate objects and **MUST NOT** be used as a substitute for the versioning process defined in section 3.4.
duplicate-of	*<STIX Domain Object>*	*<STIX Domain Object of same type>*	The referenced source and target objects are semantically duplicates of each other.

			This specification does not address whether the source or the target object is the duplicate object or what action, if any, a consumer should take when receiving an instance of this relationship. As an example, a Campaign object from one organization could be marked as a `duplicate-of` a Campaign object from another organization if they both described the same campaign.
`related-to`	*`<STIX Domain Object>`*	*`<STIX Domain Object of any type>`*	Asserts a non-specific relationship between two SDOs. This relationship can be used when none of the other predefined relationships are appropriate. As an example, a Malware object describing a piece of malware could be marked as a `related-to` a Tool if they are commonly used together. That relationship is not common enough to standardize on, but may be useful to some analysts.

3.6 Reserved Properties

This section defines property names that are reserved for future use in revisions of this document. The property names defined in this section **MUST NOT** be used for the name of any Custom Property.

Properties that are currently reserved across all STIX Objects are:

`confidence`

`severity`

`action`

`usernames`

`phone_numbers`

`addresses`

`first_seen_precision`

`last_seen_precision`

`valid_from_precision`

`valid_until_precision`

In addition, the following object names are reserved:

`incident`

`infrastructure`

4. 4 Data Markings

Data markings represent restrictions, permissions, and other guidance for how data can be used and shared. For example, data may be shared with the restriction that it must not be re-shared, or that it must be encrypted at rest. In STIX, data markings are specified using the `marking-definition` object. These definitions are applied to complete STIX Objects using object markings and to individual properties of STIX Objects via granular markings.

Some types of marking definitions or trust groups have rules about which markings override other markings or which markings can be additive to other markings. This specification does not define rules for how multiple markings applied to the same object or property should be interpreted.

4.1 Marking Definition

Type Name: `marking-definition`

The `marking-definition` object represents a specific marking. Data markings typically represent handling or sharing requirements for data, and are applied in the **`object_marking_refs`** and **`granular_markings`** properties on STIX Objects, which reference a list of IDs for `marking-definition` objects.

Two marking definition types are defined in this specification: TLP, to capture TLP markings, and Statement, to capture text marking statements. In addition, it is expected that the FIRST Information Exchange Policy (IEP) will be included in a future version once a machine-usable specification for it has been defined.

Unlike STIX Objects, Marking Definition objects cannot be versioned because it would allow for indirect changes to the markings on a STIX Object. For example, if a Statement marking is changed from "Reuse Allowed" to "Reuse Prohibited", all STIX Objects marked with that Statement marking would effectively have an updated marking without being updated themselves. Instead, a new Statement marking with the new text should be created and the marked objects updated to point to the new marking.

The JSON MTI serialization uses the JSON object type [RFC7159] when representing `marking-definition`.

4.1.1 Properties

Property Name	Type	Description
`type` (required)	`string`	The **`type`** property identifies the type of object. The value of this property **MUST** be `marking-definition`.
`id` (required)	`identifier`	The **`id`** property universally and uniquely identifies this Marking Definition. Because the object type is part of the `identifier`, it is not possible for objects of different types to share the same **`id`**.
`created_by_ref` (optional)	`identifier`	The **`created_by_ref`** property specifies the ID of the `identity` object that describes the entity that

		created this Marking Definition. If this attribute is omitted, the source of this information is undefined. This may be used by object creators who wish to remain anonymous.
`created` (required)	`timestamp`	The **`created`** property represents the time at which the Marking Definition was created. The object creator can use the time it deems most appropriate as the time the object was created.
`external_references` (optional)	`list` of type `external-reference`	The **`external_references`** property specifies a list of external references which refers to non-STIX information. This property is used to provide one or more URLs, descriptions, or IDs to records in other systems.
`object_marking_refs` (optional)	`list` of type `identifier`	The **`object_marking_refs`** property specifies a list of IDs of `marking-definition`s that apply to this Marking Definition. This property **MUST NOT** contain any references to this Marking Definition object (i.e., it cannot contain any circular references). Though uncommon, in some cases marking definitions themselves may be marked with sharing or handling guidance.
`granular_markings` (optional)	`list` of type `granular-marking`	The **`granular_markings`** property specifies a list of granular markings applied to this. This property **MUST NOT** contain any references to this Marking Definition object (i.e., it cannot contain any circular references). Though uncommon, in some cases Marking Definitions themselves may be marked with sharing or handling guidance.
`definition_type` (required)	`open-vocab`	The **`definition_type`** property identifies the type of Marking Definition. The value of the **`definition_type`** property **SHOULD** be one of the types defined in the subsections below: `statement` or `tlp` (see sections 4.1.3 and 4.1.4)
`definition` (required)	`<marking object>`	The **`definition`** property contains the marking object itself (e.g., the TLP marking as defined in section 4.1.4, the Statement marking as defined in section 4.1.3, or some other marking definition defined

		elsewhere).

4.1.2 Relationships

Data Marking is not a STIX Object and MUST NOT have any SRO relationships to it or from it. This table lists the embedded relationships by property name along with their corresponding target.

Embedded Relationships	
created_by_ref	identity
object_marking_refs	marking-definition

4.1.3 Statement Marking Object Type

The Statement marking type defines the representation of a textual marking statement (e.g., copyright, terms of use, etc.) in a definition. The value of the **definition_type** property **MUST** be statement when using this marking type. Statement markings are generally not machine-readable and this specification does not define any behavior or actions based on their values.

Content may be marked with multiple Statement marking types that do not override each other. In other words, the same content can be marked both with a statement saying "Copyright 2016" and a statement saying "Terms of use are ..." and both statements apply.

Property Name	Type	Description
statement (required)	string	A Statement (e.g., copyright, terms of use) applied to the content marked by this marking definition.

Examples

```
{
  "type": "marking-definition",
  "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
  "created": "2016-08-01T00:00:00.000Z",
  "definition_type": "statement",
  "definition": {
    "statement": "Copyright 2016, Example Corp"
  }
}
```

4.1.4 TLP Marking Object Type

The TLP marking type defines how you would represent a Traffic Light Protocol (TLP) marking in a definition property. The value of the **definition_type** property **MUST** be tlp when using this marking type.

Property Name	Type	Description

tlp (required)	string	The TLP level [TLP] of the content marked by this marking definition, as defined in this section.

The following standard marking definitions **MUST** be used to reference or represent TLP markings. Other instances of tlp-marking **MUST NOT** be used (the only instances of TLP marking definitions permitted are those defined here).

white	{
	  "type": "marking-definition",
	  "id": "marking-definition--613f2e26-407d-48c7-9eca-b8e91df99dc9",
	  "created": "2017-01-20T00:00:00.000Z",
	  "definition_type": "tlp",
	  "definition": {
	    "tlp": "white"
	  }
	}

green	{
	  "type": "marking-definition",
	  "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
	  "created": "2017-01-20T00:00:00.000Z",
	  "definition_type": "tlp",
	  "definition": {
	    "tlp": "green"
	  }
	}

amber	{
	  "type": "marking-definition",
	  "id": "marking-definition--f88d31f6-486f-44da-b317-01333bde0b82",
	  "created": "2017-01-20T00:00:00.000Z",
	  "definition_type": "tlp",
	  "definition": {
	    "tlp": "amber"
	  }
	}

red	{
	  "type": "marking-definition",
	  "id": "marking-definition--5e57c739-391a-4eb3-b6be-7d15ca92d5ed",
	  "created": "2017-01-20T00:00:00.000Z",
	  "definition_type": "tlp",
	  "definition": {

	`    "tlp": "red"`
	`  }`
	`}`

4.2 Object Markings

Object Markings apply data markings to an entire STIX Object or Marking Definition and all of its contents. Object Markings are specified as embedded relationships in the **`object_marking_refs`** property, which is an optional list of IDs for `marking-definition` objects. The referenced markings apply to that STIX Object or Marking Definition and all of its contents. Changes to the **`object_marking_refs`** property (and therefore the markings applied to the object) are treated the same as changes to any other properties on the object and follow the same rules for versioning.

Examples

This example marks the Indicator and all its properties with the Marking Definition referenced by the ID.

```
{
  "type": "indicator",
  "id": "indicator--b346b4b3-f4b7-4235-b659-f985f65f0009",
  ...
  "object_marking_refs": ["marking-definition--089a6ecb-cc15-43cc-9494-767639779123"],
  ...
}
```

4.3 Granular Markings

Whereas object markings apply to an entire STIX Object or Marking Definition and all its properties, granular markings allow data markings to be applied to individual portions of STIX Objects and Marking Definitions. Granular markings are specified in the **`granular_markings`** property, which is a list of `granular-marking` instances. Each of those instances contains a list of selectors to indicate what is marked and a reference to the `marking-definition` object to be applied. Granular markings can be used, for example, to indicate that the **`name`** property of an `indicator` should be handled as TLP:GREEN, the **`description`** property as TLP:AMBER, and the **`pattern`** property as TLP:RED.

4.3.1 Granular Marking Type

The `granular-marking` type defines how the `marking-definition` object referenced by the **`marking_ref`** property applies to a set of content identified by the list of selectors in the **`selectors`** property.

Property Name	Type	Description
`marking_ref` (required)	`identifier`	The **`marking_ref`** property specifies the ID of the `marking-definition` object that describes the marking.
`selectors` (required)	`list` of type `string`	The **`selectors`** property specifies a list of selectors for content contained within the STIX Object in which this property appears. Selectors **MUST** conform to the syntax defined in section 4.3.1.1.

		The `marking-definition` referenced in the **marking_ref** property is applied to the content selected by the selectors in this list.

4.3.1.1 Selector Syntax

Selectors contained in the **selectors** list are strings that consist of multiple components that **MUST** be separated by the `.` character. Each component **MUST** be one of:

- A property name, e.g., `description`, or;
- A zero-based list index, specified as a non-negative integer in square brackets, e.g., `[4]`

Selectors denote path traversals: the root of each selector is the STIX Object that the **granular_markings** property appears in. Starting from that root, for each component in the selector, properties and list items are traversed. When the complete list has been traversed, the value of the content is considered selected.

Selectors **MUST** refer to properties or list items that are actually present on the marked object.

As an example, consider the following STIX Object:

```
{
  "id": "vulnerability--ee916c28-c7a4-4d0d-ad56-a8d357f89fef",
  "created": "2016-02-14T00:00:00.000Z",
  "modified": "2016-02-14T00:00:00.000Z",
  "type": "vulnerability",
  "name": "CVE-2014-0160",
  "description": "The (1) TLS...",
  "external_references": [{
    "source_name": "cve",
    "external_id": "CVE-2014-0160"
  }],
  "labels": ["heartbleed", "has-logo"]
}
```

Valid selectors:

- `description` selects the **description** property ("`The (1) TLS...`").
- `external_references.[0].source_name` selects the **source_name** property of the first value of the **external_references** list ("`cve`").
- `labels.[0]` selects the first item contained within the **labels** list ("`heartbleed`").
- `labels` selects the list contained in the **labels** property. Due to the recursive nature of the selector, that includes all items in the list (`["heartbleed", "has-logo"]`).
- `external_references` selects the list contained in the **external_references** property. Due to the recursive nature of the selector, that includes all list items and all properties of those list items.

Invalid selectors:

- `pattern` and `external_references.[3]` are invalid selectors because they refer to content not present in that object.
- `description.[0]` is an invalid selector because the `description` property is a string and not a list.
- `labels.name` is an invalid selector because `labels` property is a list and not an object.

This syntax is inspired by JSONPath [Goessner 2007] and is in fact a strict subset of allowable JSONPath expressions (with the exception that the '$' to indicate the root is implicit). Care should be taken when passing selectors to JSONPath evaluators to ensure that the root of the query is the individual STIX Object. It is expected, however, that selectors can be easily evaluated in programming languages that implement list and key/value mapping types (dictionaries, hashmaps, etc.) without resorting to an external library.

Examples

This example marks the **`description`** and **`labels`** properties with the single marking definition referenced in the list.

```
{
  ...
  "granular_markings": [
    {
      "marking_ref": "marking-definition--089a6ecb-cc15-43cc-9494-767639779123",
      "selectors": ["description", "labels"]
    }
  ],
  "description": "Some description",
  "name": "Some name",
  "labels": ["first", "second"]
}
```

5. 5 Bundle

Type Name: bundle

A Bundle is a collection of arbitrary STIX Objects and Marking Definitions grouped together in a single container. A Bundle does not have any semantic meaning and Objects are not considered related by virtue of being in the same Bundle.

Bundle is not STIX Object, so it does not have any of the Common Properties other than the **type** and **id** properties. Bundle is transient and implementations should not assume that other implementations will treat it as a persistent object.

The JSON MTI serialization uses the JSON object type [RFC7159] when representing bundle.

5.1 Properties

Property Name	Type	Description
type (required)	string	The **type** property identifies the type of object. The value of this property **MUST** be bundle.
id (required)	identifier	An identifier for this Bundle. The **id** property for the Bundle is designed to help tools that may need it for processing, but tools are not required to store or track it. Consuming tools should not rely on the presence of this property or the ability to refer to bundles by ID.
spec_version (required)	string	The version of the STIX specification used to represent the content in this Bundle. This enables non-TAXII transports or other transports without their own content identification mechanisms to know the version of STIX content. The value of this property **MUST** be 2.0 for bundles containing STIX Objects defined in this specification.
objects (optional)	list of type *<STIX Object>* or marking-definition	Specifies a set of one or more STIX Objects. Objects in this list **MUST** be a STIX Object (SDO, SRO or Custom Object) or a Marking Definition object.

5.2 Relationships

Bundle is not a STIX Object and **MUST NOT** have any relationships to it or from it.

Examples

```
{
  "type": "bundle",
  "id": "bundle--5d0092c5-5f74-4287-9642-33f4c354e56d",
  "spec_version": "2.0",
  "objects": [
    {
      "type": "indicator",
      "id": "indicator--8e2e2d2b-17d4-4cbf-938f-98ee46b3cd3f",
      "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
      "created": "2016-04-29T14:09:00.000Z",
      "modified": "2016-04-29T14:09:00.000Z",
      "object_marking_refs": ["marking-definition--089a6ecb-cc15-43cc-9494-767639779123"],
      "name": "Poison Ivy Malware",
      "description": "This file is part of Poison Ivy",
      "pattern": "[file:hashes.'SHA-256' = 
'aec070645fe53ee3b3763059376134f058cc337247c978add178b6ccdfb0019f']"
    },
    {
      "type": "marking-definition",
      "id": "marking-definition--34098fce-860f-48ae-8e50-ebd3cc5e41da",
      "created": "2016-08-01T00:00:00.000Z",
      "definition_type": "tlp",
      "definition": {
        "tlp": "green"
      }
    }
  ]
}
```

6. 6 Vocabularies

The following sections provide object-specific listings for each of the vocabularies referenced in the object description sections defined in *STIX™ Version 2.0. Part 2: STIX Objects*. STIX vocabularies, which all have type names ending in '-ov', are "open": they provide a listing of common and industry accepted terms as a guide to the user but do not limit the user to that defined list.

6.1 Attack Motivation

Vocabulary Name: `attack-motivation-ov`

The attack motivation vocabulary is currently used in the following SDOs:

- Intrusion Set
- Threat Actor

Knowing a Threat Actor or Intrusion Set's motivation may allow an analyst or defender to better understand likely targets and behaviors.

Motivation shapes the intensity and the persistence of an attack. Threat Actors and Intrusion Sets usually act in a manner that reflects their underlying emotion or situation, and this informs defenders of the manner of attack. For example, a spy motivated by nationalism (ideology) likely has the patience to achieve long-term goals and work quietly for years, whereas a cyber-vandal out for notoriety can create an intense and attention-grabbing attack but may quickly lose interest and move on. Understanding these differences allows defenders to implement controls tailored to each type of attack for greatest efficiency.

This section including vocabulary items and their descriptions is based on the *Threat Agent Motivations* publication from Intel Corp in February 2015 [Casey 2015].

Vocabulary Summary	
`accidental`, `coercion`, `dominance`, `ideology`, `notoriety`, `organizational-gain`, `personal-gain, personal-satisfaction`, `revenge`, `unpredictable`	
Vocabulary Value	**Description**
`accidental`	A non-hostile actor whose benevolent or harmless intent inadvertently causes harm. For example, a well-meaning and dedicated employee who through distraction or poor training unintentionally causes harm to his or her organization.
`coercion`	Being forced to act on someone else's behalf. Adversaries who are motivated by coercion are often forced through intimidation or blackmail to act illegally for someone else's benefit. Unlike the other motivations, a coerced person does not act for personal gain, but out of fear of incurring a loss.
`dominance`	A desire to assert superiority over someone or something else.

	Adversaries who are seeking dominance over a target are focused on using their power to force their target into submission or irrelevance. Dominance may be found with ideology in some state-sponsored attacks and with notoriety in some cyber vandalism based attacks.
ideology	A passion to express a set of ideas, beliefs, and values that may shape and drive harmful and illegal acts. Adversaries who act for ideological reasons (e.g., political, religious, human rights, environmental, desire to cause chaos/anarchy, etc.) are not usually motivated primarily by the desire for profit; they are acting on their own sense of morality, justice, or political loyalty. For example, an activist group may sabotage a company's equipment because they believe the company is harming the environment.
notoriety	Seeking prestige or to become well known through some activity. Adversaries motivated by notoriety are often seeking either personal validation or respect within a community and staying covert is not a priority. In fact one of the main goals is to garner the respect of their target audience.
organizational-gain	Seeking advantage over a competing organization, including a military organization. Adversaries motivated by increased profit or other gains through an unfairly obtained competitive advantage are often seeking theft of intellectual property, business processes, or supply chain agreements and thus accelerating their position in a market or capability.
personal-gain	The desire to improve one's own financial status. Adversaries motivated by a selfish desire for personal gain are often out for gains that come from financial fraud, hacking for hire, or intellectual property theft. While a Threat Actor or Intrusion Set may be seeking personal gain this does not mean they are acting alone. Individuals can band together solely to maximize their own personal profits.
personal-satisfaction	A desire to satisfy a strictly personal goal, including curiosity, thrill-seeking, amusement, etc. Threat Actors or Intrusion Set driven by personal satisfaction may incidentally receive some other gain from their actions, such as a profit, but their primary motivation is to gratify a personal, emotional need. Individuals can band together with others toward a mutual, but not necessarily organizational, objective.
revenge	A desire to avenge perceived wrongs through harmful actions such as sabotage, violence, theft, fraud, or embarrassing certain

	individuals or the organization. A disgruntled Threat Actor or Intrusion Set seeking revenge can include current or former employees, who may have extensive knowledge to leverage when conducting attacks. Individuals can band together with others if the individual believes that doing so will enable them to cause more harm.
`unpredictable`	Acting without identifiable reason or purpose and creating unpredictable events. Unpredictable is not a miscellaneous or default category. Unpredictable means a truly random and likely bizarre event, which seems to have no logical purpose to the victims.

6.2 Attack Resource Level

Vocabulary Name: `attack-resource-level-ov`

The attack resource level vocabulary is currently used in the following SDO(s):

- Intrusion Set
- Threat Actor

Attack Resource Level is an open vocabulary that captures the general level of resources that a threat actor, intrusion set, or campaign might have access to. It ranges from individual, a person acting alone, to government, the resources of a national government.

This section including vocabulary items and their descriptions is based on the *Threat Agent Library* publication from Intel Corp in September 2007 [Casey 2007].

Vocabulary Summary	
`individual`, `club`, `contest`, `team`, `organization`, `government`	
Vocabulary Value	**Description**
`individual`	Resources limited to the average individual; Threat Actor acts independently.
`club`	Members interact on a social and volunteer basis, often with little personal interest in the specific target. An example might be a core group of unrelated activists who regularly exchange tips on a particular blog. Group persists long term.
`contest`	A short-lived and perhaps anonymous interaction that concludes when the participants have achieved a single goal. For example, people who break into systems just for thrills or prestige may hold a contest to see who can break into a specific target first. It also includes announced "operations" to achieve a specific goal, such as the original "OpIsrael" call for volunteers to disrupt all of Israel's Internet functions for a day.
`team`	A formally organized group with a leader, typically motivated by a specific goal and organized around that goal. Group persists long term and typically operates within a single geography.

organization	Larger and better resourced than a team; typically a company or crime syndicate. Usually operates in multiple geographic areas and persists long term.
government	Controls public assets and functions within a jurisdiction; very well resourced and persists long term.

6.3 Hashing Algorithm Vocabulary

Vocabulary Name: hash-algorithm-ov

An open vocabulary of hashing algorithms.

When specifying a hashing algorithm not already defined within the hash-algorithm-ov, wherever an authoritative name for a hashing algorithm name is defined, it should be used as the value. In cases where no authoritative name exists and/or where there is variance in the naming of a particular hashing algorithm, producers should exercise their best judgement.

Vocabulary Summary	
MD5, MD6, RIPEMD-160, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SSDEEP, WHIRLPOOL	
Vocabulary Value	**Description**
MD5	Specifies the MD5 message digest algorithm. The corresponding hash string for this value **MUST** be a valid MD5 message digest as defined in [RFC 1321].
MD6	Specifies the MD6 message digest algorithm. The corresponding hash string for this value **MUST** be a valid MD6 message digest as defined in the [MD6] proposal.
RIPEMD-160	Specifies the RIPEMD--160 (RACE Integrity Primitives Evaluation Message Digest) cryptographic hash function. The corresponding hash string for this value **MUST** be a valid RIPEMD-160 message digest as defined in the [RIPEMD-160] specification.
SHA-1	Specifies the SHA--1 (secure--hash algorithm 1) cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA-1 message digest as defined in [RFC 3174].
SHA-224	Specifies the SHA--224 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-224 message digest as defined in [RFC 6234].
SHA-256	Specifies the SHA--256 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-256 message digest as defined in [RFC 6234].
SHA-384	Specifies the SHA--384 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-384 message digest as defined in [RFC 6234].
SHA-512	Specifies the SHA--512 cryptographic hash function (part of the SHA-2 family). The corresponding hash string for this value **MUST** be a valid SHA-512 message digest as defined in [RFC 6234].

`SHA3-224`	Specifies the SHA3-224 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-224 message digest as defined in [FIPS202].
`SHA3-256`	Specifies the SHA3-256 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-256 message digest as defined in [FIPS202].
`SHA3-384`	Specifies the SHA3-384 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-384 message digest as defined in [FIPS202].
`SHA3-512`	Specifies the SHA3-512 cryptographic hash function. The corresponding hash string for this value **MUST** be a valid SHA3-512 message digest as defined in [FIPS202].
`ssdeep`	Specifies the ssdeep fuzzy hashing algorithm. The corresponding hash string for this value **MUST** be a valid piecewise hash as defined in the [SSDEEP] specification.
`WHIRLPOOL`	Specifies the whirlpool cryptographic hash function. The corresponding hash string for this value **MUST** be a valid WHIRLPOOL message digest as defined in [ISO10118].

6.4 Identity Class

Vocabulary Name: `identity-class-ov`

The identity class vocabulary is currently used in the following SDO(s):

- Identity

This vocabulary describes the type of entity that the Identity represents: whether it describes an organization, group, individual, or class.

Vocabulary Summary	
`individual`, `group`, `organization`, `class`, `unknown`	
Vocabulary Value	**Description**
`individual`	A single person.
`group`	An informal collection of people, without formal governance, such as a distributed hacker group.
`organization`	A formal organization of people, with governance, such as a company or country.
`class`	A class of entities, such as all hospitals, all Europeans, or the Domain Administrators in a system.
`unknown`	It is unknown whether the classification is individual, group, organization, or class.

6.5 Indicator Label

Vocabulary Name: `indicator-label-ov`

The indicator label vocabulary is currently used in the following SDO(s):

- Indicator

Indicator labels is an open vocabulary used to categorize Indicators. It is intended to be high-level to promote consistent practices. Indicator labels should not be used to capture information that can be better captured via related Malware or Attack Pattern objects. It is better to link an Indicator to a Malware object describing Poison Ivy rather than simply labeling it with "poison-ivy".

Vocabulary Summary	
`anomalous-activity`, `anonymization`, `benign`, `compromised`, `malicious-activity`, `attribution`	
Vocabulary Value	**Description**
`anomalous-activity`	Unexpected, or unusual activity that may not necessarily be malicious or indicate compromise. This type of activity may include reconnaissance-like behavior such as port scans or version identification, network behavior anomalies, and asset and/or user behavioral anomalies.
`anonymization`	Suspected anonymization tools or infrastructure (proxy, TOR, VPN, etc.).
`benign`	Activity that is not suspicious or malicious in and of itself, but when combined with other activity may indicate suspicious or malicious behavior.
`compromised`	Assets that are suspected to be compromised.
`malicious-activity`	Patterns of suspected malicious objects and/or activity.
`attribution`	Patterns of behavior that indicate attribution to a particular Threat Actor or Campaign.

6.6 Industry Sector

Vocabulary Name: `industry-sector-ov`

The industry sector vocabulary is currently used in the following SDO(s):

- Identity

Industry sector is an open vocabulary that describes industrial and commercial sectors. It is intended to be holistic; it has been derived from several other lists and is not limited to "critical infrastructure" sectors.

Vocabulary Summary	
`agriculture`, `aerospace`, `automotive`, `communications`, `construction`, `defence`, `education`, `energy`, `entertainment`, `financial-services`, `government-national`, `government-regional`, `government-local`, `government-public-services`, `healthcare`, `hospitality-leisure`, `infrastructure`, `insurance`, `manufacturing`, `mining`, `non-profit`, `pharmaceuticals`, `retail`, `technology`, `telecommunications`, `transportation`, `utilities`	
Vocabulary Value	**Description**
`agriculture`	

`aerospace`	
`automotive`	
`communications`	
`construction`	
`defence`	
`education`	
`energy`	
`entertainment`	
`financial-services`	
`government-national`	
`government-regional`	
`government-local`	
`government-public-services`	emergency services, sanitation
`healthcare`	
`hospitality-leisure`	
`infrastructure`	
`insurance`	
`manufacturing`	
`mining`	
`non-profit`	
`pharmaceuticals`	
`retail`	
`technology`	
`telecommunications`	
`transportation`	
`utilities`	

6.7 Malware Label

Vocabulary Name: `malware-label-ov`

The malware label vocabulary is currently used in the following SDO(s):

- Malware

Malware label is an open vocabulary that represents different types and functions of malware. Malware labels are not mutually exclusive; a malware instance can be both spyware and a screen capture tool.

Vocabulary Summary	
adware, backdoor, bot, ddos, dropper, exploit-kit, keylogger, ransomware, remote-access-trojan, resource-exploitation, rogue-security-software, rootkit, screen-capture, spyware, trojan, virus, worm	
Vocabulary Value	**Description**
adware	Any software that is funded by advertising. Adware may also gather sensitive user information from a system.
backdoor	A malicious program that allows an attacker to perform actions on a remote system, such as transferring files, acquiring passwords, or executing arbitrary commands [Mell2005].
bot	A program that resides on an infected system, communicating with and forming part of a botnet. The bot may be implanted by a worm or Trojan, which opens a backdoor. The bot then monitors the backdoor for further instructions.
ddos	A tool used to perform a distributed denial of service attack.
dropper	A type of trojan that deposits an enclosed payload (generally, other malware) onto the target computer.
exploit-kit	A software toolkit to target common vulnerabilities.
keylogger	A type of malware that surreptitiously monitors keystrokes and either records them for later retrieval or sends them back to a central collection point.
ransomware	A type of malware that encrypts files on a victim's system, demanding payment of ransom in return for the access codes required to unlock files.
remote-access-trojan	A remote access trojan program (or RAT), is a trojan horse capable of controlling a machine through commands issued by a remote attacker.
resource-exploitation	A type of malware that steals a system's resources (e.g., CPU cycles), such as a bitcoin miner.
rogue-security-software	A fake security product that demands money to clean phony infections.
rootkit	A type of malware that hides its files or processes from normal methods of monitoring in order to conceal its presence and activities. Rootkits can operate at a number of levels, from the application level — simply replacing or adjusting the settings of system software to prevent the display of certain information — through hooking certain functions or inserting modules or drivers

	into the operating system kernel, to the deeper level of firmware or virtualization rootkits, which are activated before the operating system and thus even harder to detect while the system is running.
`screen-capture`	A type of malware used to capture images from the target systems screen, used for exfiltration and command and control.
`spyware`	Software that gathers information on a user's system without their knowledge and sends it to another party. Spyware is generally used to track activities for the purpose of delivering advertising.
`trojan`	Any malicious computer program which is used to hack into a computer by misleading users of its true intent.
`virus`	A malicious computer program that replicates by reproducing itself or infecting other programs by modifying them.
`worm`	A self-replicating, self-contained program that usually executes itself without user intervention.

6.8 Report Label

Vocabulary Name: `report-label-ov`

The report label vocabulary is currently used in the following SDO(s):

- Report

Report label is an open vocabulary to describe the primary purpose or subject of a report. For example, a report that contains malware and indicators for that malware should have a report label of `malware` to capture that the malware is the primary purpose. Report labels are not mutually exclusive: a Report can be both a malware report and a tool report. Just because a report contains objects of a type does not mean that the report should include that label. If the objects are there to simply provide evidence or context for other objects, it is not necessary to include them in the label.

Vocabulary Summary	
`threat-report`, `attack-pattern`, `campaign`, `identity`, `indicator`, `malware`, `observed-data`, `threat-actor`, `tool`, `vulnerability`	
Vocabulary Value	**Description**
`threat-report`	Report subject is a broad characterization of a threat across multiple facets.
`attack-pattern`	Report subject is a characterization of one or more attack patterns and related information.
`campaign`	Report subject is a characterization of one or more campaigns and related information.
`identity`	Report subject is a characterization of one or more identities and related information.
`indicator`	Report subject is a characterization of one or more indicators

	and related information.
`intrusion-set`	Report subject is a characterization of one or more intrusion sets and related information.
`malware`	Report subject is a characterization of one or more malware instances and related information.
`observed-data`	Report subject is a characterization of observed data and related information.
`threat-actor`	Report subject is a characterization of one or more threat actors and related information.
`tool`	Report subject is a characterization of one or more tools and related information.
`vulnerability`	Report subject is a characterization of one or more vulnerabilities and related information.

6.9 Threat Actor Label

Vocabulary Name: `threat-actor-label-ov`

The threat actor label vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat actor label is an open vocabulary used to describe what type of threat actor the individual or group is. For example, some threat actors are competitors who try to steal information, while others are activists who act in support of a social or political cause. Actor labels are not mutually exclusive: a threat actor can be both a disgruntled insider and a spy. [Casey 2007])

Vocabulary Summary	
`activist`, `competitor`, `crime-syndicate`, `criminal`, `hacker`, `insider-accidental`, `insider-disgruntled`, `nation-state`, `sensationalist`, `spy`, `terrorist`	
Vocabulary Value	**Description**
`activist`	Highly motivated, potentially destructive supporter of a social or political cause (e.g., trade, labor, environment, etc.) that attempts to disrupt an organization's business model or damage their image. This category includes actors sometimes referred to as anarchists, cyber vandals, extremists, and hacktivists.
`competitor`	An organization that competes in the same economic marketplace. The goal of a competitor is to gain an advantage in business with respect to the rival organization it targets. It usually does this by copying intellectual property, trade secrets, acquisition strategies, or other technical or business data from a rival organization with the intention of using the data to bolster its own assets and market position.

crime-syndicate	An enterprise organized to conduct significant, large-scale criminal activity for profit. Crime syndicates, also known as organized crime, are generally large, well-resourced groups that operate to create profit from all types of crime.
criminal	Individual who commits computer crimes, often for personal financial gain and often involves the theft of something valuable. Intellectual property theft, extortion via ransomware, and physical destruction are common examples. A criminal as defined here refers to those acting individually or in very small or informal groups. For sophisticated organized criminal activity, see the crime syndicate descriptor.
hacker	An individual that tends to break into networks for the thrill or the challenge of doing so. Hackers may use advanced skills or simple attack scripts they have downloaded.
insider-accidental	A non-hostile insider who unintentionally exposes the organization to harm. “Insider” in this context includes any person extended internal trust, such as regular employees, contractors, consultants, and temporary workers.
insider-disgruntled	Current or former insiders who seek revengeful and harmful retaliation for perceived wrongs. “Insider” in this context includes any person extended internal trust, such as regular employees, contractors, consultants, and temporary workers. Disgruntled threat actors may have extensive knowledge that can be leveraged when conducting attacks and can take any number of actions including sabotage, violence, theft, fraud, espionage, or embarrassing individuals or the organization.
nation-state	Entities who work for the government or military of a nation state or who work at their direction. These actors typically have access to significant support, resources, training, and tools and are capable of designing and executing very sophisticated and effective Intrusion Sets and Campaigns.
sensationalist	Seeks to cause embarrassment and brand damage by exposing sensitive information in a manner designed to cause a public relations crisis. A sensationalist may be an individual or small group of people motivated primarily by a need for notoriety. Unlike

	the activist, the sensationalist generally has no political goal, and is not using bad PR to influence the target to change its behavior or business practices.
`spy`	Secretly collects sensitive information for use, dissemination, or sale. Traditional spies (governmental and industrial) are part of a well-resourced intelligence organization and are capable of very sophisticated clandestine operations. However, insiders such as employees or consultants acting as spies can be just as effective and damaging, even when their activities are largely opportunistic and not part of an overall campaign.
`terrorist`	Uses extreme violence to advance a social or political agenda as well as monetary crimes to support its activities. In this context a terrorist refers to individuals who target noncombatants with violence to send a message of fear far beyond the actual events. They may act independently or as part of a terrorist organization. Terrorist organizations must typically raise much of their operating budget through criminal activity, which often occurs online. Terrorists are also often adept at using and covertly manipulating social media for both recruitment and impact.

6.10 Threat Actor Role

Vocabulary Name: `threat-actor-role-ov`

The threat actor role vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat actor role is an open vocabulary that is used to describe the different roles that a threat actor can play. For example, some threat actors author malware or operate botnets while other actors actually carry out attacks directly.

Threat actor roles are not mutually exclusive. For example, an actor can be both a financial backer for attacks and also direct attacks.

Vocabulary Summary	
`agent`, `director`, `independent`, `infrastructure-architect`, `infrastructure-operator`, `malware-author`, `sponsor`	
Vocabulary Value	**Description**
`agent`	Threat actor executes attacks either on behalf of themselves or at the direction of someone else.
`director`	The threat actor who directs the activities, goals, and objectives of the malicious activities.

`independent`	A threat actor acting by themselves.
`infrastructure-architect`	Someone who designs the battle space.
`infrastructure-operator`	The threat actor who provides and supports the attack infrastructure that is used to deliver the attack (botnet providers, cloud services, etc.).
`malware-author`	The threat actor who authors malware or other malicious tools.
`sponsor`	The threat actor who funds the malicious activities.

6.11 Threat Actor Sophistication

Vocabulary Name: `threat-actor-sophistication-ov`

Threat actor sophistication vocabulary is currently used in the following SDO(s):

- Threat Actor

Threat actor sophistication vocabulary captures the skill level of a threat actor. It ranges from "none", which describes a complete novice, to "strategic", which describes an attacker who is able to influence supply chains to introduce vulnerabilities. This vocabulary is separate from resource level because an innovative, highly-skilled threat actor may have access to very few resources while a minimal-level actor might have the resources of an organized crime ring.

Vocabulary Summary	
`none`, `minimal`, `intermediate`, `advanced`, `expert`, `innovator`, `strategic`	
Vocabulary Value	**Description**
`none`	Can carry out random acts of disruption or destruction by running tools they do not understand. Actors in this category have average computer skills. Example Roles: Average User These actors: ● can not launch targeted attacks
`minimal`	Can minimally use existing and frequently well known and easy-to-find techniques and programs or scripts to search for and exploit weaknesses in other computers. Commonly referred to as a script-kiddie. These actors rely on others to develop the malicious tools, delivery mechanisms, and execution strategy and often do not fully understand the tool they are using or how they work. They also lack the ability to conduct their own reconnaissance and targeting research. Example Roles: Script-Kiddie These actors: ● attack known weaknesses;

	● use well known scripts and tools; and ● have minimal knowledge of the tools.
intermediate	Can proficiently use existing attack frameworks and toolkits to search for and exploit vulnerabilities in computers or systems. Actors in this category have computer skills equivalent to an IT professional and typically have a working knowledge of networks, operating systems, and possibly even defensive techniques and will typically exhibit some operational security. These actors rely others to develop the malicious tools and delivery mechanisms, but are able to plan their own execution strategy. They are proficient in the tools they are using and how they work and can even make minimal modifications as needed. Example Roles: Toolkit User These actors: ● attack known vulnerabilities; ● use attack frameworks and toolkits; and ● have proficient knowledge of the tools.
advanced	Can develop their own tools or scripts from publicly known vulnerabilities to target systems and users. Actors in this category are very adept at IT systems and have a background in software development along with a solid understanding of defensive techniques and operational security. These actors rely on others to find and identify weaknesses and vulnerabilities in systems, but are able to create their own tools, delivery mechanisms, and execution strategies. Example Roles: Toolkit Developer These actors: ● attack known vulnerabilities; ● can create their own tools; and ● have proficient knowledge of the tools.
expert	Can focus on the discovery and use of unknown malicious code, are is adept at installing user and kernel mode rootkits, frequently use data mining tools, target corporate executives and key users (government and industry) for the purpose of stealing personal and corporate data. Actors in this category are very adept at IT systems and software development and are experts with security systems, defensive techniques, attack methods, and operational security. Example Roles: Vulnerability Researcher, Reverse Engineer, Threat Researcher, Malware Creator These actors: ● attack unknown and known vulnerabilities; ● can create their own tools from scratch; and ● have proficient knowledge of the tools.
innovator	Typically criminal or state actors who are organized, highly technical, proficient, well-funded professionals working in teams to discover new vulnerabilities and develop exploits.

	Demonstrates sophisticated capability. An innovator has the ability to create and script unique programs and codes targeting virtually any form of technology. At this level, this actor has a deep knowledge of networks, operating systems, programming languages, firmware, and infrastructure topologies and will demonstrate operational security when conducting his activities. Innovators are largely responsible for the discovery of 0-day vulnerabilities and the development of new attack techniques. Example Roles: Toolkit Innovator, 0-Day Exploit Author These actors: ● attack unknown and known vulnerabilities; ● create attacks against 0-Day exploits from scratch; and ● create new and innovative attacks and toolkits.
strategic	State actors who create vulnerabilities through an active program to “influence” commercial products and services during design, development or manufacturing, or with the ability to impact products while in the supply chain to enable exploitation of networks and systems of interest. These actors: ● can create or use entire supply chains to launch an attack; ● can create and design attacks for any systems, software package, or device; and ● are responsible for APT-level attacks.

6.12 Tool Label

Vocabulary Name: tool-label-ov

The tool label vocabulary is currently used in the following SDO(s):

- Tool

Tool labels describe the categories of tools that can be used to perform attacks.

Vocabulary Summary	
denial-of-service, exploitation, information-gathering, network-capture, credential-exploitation, remote-access, vulnerability-scanning	
Vocabulary Value	**Description**
denial-of-service	Tools used to perform denial of service attacks or DDoS attacks, such as Low Orbit Ion Cannon (LOIC) and DHCPig.
exploitation	Tools used to exploit software and systems, such as sqlmap and Metasploit.
information-gathering	Tools used to enumerate system and network information, e.g., NMAP.
network-capture	Tools used to capture network traffic, such as Wireshark and Kismet.

`credential-exploitation`	Tools used to crack password databases or otherwise exploit/discover credentials, either locally or remotely, such as John the Ripper and NCrack.
`remote-access`	Tools used to access machines remotely, such as VNC and Remote Desktop.
`vulnerability-scanning`	Tools used to scan systems and networks for vulnerabilities, e.g., Nessus.

7. 7 Customizing STIX™

There are two primary means to customize STIX: Custom Properties, and Custom Objects. Custom Properties provides a mechanism and requirements for adding properties not defined by this specification to existing STIX Objects. Custom Objects, on the other hand, provides a mechanism and requirements to create custom STIX Objects (objects not defined by this specification).

A consumer that receives a STIX document containing Custom Properties or Objects it does not understand **MAY** refuse to process the document or **MAY** ignore those properties or objects and continue processing the document.

Producers of STIX documents that contain Custom Properties or Objects should recognize that consumers may not understand them and may ignore them. Producers should define any Custom Properties and Objects they use, along with any rules for processing them, and make these definitions and rules accessible to any potential consumers. This specification does not specify a process for doing this.

7.1 Custom Properties

There will be cases where certain information exchanges can be improved by adding properties that are neither specified nor reserved in this document; these properties are called **Custom Properties**. This section provides guidance and requirements for how producers can use Custom Properties and how consumers should interpret them in order to extend STIX in an interoperable manner.

7.1.1 Requirements

- A STIX Object **MAY** have any number of Custom Properties.
- Custom Property names **MUST** be in ASCII and **MUST** only contain the characters a–z (lowercase ASCII), 0–9, and underscore (_).
- Custom Property names **SHOULD** start with “x_” followed by a source unique identifier (such as a domain name with dots replaced by underscores), an underscore and then the name. For example, **`x_example_com_customfield`**.
- Custom Property names **MUST** have a minimum length of 3 ASCII characters.
- Custom Property names **MUST** be no longer than 250 ASCII characters in length.
- Custom Property names that do not start with “x_” may be used in a future version of the specification for a different meaning. If compatibility with future versions of this specification is required, the “x_” prefix **MUST** be used.
- Custom Properties **SHOULD** only be used when there is no existing properties defined by the STIX specification that fulfils that need.

Examples

```
{
  ...,
  "x_acme_org_confidence": 10,
  "x_acme_org_scoring": {
    "impact": "high",
    "probability": "low"
  },
  ...
}
```

7.2 Custom Objects

There will be cases where certain information exchanges can be improved by adding objects that are not specified nor reserved in this document; these objects are called **Custom Objects**. This section provides guidance and requirements for how producers can use Custom Objects and how consumers should interpret them in order to extend STIX in an interoperable manner.

7.2.1 Requirements

- Producers **MAY** include any number of Custom Objects in STIX documents.
- Custom Objects **MUST** support the Common Properties as defined in section 3.1.
    - The definitions of these properties are the same as those defined in Common Properties and therefore those properties **MUST NOT** be used to represent the custom properties in the object.
- The **`type`** property in a Custom Object **MUST** be in ASCII and **MUST** only contain the characters a–z (lowercase ASCII), 0–9, and hyphen (-).
- The **`type`** property **MUST NOT** contain a hyphen (-) character immediately following another hyphen (-) character.
- Custom Object names **MUST** have a minimum length of 3 ASCII characters.
- Custom Object names **MUST** be no longer than 250 ASCII characters in length.
- The value of the **`type`** property in a Custom Object **SHOULD** start with “x-” followed by a source unique identifier (like a domain name with dots replaced by hyphens), a hyphen and then the name. For example, `x-example-com-customobject`.
- A Custom Object whose name is not prefixed with “x-” may be used in a future version of the specification with a different meaning. Therefore, if compatibility with future versions of this specification is required, the “x-” prefix **MUST** be used.
- The value of the **`id`** property in a Custom Object **MUST** use the same format as the `identifier` type, namely, `[object-type]--[UUIDv4]`.
- Custom Objects **SHOULD** only be used when there is no existing STIX Object defined by the STIX specification that fulfils that need.

Examples

```
{
  "type": "bundle",
  "id": "bundle--f37aa79d-f5f5-4af7-874b-734d32c08c10",
  "spec_version": "2.0",
  "custom_objects": [
    {
      "type": "x-example-com-customobject",
      "id": "x-example-com-customobject--4527e5de-8572-446a-a57a-706f15467461",
      "created": "2016-08-01T00:00:00.000Z",
      "modified": "2016-08-01T00:00:00.000Z",
      "some_custom_stuff": 14,
      "other_custom_stuff": "hello"
    }
  ]
}
```

8. 8 Conformance

8.1 Producers and Consumers

A "STIX 2.0 Producer" is any software that creates STIX 2.0 content and conforms to the following normative requirements:

1. It **MUST** be able to create content encoded as JSON.
2. All properties marked required in the property table for the STIX Object or type **MUST** be present in the created content.
3. All properties **MUST** conform to the data type and normative requirements for that property.
4. It **MUST** support at least one STIX Object per the Conformance section in *STIX™ Version 2.0. Part 2: STIX Objects*.
5. It **MUST** support all features listed in section 8.2, Mandatory Features.
6. It **MAY** support any features listed in section 8.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.
7. It **MUST** support JSON as a serialization format and **MAY** support serializations other than JSON.

A "STIX 2.0 Consumer" is any software that consumes STIX 2.0 content and conforms to the following normative requirements:

1. It **MUST** support parsing all required properties for the content that it consumes.
2. It **MUST** support all features listed in section 8.2, Mandatory Features.
3. It **MAY** support any features listed in section 8.3, Optional Features. Software supporting an optional feature **MUST** comply with the normative requirements of that feature.
4. It **MUST** support JSON as a serialization format and **MAY** support serializations other than JSON.

8.2 Mandatory Features

8.2.1 Versioning

A STIX 2.0 Producer or STIX 2.0 Consumer **MUST** support versioning by following the normative requirements listed in section 3.4.

8.3 Optional Features

8.3.1 Object-Level Data Markings

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Object-Level Data Markings". Software claiming to support "Object-Level Data Markings" **MUST** follow the normative requirements listed in sections 4.1 and 4.2.

8.3.2 Granular Data Markings

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Granular Data Markings". Software claiming to support "Granular Data Markings" **MUST** follow the normative requirements listed in sections 4.1 and 4.3.

8.3.3 Custom Properties

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Custom Properties". Software claiming to support "Custom Properties" **MUST** follow the normative requirements listed in section 7.1.

8.3.4 Custom Objects

A STIX 2.0 Producer or STIX 2.0 Consumer **MAY** support "Custom Objects". Software claiming to support "Custom Objects" **MUST** follow the normative requirements listed in section 7.2.

9. Appendix A. Glossary

CAPEC - Common Attack Pattern Enumeration and Classification

Consumer - Any entity that receives STIX content

CTI - Cyber Threat Intelligence

Embedded Relationship - A link (an "edge" in a graph) between one STIX Object and another represented as a property on one object containing the ID of another object

Entity - Anything that has a separately identifiable existence (e.g., organization, person, group, etc.)

IEP - FIRST (Forum of Incident Response and Security Teams) Information Exchange Policy

Instance - A single occurrence of a STIX object version

MTI - Mandatory To Implement

MVP - Minimally Viable Product

Object Creator - The entity that created or updated a STIX object (see section 3.3)

Object Representation - An instance of an object version that is serialized as STIX

Producer - Any entity that distributes STIX content, including object creators as well as those passing along existing content

SDO - STIX Domain Object (a "node" in a graph)

SRO - STIX Relationship Object (one mechanism to represent an "edge" in a graph)

STIX - Structured Threat Information Expression

STIX Content - STIX documents, including STIX Objects, STIX Objects grouped as bundles, etc.

STIX Object - A STIX Domain Object (SDO) or STIX Relationship Object (SRO)

STIX Relationship - A link (an "edge" in a graph) between two STIX Objects represented by either an SRO or an embedded relationship

TAXII - An application layer protocol for the communication of cyber threat information

TLP - Traffic Light Protocol

TTP - Tactic, technique, or procedure; behaviors and resources that attackers use to carry out their attacks

10. Appendix B. Acknowledgments

STIX Subcommittee Chairs:

Sarah Kelley, Center for Internet Security (CIS)

John Wunder, MITRE Corporation

Cyber Observable Subcommittee Chairs:

Trey Darley, Kingfisher Operations, sprl

Ivan Kirillov, MITRE Corporation

Special Thanks:

Substantial contributions to this specification from the following individuals are gratefully acknowledged:

Sarah Kelley, Center for Internet Security (CIS)

Terry MacDonald, Cosive

Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)

Richard Struse, DHS Office of Cybersecurity and Communications

Iain Brown, GDS

Jason Keirstead, IBM

Tim Casey, Intel

Trey Darley, Kingfisher Operations, sprl

Allan Thomson, LookingGlass Cyber

Greg Back, MITRE Corporation

Ivan Kirillov, MITRE Corporation

Jon Baker, MITRE Corporation

John Wunder, MITRE Corporation

Sean Barnum, MITRE Corporation

Richard Piazza, MITRE Corporation

Christian Hunt, New Context Services, Inc.

John-Mark Gurney, New Context Services, Inc.

Aharon Chernin, Perch

Dave Cridland, Surevine

Bret Jordan, Symantec Corp.

Participants:

The following individuals were members of the OASIS CTI Technical Committee during the creation of this specification and their contributions are gratefully acknowledged:

David Crawford, Aetna

Marcos Orallo, Airbus Group SAS

Roman Fiedler, AIT Austrian Institute of Technology

Florian Skopik, AIT Austrian Institute of Technology

Russell Spitler, AlienVault

Ryan Clough, Anomali

Nicholas Hayden, Anomali

Wei Huang, Anomali

Angela Nichols, Anomali

Hugh Njemanze, Anomali

Katie Pelusi, Anomali

Dean Thompson, Australia and New Zealand Banking Group (ANZ Bank)

Alexander Foley, Bank of America

Sounil Yu, Bank of America

Vicky Laurens, Bank of Montreal

Humphrey Christian, Bay Dynamics

Ryan Stolte, Bay Dynamics

Alexandre Dulaunoy, CIRCL

Andras Iklody, CIRCL

Rapha‘l Vinot, CIRCL

Sarah Kelley, CIS

Syam Appala, Cisco Systems

Ted Bedwell, Cisco Systems

David McGrew, Cisco Systems

Mark-David McLaughlin, Cisco Systems

Pavan Reddy, Cisco Systems

Omar Santos, Cisco Systems

Jyoti Verma, Cisco Systems

Doug DePeppe, Cyber Threat Intelligence Network, Inc. (CTIN)

Jane Ginn, Cyber Threat Intelligence Network, Inc. (CTIN)

Ben Othman, Cyber Threat Intelligence Network, Inc. (CTIN)

Jeff Odom, Dell

Sreejith Padmajadevi, Dell

Ravi Sharda, Dell

Will Urbanski, Dell

Sean Sobieraj, DHS Office of Cybersecurity and Communications (CS&C)

Richard Struse, DHS Office of Cybersecurity and Communications (CS&C)

Marlon Taylor, DHS Office of Cybersecurity and Communications (CS&C)
Jens Aabol, Difi-Agency for Public Management and eGovernment
Wouter Bolsterlee, EclecticIQ
Marko Dragoljevic, EclecticIQ
Oliver Gheorghe, EclecticIQ
Joep Gommers, EclecticIQ
Sergey Polzunov, EclecticIQ
Rutger Prins, EclecticIQ
Andrei S”rghi, EclecticIQ
Raymon van der Velde, EclecticIQ
Ben Sooter, Electric Power Research Institute (EPRI)
Chris Ricard, Financial Services Information Sharing and Analysis Center (FS-ISAC)
Phillip Boles, FireEye, Inc.
Prasad Gaikwad, FireEye, Inc.
Rajeev Jha, FireEye, Inc.
Anuj Kumar, FireEye, Inc.
Shyamal Pandya, FireEye, Inc.
Paul Patrick, FireEye, Inc.
Scott Shreve, FireEye, Inc.
Jon Warren, FireEye, Inc.
Remko Weterings, FireEye, Inc.
Gavin Chow, Fortinet Inc.
Steve Fossen, Fortinet Inc.
Kenichi Terashita, Fortinet Inc.
Ryusuke Masuoka, Fujitsu Limited
Daisuke Murabayashi, Fujitsu Limited
Derek Northrope, Fujitsu Limited
Jonathan Algar, GDS
Iain Brown, GDS
Adam Cooper, GDS
Mike McLellan, GDS
Tyrone Nembhard, GDS
Chris O'Brien, GDS
James Penman, GDS
Howard Staple, GDS
Chris Taylor, GDS
Laurie Thomson, GDS
Alastair Treharne, GDS

Julian White, GDS

Bethany Yates, GDS

Robert van Engelen, Genivia

Eric Burger, Georgetown University

Allison Miller, Google Inc.

Mark Risher, Google Inc.

Yoshihide Kawada, Hitachi, Ltd.

Jun Nakanishi, Hitachi, Ltd.

Kazuo Noguchi, Hitachi, Ltd.

Akihito Sawada, Hitachi, Ltd.

Yutaka Takami, Hitachi, Ltd.

Masato Terada, Hitachi, Ltd.

Peter Allor, IBM

Eldan Ben-Haim, IBM

Allen Hadden, IBM

Sandra Hernandez, IBM

Jason Keirstead, IBM

John Morris, IBM

Laura Rusu, IBM

Ron Williams, IBM

Paul Martini, iboss, Inc.

Jerome Athias, Individual

Peter Brown, Individual

Joerg Eschweiler, Individual

Stefan Hagen, Individual

Elysa Jones, Individual

Sanjiv Kalkar, Individual

Terry MacDonald, Individual

Alex Pinto, Individual

Tim Casey, Intel Corporation

Kent Landfield, Intel Corporation

Karin Marr, Johns Hopkins University Applied Physics Laboratory

Julie Modlin, Johns Hopkins University Applied Physics Laboratory

Mark Moss, Johns Hopkins University Applied Physics Laboratory

Mark Munoz, Johns Hopkins University Applied Physics Laboratory

Nathan Reller, Johns Hopkins University Applied Physics Laboratory

Pamela Smith, Johns Hopkins University Applied Physics Laboratory

David Laurance, JPMorgan Chase Bank, N.A.

Russell Culpepper, Kaiser Permanente

Beth Pumo, Kaiser Permanente

Michael Slavick, Kaiser Permanente

Trey Darley, Kingfisher Operations, sprl

Gus Creedon, Logistics Management Institute

Wesley Brown, LookingGlass

Jamison Day, LookingGlass

Kinshuk Pahare, LookingGlass

Allan Thomson, LookingGlass

Ian Truslove, LookingGlass

Chris Wood, LookingGlass

Greg Back, Mitre Corporation

Jonathan Baker, Mitre Corporation

Sean Barnum, Mitre Corporation

Desiree Beck, Mitre Corporation

Michael Chisholm, Mitre Corporation

Nicole Gong, Mitre Corporation

Ivan Kirillov, Mitre Corporation

Michael Kouremetis, Mitre Corporation

Chris Lenk, Mitre Corporation

Richard Piazza, Mitre Corporation

Larry Rodrigues, Mitre Corporation

Jon Salwen, Mitre Corporation

Charles Schmidt, Mitre Corporation

Alex Tweed, Mitre Corporation

Emmanuelle Vargas-Gonzalez, Mitre Corporation

John Wunder, Mitre Corporation

James Cabral, MTG Management Consultants, LLC.

Scott Algeier, National Council of ISACs (NCI)

Denise Anderson, National Council of ISACs (NCI)

Josh Poster, National Council of ISACs (NCI)

Mike Boyle, National Security Agency

Joe Brule, National Security Agency

Jessica Fitzgerald-McKay, National Security Agency

David Kemp, National Security Agency

Shaun McCullough, National Security Agency

John Anderson, NC4

Michael Butt, NC4

Mark Davidson, NC4

Daniel Dye, NC4

Angelo Mendonca, NC4

Michael Pepin, NC4

Natalie Suarez, NC4

Benjamin Yates, NC4

Daichi Hasumi, NEC Corporation

Takahiro Kakumaru, NEC Corporation

Lauri Korts-P_rn, NEC Corporation

John-Mark Gurney, New Context Services, Inc.

Christian Hunt, New Context Services, Inc.

Daniel Riedel, New Context Services, Inc.

Andrew Storms, New Context Services, Inc.

Stephen Banghart, NIST

David Darnell, North American Energy Standards Board

Cory Casanave, Object Management Group

Aharon Chernin, Perch

Dave Eilken, Perch

Sourabh Satish, Phantom

Josh Larkins, PhishMe Inc.

John Tolbert, Queralt Inc.

Ted Julian, Resilient Systems, Inc..

Igor Baikalov, Securonix

Joseph Brand, Semper Fortis Solutions

Duncan Sparrell, sFractal Consulting LLC

Thomas Schreck, Siemens AG

Rob Roel, Southern California Edison

Dave Cridland, Surevine Ltd.

Bret Jordan, Symantec Corp.

Curtis Kostrosky, Symantec Corp.

Juha Haaga, Synopsys

Masood Nasir, TELUS

Greg Reaume, TELUS

Alan Steer, TELUS

Crystal Hayes, The Boeing Company

Wade Baker, ThreatConnect, Inc.

Cole Iliff, ThreatConnect, Inc.

Andrew Pendergast, ThreatConnect, Inc.

Ben Schmoker, ThreatConnect, Inc.

Jason Spies, ThreatConnect, Inc.

Ryan Trost, ThreatQuotient, Inc.

Patrick Coughlin, TruSTAR Technology

Chris Roblee, TruSTAR Technology

Mark Angel, U.S. Bank

Brian Fay, U.S. Bank

Joseph Frazier, U.S. Bank

Mark Heidrick, U.S. Bank

Mona Magathan, U.S. Bank

Yevgen Sautin, U.S. Bank

Richard Shok, U.S. Bank

James Bohling, US Department of Defense (DoD)

Eoghan Casey, US Department of Defense (DoD)

Gary Katz, US Department of Defense (DoD)

Jeffrey Mates, US Department of Defense (DoD)

Evette Maynard-Noel, US Department of Homeland Security

Robert Coderre, VeriSign

Kyle Maxwell, VeriSign

Eric Osterweil, VeriSign

Patrick Maroney, Wapack Labs LLC

Anthony Rutkowski, Yanna Technologies LLC

11. Appendix C. Revision History

Revision	Date	Editor	Changes Made
01	2017-01-20	Bret Jordan, John Wunder, Rich Piazza, Ivan Kirillov, Trey Darley	Initial Version
02	2017-04-24	Bret Jordan, John Wunder, Rich Piazza, Ivan Kirillov, Trey Darley	Changes made from first public review

부 록 Ⅰ-1

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

지식재산권 확약서 정보

Ⅰ-1.1 지식재산권 확약서(1)

- 해당 사항 없음

Ⅰ-1.2 지식재산권 확약서(2)

- 해당 사항 없음

※ 상기 기재된 지식재산권 확약서 이외에도 본 표준이 발간된 후 접수된 확약서가 있을 수 있으니, TTA 웹사이트에서 확인하시기 바랍니다.

부 록 Ⅰ-2

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

시험인증 관련 사항

Ⅰ-2.1 시험인증 대상 여부

- 해당 사항 없음

Ⅰ-2.2 시험표준 제정 현황

- 해당 사항 없음

부 록 Ⅰ-3

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

본 표준의 연계(family) 표준

Ⅰ-3.1 TTA, 구조화된 위협 정보 표현 규격(STIX™) 버전 2.0 - 제2부: STIX 객체

STIX 도메인 객체의 집합을 정의하는 문서로 객체들의 구성요소와 구성요소에 대한 설명을 제공

Ⅰ-3.2 TTA, 구조화된 위협 정보 표현 규격(STIX™) 버전 2.0 - 제3부: STIX 사이버 관측 코어 개념

STIX의 Observable 핵심 개념을 정의하는 문서로 관측 가능한 객체를 구성하는 필드와 필드에 대한 설명을 제공

Ⅰ-3.3 TTA, 구조화된 위협 정보 표현 규격(STIX™) 버전 2.0 - 제4부: STIX 사이버 관측 객체

STIX의 관측 가능한 객체 집합을 정의하는 문서로 관측 가능한 객체의 구성요소와 구성요소에 대한 설명을 제공

Ⅰ-3.4 TTA, 구조화된 위협 정보 표현 규격(STIX™) 버전 2.0 - 제5부: STIX 패터닝

STIX의 Indicator 지원 패턴을 정의하는 문서로 Indicator 지원 패턴을 구성하는 필드와 필드에 대한 설명을 제공

부 록 Ⅰ-4

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

참고 문헌

[1] IEEE 754-2008, “IEEE Standard for Floating-Point Arithmetic”, August 2008. [Online]. Available: http://ieeexplore.ieee.org/document/4610935/

[2] ISO 10646, “ISO/IEC 10646:2014 Information technology -- Universal Coded Character Set (UCS)”, 2014. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c063182_ISO_IEC_10646_2014.zip

[3] RFC 2119, Bradner, S., “"Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, http://www.rfc-editor.org/info/rfc2119.

[4] RFC 3339, Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, http://www.rfc- editor.org/info/rfc3339.

[5] RFC 3986, Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, http://www.rfc-editor.org/info/rfc3986.

[6] RFC4122, Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, http://www.rfc-editor.org/info/rfc4122.

[7] RFC7159, Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014. http://www.rfc-editor.org/info/rfc7159.txt.

[8] TLP, Traffic Light Protocol, Version 1.0 (TLP). (2016, Aug. 25). FIRST. [Online]. Available: https://first.org/tlp

부 록 Ⅰ-5

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

영문표준 해설서

Ⅰ-5.1 개요

Ⅰ-5.1.1 그래프 기반 모델(Graph-Based Model)

STIX 2.0은 노드와 엣지의 연결된 그래프이다. STIX 2.0에서는 도메인 객체를 통해 그래프 노드를 정의하고 STIX 관계(STIX 관계 객체와 포함된 관계 포함)를 통해 엣지를 정의한다. STIX 도메인 객체와 STIX 관계 객체를 통틀어 STIX 객체라 한다. 이 그래프 기반 언어는 일반적인 분석 접근방식을 준수하며 이 언어를 통해 유연한 모듈형의 구조화되고 일관성 있는 CTI 표시가 가능하다.

Ⅰ-5.1.2 STIX™ 도메인 객체(STIX™ Domain Objects)

STIX 2.0은 STIX 도메인 객체(SDO, STIX Domain Object) 집합: 공격 패턴, 캠페인, 조치, ID, Indicator, 침입 집단, 멀웨어, 관측 데이터, 보고서, 위협 행위자, 도구 및 취약점을 정의한다. 이러한 각 객체는 CTI에 널리 사용되는 개념에 해당한다. STIX 관계를 따라 SDO의 구성 요소를 사용하여 폭 넓고 포괄적인 CTI를 만들고 공유할 수 있다.

STIX 도메인 객체는 모두 공통의 속성 집합을 공유한다. 이러한 공통 속성은 버전 관리, 데이터 표시(데이터를 공유하고 사용할 수 있는 방법을 나타냄) 및 확장성 등의 표준 기능을 제공한다.

Ⅰ-5.1.3 STIX™ 관계(STIX™ Relationships)

관계는 객체들이 관련되는 방법을 설명하는 STIX 객체 간의 링크이다. 대부분의 관계는 STIX 관계 객체(SRO, STIX Relationship Object)를 사용하여 표시되지만, ID 참조로 표시되는 특수한 포함된 관계도 있다.

일반적인 객체 관계는 두 SRO 중의 하나이며 STIX에서 대부분의 관계에 사용된다. 이 일반 관계 객체는 관계가 나타내는 대상을 더 구체적으로 설명하는 relationship_type이라는 속성을 포함하고 있다. 이 표준은 특정 형식의 SDO 간의 relationship_type 속성에 사용하기 위한 알려진 용어 집합을 정의한다. 예를 들어 Indicator SDO는 Indicator를 사용하여 해당 멀웨어를 검색할 수 있는 방법을 설명하기 위해서 indicates의 relationship_type을 통해 멀웨어에 대한 관계를 정의한다. 표준에 정의된 용어 외에, 관계 형식으로 사용할 사용자 지정 용어도 STIX에서 허용된다.

현재 유일한 다른 SRO(일반 관계 외에)는 발견(Sighting) 관계 객체이다. 발견 객체는 엔터티(예:IP주소)가 Indicator 발견 등과 같이 SDO를 "발견"한 경우를 포착하기 위해 사용된다. 발견은 발견 객체에만 적용할 수 있는 count 등의 추가 속성을 포함하고 있기 때문에 별도의 SRO이다. 다른 SRO는 미래의 STIX 버전에서 일반 관계 객체에 없는 추가 속성도 필요한 새로운 관계가 식별될 경우 정의될 수 있다.

Ⅰ-5.1.4 사이버 관측 가능 객체(Cyber Observables)

STIX 언어의 어떤 부분에 대해서는 관측 대상 객체의 구조화된 표시와 사이버 도메인에서 해당 객체의 속성을 설명해야 한다. 이러한 기능은 더 높은 수준의 개념을 설명하는데 사용되는 STIX 부분과 여러 가지 면에서 다르다. 사이버 관측 가능 객체 섹션에서는 하나 이상의 관측 대상 데이터 요소, 예를 들어 존재하는 파일, 실행 중인 것으로 관측된 프로세스 또는 두 IP 간에 네트워크 트래픽이 발생한 사실에 관한 정보를 설명한다. 즉, 발생한 일에 관한 사실을 설명하지만, 누가 발생시켰는지 또는 언제 발생했는지에 대해서는 설명하지 않을 수도 있으며 발생한 이유는 절대 설명하지 않는다.

Ⅰ-5.1.5 STIX™ 패턴화(STIX™ Patterning)

네트워크와 끝점에서 악의적일 수 있는 활동의 검색을 강화하려면 사이버 환경에서 찾을 대상을 설명하는 표준 언어가 필요하다. STIX 패턴화 언어를 사용하면 위협 인텔리전스 플랫폼 또는 기타 유사한 시스템에 의해 타임스탬프가 포함된 사이버 관측 가능 객체 데이터(STIX 관측 대상 데이터 객체 등)를 수집하고 이 데이터와 대조함으로써 다른 분석 도구와 시스템에 발생 가능한 침해사고에 대응하고 처리하도록 구성할 수 있다.

Ⅰ-5.1.6 어휘(Vocabularies)

많은 STIX 객체는 정의된 값의 집합에서 그 값을 선택할 수 있는 속성을 포함하고 있다. 이러한 값의 집합을 어휘라 하며 정확히 같은 문자열을 사용하여 동일한 개념을 나타낼 가능성을 높여서 상호운용성을 향상시키기 위해 STIX 에 정의되어 있다. 어휘를 일관성 있게 사용하면 예를 들어 한 엔터티가 에너지 부문을 "에너지"로 참조하고 다른 엔터티가 "에너지 부문"으로 참조하는 혼란의 가능성이 감소하므로 비교와 상관성 확인이 더 쉬워진다. STIX 어휘의 미리 정의된 값을 사용하는 것은 권장되지만 경우에 따라 이러한 사용이 불가능하거나 바람직하지 않을 수 있다. STIX는 어휘를 "개방형"으로 정의하여 이 기능을 지원하는데, 어휘를 이렇게 정의하면 엔터티에서 제안 어휘에서 벗어난 값을 사용할 수 있게 된다.

Ⅰ-5.1.7 직렬화(Serialization)

STIX는 특정 저장소 또는 직렬화를 정의하지 않는다. 그러나 STIX 2.0에는 JSON[RFC7159]이라는 구현 의무(MTI, mandatory-to-implement) 직렬화가 있다. 다시 말해서 모든 STIX 표준 도구는 JSON 지원을 구현해야 하며 다른 직렬화 지원을 구현할 수 있다.

Ⅰ-5.1.8 STIX™ 전송(Transporting STIX™)

STIX 2.0은 전송에 대해 알 수 없다. 즉, 구조와 직렬화가 특정 전송 메커니즘에 의존하지 않는다. 컴패니언 CTI 표준 TAXII™는 STIX 객체를 전송하도록 특별히 설계되었다. STIX는 특히 TAXII 이외의 통신 메커니즘에 대해 벌크 STIX 데이터를 전송할 수 있도록 하는 STIX 객체용 컨테이너로 번들을 제공한다.

Ⅰ-5.2 공통 데이터 형식

STIX의 공통 데이터 형식은 부울 형식, 외부 참조 형식, 부동 소수점 수 형식, 해시 형식, 식별자 형식, 정수 형식, 킬 체인 단계 형식, 목록 형식, 개방형 어휘 형식, 문자열 형식, 타임스탬프 형식 등 11가지 형식을 활용한다.

<표 Ⅰ-5.2-1> 공통 데이터 형식

형식	설명
boolean	true(참)또는 false(거짓)인 값이다.
external-reference	STIX 이외의 ID 또는 다른 관련 외부 콘텐츠에 대한 참조이다.
float	배정밀도(double precision) 형태의 수이다.
hashes	하나 이상의 암호화 해시이다.
identifier	STIX 도메인 객체, STIX 관계 객체, 번들 또는 표시 정의에 대한 식별자(ID)이다.
integer	자연수이다.
kill-chain-phase	킬 체인 단계를 나타낸다.
list	목록에 나타나는 방법에 따라 순서가 정해지는 값의 시퀀스이다. 목록 안의 모든 값이 지정된 형식을 준수해야 함을 나타내려면 “<type> 형식의 목록”이라는 문구를 사용한다.
open-vocab	STIX 개방형(open-vocab) 또는 제안 어휘의 값이다.
string	일련의 유니코드 문자이다.
timestamp	시간 값(날짜 및 시간)이다.

Ⅰ-5.2.1 부울(Boolean)

부울(boolean)은 true 또는 false인 값이다. 이 형식의 속성은 true 또는 false인 값을 가져야 한다. JSON MTI 직렬화는 리터럴(따옴표 없는) true 또는 false인 JSON 부울 형식[RFC7159]을 사용한다.

Ⅰ-5.2.2 외부 참조(External Reference)

외부 참조는 STIX 외부에 표시되는 정보를 가리킨다는 것을 설명하기 위해 사용된다. 예를 들어 멀웨어 객체는 외부 참조를 사용하여 외부 데이터베이스 또는 보고서의 멀웨어가 참조를 통해 소스 자료를 표시할 수 있음을 나타낼 수 있다. JSON MTI 직렬화는 외부 참조를 표시할 때 JSON 객체 형식[RFC7159]을 사용한다.

Ⅰ-5.2.3 부동 소수점 수(Float)

부동 소수점 수 데이터 형식은 IEEE 754[IEEE 754-2008] 배정밀도 수(예: 소수점 부분이 있는 수)를 표시한다. 그러나 값 ±Infinity와 NaN은 JSON에서 표시할 수 없기 때문에 STIX에서는 유효한 값이 아니다. JSON MTI 직렬화에서 부동 소수점 값은 JSON 수 형식[RFC7159]에 의해 표시된다.

Ⅰ-5.2.4 해시(Hashes)

해시 형식은 하나 이상의 암호화 해시를 특수한 키/값 쌍 집합으로 표시한다. 따라서 각 해시 알고리즘의 이름은 사전의 키로 지정해야 하며 해당 값을 생성하는 데 사용되는 해시 알고리즘의 이름을 식별해야 한다. 이 이름은 hash-algorithm-ov에 정의된 값 또는 앞에 “x_”가 붙는 사용자 지정 값(예: “x_custom_hash”) 중 하나이어야 한다.

키는 각 hashes 속성 내에서 고유해야 하고 ASCII 형식이어야 하며 a~z(소문자 ASCII), A~Z(대문자 ASCII), 숫자 0~9, 하이픈(-) 및 밑줄(_) 문자로 제한된다. 키는 길이가 ASCII 문자 최소 3자 이상이어야 하고 30자 이하인 것이 바람직하며 256자 이하이어야 한다.

Ⅰ-5.2.5 식별자(Identifier)

식별자(identifier)는 SDO, SRO, 번들 또는 표시 정의를 전반적으로 그리고 고유하게 식별한다. ID는 object-type--UUIDv4 형태를 따라야 하며, 여기서 object-type은 식별 또는 참조되는 객체의 type 속성에서 가져온 정확한 값(모든 형식 이름은 정의에 의해 소문자 문자열), UUIDv4는 RFC 4122 표준 버전 4 UUID이다. UUID는 RFC 4122 에 정의된 알고리즘이어야 한다. JSON MTI 직렬화는 식별자(ID)를 표시할 때 JSON 문자열 형식[RFC7159]을 사용한다.

Ⅰ-5.2.6 정수(Integer)

정수 데이터 형식은 자연수를 표시한다. 달리 지정하지 않은 한, 모든 정수는 부호 포함 64비트 값([-(2**53)+1, (2**53)-1])으로 표시할 수 있어야 한다. 추가 제한을 사용하는 경우 설명에 따라 형식에 대해 해당 제한을 적용할 수 있다. JSON MTI 직렬화에서 정수는 JSON 수 형식[RFC7159]에 의해 표시된다.

Ⅰ-5.2.7 킬 체인 단계(Kill Chain Phase)

킬 체인 단계(kill-chain-phase)는 킬 체인(공격자가 자신의 목적을 달성하기 위해 실행할 수 있는 여러 단계를 말함)의 단계를 표시한다. JSON MTI 직렬화는 킬 체인 단계를 표시할 때 JSON 객체 형식[RFC7159]을 사용한다. 록히드 마틴 사의 사이버 킬 체인(Cyber Kill Chain™)을 참조하는 경우 kill_chain_name은 반드시 lockheed-martin-cyber-kill-chain 이어야 한다.

Ⅰ-5.2.8 목록(List)

list 형식은 목록에 나타나는 방법에 따라 순서가 정해지는 값의 시퀀스를 정의한다. 목록 안의 모든 값이 지정된 형식을 준수해야 함을 나타내려면 “<type> 형식의 목록”이라는 문구를 사용한다. 예를 들어 integer 형식의 목록은 목록의 모든 값이 integer 형식이어야 함을 의미한다. 이 표준은 목록에 허용되는 값의 최대 수를 지정하지 않지만 목록의 모든 인스턴스는 적어도 하나의 값을 가져야 한다. 특정 STIX 객체 속성은 목록의 길이에 대한 더 제한적인 상위 및/또는 하위 경계를 정의할 수 있다.

빈 목록은 STIX에서는 금지되며 선택 사항인 경우 속성을 생략하는 대안으로 사용하지 않아야 한다. 속성이 필수인 경우, 목록은 반드시 존재해야 하며 적어도 하나의 값을 가져야 한다. JSON MTI 직렬화는 0개 이상의 값의 순서가 정해진 목록인 JSON 배열 형식[RFC7159]을 사용한다.

Ⅰ-5.2.9 개방형 어휘(Open Vocabulary)

open-vocab 형식은 문자열(string)로 표시된다. 이 형식을 사용하는 속성에 대해서는 해당 속성의 정의에서 식별되는 제안 값의 목록(일명 제안 어휘)이 있다. 속성의 값은 제안 어휘에서 선택하는 것이 바람직하지만 다른 string 값일 수 있다. 제안 어휘에서 가져오지 않은 값은 모두 소문자인 것이 바람직하며(소문자는 지역성 규칙에 의해 정의됨) 단어 구분 기호로 공백이나 밑줄 대신 대시를 사용하는 것이 바람직하다.

제안 어휘에 정의되지 않은 하나 이상의 open-vocab 용어와 함께 STIX 콘텐츠를 받는 소비자는 이러한 값을 무시할 수 있다. JSON MTI 직렬화는 open-vocab를 표시할 때 JSON 문자열 형식[RFC7159]을 사용한다.

Ⅰ-5.2.10 문자열(String)

string 데이터 형식은 유니코드로 코딩한 문자 집합[ISO10646]에서 가져온 유효한 문자의 유한 길이 문자열을 나타낸다. 유니코드는 ASCII 및 많은 다른 국제 문자 집합의 문자를 포함하고 있다. JSON MTI 직렬화는 JSON 문자열 형식[RFC7159]을 사용하며, 이 형식은 유니코드를 지원하기 위해 UTF-8 인코딩을 의무화한다.

Ⅰ-5.2.11 타임스탬프(Timestamp)

timestamp 형식은 STIX에서 날짜 및 시간을 표시하는 방법을 정의한다. JSON MTI 직렬화는 타임스탬프(timestamp)를 표시할 때 JSON 문자열 형식[RFC7159]을 사용한다.

timestamp 속성은 YYYY-MM-DDTHH:mm:ss[.s+]Z 형식을 사용하는 유효한 RFC 3339 형식의 타임스탬프[RFC3339]이어야 하며, 여기서 “s+”는 한 개 이상의 1초 미만 값을 표시한다. 대괄호는 1초 미만의 정밀도가 선택 사항이고 자릿수가 제공되지 않은 경우 소수 자릿수가 존재하지 않아야 한다. 타임스탬프는 UTC 표준 시간대로 표시해야 하며 “Z” 지정을 사용하여 이 점을 나타내야 한다.

Ⅰ-5.3 STIX^{TM} 객체

모든 STIX 객체를 표시할 때 JSON 객체 형식을 사용한다. STIX 객체는 공통 속성, ID 및 참조, 객체 작성자, 버전 관리, 공통 관계, 예약된 속성 등 6가지로 구성된다.

Ⅰ-5.3.1 공통 속성(Common Properties)

<표 Ⅰ-5.3-1> STIX 객체 공통 속성

속성 이름	형식	설명
type(필수)	string	type 속성은 STIX 객체의 형식을 식별한다. type 속성의 값은 STIX^{TM} Version 2.0.

		Part 2: STIX Objects의 섹션 2와 3에 정의된 STIX 객체의 형식 중 하나의 이름(예: indicator) 또는 섹션 7.2에 정의된 사용자 지정 객체의 이름이어야 한다.
id(필수)	identifier	id 속성은 이 객체를 전반적으로 그리고 고유하게 식별한다. 같은 id를 가진 모든 객체는 같은 객체의 다른 버전으로 간주한다. 객체 형식은 identifier의 일부이므로 서로 다른 형식의 객체에 대해 같은 id를 공유하는 것은 유효하지 않다.
created_by_ref(선택 사항)	identifier	created_by_ref 속성은 이 객체를 만든 엔터티를 설명하는 ID 객체의 ID를 지정한다. 이 속성이 생략되었다면 이 정보의 출처가 정의되어 있지 않은 것이다. 익명으로 남아있기를 바라는 객체 작성자가 이 속성을 사용할 수 있다.
created(필수)	timestamp	created 속성은 이 객체의 첫 번째 버전이 만들어진 시간을 표시한다. 객체 작성자는 객체가 만들어질 때 가장 적절해 보이는 시간을 사용할 수 있다. created 속성은 객체의 새 버전을 만들 때 변경되지 않아야 한다. created 타임스탬프는 가장 가까운 밀리초(초의 소수점 뒤에 정확히 세 자리)의 정밀도를 가져야 한다. 버전 관리에 대한 추가 정의는 3.4 섹션을 참조한다.
modified(필수)	timestamp	modified 속성은 객체의 이 특정 버전이 만들어진 시간을 표시한다. 객체 작성자는 객체의 이 버전이 수정될 때 가장 적절해 보이는 시간을 사용할 수 있다. 지정된 객체 버전에 대한 modified 속성의 값은 created 속성의 값 이상이어야 한다. 객체 작성자는 객체의 새 버전을 만들 때 modified 속성을 설정해야 한다. modified 타임스탬프는 가장 가까운 밀리초(초의 소수점 뒤에 정확히 세 자리)의 정밀도를 가져야 한다. 버전 관리에 대한 추가 정의는 3.4 섹션을 참조한다.
revoked(선택 사항)	boolean	revoked 속성은 객체가 철회되었는지 여부를 나타낸다. 철회된 객체는 객체 작성자가 더 이상 유효하다고 생각하지 않는다. 객체 철회는 영구적이며, 이 id를 가진 객체의 미래 버전을 만들면 안 된다. 이 속성의 기본값은 false이다. 버전 관리에 대한 추가 정의는 3.4 섹션을 참조한다.
labels(선택 사항)	string 형식의 list	labels 속성은 분류의 집합을 지정한다. 각 STIX 객체는 labels 속성에 대한 제안 어휘를 정의할 수 있다. 예를 들어 STIX™ Version 2.0. Part 2: STIX Objects의 섹션 2.5에 정의되어 있는 indicator 객체는 섹션 6.5에 정의된 indicator 레이블 어휘를 사용한다. 어휘가 정의된 경우 이 목록의 항목은 어휘에서 가져오는 것이 바람직하다. 추가 레이블은 제안 어휘 내에 있는 내용을 초과하여 추가할 수 있다.
external_references	external-reference 형식	external_references 속성은 STIX 이외의

(선택 사항)	의 list	정보를 참조하는 외부 참조의 목록을 지정한다. 이 속성은 다른 시스템의 레코드에 대해 하나 이상의 URL, 설명 또는 ID를 제공하기 위해 사용된다.
object_marking_refs(선택 사항)	identifier 형식의 list	object_marking_refs 속성은 이 객체에 적용되는 marking-definition 객체의 ID의 목록을 지정한다. 데이터 표시에 대한 추가 정의는 4 섹션을 참조한다.
granular_markings(선택 사항)	granular-marking 형식의 list	granular_markings 속성은 이 객체에 적용된 세분화된 표시의 목록을 지정한다. 데이터 표시에 대한 추가 정의는 4 섹션을 참조한다.

Ⅰ-5.3.2 ID 및 참조(IDs and References)

id 속성은 SDO, SRO, 번들 또는 표시 정의를 전반적으로 그리고 고유하게 식별하며, identifier 형식의 요구사항을 만족해야 한다. 모든 STIX 객체(및 번들과 표시 정의)는 identifier 형식에 의해 정의된 ID를 사용한다. identifier 형식은 다른 구성체(모든 STIX 객체의 created_by_ref 속성 등)에 대한 ID 참조인 속성을 정의하는 데에도 사용된다. ID 참조의 확인은 ID 참조 속성에 의해 참조한 실제 객체를 식별하고 가져오는 프로세스이다. ID 참조는 ID 참조 속성(예: created_by_ref)의 값이 다른 객체의 id 속성과 정확히 일치할 때 객체로 확인된다. 소비자가 객체의 여러 버전에 대한 액세스 권한을 가진 경우, 소비자는 해당 객체에 대한 참조를 I-5.3.4에 정의된 내용을 참조하는 것으로 해석한다. ID 참조를 통해 소비자/생산자가 현재 가지고 있지 않은 객체를 참조할 수 있다.

Ⅰ-5.3.3 객체 작성자(Object Creator)

객체 작성자는 지정된 객체에 대한 id 속성을 생성하는 엔터티(예: 시스템, 조직, 도구의 인스턴스)이다. 객체 작성자는 ID 객체로 표시된다. 객체 작성자를 표시하는 ID 객체에 대한 포함된 관계는 created_by_ref 속성에 포착될 수 있다. (또는 속성을 비워 둘 수 있다. 즉, 객체 작성자가 익명일 수 있다.)

객체를 변경하지 않고, 즉 원래 id를 유지하면서 다른 엔터티에서 객체를 다시 게시하는 엔터티는 객체 작성자로 간주되지 않으며 created_by_ref 속성을 변경해서는 안 된다. 객체를 수락하고 수정, 추가 또는 생략하여 다시 게시하는 엔터티는 해당 객체에 대한 새 id를 만들어야 한다. 버전 관리 목적을 위한 경우 이러한 엔터티는 새 객체의 객체 작성자로 간주된다.

Ⅰ-5.3.4 버전 관리(Versioning)

버전 관리는 객체 작성자가 자신이 만드는 STIX 객체를 업데이트하고 철회하는 데 사용하는 메커니즘이다. STIX 객체는 revoked, created 및 modified 속성을 사용하여 버전을 관리한다.

STIX 객체는 정보를 업데이트, 추가 또는 제거하기 위해 버전 관리를 할 수 있다. STIX 객체의 버전은 해당 객체의 id 속성과 modified 속성 조합에 의해 고유하게 식별된다. 객

체의 첫 번째 버전은 created 및 modified 속성에 대해 동일한 타임스탬프를 가져야 한다. modified 속성의 최근 값이 객체의 최신 버전을 나타낸다. 구현에서는 최근 modified 값을 가진 STIX 객체의 버전을 해당 객체의 최근 상태로 생각해야 한다. 객체의 모든 새 버전에 대해 modified 속성은 새 버전이 만들어진 시기를 나타내도록 업데이트된다. 소비자가 서로 다르지만 id와 modified 타임스탬프가 동일한 두 객체를 수신하는 경우, 소비자가 객체를 처리하는 방법은 정의되지 않는다.

STIX 객체는 단일 객체 작성자를 가지며, 이 작성자는 객체에 대한 id를 생성하고 첫 번째 버전을 만드는 엔터티이다. 객체 작성자는 객체의 created_by_ref 속성에서 식별될 수도 있지만 반드시 그런 것은 아니다. 객체 작성자에 대해서만 STIX 객체의 새 버전을 만드는 것이 허용된다. 객체 작성자 이외의 생산자는 이 객체의 새 버전을 만들어서는 안된다. 객체 작성자 이외의 생산자가 새 버전을 만들고 싶으면 새 id를 가진 새 객체를 만들어야 한다. 또한 새 객체를 해당 객체가 파생된 원본 객체에 관련시키는 derived-from 관계 객체를 만드는 것이 바람직하다.

객체 버전(객체의 id 및 modified 속성에 의해 식별된)의 모든 표현(객체 버전이 직렬화 및 공유될 때마다)은 언제나 같은 속성 집합과 각 속성에 대한 같은 값을 가져야 한다. 속성의 값을 변경하거나 속성을 추가 또는 제거하려면 변경 시간으로 modified 속성을 업데이트하여 새 버전임을 나타내야 한다.

Ⅰ-5.3.5 공통 관계(Common Relationships)

각 SDO는 해당 SDO의 정의에 지정된 자체의 관계 형식 집합을 가지고 있다. 다음의 공통 관계 형식은 모든 SDO에 대해 정의된다.

Ⅰ-5.3.6 예약된 속성(Reserved Properties)

향후 이 문서의 개정판에서 사용하도록 예약된 속성 이름을 정의한다. 이 섹션에 정의된 속성 이름을 사용자 지정 속성의 이름에 사용해서는 안 된다.

Ⅰ-5.4 데이터 표시

데이터 표시는 제한, 사용 권한 및 데이터를 사용하고 공유할 수 있는 방법에 대한 다른 지침을 표시한다. 예를 들어 데이터를 공유를 금지하거나 암호화해야 하는 제한을 적용한 상태로 공유할 수 있다. STIX 데이터 표시는 표시 정의(marking-definition) 객체를 사용하여 지정된다. 이러한 정의는 객체 표시를 사용하여 STIX 객체를 완성하기 위해, 그리고 세분화된 표시를 통해 STIX 객체의 개별 속성에 적용된다.

일부 형식의 표시 정의 또는 신뢰 그룹은 다른 표시를 재정의하는 표시 또는 다른 표시에 추가할 수 있는 표시에 관한 규칙을 가지고 있다. 이 표준은 같은 객체 또는 속성에 적용되는 여러 표시를 해석하는 방법에 관한 규칙을 정의하지 않는다.

Ⅰ-5.4.1 표시 정의(Marking Definition)

표시 정의(marking-definition) 객체는 특정 표시를 나타낸다. 일반적으로 데이터 표시는 데이터에 대한 처리 또는 공유 요구사항을 표시하며 표시 정의(marking-definition) 객체

에 대한 ID의 목록을 참조하는 STIX 객체의 object_marking_refs 및 granular_markings 속성에 적용된다.

두 표시 정의 형식은 이번 장에 정의되어 있다. 즉, TLP(Traffic Light Protocol) 표시를 포착하는 TLP와 텍스트 표시 설명문을 포착하는 Statement가 그것이다. 또한 국제 침해 사고대응 팀 협의회(FIRST, Forum of Incident Response and Security Teams) 정보 교환 정책(IEP, Information Exchange Policy)은 시스템에 사용 가능한 표준이 정의된 후 향후 버전에 포함될 것으로 예상된다.

STIX 객체와 달리 표시 정의 객체는 STIX 객체에 대한 표시를 간접적으로 변경할 수 있기 때문에 버전 관리가 불가능하다. 예를 들어 Statement 표시가 "Reuse Allowed"(재사용 허용)에서 "Reuse Prohibited"(다시 사용 금지)로 변경된다면 해당 Statement 표시로 표시된 모든 STIX 객체가 객체 자체를 업데이트하지 않은 상태에서 결과적으로 업데이트된다. 그래서 그 대신에 새 텍스트를 가진 새 Statement 표시를 만들고 표시된 객체가 새 표시를 가리키도록 업데이트하는 것이 바람직하다. JSON MTI 직렬화는 표시 정의(marking-definition)를 표시할 때 JSON 객체 형식[RFC7159]을 사용한다.

<표 I-5.4-1> 표시 정의 속성

속성 이름	형식	설명
type(필수)	string	type 속성은 객체의 형식을 식별한다. 이 속성의 값은 marking-definition이어야 한다.
id(필수)	identifier	id 속성은 이 표시 정의를 전반적으로 그리고 고유하게 식별한다. 객체 형식은 identifier의 일부이므로 서로 다른 형식의 객체에 대해 같은 id를 공유하는 것은 불가능하다.
created_by_ref(선택 사항)	identifier	created_by_ref 속성은 이 표시 정의를 만든 엔터티를 설명하는 identity 객체의 ID를 지정한다. 이 속성이 생략되었다면 이 정보의 출처가 정의되어 있지 않은 것이다. 익명으로 남아 있기를 바라는 객체 작성자가 이 속성을 사용할 수 있다.
created(필수)	timestamp	created 속성은 표시 정의가 만들어진 시간을 표시한다. 객체 작성자는 객체가 만들어질 때 가장 적절해 보이는 시간을 사용할 수 있다.
external_references (선택 사항)	external-reference 형식의 list	external_references 속성은 STIX 이외의 정보를 참조하는 외부 참조의 목록을 지정한다. 이 속성은 다른 시스템의 레코드에 대해 하나 이상의 URL, 설명 또는 ID를 제공하기 위해 사용된다.
object_marking_refs(선택 사항)	identifier 형식의 list	object_marking_refs 속성은 이 표시 정의에 적용되는 marking-definition 객체의 ID의 목록을 지정한다. 이 속성은 이 표시 정의 객체에 대한 참조를 포함해서는 안 된다(즉, 순환 참조를 포함할 수 없음). 일반적이지는 않지만 경우에 따라 표시 정의 자치가 공유 또는 처리 지침으로 표시될 수 있다.

granular_markings(선택 사항)	granular-marking 형식의 list	granular_markings 속성은 이 객체에 적용된 세분화된 표시의 목록을 지정한다. 이 속성은 이 표시 정의 객체에 대한 참조를 포함해서는 안 된다(즉, 순환 참조를 포함할 수 없음). 일반적이지는 않지만 경우에 따라 표시 정의 자치가 공유 또는 처리 지침으로 표시될 수 있다.
definition_type(필수)	open-vocab	definition_type 속성은 표시 정의의 형식을 식별한다. definition_type 정의의 값은 아래 하위 섹션에 정의된 statement 또는 tlp 형식 중 하나이어야 한다(4.1.3 및 4.1.4 참조).
definition(필수)	<marking object>	definition 속성은 표시 객체 자체를 포함하고 있다(예: 4.1.4 섹션에 정의된 TLP 표시, 4.1.3 섹션에 정의된 Statement 표시 또는 다른 곳에 정의된 몇몇 다른 표시 정의).

데이터 표시는 STIX 객체가 아니며 STIX 객체에 대해 또는 STIX 객체로부터 SRO 관계를 가져서는 안 된다.

Ⅰ-5.4.2 객체 표시(Object Markings)

객체 표시는 데이터 표시를 전체 STIX 객체나 표시 정의 및 객체의 모든 콘텐츠에 적용된다. 객체 표시는 표시 정의(marking-definition) 객체에 대한 ID 목록(선택 사항)인 object_marking_refs 속성에 포함된 관계로 지정된다. 참조된 표시는 해당 STIX 객체나 표시 정의 및 해당 객체의 모든 콘텐츠에 적용된다.
object_marking_refs 속성에 대한 변경 내용(및 따라서 객체에 적용된 표시)은 객체의 다른 속성에 대한 변경 내용과 같이 처리되며 같은 버전 관리 규칙을 따른다.

Ⅰ-5.4.3 세분화된 표시(Granular markings)

객체 표시는 전체 STIX 객체나 표시 정의 및 해당 객체의 모든 속성에 적용되는 반면에, 세분화된 표시를 사용하면 데이터 표시를 STIX 객체와 표시 정의의 개별적인 부분에 적용할 수 있다. 세분화된 표시는 granular-marking 인스턴스의 목록인 granular_markings 속성에 지정된다. 이러한 인스턴스 각각은 표시되는 대상과 적용할 marking-definition 객체에 대한 참조를 나타내는 선택기의 목록을 포함하고 있다. 예를 들어 세분화된 표시를 사용하여 indicator의 name 속성이 TLP:GREEN으로, description 속성이 TLP:AMBER로, pattern 속성이 TLP:RED로 처리되어야 한다는 것을 나타낼 수 있다.

Ⅰ-5.5 번들

번들은 임의의 STIX 객체와 표시 정의(marking-definition) 객체를 단일 형식에 함께 모아 놓은 집합이다. 번들에 들어있는 데이터는 큰 의미가 없으며, 같은 번들 안에 있다는 이유로 객체라고 간주되지도 않는다. 번들은 STIX 객체가 아니므로 type과 id 속성 이외의 어떤 공통 속성도 갖지 않는다. 또한 일시적인 형식으로 구현에서는 다른 구현이 번들을 영구적 객체로 취급하거나 가정하지 않아야 한다. JSON MTI 직렬화는 번들

(bundle)을 표시할 때 JSON 객체 형식[RFC7159]을 사용한다. 또한, 번들은 STIX 객체가 아니며 STIX 객체에 대해 또는 STIX 객체로부터 관계를 가져서는 안 된다.

<표 I-5.5-1> 번들 속성

속성 이름	형식	설명
type(필수)	string	type 속성은 객체의 형식을 식별한다. 이 속성의 값은 bundle이어야 한다.
id(필수)	identifier	이 번들에 대한 ID이다. Bundle에 대한 id 속성은 처리를 위해 필요할 수 있지만 저장하거나 추적할 필요가 없는 도구를 돕도록 설계되었다. 도구 소비는 이 속성의 존재 또는 ID에 의해 번들을 참조하는 기능에 의존해서는 안 된다.
spec_version(필수)	string	이 번들의 콘텐츠를 표시하는 데 사용되는 STIX 표준의 버전. 이 속성을 사용하여 자체의 콘텐츠 식별 메커니즘이 없어도 TAXII가 아닌 전송 또는 다른 전송에서 STIX 콘텐츠의 버전을 알 수 있다. 이 표준에 정의된 STIX 객체가 포함된 번들의 경우 이 속성의 값은 2.0이어야 한다.
objects(선택 사항)	형식 <STIX Object> 또는 marking-definition의 list	STIX 객체 하나 이상의 집합을 정의한다. 이 목록의 객체는 STIX 객체(SDO, SRO 또는 사용자 지정 객체) 또는 표시 정의 객체이어야 한다.

Ⅰ-5.6 어휘

STIX 어휘는 STIX Objects 에 정의된 객체 설명 섹션에 참조된 각 어휘에 대한 객체별 형식을 제공한다. 모든 '-ov' 로 끝나는 형식 이름을 가진 STIX 어휘는 "개방형"이며, 보안 분야에서 수용하는 공통 용어의 목록을 사용자에 대한 가이드로 제공하되, 사용자에 대해서는 정의된 형식으로 제한하지 않는다.

Ⅰ-5.6.1 공격 동기(Attack Motivation)

공격 동기 어휘는 현재 다음과 같은 SDO에 사용된다.

- 침입 집단 (Intrusion Set)
- 위협 행위자 (Threat Actor)

분석자 또는 방어자는 위협 행위자 또는 침입 집단의 동기를 활용하여 유력한 목표와 동작을 더 잘 이해할 수 있다. 동기는 공격의 강도와 지속성을 형상화한다. 위협 행위자와 침입 집단은 대개 기반을 이루는 정서나 상황을 반영하는 방법으로 행동하며 이러한 행동이 방어자에게 공격의 방법을 알려 준다. 예를 들어 국수주의(이데올로기) 기반의 동기를 가진 스파이는 장기적인 목표를 달성하기 위해 여러 해 동안 조용히 작업할 인내력을 가지고 있을 가능성이 있는 반면에, 명예욕을 추구하는 사이버 범죄자는 강력하고 주의를 끌 수 있는 공격을 만들 수 있지만 머지않아 관심에서 멀어져 다른 곳으로 이동할 수 있다. 이러한 차이점을 이해하면 방어자가 각 공격 유형에 맞게 조정된 대응책을 구현하여 방어의 효율을 최대로 높일 수 있다.

Ⅰ-5.6.2 공격 리소스 레벨(Attack Resource Level)

공격 리소스 레벨 어휘는 공격 동기와 동일하게 다음과 같은 SDO에 사용된다.

● 침입 집단 (Intrusion Set)

● 위협 행위자 (Threat Actor)

공격 리소스 레벨은 위협 행위자, 침입 집단 또는 캠페인이 접근하고자 하는 일반적 레벨의 리소스를 수용하는 개방형 어휘이다. 단독으로 행동하는 개인에서 국가 정부의 자원인 정부에 이르기까지 광범위하다.

Ⅰ-5.6.3 해시 알고리즘 어휘(Hashing Algorithm Vocabulary)

해시 알고리즘의 개방형 어휘. 아직 hash-algorithm-ov 내에 정의되지 않은 해시 알고리즘을 지정하는 경우 해시 알고리즘 이름에 대한 권한 이름이 정의되는 곳마다 해당 이름을 값으로 사용해야 한다. 권한 이름이 존재하지 않거나 특정 해시 알고리즘의 네이밍에 변화성이 있는 경우 생산자가 최선의 결정을 내려야 한다.

Ⅰ-5.6.4 ID 클래스(Identity Class)

ID 클래스 어휘는 현재 다음과 같은 SDO에 사용된다.

● Identity

이 어휘는 ID가 표시하는 엔터티의 형식, 즉 조직, 그룹, 개인 또는 클래스 중 무엇에 대한 것인지를 설명한다.

Ⅰ-5.6.5 Indicator 레이블(Indicator Label)

Indicator 레이블 어휘는 현재 다음과 같은 SDO에 사용된다.

● Indicator

Indicator 레이블은 indicator를 분류하는 데 사용되는 개방형 어휘이다. 일관된 관례를 촉진하기 위한 고급 어휘이다. indicator 레이블은 관련 멀웨어 또는 공격 패턴 객체를 통해 더 잘 포착할 수 있는 정보를 포착하는 데 사용해서는 안 된다. 단순히 "poison-ivy"로 레이블을 표시하기보다는 Poison Ivy를 설명하는 멀웨어 객체에 indicator를 연결하는 것이 적절하다.

Ⅰ-5.6.6 산업 부문(Industry Sector)

산업 부문 어휘는 현재 다음과 같은 SDO에 사용된다.

● Identity

산업 부문은 산업 및 상업적 부문을 설명하는 개방형 어휘이다. 전체적인 목적을 지향하며 여러 다른 목록에서 파생되었고 "중요 인프라" 부문으로 제한되지 않는다.

Ⅰ-5.6.7 멀웨어 레이블(Malware Label)

멀웨어 레이블 어휘는 현재 다음과 같은 SDO에 사용된다.

● 멀웨어 (Malware)

멀웨어 레이블은 서로 다른 형식과 기능의 멀웨어를 나타내는 개방형 어휘이다. 멀웨어 레이블은 상호 배타적이지 않으며, 멀웨어 인스턴스가 스파이웨어 겸 화면 캡처 도구일 수 있다.

Ⅰ-5.6.8 보고서 레이블(Report Label)

보고서 레이블 어휘는 현재 다음과 같은 SDO에 사용된다.

● 보고서 (Report)

보고서 레이블은 보고서의 주 목적 또는 주제를 설명하는 개방형 어휘이다. 예를 들어 멀웨어와 해당 멀웨어에 대한 indicator 레이블을 포함하고 있는 보고서는 멀웨어가 주 목적임을 포착하기 위한 멀웨어 보고서 레이블을 가지고 있어야 한다. 보고서 레이블은 상호 배타적인 것이 아니다. 보고서는 멀웨어 보고서일 수도 있고 도구 보고서일 수도 있다. 단지 보고서가 어떤 형식의 객체를 포함하고 있다는 이유만으로 해당 보고서가 해당 레이블을 포함해야 한다는 것을 의미하지는 않는다. 객체가 단지 다른 객체에 대한 증거와 컨텍스트를 제공하기 위해 존재한다면 레이블에 포함시킬 필요는 없다.

Ⅰ-5.6.9 위협 행위자 레이블(Threat Actor Label)

위협 행위자 레이블 어휘는 현재 다음과 같은 SDO에 사용된다.

● 위협 행위자 (Threat Actor)

위협 행위자 레이블은 개인 또는 그룹이 어떤 종류의 위협 행위자인지 설명하는 데 사용되는 개방형 어휘이다. 예를 들어 어떤 위협 행위자와 경쟁자는 정보의 절취를 시도하는가 하면, 사회적 또는 정치적 원인을 지원하여 행동하는 활동가도 있다. 행위자 레이블은 상호 배타적인 것이 아니다. 즉, 위협 행위자는 악의를 가진 내부자일 수도 있고 스파이일 수도 있다.

Ⅰ-5.6.10 위협 행위자 역할(Threat Actor Role)

위협 행위자 역할 어휘는 현재 다음과 같은 SDO에 사용된다.

● 위협 행위자 (Threat Actor)

위협 행위자 역할은 위협 행위자가 할 수 있는 여러 가지 역할을 설명하는 데 사용되는 개방형 어휘이다. 예를 들어 일부 위협 행위자는 멀웨어를 작성하거나 봇넷을 운영하는가 하면, 다른 행위자는 실제로 공격을 직접 수행한다. 위협 행위자 역할은 상호 배타적인 것이 아니다. 예를 들어 행위자는 공격을 위한 재정 후원자일 수도 있고 직접 공격일 수도 있다.

Ⅰ-5.6.11 위협 행위자 정교화(Threat Actor Sophistication)

위협 행위자 정교화 어휘는 현재 다음과 같은 SDO에 사용된다.

● 위협 행위자 (Threat Actor)

위협 행위자 정교화 어휘는 위협 행위자의 숙련도 수준을 포착한다. 이는 완전 초보자를 설명하는 "없음"에서 취약점을 유입시키기 위해 공급망에 영향을 미칠 수 있는 공격자를 설명하는 "전략"까지 광범위하다. 이 어휘는 혁신적이고 고도로 숙련된 위협 행위자가 매

우 적은 리소스에 접근하면서도 최소 수준 행위자가 조직 범위의 리소스를 가질 수 있기 때문에 리소스 레벨과 구분된다.

Ⅰ-5.6.12 도구 레이블(Tool Label)

도구 레이블 어휘는 현재 다음과 같은 SDO에 사용된다.

● 도구 (Tool)

도구 레이블은 공격을 수행하는 데 사용할 수 있는 도구의 범주를 설명한다.

Ⅰ-5.7 STIX^{TM} 사용자 지정

STIX 사용자를 지정하는 두 가지 방식으로 사용자 지정 속성과 사용자 지정 객체가 있다. 사용자 지정 속성은 해당 형식에 정의되지 않은 속성을 기존 STIX 객체에 추가하기 위한 메커니즘과 요구사항을 제공한다. 반면, 사용자 지정 객체는 사용자 지정 STIX 객체(해당 형식에 정의되지 않은 객체)를 만드는 메커니즘과 요구사항을 제공한다.

자신이 이해하지 못하는 사용자 지정 속성 또는 객체를 포함하고 있는 STIX 문서를 수신하는 소비자는 문서 처리를 거부할 수 있으며, 또는 해당 속성이나 객체를 무시하고 문서 처리를 계속할 수 있다.

사용자 지정 속성 또는 객체를 포함하고 있는 STIX 문서의 생산자는 소비자가 이러한 문서를 이해하지 못하여 무시할 수 있다는 것을 인식해야 한다. 생산자는 사용자 지정 속성과 객체를 처리하는 규칙과 함께 자신이 사용하는 속성이나 객체를 정의해야 하며 이러한 정의와 규칙을 잠재적 소비자가 액세스할 수 있게 만들어 주어야 한다. 이 사양은 이를 수행하는 프로세스를 지정하지 않는다.

Ⅰ-5.7.1 사용자 지정 속성(Custom Properties)

이 문서에 지정되지도 않고 예약되지도 않은 속성을 추가하여 특정 정보 교환을 개선할 수 있는 경우가 있는데, 이러한 속성을 사용자 지정 속성이라 한다.

Ⅰ-5.7.2 사용자 지정 객체(Custom Objects)

이 문서에 지정되지도 않고 예약되지도 않은 객체를 추가하여 특정 정보 교환을 개선할 수 있는 경우가 있는데, 이러한 객체를 사용자 지정 객체라 한다.

Ⅰ-5.8 적합성

STIX 2.0 생산자는 STIX 2.0 콘텐츠를 만드는 소프트웨어이고, STIX 2.0 소비자는 STIX 2.0 콘텐츠를 소비하는 소프트웨어를 말하며 표준 요구사항을 준수해야 한다. 특히, STIX 2.0 생산자 또는 STIX 2.0 소비자는 버전 관리에 대한 표준 요구사항을 준수하고 지원해야 한다.

Ⅰ-5.8.1 생산자와 소비자(Producers and Consumers)

"STIX 2.0 생산자"는 STIX 2.0 콘텐츠를 만드는 소프트웨어이며 다음과 같은 표준 요구사항을 준수한다.

- JSON으로 인코딩된 콘텐츠를 만들 수 있어야 한다.
- STIX 객체 또는 형식의 속성 테이블에 필수로 표시된 모든 속성은 작성된 콘텐츠에 반드시 존재해야 한다.
- 모든 속성은 해당 속성에 대한 데이터 형식과 표준 요구사항을 준수해야 한다.
- STIX™ Version 2.0. Part 2: STIX Objects의 적합성 섹션에 따라 적어도 한 개의 STIX 객체를 지원해야 한다.
- 의무 기능에 열거한 모든 기능을 지원해야 한다.
- 선택 기능에 열거한 기능을 지원할 수 있다. 선택 기능을 지원하는 소프트웨어는 해당 기능에 대한 표준 요구사항을 준수해야 한다.
- JSON을 직렬화 형식으로 지원해야 하며 JSON 이외의 직렬화를 지원할 수 있다.

"STIX 2.0 소비자"는 STIX 2.0 콘텐츠를 소비하는 소프트웨어이며 다음과 같은 표준 요구사항을 준수한다.

- 자신이 소비하는 콘텐츠에 대한 모든 필수 속성에 대한 구문 분석을 지원해야 한다.
- 의무 기능에 열거한 모든 기능을 지원해야 한다.
- 선택 기능에 열거한 기능을 지원할 수 있다. 선택 기능을 지원하는 소프트웨어는 해당 기능에 대한 표준 요구사항을 준수해야 한다.
- JSON을 직렬화 형식으로 지원해야 하며 JSON 이외의 직렬화를 지원할 수 있다.

Ⅰ-5.8.2 의무 기능(Mandatory Features)

STIX 2.0 생산자 또는 STIX 2.0 소비자는 버전 관리를 지원해야 한다.

Ⅰ-5.8.3 선택 기능(Optional Features)

STIX 2.0 생산자 또는 STIX 2.0 소비자는 아래 사항을 지원할 수 있다.

- 객체 레벨 데이터 표시(Object-Level Data Markings)
- 세분화된 데이터 표시(Granular Data Markings)
- 사용자 지정 속성(Custom Properties)
- 사용자 지정 객체(Custom Objects)

부 록 Ⅰ-6

(본 부록은 표준을 보충하기 위한 내용으로 표준의 일부는 아님)

표준의 이력

판수	채택일	표준번호	내용	담당 위원회
제1판	2018.XX.XX	제정 TTAE.OT-xx.xxxx	-	사이버보안 프로젝트 그룹 (PG503), 정보보호 기술위원회(TC5)