Computer Security Symposium 2018
22 - 25 October 2018

脅威情報の共有に向けた グラフ記述のための軽量マークアップ言語の提案

山嵜 麿与[†1]

概要：構造化された脅威情報を共有するために，STIX や MISP 等の標準化形式が提案されている．しかし，これらの形式は機械間での共有を目的としているために人手による記述が難しく，実運用では構造化形式が広く用いられていない．本論文ではこの問題に対処するために，人間と機械の双方にとって読み書きしやすいグラフ記述のための軽量マークアップ言語を提案する．提案手法では，更新頻度が少ないグラフのスキーマを予め共有しておくことで，共有の度に異なるグラフ構造の情報を軽量マークアップ言語として記述できる．提案手法を用いることで，STIX 2.0 準拠の脅威情報を軽量マークアップ言語により記述可能であることを示す．また JSON 形式の STIX と比較して 2%，DOT 言語と比較して 19%の文字編集量で，脅威情報を記述可能であることを実験にて示す．

キーワード：脅威情報，グラフ記述言語，軽量マークアップ言語，知識表現

A Lightweight Markup Language for Graph Description towards Threat Information Sharing

Mayo YAMASAKI[†1]

Abstract: To share structured threat information, standardized formats such as STIX and MISP have been proposed. However, it is hard to describe them manually because these formats are designed to mainly share between machines. To tackle this problem, this paper proposes a lightweight markup language for graph description that is easy to read and write for both humans and machines. In the proposed method, by preliminarily sharing the schema of the graph with a low update frequency, graph-structured data for each document can be described as a lightweight markup language. I show that threat information compliant with STIX 2.0 can be described by using the proposed method. Further, I experimentally demonstrate that the proposed method can describe threat information with 2% editing cost compared to STIX in JSON and 19% cost compared with the DOT language.

Keywords: Threat Intelligence, Graph Description Language, Lightweight Markup Language, Knowledge Representation

1. はじめに

情報セキュリティコミュニティでは，脅威情報を共有する必要性が広く認知されており，STIX(Structured Threat Information eXpression)[1]や MISP(Malware Information Sharing Platform)[2]等の構造化形式が提案されている．しかしその一方で，60%の組織が標準的な構造化形式を用いておらず[3]，構造化形式が用いられている場合においても記述が複雑であるという問題が存在するために，検知指標(Indicator of Compromise)や観測情報(Observable)以外の情報が欠如している [4].

そこで本稿では，記述が複雑であるという問題に対処するために，構造化された脅威情報を効率的に作成可能な，グラフ記述のための軽量マークアップ言語を提案する．提案言語では，更新頻度が少ないグラフのスキーマを作成し予め共有しておくことで，共有の度に異なるグラフ構造の情報を軽量マークアップ言語として記述できる.スキーマは，適用領域に合わせて定義した頂点の種別と，それらの間に成り立つ辺である． Figure 1 に提案言語による脅威情報の記述例を示す．スキーマとして，マルウェアを意味する頂点の種別 *malware* と検知指標の IPv4 アドレスを意味する頂点の種別 *ioc-ipv4* とこれらの間に辺 *indicates* が定義されているとする．頂点は”[頂点名]{頂点の種別}”の様に記述し，空行で分けられた段落内の全ての頂点に，スキーマで定義されている全ての辺が成り立つことを仮定する．この仮定により，Figure 1 に示すグラフを決定的に抽出できる．

本研究の貢献は次の通りである．

- 人間と機械の双方にとって読み書きしやすい，グラフ記述のための軽量マークアップ言語を提案する．
- STIX 2.0 準拠の脅威情報を，提案言語を用いて記述できることを，スキーマと変換規則と共に示す．
- 構造化情報の作成コストを比較するために，評価実験を実施した．比較の結果，既存のグラフ記述言語 DOT[5]と比較して 19%，JSON[a]形式の STIX と比較して 2%の文字編集量で STIX 準拠の脅威情報を作成できることを示す．

†1 NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories

a http://www.json.org

原文書

EvilRatマルウェアは、収集した情報を192.168.0.0に送信している。
このIPアドレスは、EvilTrojanマルウェアのC2サーバとしても利用されていた。

また、Trojan.EvilRansomマルウェアは、192.168.0.1と通信している。

提案言語で記述した文書

[EvilRat]{malware}は、収集した情報を[192.168.0.0]{ioc-ipv4}に送信している。
このIPアドレスは、[EvilTrojan]{malware}のC2サーバとしても利用されていた。

また、[Trojan.EvilRansom]{malware}は、[192.168.0.1]{ioc-ipv4}と通信している。

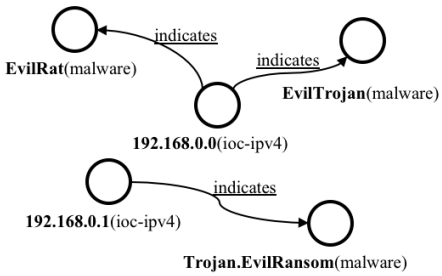

Figure 1 提案言語を用いて記述した脅威情報の例

2. 関係研究

既存のグラフ記述言語には DOT 言語や，XML を用いて記述する GXL[6]，GraphML[7]等がある．これらの言語用いた場合，人間が読み書きしやすい文書とは独立した方法で，文章から得られる構造化情報を記述する必要がある．Figure 1 と同じグラフ構造の情報を，DOT 言語を用いて記述した例を以下に示す．

```
digraph G {
   a [label = "malware:EvilRat"];
   b [label = "malware:EvilTrojan"]
   c [label = "malware:Trojan.EvilRansom"]
   d [label = "ioc-ipv4:192.168.0.0"];
   e [label = "ioc-ipv4:192.168.0.1"];
   a -> d [label = "indicates"];
   b -> d [label = "indicates"];
   c -> e [label = "indicates"];
}
```

また RDF(Resource Description Framework)[8]では，主語，述語，目的語のトリプルでグラフを表現する．具体的な記述形式には，XML や JSON，テキスト形式の N-Triples，読み書きのしやすさを考慮した Turtle 等がある．RDF においても，DOT 等のグラフ記述言語と同様の問題がある．

人間が読み書きしやすい軽量マークアップ言語としては，Markdown[9][10]や reStructuredText[11]等があり，HTML 等の構造化形式への変換に広く用いられている．これらの軽量マークアップ言語では，箇条書きや表などの簡易な構造を文書中に記述することはできるが，グラフ構造の情報を記述することができない．

本研究では，人間が読み書きしやすい軽量マークアップ言語として，文書中にグラフ構造の情報を記述する手法を提案する．

3. 提案言語

3.1 概要

提案言語では，文書中にグラフ構造の情報を記述するために，頂点に関する情報を”[頂点名]{頂点の種別}”の様に記述した文書と，頂点種別間に辺を定義したスキーマとを分離している．これに加えて，提案言語では，以下の 2 つの制限を設けている．

- **頂点種別間に成り立つ辺は高々一つである．**
- **スキーマで辺が定義されている場合，同一段落内にある全ての頂点について，その辺が成り立つ．**

例えば Figure 1 では，2 つの頂点種別(*malware*，*ioc-ipv4*)と 1 つの辺(*indicates*)が定義されたスキーマを用いて，文書を記述している．Table 1 に行から列への有向辺として，スキーマを示す．

Table 1　Figure 1 におけるグラフのスキーマ

	malware	*ioc-ipv4*
malware	-	-
ioc-ipv4	*indicates*	-

辺の情報を文書中に付与しないため，決定的にグラフ構造の情報を抽出するためには，**頂点の種別間に成り立つ辺は高々一つでなければならない**．従って，Table 1 のスキーマでは *ioc-ipv4* から *malware* への辺 *indicates* を定義しているため，これとは別に *malware* から *ioc-ipv4* への辺は定義できない．STIX 2.0 等のグラフ構造では，ある種別の頂点間に複数の辺が成り立つことがあるが，頂点の種別を分割することで，成り立つ辺を高々一つにすることができる．

また，一つの文書に複数のグラフ構造の情報を記述したい場合がある．このため，**スキーマで辺が定義されている場合は，同一段落内にある全ての頂点について，その辺が成り立つことを仮定している**．Figure 1 では，2 つの段落に文書を分けることで，2 つのグラフを表現している．段落が分けられていない場合，全ての *ioc-ipv4* と *malware* 間に関係 *indicates* が成立する．例えば，”192.168.0.1” から”EvilTrojan”への関係が成立する．

このように，提案言語を用いる場合，記述するグラフのスキーマと，段落の分け方について注意が必要である．但し，既存のグラフ記述言語や STIX を利用する場合においても，情報の構造や境界について注意が必要である．

3.2 文法

3.2.1 段落

段落は 1 つ以上の行であり，1 つ以上の空行で分けられる．空白文字またはタブ文字だけを含む行は空行である．

3.2.2 頂点

グラフの頂点は，頂点名と頂点情報から構成される．頂点は以下の規則を満たす．

- 頂点は，頂点名を囲む角括弧”[]”と，それに連続した頂点情報を囲む波括弧”{}”である．
- 頂点が入れ子に記述されている場合，最も内側の定義が使用される．
- 角括弧内では，バックスラッシュ文字でエスケープされているか，角括弧がペアで使われている場合にのみ，角括弧が使用できる．波括弧についても同様である．
- 頂点情報は頂点の種別または，区切り文字”,”で分けられた頂点の種別と頂点の属性である．また，区切り文字前後の空白文字とタブ文字は削除される．

以下に頂点の例を示す．

```
[name a]{type1}
[name b]{type 2, attr-name:attr-value}
[name[c]{type-3, , attr-1, attr-2}]{type-4}
```

一行目は，頂点名が"name a"で種別が"type1"，属性は省略されている．二行目は，頂点名が"name b"で種別が"type 2"，属性が"attr-name:attr-value"である．三行目は，頂点名が"c"で種別が”typel-3”，属性が" attr-1, attr-2"である．

3.2.3 辺

辺は，スキーマによる辺の指定がない場合は，同一段落にある全ての頂点間にただ一つだけ貼られる．

スキーマにより辺が 1 つ以上定義されている場合は，スキーマに定義されている辺に従い，同一段落にある全ての頂点間に辺が貼られる．

3.2.4 スキーマ

スキーマは，提案言語で記述された文書とは異なる方法で記述し共有する．スキーマでは，頂点の種別とそれらの間の辺を定義する．スキーマは以下の規則を満たす．

- 頂点の種別は文字列で定義される．
- 頂点の種別に波括弧”{}”を用いる場合，バックスラッシュ文字でエスケープされているか，波括弧がペアで使われている必要がある．
- 任意の頂点種別間に定義されている辺は，高々一つである．
- 辺には，辺の方向と辺の名称を表す文字列(ラベル)を付与してもよい．

4. 提案言語による STIX 2.0 の記述

4.1 STIX 2.0 の概要

STIX 2.0 は，脅威情報を記述するための言語であり，JSON 形式のデータとして記述できる．STIX では，SDO(STIX Domain Object)と呼ばれる頂点と，SRO(STIX Relationship Object)と呼ばれる辺を用いて情報を表現する．

SDO と SRO には複数の属性があり，属性名と属性値から成る．属性値には，JSON の値(数値，文字列，真偽値，配列，key-value 型のオブジェクト)が主に用いられるが，Indicator SDO の pattern 属性の様に，独自の文法で記述する属性値も存在する．また，必須の属性と必須でない属性が存在する．

STIX 2.0 を用いた Figure 1 の脅威情報の記述は冗長であるため，”EvilRat”マルウェアの記述例のみを以下に示す．

```
{
    "type": "malware",
    "id": "malware--395237ba-9175-47ed-… ",
    "created": "2018-08-16T01:00:17.017Z",
    "modified": "2018-08-16T01:00:17.017Z",
    "name": "evilrat",
    "labels": [
        "unknown"
    ]
}
```

4.2 STIX 2.0 のためのスキーマ

4.2.1 頂点の定義

頂点には，SDO の種別を用いる．頂点の一覧と，SDO に変換した際に用いる各属性の初期値を Table 2 に示す. Table 2 の”${NAME}”は頂点名，”${NOW}”は SDO に変換した時点の時刻である. また Table 2 の *1 で示す頂点の種別では，属性 first_observed，last_observed，number_observed の初期値として *observed-data* の行で示す初期値を用いる. *2 で示す属性 object_refs の初期値は, 同一段落内の全ての SDO を用いる．

頂点種別のいくつかは，SDO の種別と一対一に対応していない．SRO では，Threat Actor と Identity との間に，標的(targets)，装う対象(impersonates)，帰属(attributed-to)の 3 種類の関係が定義されているため，**頂点の種別間に成り立つ辺は高々一つでなければならない**という制約を満たすために，Identity を 3 つの頂点種別に分割している．

また Malware については，Malware から Malware 自身への関係 varient-of が SRO 内で唯一定義されている．同一段落に複数個の Malware を記述する場合に varient-of の関係は成り立たないことが多いため，段落を分割する必要があった．このため利便性を向上させるために，varient-of を表す頂点種別 *original-malware* を新たに追加した．

最後に，Indicator と Observed Data，Vulnerability には，記述規則が複雑で記述が煩雑になる属性が存在するため，糖衣構文に相当する頂点種別を追加した．例えば，*ioc-ipv4*, *ipv4*，*cve* 等を頂点種別として新たに追加した．

Table 2 STIX 記述用スキーマの頂点一覧

SDO	頂点の種別	SDO 属性の初期値
Attack Pattern	*attack-pattern*	name=${NAME}
Campaign	*campaign*	name=${NAME}
Course of Action	*coa*	name=${NAME}
Identity	*victim*	name=${NAME}, identity_class=” unknown”, labels=[“unknown”]
	criminal	name=${NAME}, identity_class=” unknown”, labels=[“unknown”]
	persona	name=${NAME}, identity_class=” unknown”, labels=[“unknown”]
Indicator	*ioc*	pattern=${NAME}, labels=[“unknown”]
	ioc-as-number	pattern="[autonomous-system:number=${NAME}]", labels=[“unknown”]
	ioc-directory	pattern="[directory:path=${NAME}]", labels=[“unknown”]
	ioc-domain-name	pattern="[domain-name:value=${NAME}]", labels=[“unknown”]
	ioc-email-addr	pattern="[email-addr:vlaue=${NAME}]", labels=[“unknown”]
	ioc-file-name	pattern="[file:name=${NAME}]", labels=[“unknown”]
	ioc-file-sha256	pattern="[file:hashes.'SHA-256'=${NAME}], labels=[“unknown”]
	ioc-file-md5	pattern="[file:hashes.'MD5'=${NAME}]", labels=[“unknown”]
	ioc-ipv4	pattern="[ipv4-addr:vlaue=${NAME}]", labels=[“unknown”]
	ioc-ipv6	pattern="[ipv6-addr:vlaue=${NAME}]” , labels=[“unknown”]
	ioc-mac-addr	pattern="[mac-addr:vlaue=${NAME}]", labels=[“unknown”]
	ioc-mutex	pattern="[mutex:name=${NAME}]", labels=[“unknown”]
	ioc-process-name	pattern="[process:name=${NAME}]", labels=[“unknown”]
	ioc-process-id	pattern="[process:pid=${NAME}]", labels=[“unknown”]
	ioc-url	pattern="[url:vlaue=${NAME}]", labels=[“unknown”]
	ioc-user-id	pattern="[user-account:user_id=${NAME}]", labels=[“unknown”]
	ioc-registry-key	pattern="[windows-registry-key:key=${NAME}]", labels=[“unknown”]
	ioc-x509-ca	pattern="[x509-certificate:issuer=${NAME}]", labels=[“unknown”]
	ioc-x509-serial	pattern="[x509-certificate:serial_number=${NAME}]", labels=[“unknown”]
Intrusion Set	*intrusion-set*	name=${NAME}
Malware	*malware*	name=${NAME}, labels=[“unknown”]
	original-malware	name=${NAME}, labels=[“unknown”]
Observed Data	*observed-data*	objects=${NAME}, first_observed=${NOW}, last_observed=${NOW}, number_observed=1
	as-number	objects={"0":{"type":"autonomous-system","number": "${NAME}"}}, *1
	directory	objects={"0":{"type":"directory","path": "${NAME}"}}, *1
	domain-name	objects={"0":{"type":"domain-name","value": "${NAME}"}}
	email-addr	objects={"0":{"type":"email-addr","value": "${NAME}"}}, *1
	file-name	objects={"0":{"type":"file","name": "${NAME}"}}, *1
	file-sha256	objects={"0":{"type":"file","hashes": { "SHA-256": "${NAME}"}}}, *1
	file-md5	objects={"0":{"type":"file","hashes": { "MD5": "${NAME}"}}}, *1
	ipv4	objects={"0":{"type":"ipv4-addr","value": "${NAME}"}}, *1
	ipv6	objects={"0":{"type":"ipv6-addr","value": "${NAME}"}}, *1
	mac-addr	objects={"0":{"type":"mac-addr","value": "${NAME}"}}, *1
	mutex	objects={"0":{"type":"mutex","name": "${NAME}"}}, *1
	process-name	objects={"0":{"type":"process","name": "${NAME}"}}, *1
	process-id	objects={"0":{"type":"process","id": "${NAME}"}}, *1
	url	objects={"0":{"type":"url","value": "${NAME}"}}, *1
	user-id	objects={"0":{"type":"user-account","user_id": "${NAME}"}}, *1
	registry-key	objects={"0":{"type":"windows-registry-key","key": "${NAME}"}}, *1
	x509-ca	objects={"0":{"type":"x509-certificate","issuer": "${NAME}"}}, *1
	x509-serial	objects={"0":{"type":"x509-certificate","serial_number": "${NAME}"}}, *1
Report	*report*	name=${NAME}, labels=[“unknown”], published=${NOW}, object_refs*2
Threat Actor	*threat-actor*	name=${NAME}, labels=[“unknown”]
Tool	*tool*	name=${NAME}, labels=[“unknown”]
Vulnerability	*vulnerability*	name=${NAME}
	cve	name=${NAME}, external_references=[{"source_name": "cve", "external_id": ${NAME}}]

4.2.2 辺の定義

Table 3 に頂点種別間の有向辺の定義を示す．Table 3 では始点と終点の全ての組に対して，辺を定義している．また SDO の種別を記載している場合は，その SDO 種別に対応する全ての頂点種別に対して辺を定義しているものとする．

Table 3　STIX 記述用スキーマの辺一覧

辺	始点の頂点種別または SDO	終点の頂点種別または SDO
targets	*attack-pattern*, *campaign*, *intrusion-set*, *threat-actor*, *malware*, *tool*	Vulnerability, *victim*
uses	*attack-pattern*	*malware*, *tool*
uses	*campaign*, *intrusion-set*, *threat-actor*	*attack-pattern*, *malware*, *tool*
uses	*malware*	*tool*
attributed-to	*campaign*	*intrusion-set*, *threat-actor*
attributed-to	*intrusion-set*	*threat-actor*
attributed-to	*threat-actor*	*criminal*
mitigates	*coa*	*attack-pattern*, *malware*, *tool*, Vulnerability
indicates	Indicator	*attack-pattern*, *campaign*, *intrusion-set*, *threat-actor*, *malware*, *tool*
variant-of	*malware*	*original-malware*
impersonates	*threat-actor*	*persona*
object-refs	*report*	*attack-pattern*, *campaign*, *coa*, *victim,* Indicator, *intrusion-set*, *malware*, Observed Data, report, *threat-actor*, *tool*, Vulnerability
with	*ioc-file-name*	*ioc-file-sha256, ioc-file-md5*
with	*file-name*	*file-sha256, file-md5*
with	*ioc-x509-ca*	*ioc-x509-serial*
with	*x509-ca*	*x509-serial*
with	*ioc-process-name*	*ioc-process-id*
with	*process-name*	*process-id*

SRO に定義されていない関係として，”object-refs”と“with”を新たに追加した．”object-refs”は Report の属性であり，値として SDO の ID の配列をとるため，SDO 間の関係を意味する．“with”は，Indicator と Observed Data の糖衣構文の内，組み合わせて用いる場合が多い頂点種別間に定義した．

4.3 STIX 2.0 への変換

提案言語により記述された文書と，Table 2 と 3 で示したスキーマから，グラフ構造の情報を抽出し STIX 2.0 への変換を行った．頂点属性の文字列は，トップレベルが”{}”を省略した key-value 型のオブジェクトで始まる flow style の YAML[b]で記述されているものとし，頂点属性を SDO の属性に変換する．

また，同一文書内に存在する頂点の内，頂点名と頂点種別の組が等しい頂点は同一の頂点とみなし結合した．

全ての頂点と関係を含む文書を記述し，STIX 2.0 への変換を行った．出力された STIX 情報を，STIX Validator[c]を用いて検証し，仕様に準拠していることを確認した．

5. 評価実験

提案言語が文書記述に与える影響の評価と，既存のグラフ記述言語との記述コストの比較を行うために，評価用のデータセットを構築した．

提案言語を意識せずに作成された，既存の脅威レポート 10 文書を収集し，頂点情報の付与を行った．収集した文書の平均文字数は 13,548 文字で，STIX 2.0 に変換した後の平均 SDO 数が 13.9 個，SRO 数が 14.5 個であった．

頂点情報の付与には，文書中に記述されているグラフをリアルタイムにフィードバックする GUI を用いた．Figure 2 に GUI の画面例を文書例と共に示す．

頂点情報を付与した脅威レポートの記述コストは，元の脅威レポートとのレーベンシュタイン距離を用いて算出した．レーベンシュタイン距離は，文字の挿入，削除，置換によって，一方の文字列をもう一方の文字列に変換する場合の最小手順数である．

また，段落構成の変更や文の追加・入れ替えなど，頂点情報の付与以外で生じた編集量を計測するために，頂点情報を除外した編集後の脅威レポートと元の脅威レポートとのレーベンシュタイン距離も計測した．

ベースラインとして，既存手法である DOT 言語，RDF の N-Triples 形式，RDF の Turtle 形式，JSON 形式の STIX，YAML 形式の STIX を用いた．いずれの手法でも，提案言語により作成したグラフ構造の脅威情報と同じ内容の情報を記述した．これらの手法では，文書とは独立した方法で記述する必要があるため，単純に文字数を編集量とした．各手法の平均編集量の比較を Table 4 に示す．

Table 4 の結果から，提案言語を用いることで，STIX(JSON)の 2%，DOT 言語の 19%の編集量でグラフ構造の脅威情報を記述できることがわかった．

Table 4　平均編集量の比較結果

	平均編集量	STIX(JSON)に対する平均編集量の割合[%]
STIX(JSON)	12593	-
STIX(YAML)	10033	80
RDF(N-Triples)	2227	18
RDF(Turtle)	1904	15
DOT 言語	1382	11
提案言語	**268**	**2**
提案言語(頂点を除く)	38	-

b http://yaml.org/spec/1.2/spec.html

c https://github.com/oasis-open/cti-stix-validator

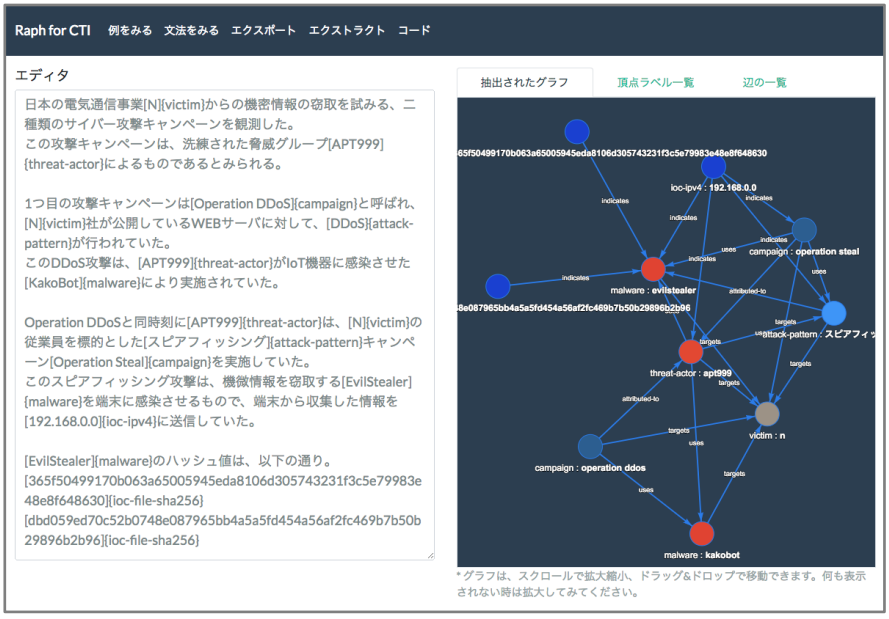

Figure 2 提案言語を用いて STIX 形式の脅威情報を作成するための GUI

6. 考察

比較実験の結果から，文書中にグラフ構造の情報を記述する提案言語は，既存手法と比べて少ない編集量で記述できることがわかった．同時に提案言語では，Figure 2 に示すように，可読性を大きく損なうことなくグラフ構造を記述できていることがわかる．また提案言語を用いた場合に発生した，頂点情報の付与を除く編集量は 38 に止まり，少ない編集量であると言える．これは，原文書が関係のある内容を同一段落に記述しているためであると考えられる．

一方で，提案言語で文書を記述する際には，予め共有されたスキーマに注意する必要がある．同一の頂点情報を含む文章でも，段落構成が変わることで，グラフ構造が変化するため，視覚的なフィードバック無しに，正確にグラフ構造を記述することは容易ではない．特に，スキーマが複雑になるグラフ構造を記述する場合には注意が必要になる．頂点種別と辺の数が多い場合，段落構成の変更により，大きくグラフ構造が変化してしまう．例えば，各頂点に階層的な属性がある STIX 2.0 を，単純な頂点種別と辺として表現した場合に，Table 2 や 3 に示す様に，膨大な頂点種別と辺の数となってしまう．

提案言語にはこのような問題が残されているが，STIX 2.0 の主要な属性を記述する範囲では，テキストファイルや電子メール等を用いて，容易に構造化された情報を共有できるという利点がある．

7. おわりに

本稿では，構造化された脅威情報の共有に向けて，グラフ記述のための軽量マークアップ言語を提案した．構造化情報の作成コストを比較する実験の結果，JSON 形式の STIX と比較して 2%の， DOT 言語と比較して 19%との文字編集量で記述可能であることがわかった．今後は，実運用を通じて，利便性の評価と課題の調査を進めていきたい．

参考文献

[1] “STIX - Structured Threat Information Expression”. https://stixproject.github.io/, (参照 2018-08-13).

[2] “MISP standards”. https://github.com/MISP/misp-rfc, (参照 2018-08-13).

[3] “The Value of Threat Intelligence: The Second Annual Study of North American & United Kingdom Companies”. https://anomali.cdn.rackfoundry.net/files/white-papers/2017-anomali-research-report.pdf, (参照 2018-08-13).

[4] “Exploring the opportunities and limitations of current Threat Intelligence Platforms”. https://www.enisa.europa.eu/publications/exploring-the-opportunities-and-limitations-of-current-threat-intelligence-platforms, (参照 2018-08-13).

[5] Gansner, Emden R., and Stephen C. North . An open graph visualization system and its applications to software engineering. Software: practice and experience. 2000, 30.11, p. 1203-1233.

[6] Holt, Richard C., Andreas Winter, and Andy Schurr. GXL: Toward a standard exchange format. Reverse Engineering, 2000. Proceedings. Seventh Working Conference on. IEEE. 2000, p. 162-171.

[7] Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., & Marshall, M. S. GraphML progress report structural layer proposal. International Symposium on Graph Drawing. Springer, Berlin, Heidelberg. 2001. p. 501-512.

[8] “RDF 1.1 Concepts and Abstract Syntax”. https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, (参照 2018-08-13).

[9] “Markdown” https://daringfireball.net/projects/markdown/, (参照 2018-08-13).

[10] “CommonMark”, https://commonmark.org/, (参照 2018-08-13).

[11] “reStructuredText Markup Specification”, http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html, (参照 2018-08-13). (参照 2018-08-13).