Supporting Situationally Aware Cybersecurity Systems

30th September 2015

Zareen Syed, Tim Finin, Ankur Padia and Lisa Mathews

University of Maryland Baltimore County
1000 Hilltop Circle, MD, USA 21250
zsyed@umbc.edu, finin@cs.umbc.edu, pankur1@umbc.edu,
math1@umbc.edu

Summary

In this report, we describe the Unified Cyber Security ontology (UCO) to support situational awareness in cyber security systems. The ontology is an effort to incorporate and integrate heterogeneous information available from different cyber security systems and most commonly used cyber security standards for information sharing and exchange. The ontology has also been mapped to a number of existing cyber security ontologies as well as concepts in the Linked Open Data cloud. Similar to DBpedia which serves as the core for Linked Open Data cloud, we envision UCO to serve as the core for the specialized cyber security Linked Open Data cloud which would evolve and grow with the passage of time with additional cybersecurity data sets as they become available. We also present a prototype system and concrete use-cases supported by the UCO ontology. To the best of our knowledge, this is the first cyber security ontology that has been mapped to general world ontologies to support broader and diverse security use-cases. We compare the resulting ontology with previous efforts, discuss its strengths and limitations, and describe potential future work directions.

1 Introduction

Cybersecurity data and information is generated by different tools, sensors and systems, expressed using different standards and formats, published by different sources and often scattered as isolated pieces of information. Furthermore, cybersecurity data is available in structured, semi-structured and unstructured forms from both, internal sources i.e. within the organization, and external sources i.e. outside the organization. Unifying such scattered information will help provide better visibility and situational awareness to cybersecurity analysts. Also, such unification can support deep investigation and help transition from reactive approach to a more proactive and eventually a predictive approach.

Semantic Web technologies provide a common framework that allows data to be shared and reused across application, enterprise as well as community boundaries. Semantic Web languages such as RDF and OWL enable representing meaning by representing things or concepts rather than strings of words. They provide rich constructs to represent information that is not only machine readable but also machine understandable, thus aiding in semantic integration and sharing of information from heterogeneous sources. OWL language has constructs for defining mappings between classes and instances which aides in linking internal information to external knowledge sources, thus making available a larger pool of knowledge and helping in providing a more complete picture and situational awareness.

Semantic Web technologies represent real world entities as concepts rather than strings, as strings are lexical and ambiguous. Concepts are associated with a globally unique identifier called "uri". For example, the string "Georgia" may refer to "Georgia state" in the United States or "Georgia country" (Figure 1). Furthermore, concepts can be associated with attributes and can have relations with other concepts. The attributes and relations build up context for the concept. For example "Georgia country" can have "longitude" and "latitude" as attributes, which provide information about its location on the map and its neighboring countries. Moreover, such information can be used to make inference. For example, If an incident originates from "Georgia country" and it's a neighbor of Russia, it may raise more alarms as most cybersecurity attacks have originated from Russia in the past (Figure 2). Furthermore, these relations can help in connecting the dots and relating incidents with other similar incidents providing more insight into the source and motivation for the attack.

Fig. 1. Things vs. Strings, "Strings" are ambiguous and can refer to different concepts in the real world. "Things" are precise and reference unique concepts using "uri" (Uniform Resource Identifier).

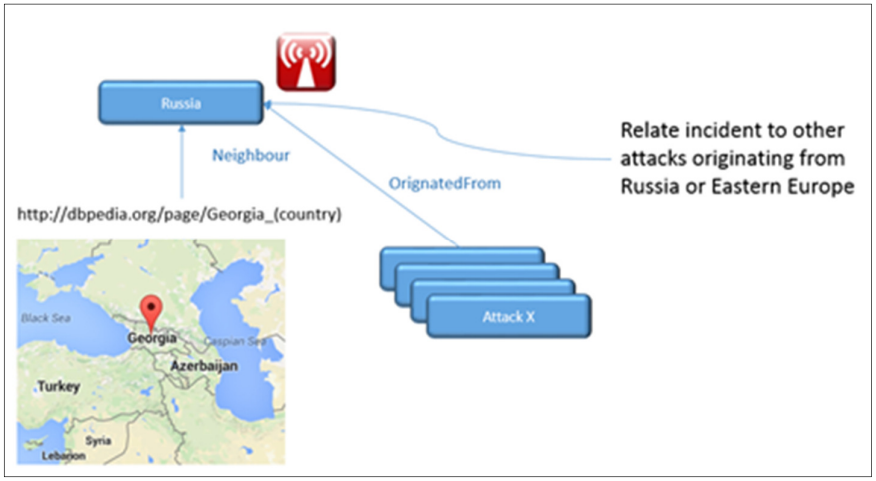

Fig. 2. Semantic Relations enable supporting complex security use-cases, for example, if "Georgia_(country)" has "neighbor" relation with "Russia" it may raise more alarms for a cybersecurity incident.

Semantic technologies are used and supported by Big Data companies like Google, Microsoft, Facebook and Apple for information sharing and interoperability and supporting high level functions like analyzing queries, providing semantic search and answering questions. In order to achieve situational awareness, cybersecurity systems need to transition to produce and consume semantic information about likely entities, relations, actions, events, intentions and plans. Through this seedling project we have developed Unified Cybersecurity Ontology (UCO) as an effort to help evolve the cybersecurity standards from a syntactic representation to a more semantic representation.

1.1 Contributions

We see several contributions that our work has to offer:

1. We have developed a comprehensive catalogue of cybersecurity standards that we surveyed and reviewed, the catalogue is included in Appendix A.
2. We have developed the UCO ontology which provides a common understanding of cybersecurity domain and unifies most commonly used cybersecurity standards.
3. Compared to existing cybersecurity ontologies which have been developed independently, UCO has also been mapped to a number of existing publicly available cybersecurity ontologies to promote ontology sharing, integration and re-use.
4. UCO is the first cybersecurity ontology to map concepts to general world knowledge sources i.e. Linked Open Data to support diverse use-cases.

5. We describe important use-cases that can be supported by unifying cybersecurity data and existing general world knowledge through the UCO ontology.

This report is organized as follows, in section 2 we outline our approach to ontology construction and describe the UCO ontology and other related ontologies. In section 3 we present the design and implementation of a demo system that uses the UCO ontology to support a number of use-cases with real world cybersecurity data. We review existing work in section 4 and conclude with future work directions in section 5.

2 Approach

Our approach to support cyber-situational awareness has been through the development of a core cybersecurity ontology that facilitates data sharing across different formats and standards and allows reasoning to infer new information. We have surveyed, reviewed and cataloged existing cybersecurity standards and ontologies and selected the most common and widely used standards to incorporate in the UCO ontology. In this section, we first briefly outline the advantages of using semantic web languages in more detail and describe the UCO ontology along with it's design considerations. We also describe the feasibility to support diverse and complex use-cases by linking cybersecurity information to external knowledge sources.

2.1 Advantages of RDF, RDFS and OWL

RDF is a directed graph and unambiguous compared to XML, which is tree based and has multiple representation for the same information. As RDF and OWL have formal semantics grounded in first order logic they are more preferable for dealing with security situations. RDF and OWL have a decentralized philosophy which allows incremental building of knowledge, and its sharing and reuse. For example, properties can be defined separately from classes (unlike OOP). OWL facilitates information integration by providing rich semantic constructs for schema mapping such as Sub Class, Sub Property, Equivalent Class, Equivalent Property, Same As, Union Of, Intersection Of etc. Furthermore, OWL has powerful reasoners, which enable detecting inconsistencies during data sharing. For example, if there is a constraint for two classes, "Malware" and "Virus", to be disjoint and the data sets imported from different sources mention the same software to be both a Malware and Virus, in such cases the reasoner will infer an inconsistency. In addition, with the support of off-the-shelf reasoners, like Fact++ new facts can be inferred from the given facts. There are powerful reasoners available both as Open Source Software and Commercial products.

2.2 Unified Cyber Security Ontology

The Unified Cybersecurity Ontology (UCO) is an extension to Intrusion Detection System ontology (IDS) [UN04b] developed earlier by our group to describe events related to cybersecurity. Our group has been working on a number of projects that focus on

individual components of a unified cybersecurity framework to analyze different data streams and assert facts in a triple store. The UCO ontology is essential for unifying information from heterogeneous sources and supporting reasoning and rule writing. The ontology supports reasoning and inferring new information from existing information. The ontology also supports capturing specialized knowledge of a cybersecurity analyst using ontology classes and terms as well as rules. Figure 3 shows a generic rule which uses terms from the ontology. The rule connects information which is external to the organization network with the evidences and network information available within the organization to alert the host.

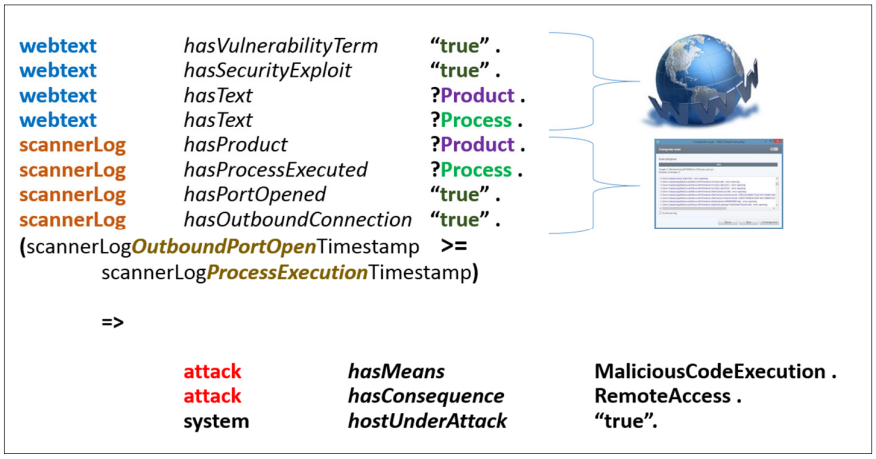

Fig. 3. The UCO ontology facilitates writing generic rules and combining evidence from multiple sources.

UCO ontology provides a common understanding of cybersecurity domain and maps to top level classes defined in STIX (Structured Threat Information eXpression) architecture and schema. STIX is the most comprehensive effort to unify cybersecurity information sharing and enables extensions by incorporating vocabulary from several other standards. However, in STIX the information is represented in XML and therefore cannot support reasoning which is supported by UCO. We have created Unified Cybersecurity Ontology as a semantic version of STIX. In addition to mapping to STIX, UCO has also been extended with a number of relevant cybersecurity standards, vocabularies and ontologies such as CVE, CCE, CVSS, CAPEC, CYBOX, KillChain and STUCCO. To support diverse use-cases, UCO ontology has been mapped to general world knowledge available through Google's knowledge graph, DBpedia knowledge base, Yago knowledge base etc. Linking to these knowledge sources provides access to large number of datasets for different domains (for e.g. geonames) as well as terms in different languages (e.g Russian).

Below we describe the list of important classes present in UCO ontology:

1. Means

The 'means' class describes various methods of executing an attack. For instance, the 'means' class consists of sub-classes like 'BufferOverFlow', 'synFlood', 'LogicExploit', 'tcpPortScan', etc., which can further consist of their own sub-classes.

2. Consequences

The 'consequences' class describes the possible outcomes of an attack. The 'consequences' class consists of sub-classes like 'DenialOfService', 'LossOfConfiguration', 'PrivilegeEscalation', 'UnauthUser', etc.

3. Address

The 'address' class represents the address of the machine expressed using IPV4 or IPV6.

4. AttackPattern

Attack Patterns are descriptions of common methods for exploiting software providing the attacker's perspective and guidance on ways to mitigate their effect. An example of attack pattern is "Phishing".

5. CCE

The CCE class is a top level class mapped to CCE ontology that we developed independently. The CCE ontology represents the Common Configuration Enumeration standard which assigns unique entries (called CCEs) to common system configuration issues.

6. CVE

The CVE class is a top level class mapped to CVE ontology that we developed independently. CVE is a dictionary of publicly known information on security vulnerabilities and exposures. CVE's common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services.

7. CVSS

The CVSS class is a top level class mapped to CVSS ontology that we developed independently. The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities.

8. Weakness

The Weakness class maps to CWE ontology that we developed independently. The Common Weakness Enumeration (CWE) is a formal list of software weakness types created to serve as a common language for describing software security weaknesses in architecture, design, or code.

9. Exploit

This class characterizes description of an individual exploit and maps to 'ExploitType' in STIX schema.

10. Exploit Target

Exploit Targets are vulnerabilities or weaknesses in software, systems, networks or configurations that are targeted for exploitation by the TTP (cyber threat adversary Tactic, Technique or Procedure).

11. File

The File class represents a file and it's properties on a computer system.

12. Hardware

Represents hardware component of the computer.

13. Indicator

A cyber threat indicator is made up of a pattern identifying certain observable conditions as well as contextual information about the patterns meaning, how and when it should be acted on, etc. This class is mapped to "IndicatorType" in STIX schema and "Indicator" class in CAPEC ontology.

14. Kill Chain

This class is mapped to the top class of Kill chain ontology. Network intrusions can be seen as a series of actions taken in sequence, each relying on the success of the last. Stages of the intrusion progress linearly - starting with initial reconaissance and ending in compromise of sensitive data. These concepts are useful in coordinating defensive measures and defining the behavior of malicious actors. For instance, the behavior of a financially motivated intruder may appear similar to an espionage-motivated one until the final stage where they execute actions to steal their preferred type of information from the target. This concept is often called a "kill chain" or a "cyber attack lifecycle".

15. Malware

This class represents Malware instances.

16. Network state

This class is used to capture network related properties.

17. OSVDB

This class represents instances of open source vulnerability database. OSVDB is an independent and open-sourced database with information on security vulnerabilities.

18. Process

This class represents computer processes and associated properties such as code size, open ports, open files, child processes etc.

19. Source

This class represents information source like domain expert, IDPS or Web.

20. Attacker

This class represents identification or characterization of the adversary and is mapped to "ThreatActor" in STIX.

21. Means

The Means class maps to TTP in STIX. The TTP field characterizes specific details of observed or potential attacker Tactics, Techniques and Procedures.

22. KillChain Phase

This class maps to "KillChainPhase" in STIX and characterizes an individual phase within a kill chain definition.

23. Attack

This class Identifies or characterizes a single cyber threat attack and is mapped to "Incident" in STIX.

Table 1. Statistics for UCO and related ontologies created in this seedling project.

	CCE	**CVE**	**CVSS**	**UCO**	**Total**
Axiom	11	21	197	633	862
Class Count	1	3	35	106	145
Object Property Count	0	2	32	59	93
Data Property Count	5	6	3	45	59
Individual Count	0	0	23	7	30
Equivalent classes	0	0	3	16	19

DL Expressivity	AL	ALUHO (D)	ALUHOQ (D)	ALCROIQ (D)	

Table 2. Statistics for existing Cybersecurity Ontologies that have been mapped to UCO.

	Capec	**Cybox**	**Cybox Common**	**Data Marking**	**Kill Chain**	**MAEC**	**STIX**
Axiom	7915	296	117	2	63	2	8808
Class Count	1219	21	10	1	12	1	1303
Object Property Count	6	22	5	0	5	0	114
Data Property Count	3	19	13	0	0	0	47
Individual Count	10	70	25	0	0	0	91
Equivalent Class	2	4	2	0	26	0	17
DL Expressivity	AL CHO (D)	AL UOQ (D)	AL UOQ (D)	AL	SIQ	AL	SHOIQ(D)

The statistics for UCO and related ontologies that have been created as a part of this seedling project are given in Table 1. Statistics related to other existing cybersecurity ontologies to which UCO has been mapped, are given in Table 2. These ontologies are independent and do not have any overlapping classes. However, to generate a connected graph, the UCO ontology has a few classes representing parent class of each of the other ontology. Such a design allows easy maintenance of the ontology. As different ontologies are loosely coupled each of the ontology can evolve independently. As shown in table 1, CCE contains a class and 5 data type properties like description, references, platform etc. CVSS, ontology contained 35 classes and includes classes like base group, environmental group and temporal group, which are represented as the combination of other classes. Such an increase in number of class was possible as the XML

schema was rich in semantics. Some of the individuals of CVSS were associated with more than one class. For example, “High” and “Low”, individuals were assigned to “Modified Attack Complexity” and “Attack Complexity” making it count 4 individual assertions instead of 2. As compared to other ontologies, we designed UCO to considerably extend STIX framework with additional classes and defined relations among them. For example “IPAddress”, “Software”, “WebBrowser” are a few classes with “WebBrowser” being the subclass of “Software”. Moreover, there are 16 classes in the ontology which are defined as the combination of other classes. For example, “Product” is represented as the union of “Software” and “Hardware”.

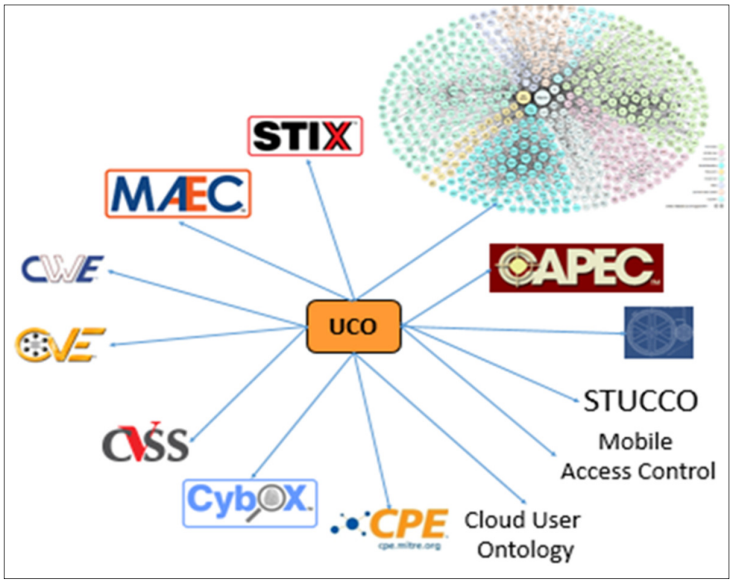

Fig. 4. UCO ontology serves as the core for Cybersecurity Linked Open Data Cloud.

Figure 4 shows UCO ontology serving as the core ontology for linking with other cybersecurity ontologies and LOD cloud. To facilitate data integration from multiple freely available knowledge base, we mapped UCO to Linked Open Data (LOD) cloud. Such an extension allows a client to fetch data from multiple freely available data sources with different schema but represented using semantic web technologies. An example of such a mapping is shown below:

uco:acrobat_reader owl:sameAs dbr:Adobe_Acrobat

Here, “uco:” is the namespace used for Unified Cybersecurity Ontology and the mapping asserts the Adobe Acrobat from DBpedia and the acrobat reader present in our ontology to be same.

3 Demo and Use-Cases

To demonstrate the benefits and effectiveness of the UCO ontology we have designed and implemented a number of use-cases with real world cybersecurity data. Below we describe our prototype system design and present a number of implemented use-cases.

3.1 Prototype System Design

To perform experiments we used STUCCO **extractors**[1] to extract entities from the NVD XML file. We developed our code to combine the extracted terms and associate each term with corresponding object to generate <subject, predicate, object> tuples. We support federated queries permitting data integration from multiple sources like DBpedia and Yago. We defined necessary mappings from terms in NVD data to DBpedia and our unified ontology. We enabled the reasoner of Fuseki server, to support reasoning.

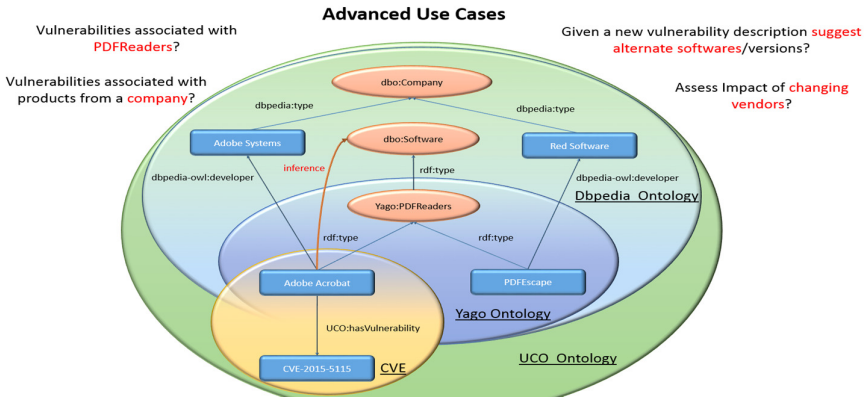

Fig. 5. Advanced use-cases that can be supported using mappings between UCO and general world ontologies

3.2 Unified Cybersecurity Ontology Use-Cases

In figure 5 we show several advanced use-cases that can be supported using mappings between UCO and general world ontologies that cannot be supported by individual ontologies alone. Below we describe each individual use-cases and present corresponding SPARQL queries. We also show a snippet of results obtained by executing the same query over sample NVD data set that was loaded into fuseki triple store.

Use-Case #1: Vulnerabilities associated with PDF Readers

[1] We are thankful to the authors for sharing useable code on github https://github.com/stucco/

An organization or a security analyst may be interested in finding out what kind of vulnerabilities may be associated with a specific type of software. The CVE entries only mention software, however if these software are linked to external knowledge sources such as Google's knowledge graph or DBpedia, one can also retrieve the associated type of software. For example, from CVE we have the information that "Adobe Acrobat" has a certain vulnerability identified with CVE entry reference "CVE-2015-5115". Mapping "Adobe Acrobat" instance to the corresponding instance in DBpedia resource will provide additional information that it is a type of "Yago:PDFReaders". This mapping with enable answering queries asking for vulnerabilities associated with specific category of software such as PDF readers and can also suggest alternate software of the same type that do not have a specific vulnerability. The following SPARQL query demonstrates this use-case.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago/>

prefix uco: <http://ffrdc.ebiquity.umbc.edu/ns/ontology/>

select distinct ?oberservable ?reader {
  SERVICE <http://dbpedia.org/sparql> {
    ?reader rdf:type yago:PDFReaders .
  }
  ?vulnerability uco:hasObservable ?observable .
  ?oberservable uco:hasSoftware ?reader .
}
order by ?reader
```

vulnerability	reader
uco:CVE-2002-2435	<http://dbpedia.org/resource/Adobe_Acrobat>
uco:CVE-2002-2436	<http://dbpedia.org/resource/Adobe_Acrobat>
uco:CVE-2002-2437	<http://dbpedia.org/resource/Adobe_Acrobat>

Use-Case #2: Vulnerabilities associated with products from a given company

Another interesting use-case is to explore vulnerabilities associated with products from a given company. Again by mapping software instances to external knowledge sources, one can find the name of the company which developed the software. The following SPARQL query retrieves vulnerabilities for products along with information about the source company.

```
prefix dbp: <http://dbpedia.org/property/>
prefix uco:  <http://ffrdc.ebiquity.umbc.edu/ns/ontology/>

select distinct ?product ?company ?vulnerability
where {
    SERVICE <http://dbpedia.org/sparql> {
        ?product dbp:developer ?company
    }
    ?vulnerability uco:hasObservable ?observable .
    ?blankNode uco:hasSoftware ?product .
}
order by ?product
```

product	company	vulnerability
<http://dbpedia.org/resource/Adobe_Acrobat>	<http://dbpedia.org/resource/Adobe_Systems>	uco:CVE-2002-2435
<http://dbpedia.org/resource/Adobe_Acrobat>	<http://dbpedia.org/resource/Adobe_Systems>	uco:CVE-2002-2436
<http://dbpedia.org/resource/Adobe_Acrobat>	<http://dbpedia.org/resource/Adobe_Systems>	uco:CVE-2002-2437

Use-Case #3: Suggest similar software to given software

After knowing information about a certain vulnerability a security analyst may be interested in finding alternate software that doesn't have the given vulnerability. For example in case of a PDF readers the following SPARQL query retrieves software of the same type filtering out "acrobat" software.

```
prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX yago: <http://dbpedia.org/class/yago/>
prefix uco:  <http://ffrdc.ebiquity.umbc.edu/ns/ontology/>

select ?similarSoftware ?product where {

    SERVICE <http://dbpedia.org/sparql> {
        ?similarSoftware rdf:type yago:PDFReaders .
    }

FILTER (!regex(str(?similarSoftware), 'acrobat', 'i'))
}
```

similarSoftware
<http://dbpedia.org/resource/STDU_Viewer>
<http://dbpedia.org/resource/Preview_(Mac_OS)>
<http://dbpedia.org/resource/Pdfescape>
<http://dbpedia.org/resource/Adobe_Digital_Editions>
<http://dbpedia.org/resource/Adobe_Creative_Suite>

Use-Case #4: Assess impact of changing vendors

In case an organization is interested in changing venders, they can assess the impact by using the following SPARQL query which creates a summary of vulnerability counts associated with products from different vendors.

```
prefix dbp: <http://dbpedia.org/property/>
prefix uco:  <http://ffrdc.ebiquity.umbc.edu/ns/ontology/>

select ?company (count(?vulnerability) as ?vulnerability_count) where {
    SERVICE <http://dbpedia.org/sparql> {
        ?software dbp:developer ?company
    }

    ?vulnerability uco:hasObservable ?observable .
    ?observable uco:hasSoftware ?software .
}
group by ?company
```

company	vulnerability_count
<http://dbpedia.org/resource/Opera_Software>	864
<http://dbpedia.org/resource/Adobe_Systems>	74

4 Related Work

One of the earliest efforts for developing ontologies for cybersecurity was by our group in 2003 by Undercoffer et al. They implemented a target centric ontology for the domain of intrusion detection composed of 23 classes and 190 properties and attributes. More et al. (2012) from our group extended this IDS ontology to incorporate and integrate cybersecurity related information from heterogenous sources. The UCO ontology is a further extension and enhancement of the IDS ontology to represent and map different publicly available standards and ontologies in the cybersecurity domain. STIX is the most comprehensive standard to unify cybersecurity information sharing and enables extensions by incorporating vocabulary from several other standards. Ulicny et al.

(2014) created a STIX ontology based on the STIX schema along with a number of related ontologies. We have defined mappings between UCO and STIX ontology classes. Iannacone et al. (2015) developed STUCCO ontology for integrating different structured and unstructured data sources along with data extractors. The ontology is composed of 15 entity types and 115 properties and is defined using JSON-schema. We translated STUCCO from JSON to OWL in order to map it to UCO ontology. We came across a significant number of papers on using ontologies for cyber security, however, the ontology or the details were not discussed in sufficient depth to enable reuse.

5 Conclusions and Future Work Directions

The UCO ontology provides a common understanding of cybersecurity domain and unifies most commonly used cybersecurity standards. Unlike existing independent and isolated cybersecurity ontologies, UCO has been mapped to publicly available ontologies in the cybersecurity domain and hence offers more coverage. In addition to that, UCO is also mapped to concepts in general world knowledge sources to support diverse use-cases. To the best of our knowledge this is the first such effort in the area of cybersecurity ontologies to unify cybersecurity information with general world knowledge about entities and relations. The different use-cases discussed demonstrate the utility and value of the UCO ontology in supporting diverse security scenarios. We briefly discuss promising future work directions below.

5.1 Temporal Representation and Reasoning

Cybersecurity data and information may have a temporal component for example timestamps associated with files, system logs and network events etc. The current version of UCO ontology uses a very basic representation of time and is represented as a data property associated with classes that represent events. In the future, we plan to represent time instances and intervals. A number of frameworks and representations have been proposed in research such as OWL-Time [HO04] and time-entry [PA04] which provide vocabularies for stating facts about temporal instants and intervals. Future work directions include reviewing different approaches, identifying and analyzing shortcomings and encountered challenges followed by the choice of suitable representation to extend UCO ontology.

5.2 Modeling Uncertainty and Confidence

While ontologies are widely used to capture the knowledge about concepts and their relations defined in a domain for information exchange and knowledge sharing requires crisp logic i.e. any sentences in these languages, being asserted facts, domain knowledge, or reasoning results, must be either true or false and nothing in between. However, most of the real world domains contain uncertain knowledge because of incomplete or partial information that is true only to a certain degree. Probability theory is a natural choice for dealing with this kind of uncertainty. Incorporating probability theory into existing ontology languages will strengthen these languages with additional

expressive power to quantify the degree of overlap or inclusion between concepts, to support probabilistic queries such as finding the most similar concept that a given description belongs to, and to make more accurate semantic integration possible.

5.3 Cybersecurity Information Extraction from Unstructured Data

Cybersecurity vulnerabilities are typically identified and published publicly but response has always been slow in covering up these vulnerabilities because there is no automatic mechanism to understand and process this unstructured text that is published on internet. There is a strong need for systems that can automatically analyze unstructured text and extract vulnerability entities and concepts from various non-traditional unstructured data sources such as Cybersecurity blogs, security bulletins and hackers forums. This information extraction task will help expediting the process of understanding and realizing the vulnerabilities and thus making systems secure at faster rate. We have developed a number of preliminary prototype systems for cybersecurity information extraction in our lab that can be further refined and extended to support situational awareness.

6 References

[HA13a] Lushan Han, Tim Finin, Paul McNamee, Anupam Joshi, and Yelena Yesha, Improving Word Similarity by Augmenting PMI with Estimates of Word Polysemy, IEEE Transactions on Knowledge and Data Engineering, IEEE Computer Society, v25n6, pp. 1307-1322, 2013.

[HA15] Lushan Han, Tim Finin, Anupam Joshi and Doreen Cheng, Querying RDF Data with Text Annotated Graphs, 27th Int. Conf. on Scientific and Statistical Database Management, San Diego, June 2015.

[JO13] Arnav Joshi, Ravendar Lal, Tim Finin and Anupam Joshi, Extracting cybersecurity related linked data from text, 7th IEEE Int. Conf. on Semantic Computing, September 2013.

[LA13] Ravendar Lal, "Information Extraction of Security related entities and concepts from unstructured text.", M.S. Thesis, University of Maryland Baltimore County, May 2013.

[LI15] Wenjia Li, Anupam Joshi and Tim Finin, SVM-CASE: An SVM-based Context Aware Security Framework for Vehicular Ad-hoc Networks, IEEE 82nd Vehicular Technology Conf., Boston, Sept. 2015.

[MA12] M. Lisa Mathews, Paul Halvorsen, Anupam Joshi and Tim Finin, A Collaborative Approach to Situational Awareness for CyberSecurity, 8th IEEE Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing, Pittsburgh PA, 14-17 Oct 2012.

[MA14] James Mayfield, Paul McNamee, Craig Harman, Tim Finin and Dawn Lawrie, Kelvin: Extracting Knowledge from Large Text Collections, AAAI Fall Symposium on Natural Language Access to Big Data, Nov. 2014.

[MO12a] Sumit More, Mary Mathews, Anupam Joshi, and Tim Finin, A Semantic Approach to Situational Awareness for Intrusion Detection, National Symposium on Moving Target Research, June 2012.

[MO12b] Sumit More, Mary Mathews, Anupam Joshi and Tim Finin, A Knowledge-Based Approach To Intrusion Detection Modeling, IEEE Workshop on Semantic Computing and Security, pp. 75-81, IEEE Computer Society, May 2012.

[MU11] Varish Mulwad, Wenjia Li, Anupam Joshi, Tim Finin, and Krishnamurthy Viswanathan, Extracting Information about Security Vulnerabilities from Web Text, Web Intelligence for Information Security Workshop, IEEE Computer Society Press, August 2011,

[MU13] Varish Mulwad, Tim Finin and Anupam Joshi, Semantic Message Passing for Generating Linked Data from Tables, 12th Int. Semantic Web Conf., Sydney, Oct. 2013.

[SH13] Puneet Sharma, Anupam Joshi and Tim Finin, Detecting Data Exfiltration by Integrating Information Across Layers, IEEE 14th Int. Conf. on Information Reuse and Integration, 2013.

[SL14] Jennifer Sleeman and Tim Finin, Taming Wild Big Data, AAAI Fall Symposium on Natural Language Access to Big Data, Nov. 2014.

[SL15] Jennifer Sleeman and Tim Finin and Anupam Joshi, Entity Type Recognition for Heterogeneous Semantic Graphs, AI Magazine, v36n1, pp. 75-86, April 2015, AAAI Press.

[ST12] Veselin Stoyanov, James Mayfield, Tan Xu, Douglas W. Oard, Dawn Lawrie, Tim Oates and Tim Finin, A Context-Aware Approach to Entity Linking, Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction, NAACL-HLT, June 2012.

[SY15] Zareen Syed, Lushan Han, Muhammad Rahman, Tim Finin, James Kukla, Jeehye Yun, UMBC_Ebiquity-SFQ: Schema Free Querying System, Schema-agnostic Queries Semantic Web Challenge, 12th Extended Semantic Web Conference, Portoroz Slovenia, June 2015. Note: Received 2015 ESWC Schema Agnostic Query Challenge Award.

[UN03] Jeffrey Undercoffer, John Pinkston, Anupam Joshi and Timothy Finin, A Target-Centric Ontology for Intrusion Detection, IJCAI Workshop on Ontologies and Distributed Systems, Aug. 2003, Acapulco MX.

[UN04a] J. Undercofer, Intrusion Detection: Modeling System State to Detect and Classify Aberrant Behavior, Ph.D. dissertation, Computer Science and Electrical Engineering, University of Maryland, Baltimore County, February 2004.

[UN04b] Jeffery Undercoffer, Anupam Joshi, Tim Finin and John Pinkston, A Target Centric Ontology for Intrusion Detection: Using DAML+OIL to Classify Intrusive Behaviors,

Knowledge Engineering Review, Special Issue on Ontologies for Distributed Systems, 2004.

[WO15] Travis Wolfe, Mark Dredze, James Mayfield, Paul McNamee, Craig Harman, Tim Finin and Benjamin Van Durme, Interactive Knowledge Base Population, arXiv:1506.00301, 2015

7 Appendix - A

Catalogue of Cybersecurity Standards

1. High level descriptions and frameworks

1.1 Structured Threat Information eXpression (STIX)

1.2 Open Vulnerability and Assessment Language (OVAL)

1.3 The Vocabulary for Event Recording and Incident Sharing (VERIS)

1.4 The Incident Object Description Format (IODEF)

2. Actionable observables

2.1 Cyber Observables eXpression (CybOX)

2.2 Mandiant's Open Indicators of Compromise (OpenIOC)

2.3 Malware Attribute Enumeration and Characterization (MAEC)

3. Enumerations

3.1 Common Attack Pattern Enumeration and Classification (CAPEC)

3.2 Common Vulnerabilities and Exposures (CVE)

3.3 Common Weakness Enumeration (CWE)

3.4 The Common Configuration Enumeration (CCE)

3.5 Common Platform Enumeration (CPE)

4. Scoring and measurement frameworks

4.1 Common Vulnerability Scoring System (CVSS)

4.2 Common Weakness Scoring System (CWSS)

4.3 The Extensible Configuration Checklist Description Format (XCCDF)

4.4 Common Configuration Scoring Scheme (CCSS)

5. Process frameworks

5.1 The Security Content Automation Protocol (SCAP)

5.2 Cybersecurity Information Exchange, Recommendation ITU-T X.1500 (CYBEX)

6. Transport

6.1 Trusted Automated eXchange of Indicator Information (TAXII)

6.2 The OASIS Customer Information Quality (CIQ)

Catalog of Cybersecurity Standards

1. High level descriptions and frameworks

These standards combine multiple types of information for e.g. including indicators, affected assets, actions that were taken, and other contextual information.

1.1 Structured Threat Information eXpression (STIX)

STIX™ is a collaborative community-driven effort to define and develop a standardized language to represent structured cyber threat information. The STIX Language intends to convey the full range of potential cyber threat information and strives to be fully expressive, flexible, extensible, automatable, and as human-readable as possible. STIX provides a unifying architecture tying together a diverse set of cyber threat information including:

1. Cyber Observables
2. Indicators
3. Incidents
4. Adversary Tactics, Techniques, and Procedures (including attack patterns, malware, exploits, kill chains, tools, infrastructure, victim targeting, etc.)
5. Exploit Targets (e.g., vulnerabilities, weaknesses or configurations)
6. Courses of Action (e.g., incident response or vulnerability/weakness remedies or mitigations)
7. Cyber Attack Campaigns
8. Cyber Threat Actors

Overseeing Organization: STIX is being transitioned from MITRE and DHS to OASIS.

1.2 Open Vulnerability and Assessment Language (OVAL)

OVAL is an information security community effort to standardize how to assess and report upon the machine state of computer systems. OVAL includes a language to encode system details, and an assortment of content repositories held throughout the community. Tools and services that use OVAL for the three steps of system assessment — representing system information, expressing specific machine states, and reporting the results of an assessment — provide enterprises with accurate, consistent, and actionable information so they may improve their security. Use of OVAL also provides for reliable and reproducible information assurance metrics and enables interoperability and automation among security tools and services.

Overseeing Organization: OVAL has been transitioned from MITRE to Center for Internet Security (CIS).

1.3 The Vocabulary for Event Recording and Incident Sharing (VERIS)

The Vocabulary for Event Recording and Incident Sharing (VERIS) is a metrics framework designed to provide a common language for describing security incidents and their effects in a structured manner. The difference between STIX incidents and VERIS is in purpose and use: VERIS is an after-the-fact characterization of cyber incidents intended for post-incident strategic trend analysis and risk management. STIX provides the capability to capture information about security incidents and their effects but does so in the context of a broader threat intelligence framework.

Overseeing Organization: Verizon

1.4 The Incident Object Description Format (IODEF)

The Incident Object Description Format (IODEF) is an Internet Engineering Task Force (IETF) standard developed for exchange of incident information. There is no formal relationship between STIX and IODEF, although it is possible to leverage IODEF within STIX in order to represent incident information. Doing so, however, would lose the richness and architectural alignment provided by the STIX Incident structure.

Overseeing Organization: IETF, Managed Incident Lightweight Exchange (MILE) working group

2. Actionable observables

Standards for cyber observables represent information used to detect attacks or malicious activity (such as system libraries used by a malware). A cyber observable is a measurable event or stateful property in the cyber context. Examples of measurable events include registry key creation, file deletion, and the sending of an HTTP GET request; examples of stateful properties include the MD5 hash of a file, the value of a registry key, and the name of a process.

2.1 Cyber Observables eXpression (CybOX)

The Cyber Observable eXpression is a standardized schema for the specification, capture, characterization and communication of events or stateful properties that are observable in the operational domain. A wide variety of high-level cyber security use cases rely on such information including: event management/logging, malware characterization, intrusion detection, incident response/management, attack pattern characterization, etc. CybOX provides a common mechanism (structure and content) for addressing cyber observables across and among this full range of use cases improving consistency, efficiency, interoperability and overall situational awareness. STIX leverages CybOX for this purpose, such as in indicator patterns, infrastructure descriptions, and course of action parameters.

Overseeing Organization: Cybox is being transitioned from MITRE to OASIS.

2.2 Mandiant's Open Indicators of Compromise (OpenIOC)

The STIX Indicator's test mechanism field is an extensible alternative to providing an indicator signature in something other than CybOX. Mandiant's Open Indicators of Compromise, Open Vulnerability and Assessment Language (OVAL), SNORT rules, and YARA rules are supported as default extensions to that test mechanism field.

Overseeing Organization: MANDIANT

2.3 Malware Attribute Enumeration and Characterization (MAEC)

MAEC is a standardized language for encoding and communicating high-fidelity information about malware based upon attributes such as behaviors, artifacts, and attack patterns. By eliminating the ambiguity and inaccuracy that currently exists in malware descriptions and by reducing reliance on signatures, MAEC aims to improve human-to-human, human-to-tool, tool-to-tool, and tool-to-human communication about malware; reduce potential duplication of malware analysis efforts by researchers; and allow for the faster development of countermeasures by enabling the ability to leverage responses to previously observed malware instances. STIX leverages MAEC via the TTP construct for this purpose, and additionally both STIX and MAEC use CybOX.

Overseeing Organization: MITRE, DHS

3. Enumerations

Enumerations define global identifiers to reference shared data objects for e.g. Common Vulnerabilities and Exposures (CVE).

3.1 Common Attack Pattern Enumeration and Classification (CAPEC)

CAPEC is a publicly available, community developed list of common attack patterns along with a comprehensive schema and classification taxonomy. Attack patterns are descriptions of common methods for exploiting software systems. They derive from the concept of design patterns applied in a destructive rather than constructive context and are generated from in-depth analysis of specific real-world exploit examples. STIX can utilize Common Attack Pattern Enumeration and Classification (CAPEC) for structured characterization of tactics, techniques, and procedures (TTP) attack patterns through use of the CAPEC schema extension.

Overseeing Organization: MITRE, DHS

3.2 Common Vulnerabilities and Exposures (CVE)

CVE is a dictionary of publicly known information security vulnerabilities and exposures. CVE's common identifiers enable data exchange between security products and provide a baseline index point for evaluating coverage of tools and services.

Overseeing Organization: MITRE, DHS

3.3 Common Weakness Enumeration (CWE)

CWE provides a unified, measurable set of software weaknesses that is enabling more effective discussion, description, selection, and use of software security tools and services that can find these weaknesses in source code and operational systems as well as better understanding and management of software weaknesses related to architecture and design.

Overseeing Organization: MITRE, DHS

3.4 The Common Configuration Enumeration (CCE)

The Common Configuration Enumeration, or CCE, assigns unique entries (also called CCEs) to configuration guidance statements and configuration controls to improve workflow by facilitating fast and accurate correlation of configuration issues present in disparate domains. In this way, it is similar to other comparable data standards such as the Common Vulnerability and Exposure (CVE) List, which assigns identifiers to publicly known system vulnerabilities.

Overseeing Organization: Transitioned from MITRE to NIST

3.5 Common Platform Enumeration (CPE)

Common Platform Enumeration (CPE) is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise's computing assets. CPE does not identify unique instantiations of products on systems, such as the installation of XYZ Visualizer Enterprise Suite 4.2.3 with serial number Q472B987P113. Rather, CPE identifies abstract classes of products, such as XYZ Visualizer Enterprise Suite 4.2.3, XYZ Visualizer Enterprise Suite (all versions), or XYZ Visualizer (all variations).

Overseeing Organization: Transitioned from MITRE to NIST

4. Scoring and measurement frameworks

These standards define quantitative description of threats.

4.1 Common Vulnerability Scoring System (CVSS)

Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. It attempts to establish a measure of how much concern a vulnerability warrants, compared to other vulnerabilities, so efforts can be prioritized. The scores are based on a series of measurements (called metrics) based on expert assessment. CVSS consists of three metric groups: Base, Temporal, and Environmental. The Base group represents the intrinsic qualities of a vulnerability, the Temporal group reflects the characteristics of a vulnerability that change over time, and the Environmental group represents the characteristics of a vulnerability that are unique to a user's environment. The Base metrics produce a score ranging from 0 to 10, which can then be modified by scoring the Temporal and Environmental metrics. A CVSS score is also represented as a vector string, a compressed textual representation of the values used to derive the score.

Overseeing Organization: It is under the custodianship of the Forum of Incident Response and Security Teams (FIRST).

4.2 Common Weakness Scoring System (CWSS)

The Common Weakness Scoring System (CWSS) provides a mechanism for prioritizing software weaknesses in a consistent, flexible, open manner. It is a collaborative, community-based effort that is addressing the needs of its stakeholders across government, academia, and industry. CWSS standardizes the approach for characterizing weaknesses. Users of CWSS can invoke attack surface and environmental metrics to apply contextual information that more accurately reflects the risk to the software capability, given the unique business context it will function within and the unique business capability it is meant to provide. This allows stakeholders to make more informed decisions when trying to mitigate risks posed by weaknesses. CWSS is distinct from - but not a competitor to - the Common Vulnerability Scoring System (CVSS). These efforts have different roles, and they can be leveraged together.

Overseeing Organization: MITRE, DHS

4.3 The Extensible Configuration Checklist Description Format (XCCDF)

XCCDF was created to document technical and non-technical security checklists using a standardized format. The general objective is to allow security analysts and IT experts to create effective, interoperable automated checklists, and to support the use of these checklists with a wide variety of tools. A checklist is an organized collection of rules about a particular kind of system or platform. Automation is necessary for consistent and rapid verification of system security because of the sheer number of things to check and the number of hosts within an organization that need to be assessed (often many thousands). XCCDF enables easier, more uniform creation of security checklists, which in turn helps to improve system security by more consistent and accurate application of sound security practices. Adoption of XCCDF lets security professionals, security tool vendors, and system auditors exchange information more quickly and precisely, and also permits greater automation of security testing and configuration assessment. XCCDF development is being pursued by NIST, the NSA, The MITRE Corporation, and the US Department of Homeland Security. XCCDF is intended to serve as a replacement for the security hardening and analysis documentation written in prose. XCCDF is used by the Security Content Automation Protocol.

Overseeing Organization: NIST

4.4 Common Configuration Scoring Scheme (CCSS)

CCSS addresses software security configuration issue vulnerabilities. CCSS is largely based on CVSS and CMSS, and it is intended to complement them.

Overseeing Organization: NIST

5. Process frameworks

These are frameworks for exchanging security information, they leverage formats and protocols defined by other standards.

5.1 The Security Content Automation Protocol (SCAP)

The Security Content Automation Protocol (SCAP) is a method for using specific standards to enable automated vulnerability management, measurement, and policy compliance evaluation (e.g., FISMA compliance). The National Vulnerability Database (NVD) is the U.S. government content repository for SCAP. SCAP combines a number of open standards that are used to enumerate software flaws and configuration issues related to security. They measure systems to find vulnerabilities and offer methods to score those findings in order to evaluate the possible impact. It is a method for using those open standards for automated vulnerability management, measurement, and policy compliance evaluation. SCAP defines how the following standards (referred to as SCAP 'Components') are combined:

1. Common Vulnerabilities and Exposures (CVE)

2. Common Configuration Enumeration (CCE)
3. Common Platform Enumeration (CPE)
4. Common Weakness Enumeration (CWE)
5. Common Vulnerability Scoring System (CVSS)
6. Extensible Configuration Checklist Description Format (XCCDF)
7. Open Vulnerability and Assessment Language (OVAL)

Overseeing Organization: NIST

5.2 Cybersecurity Information Exchange, Recommendation ITU-T X.1500 (CYBEX)

Recommendation ITU-T X.1500 describes techniques for the exchange of security information. It includes guidance in on several key functions related to information exchange: structuring security information, identifying security information and entities; establishment of trust between entities; requesting and responding with security information; and assuring the integrity of the security information exchange.

Overseeing Organization: ITU-T

6. Transport

6.1 Trusted Automated eXchange of Indicator Information (TAXII)

TAXII (Trusted Automated eXchange of Indicator Information) is the main transport mechanism for cyber threat information represented in STIX. Through the use of TAXII services, organizations can share cyber threat information in a secure and automated manner. Like STIX, TAXII is led by DHS and the STIX and TAXII communities work closely together (and in fact consist of many of the same people) to ensure that they continue to provide a full stack for sharing threat intelligence.

Overseeing Organization: OASIS

6.2 The OASIS Customer Information Quality (CIQ)

The OASIS Customer Information Quality (CIQ) is a language for representing information about individuals and organizations. The STIX Identity structure uses an extension mechanism to represent identify information used to characterize malicious actors, victims and intelligence sources. The STIX-provided extension leverages CIQ. CIQ Specifications enables organisations to have a unified and consistent representation and standardization of their party data (e.g. employee, members, suppliers, partners, customers, etc) and use it to support various application requirements in the organisation that deal with party data (e.g. Master Data Management (MDM), Customer/Party Data Integration, Party identification/recognition/identity management,

HR, billing, sales, marketing, data quality and integrity, e-commerce/e-business, party data exchange, postal services, customer/party views, etc). By representing and managing party data consistently using CIQ specifications as the base scheme for an organisation, and extending it to support specific business requirements, unique identitification, integration, standardisation, matching, synchonization and management of quality party information/data is possible.

Overseeing Organization: OASIS