A Unified Data Model for Cyber Threat Intelligence in Operational Technology Networks

Vijayakumar, S.

Award date:
2020

Department of Mathematics and Computer Science
Security and Privacy Research Group

A Unified Data Model for Cyber Threat Intelligence in Operational Technology Networks

Master Thesis

Sangavi Vijayakumar

Supervisors:
Dr. Luca Allodi
Dr. Mario Dagrada

Eindhoven, February 2020

Abstract

Intrusion Detection Systems are devices (both hardware and software) that monitor network and host activity in an organization. These systems generate a large number of security events every minute, which correspond to a wide range of activity from attempted breach by accessing the internet to system reconfiguration of the critical assets in the network. Some of these alerts might be a false positive and the security analyst looking at the IDS would be drowned in a sea of false positives generated every minute. This situation raises the concern for better data organization for the security analyst to navigate through the sea of alerts and investigate a particular incident.

The IDS has a lot of sensors that monitor the network and feed the data to a central command center, which the analyst views. This data is raw and unstructured, and present in different locations in the IDS. For instance, the alerts might be available separately and the asset inventory (if it exists), in another page/tab. Thus, it is not efficient to view the alerts/activities of a particular asset in one place. The analyst has to navigate through these different data structures to find the relevant information during investigation. The Unified Data Model proposed in this thesis attempts to solve this hassle, by collecting and organizing the raw data in an IDS in a structured graph with nodes as entities from the IDS and edges as relationships linking these entities together. In this way, an analyst can visually interact with the structured IDS data, explore the context around entities and so on.

This Data Model was then used to analyze the infamous Stuxnet Incident and verify the previous analysis done by other organizations. The data model was then validated against multiple IDS architectures, fitting data from different IDSs and presenting them in a graphical manner for Cyber Threat Intelligence (CTI) in both Information and Operational Technology (IT and OT) Networks.

The main utility of this data model is its ability to represent entities from the IT and the OT domain, enabling analysts to investigate incidents even in critical infrastructure industries such as Water, Nuclear and Oil and Gas. Moreover, the data model is abstract and decoupled from the underlying data storage technologies, thus allowing a high degree of customization and extensibility to support different IDSs. However, this data model has to be configured and installed for each IDS and this can be expensive in terms of effort and time. Nevertheless, once it is installed, it can be used indefinitely with the IDS.

Acknowledgements

This work would have not been possible without two main entities - my supervisor dr. Luca Allodi and the Research Team at Forescout Tecnhologies, Eindhoven.

I would like to express my heartfelt gratitude towards dr. Luca Allodi, for his constant motivation and feedback during the Master Thesis. He was an inspiration and helped me contribute my best efforts towards the Thesis. I thank the Eindhoven University for giving me this opportunity to pursue my Master's Degree here. I would be proud to call myself a TU/e Graduate.

I would like to thank immensely the Research Team in Forescout for providing me the infrastructure and support for completing this Thesis successfully. Special mention to my inspiration - dr. Elisa Costante and dr. Mario Dagrada, without whom I would not be here. Their constant support and constructive feedback boosted my productivity. I would also like to thank Qasim Albaqali and Alessandro Manzi for their support and collaboration.

My parents and my brother Sanjay, from India, without whom this Masters Degree would not have been possible. I am here, thanks to their emotional and financial support.

Last but not the least, I would like to express my ultimate gratitude to my best friend and fellow student Sashaank, who walked the bed of roses and thorns with me side by side, encouraging and supporting me throughout this journey of my Master's Degree. I would not have completed this without him, my pillar of strength and support.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

The convergence of the Information Technology (IT) and Operational Technology (OT) landscapes has gained momentum recently, after witnessing the huge volume of data generated by industries and the hardships associated with maintaining this data. These two environments have been independent of each other with different focus areas. For instance, the IT landscape was more data-driven than the OT landscape, which was expertise-driven. The prioritization of security principles for these two landscapes were also different, one focusing on the security of assets (OT) and the other on the security and privacy of data (IT) [21]. However, the critical need to secure the industrial automation devices and the sheer complexity of the OT processes has driven the IT and OT environments to interoperate. While this integration may have a lot of benefits such as improved visibility and performance, it also raise concerns on the privacy of the data shared by these devices and their operational security. Operational security of the devices refers to the integrity and correctness of the operations or automation tasks performed by them.

Industrial Control Systems (ICS) are logical control systems, that perform automation tasks, found in critical infrastructure industries such as oil and gas, nuclear, and electrical power, which are major corporations that provide the basic necessities such as fuel, weapons, and trade that drive a nation's economic needs. These industries have been under attack by targeted and state-sponsored groups for quite some time. The importance of security in these OT networks was established after the infamous Stuxnet attack in the early 2000s, when a group of state-sponsored American hackers compromised the nuclear power industry in Iran, using Zero Day vulnerabilities in Windows Operating Systems (OS) [38]. This attack inspired a lot of different attacks of the same flavor to follow suit. This chain of attacks on the ICS industries has led security researchers to investigate and evaluate their resilience against cyber-attacks to reduce its impact and strengthen the resilience of the networks. For this purpose, it becomes paramount to represent these OT enterprises' data in a coherent manner for analysis and knowledge discovery. The modelling and unification of the data in a comprehensive manner can drive security researchers in the right direction to perform security investigations on the the incidents in these OT networks. For instance, the researchers may wish to identify vulnerable assets, patch them, identify possible attack paths and strengthen their defenses to keep their networks safer than before. The following section details on the research question and the motivation behind it.

1.1 The Problem Statement

This convergence of the IT and OT landscapes has attracted a lot of threats as explained in section 1.2. Threat Intelligence (or Threat Intel) in cyber-security is defined as the accumulation

of threat data for knowledge sharing and analysis. Threat Intel data is a subset of an enterprise's data. Threat Intel data guides enterprises to be more proactive in defending their organization against cyber attacks. It helps them understand the mechanism of a threat and its impact on the organization. For this purpose, analysis of threat data has gained importance. This has led to several efforts to aggregate threat data and express them in a standard format for exchange and analysis. Following this, MITRE and Verizon introduced their threat data models [26, 35]. MITRE has gone a step further than simply defining a data model to express threats. These two models however focus on the IT data and do not have support for OT data in the data model. Moreover, all these data are generally present in the IDS where an organization's network is monitored for threats and suspicious activity. These IDS' collect data from different sensors and store them in different raw formats split across different locations, without any context. This emphasizes the need for a data model to organize and structure the data for analysis and threat intelligence purposes. This problem leads us to the following research question of this thesis. Given the detailed problem statement, the following section presents the proposed solution we aim to build for addressing the research question.

1.1.1 Research Question

The primary research question that this thesis aims to address is that,

RQ1. How can the entities monitored by an Intrusion Detection System (IDS) be presented to a security analyst for an efficient intrusion detection and incident investigation in an Operational Technology (OT) network?

The following sub-section elaborates on how this research question could possibly be addressed.

1.1.2 The Proposed Solution

The basis for analyzing threat data is to define a rich data model that can capture the most important concepts in the data and create relationships between them to understand how these concepts interact. For instance, to analyze the Stuxnet[1] attack on ICS industries, it is essential for us to capture the different stages of the attack - cyber kill chain phases, the devices affected and the vulnerabilities in those devices. These devices, vulnerabilities, cyber kill chain phases are the concepts of the data model and the relationships become the edges linking these concepts together. Thus, it is paramount to have a rich data model with concepts and relationships to represent, analyze and share the threat data.

The high-level goal of this research is to create a domain-specific unified security data model that can be used to support security analysts with their OT investigation activities. More specifically, this research defines an OT security domain model that represents the concepts and its relationships of the OT domain such as hosts, controllers, engineering work stations, users, etc. and connects them to concepts related to the cyber-threats such as security alerts, operational alerts, network operations, vulnerabilities, Indicators of Compromise (IoC) and so on. The data model should allow the exploration of the relationships among the different concepts, both IT and OT. Additionally, the data model should be annotated with metadata, contextual and semantic information to aid the security analyst in incident response activities. This research focuses on developing such a data model, that serves the following purposes:

1. Unify data from different sources/domains to give a holistic representation of the enterprise's data, as stated in section 2.2.1.

[1] https://en.wikipedia.org/wiki/Stuxnet

2. Capture concepts and relationships related to the OT security domain - such as Devices, Network Operations, Links, sensor information and so on for threat analysis. So far, the existing threat models mentioned in chapter 3 only focus on the IT security domain. This research extends the existing literature by adding support for expressing the OT concepts.

3. Perform qualitative evaluation of the OT security data model and verify that it can capture most of the concepts present in an OT enterprise and satisfy the use-cases defined in section 5.1.

The novelty of the proposed unified data model lies in its ability to combine a general representation of Intrusion Detection System (IDS) data and the OT specific attributes together as a single unified data model. This process will help an analyst to query the knowledge graph to get more insight into some of the concepts and allow them to extract knowledge which might have been vague before modelling the knowledge graph. This is an important contribution to OT Threat Intelligence - to look at OT data in a more structured and comprehensive way for efficient incident investigation.

1.2 Motivation

ICS industries have been targeted by cyber-attacks for a long time now. The first multi-stage large scale ICS attack was the Stuxnet attack by the Americans on Iranian nuclear power systems. The Equation Group[2] is the APT Threat Actor behind this attack and other well known attacks such as the Flame malware that leveraged a cryptographic vulnerability to create false certificates and masquerade as a legitimate software to avoid being detected by the anti-virus software[3]. The Flame malware leverages the vulnerabilities in Windows machines that were responsible for the Stuxnet attack and was targeted to infect the Iranian Oil Ministry's systems.

Similarly DuQu was another targeted Cyber-Espionage attack on ICS Networks that also used the vulnerabilities that led to Flame and Stuxnet's success. Although, it was targeted on Iranian systems just like Stuxnet and Flame, its main purpose was to steal information and was more of a spyware than a sabotaging malware like Stuxnet.

Figure 1.1: The timeline of cyber-attacks against the ICS Network Enterprises, Source: [27]

McAfee's blog post [27] on the different ICS attacks so far offer an interesting insight as to why it is important to focus on the security of OT networks. The figure 1.1 above shows the timeline of the major ICS attacks that have been identified so far. The author of the blog post [27] discusses in detail the mechanism of the recent Triton malware that targeted a specific product, namely the Triconex safety controller by Schneider Electric. This safety controller component is prominently used in most ICS enterprises such as the oil and nuclear industries. This malware is said to have been propogated via a phishing attack and masqueraded as an innocent logging software `trilog.exe` , which is an IoC, that executed the attack in stages.

[2] https://en.wikipedia.org/wiki/Equation_Group
[3] https://en.wikipedia.org/wiki/Flame_(malware)#Operation

The analysis of such attacks require the need to have a data model that captures the different concepts involved in the attack - Indicators, Cyber-Kill-Chain-Phases, Devices (SIS Workstation, SIS Controllers), Protocols, Malware and so on for effective incident response. If these concepts are present in an ICS Network Enterprise's data, a security researcher can explore the attack paths, vulnerability of devices and protocols (in the Triton case) and assess the impact of a threat/risk to the organization. This can lead to proactive measures for handling cyber-attacks to be formulated, equipping experts to stay vigilant for such attacks.

1.3 Structure of the Thesis

The document is organized as follows, the first chapter explains the introduction to the research question and the problem statement it attempts to solve. Chapter 2 discusses information relevant to understand the background of this document's purpose and technicalities. Chapter 3 details on the existing data models and databases (unified data stores) for storing the knowledge graph, following by a qualitative comparison of the data models relevant for CTI. Chapter 4 provides a detailed overview of the prospective data model, which this work wishes to extend to address the problem statement discussed above. Chapter 5 elaborates on the proposed solution for solving the research question by leveraging the existing STIX data model. Chapter 6 discusses the case studies used to evaluate the unified data model described in chapter 5 on its abilities to express threats using real-time OT datasets. The chapter 7 discusses the results of the evaluation and the final chapter 8 concludes the research with the scope for future work.

Chapter 2

Background

This chapter elaborates on some relevant background information to understand the flow of the thesis and the motivation behind the research question of this thesis. The first section discusses the OT Networks in detail, followed by section 2.2, which introduces the concept of knowledge graphs and its application in the cyber-security domain for investigating the security incidents.

2.1 Operational Technology Networks

Operational Technology (OT) networks include the ICS devices that perform the automation and operational tasks for industrial processes. The OT industries are a combination of IT and OT networks, with a clear hierarchical division between them. This hierarchy is none other than the Purdue Levels, introduced by The PERA [37] by Purdue University. OT environments, unlike IT, are composed of more than just workstations and servers, they are composed of physical and logical devices such as sensors and Programmable Logic Controllers (PLCs). There are 5 Purdue Levels, numbered from 0 through 4, which consists of different devices in each level.

The Purdue Level is an enterprise architecture model, defined in 1992, that separate devices into logical partitions depending on their functions [37]. For instance, logistics and employee workstations are on level 4, which is the highest layer. Devices such as PLCs and sensors are on the lowest layer, since they perform physical tasks commanded by one or more master SCADA systems in the higher levels. This is illustrated in the figure 2.1 below. The data model for such an OT environment must consider this enterprise architecture and the dynamic network topology in order to model the different security concepts including threats. This data model can be used to aid an Security Operations Center (SOC) expert to make time-critical decisions on incoming security alerts.

This leads us to question the security of these assets in the different Purdue Levels and the unification of these assets in an overall enterprise view in order to monitor for threats and gain deeper insight into the ICS enterprise's data.

[1] https://www.win.tue.nl/~setalle/CCD/

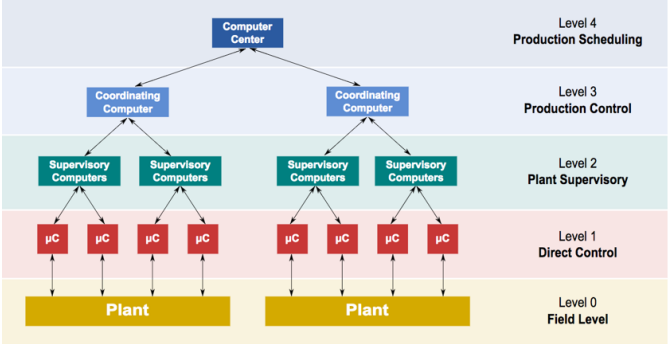


Figure 2.1: The architecture of systems in an OT environment, Source: [1]

2.2 Enterprise Knowledge Graphs

In order to safeguard an enterprise's assets and monitor them for threats, some kind of unification and data modelling might be required. This is because enterprises in general, have data residing in various databases that serve multiple purposes depending on the application they are designed for. For instance, enterprises may store their legal and audit data and their customer data in separate databases to enforce access control and logical data separation. In order to cater to their business needs, a holistic view of their data might be needed for a better clarity of decision making. This can involve integrating data from various sources and presenting them in a neat and understandable way that can drive the decision making process. For this reason, the enterprise knowledge graphs became popular, when Google first introduced their knowledge graph data model in 2012 [31]. From then on, enterprises are adopting this model for corporate data management, knowledge representation and sharing for efficient decision making purposes [9].

The author of [14] defines enterprise knowledge graphs as a centralized data model that can be used for knowledge representation, discovery and sharing. This enterprise knowledge graph is tailored for every enterprise to suit its applications and its domain in the industry market. The author of [14] discusses the components of an enterprise knowledge graph which is shown in the figure 2.2 below. An enterprise knowledge graph typically consists of a business taxonomy - that explains the enterprise's needs, concepts and vocabulary, business data sources, a Unified Data Model (UDM) and a graph database. The UDM is a centralized enterprise data schema that is independent of any database technologies and is vital for a good understanding of how the data is organized in the database. This UDM can be used to represent knowledge on a very high level that can provide a quick overview of the data and its relationships. The enterprise knowledge graph is created by validating the enterprise's data against the UDM to provide structure and semantics to it. The enterprise knowledge graph should enable business users to explore the relationships between data and the evolution of data with attached semantics. Thus, having this enterprise knowledge graph can help understand and operate on the data in a more efficient manner than disparate data with no semantics.

The graph data model stores the schema in an unstructured format, captured as nodes with properties and relationships between the nodes. A category of the graph data model is the knowledge graph model, which is a property graph augmented with context and semantics about the data. The knowledge graph schema is continuously evolving and thus dynamic. Knowledge graphs are structured, with high scalability and embedded semantics that are self-descriptive and enable an analyst understand the data better. The self-descriptive nature of these graphs

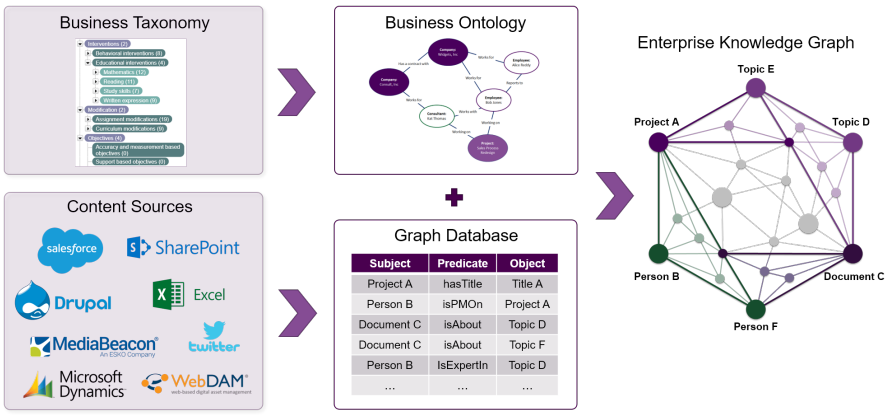


Figure 2.2: The components of an Enterprise Knowledge Graph, Source: [14]

provides contextual knowledge of the nodes and their relationships, hence the term - knowledge graphs. The figure 2.3 below represents a simple knowledge graph that connects various entities involved in Star Trek in different ways, much like the human brain does [5]. For instance, Spock is a character in Star Trek that was played by Leonard Nimoy. While regular graphs may stop with Star Trek's genre as science fiction, a knowledge graph will recommend "Star Wars" to a user who search for "Star Trek". Tech giants such as Google, Facebook and LinkedIn use this knowledge graph data model to offer recommendations to their users based on their interests and strengths.

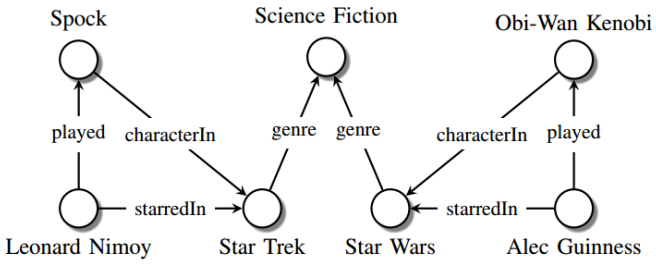


Figure 2.3: Example of a knowledge graph modeling the Star Trek franchise, Source: [22]

2.2.1 Enterprise Knowledge Graphs for Cybersecurity

The previous paragraph highlighted the importance of having a knowledge graph for enterprises in general. This problem is partly addressed by [30] which is a framework for building and maintaining enterprise knowledge graphs using semantic web ontologies and graph databases. The present work however focuses on developing a rich data model for an enterprise knowledge graph in the OT security domain. The OT security enterprises are those that provide security solutions to ICS industries, such as Forescout Technologies[2] who develop solutions that offer device visibility and control for Industrial Control Systems (ICS) networks. The low-level sensors in ICS networks periodically send signals to indicate normal and abnormal behaviour to their

[2] https://www.forescout.com/company/

master SCADA systems. OT security enterprises collect all this information from their Command Center (CC), which is a front end interface for their underlying Security Incident and Event Management (SIEM) tool, and provide aggregate statistics of the data. The data from the CC can then be fed into popular database management systems for storage and processing. Within enterprises, there may be various divisions such as research, engineering, and product that work with the same data differently. This flow of data in an OT enterprise is shown in figure 2.4 below. It is paramount to also provide a holistic view of the data that can drive new research possibilities and also evaluate the current system. Thus, an enterprise-wide knowledge graph, that models data from different landscapes (IT and OT) with rich relationships, would fulfill this purpose.

The Data Flow Architecture

Unifying the data from various sources and fitting them against the UDM could look like the image shown in figure 2.4 below. The data sources mentioned in the image are raw data sources that can either be structured or unstructured. The UDM defined in this work could organize and structure these raw data to build the enterprise knowledge graph for an ICS enterprise and finally store it in a unified data store such as Neo4j, since we intend to store data as a graph. The choice of Neo4j as a data storage technology will be elaborated in the next chapter 3.

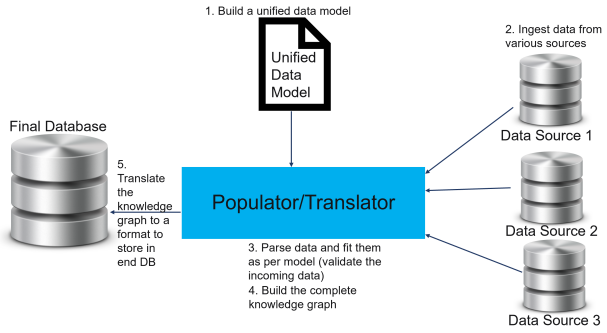


Figure 2.4: The Proposed Architecture for Building an Enterprise Knowledge Graph

A more detailed example of how this architecture can be used to fit this research is shown in figure 2.5 below. The different concepts which we wish to express in the data model are present in various data sources. All these raw data have to be aggregated and fitted against the data model. The UDM defines the concepts and relationships that can exist in the knowledge graph. The Populator populates the knowledge graph as shown in the yellow boxes in the image 2.5. This knowledge graph is a structured representation of the raw data and will then be fed into a unified data store such as Neo4j for visualization and analytics.

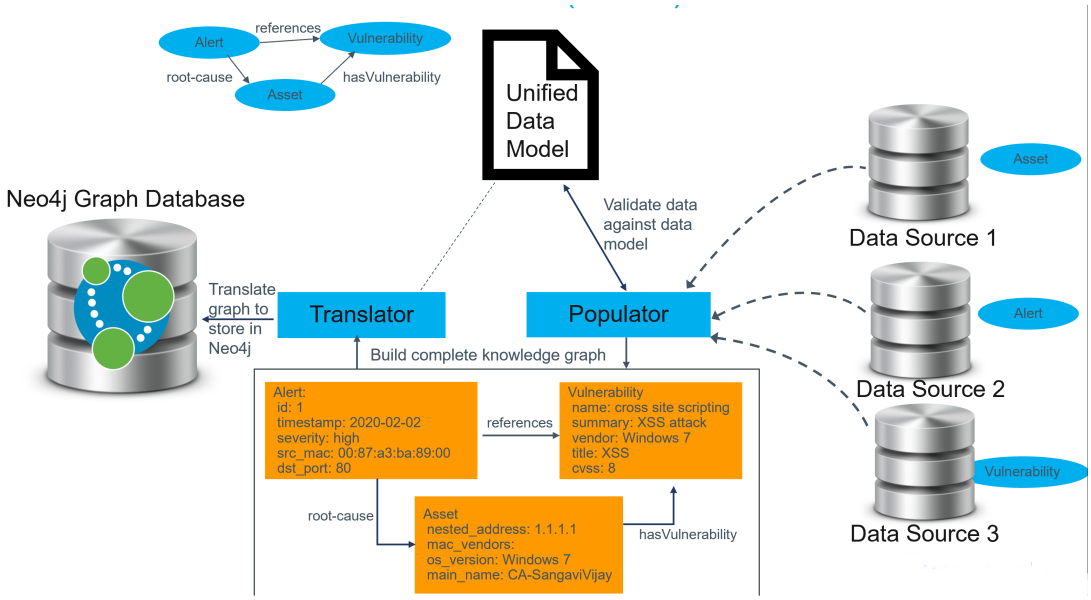


Figure 2.5: The Refined Architecture for Building the Enterprise Knowledge Graph for OT Security domain

Chapter 3

State of the Art

This chapter introduces the idea of a data model and reviews some of the previous work done for developing data models for the cyber-security domain and specifically for threat intelligence and information exchange. The first part of the chapter reviews the existing data models and compares them in terms of human-readability, expressiveness and community support and adoption. The second part of the chapter reviews some implementation and storage technologies used for storing the data as a knowledge graph for an interactive and effective security incident investigation.

3.1 Existing Data Models in Cyber-Security

A data model is an abstract representation of data as entities and relationships linking those entities. They define how the data is structured and connected in the database. There are several data models that serve different domain-specific data. Choosing the most appropriate model for the given domain, Industrial Control Systems (ICS), is a crucial task. This research focuses on building a unified data model for OT networks, that gives a wholesome and consolidated view of all the data and their relationships. The chosen data model will represent the OT security concepts, such as alerts, host change logs, assets, users, and their relationships along with a security risk factor that can help the SOC analyst make better decisions when alerts are raised. For this purpose and to understand the previous research in this domain, the following types of data models are explored to discover the most suitable data model for this research.

3.1.1 The STIX Data Model

MITRE, which is an industry leader in security solutions, established a new data model, modelling security concepts, in JSON format, known as Structured Threat Information Expression (STIX) [26]. This data model was developed for exchanging threat intelligence information in a standard format that was readable and expressive. The concepts in this model have properties (attributes) that describe them, along with a unique identifier. Relationships are modelled as separate objects with unique identifiers. Thus, this allows for dynamic creation or modification of relationships between data objects. Relationship objects are associated with properties such as source, target, label and type. The "relationship_type" property allows the users to specify one of the existing relationship types given in STIX, or create custom relationship types using the APIs they provide. Similarly, new data objects and properties can be created using their APIs. These APIs are available in the Python language. The figure 3.1 below shows a small

example of the relationship between an “indicator” (IoC) and a “malware” object. The following figure 3.2 summarizes the different concepts and the relationships between them.

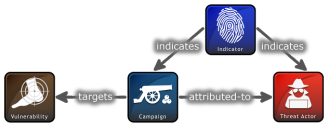


Figure 3.1: An example of a relationship between an IoC object, vulnerability, campaign and threat actor in STIX, Source: [26]

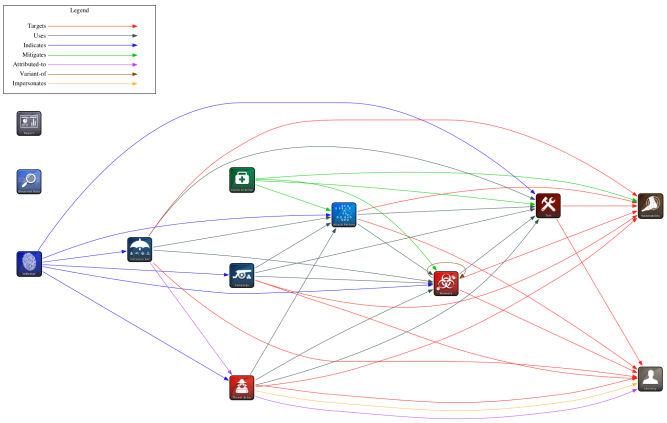


Figure 3.2: The concepts and relationships of domain objects in STIX, Source: [26]

The CyGraph Project

CyGraph is a military-grade effort by MITRE to model cyber-security data as logical concepts and link them using relationships as a graph. This graph is what they call an “attack graph”. CyGraph builds such a dynamic attack graph, as shown in figure 3.3, that maps the entire path of a cyber attack on an enterprise’s system. It also includes back propagation to identify the locus of an attack and forward propagation capabilities that predict how a system entity such as a host workstation’s vulnerabilities can be leveraged in an attack. For instance, the following query 3.1 was used to mine the subgraph shown in figure 3.3. It matches all different paths from one alert to the other. CyGraph has its own query language called CyGraph Query Language (CyQL), which is based on Neo4j’s Cypher.

Listing 3.1: Query mining the relationship between two alerts, to generate an attack graph. [23]

```
MATCH paths = (:Alert {name:"Snort 33022"})-
[:SRC|DST|DETECTION|ON|ENABLES|AGAINST|PREPARES*]->
(:Alert {name:"Snort 1576"})
RETURN paths
```

The authors of [23] use the Neo4j graph database, to model entities in attack graphs for analysis and visualization. The CyGraph ingests data from various sources such as the network

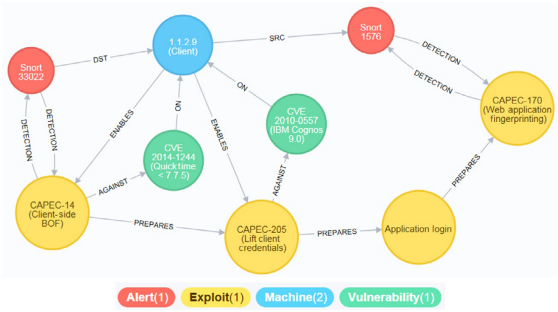


Figure 3.3: Example of an attack graph mined dynamically by CyGraph relating two different alerts, Source: [23]

infrastructure layer, vulnerability databases (NVD, CAPEC), sensors and so on, and models them as an undirected knowledge graph which relates nodes of different origins and types contextually. This knowledge graph is represented in figure 3.4. Relational databases were deemed to be inefficient for evolving network environments, since all relational models must possess a schema, and schema modification is a cumbersome process [23]. Moreover, the expensive join operations were eliminated in the graph model, thus increasing query response time to a great extent.

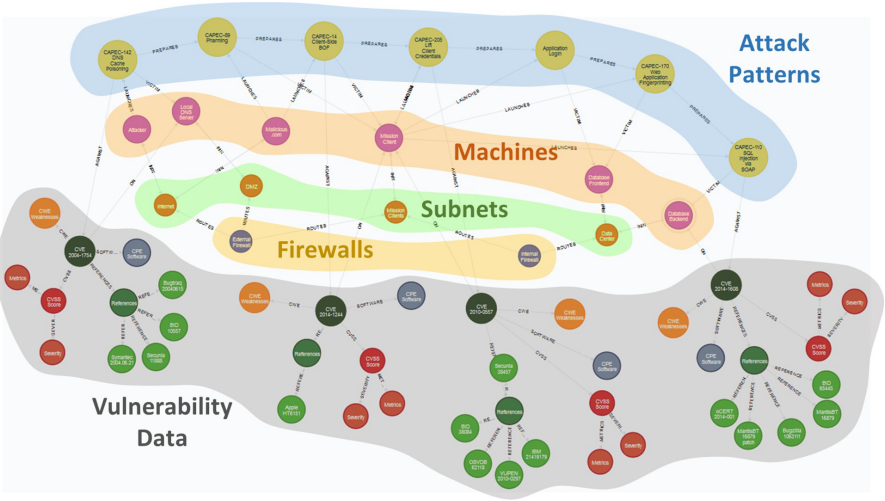


Figure 3.4: Complete knowledge graph of entities developed by CyGraph, Source: [23]

3.1.2 The SEPSES Data Model

The SEPSES data model [15] attempts to build a knowledge graph for the security domain using the Resource Description Framework (RDF), SPARQL for querying and public data sources such as the Common Vulnerability Exposures (CVE), National Vulnerability Database (NVD) and Common Attack Pattern Enumeration and Classification (CAPEC). The knowledge graph, as shown in figure 3.5 below, models the vulnerabilities for each asset and links the vulnerabilities with their Common Vulnerability Scoring System (CVSS) score. The SEPSES graph does not map attack paths to the network assets and help in proactive incident response, since it only helps

in identifying vulnerable assets in the network. Moreover, the section on the Intrusion Detection case study in [15] shows an example of the query to discover vulnerable assets with CVE identities using Snort rules. The query spans more than 10 lines and the results are displayed as a table with disparate rows to be interpreted by a SOC analyst. While the idea of a data model for security and building a knowledge graph with concepts and semantics, [15] fails to visualize and process the results in an intuitive manner. The results can be confusing and may slow down the SOC analyst while making time-critical decisions. CyGraph [23] on the other hand, links different alerts and enables a SOC analyst to understand if an asset is vulnerable and prone to attack and what can be done to mitigate or prevent it. The goal of this research is to build such a data model, like [15, 23] does and represent it in an interactive manner, such as [23] does, for the OT environment.

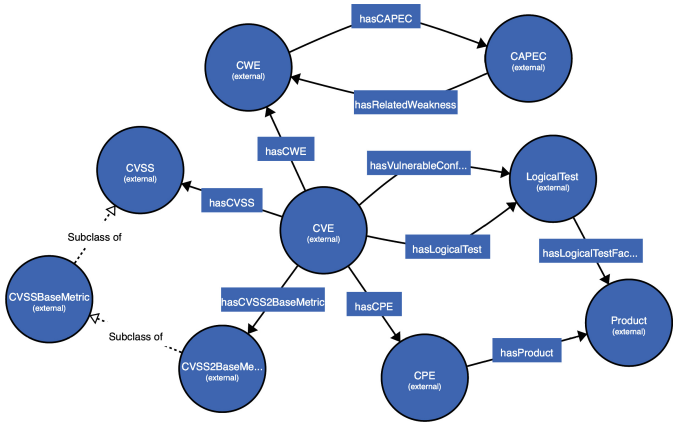


Figure 3.5: The SEPSES knowledge graph modeling the vulnerabilities that each asset in a system might possess along with their severity and possible mitigations, Source: [15]

The STUCCO Data Model

The SEPSES knowledge graph [15] was based on the STUCCO project [12], that also obtained data from more than 13 structured data sources including the NVD database. The STUCCO knowledge graph is shown in the figure 3.6 below. Their knowledge graph is a high level abstraction of the concepts in a security domain [12]. This research aims to build such a data model for the operational technology environment that involves data from Programmable Logic Controllers (PLCs) and Supervisory Control And Data Acquisition (SCADA) systems. The authors of [12] assert that using a JSON data model helped them to model their security concepts as a graph and also validate their datasets which are present in their Titan graph database.

3.1.3 The VERIS Data Model

The Verizon Community [35] developed a security data model, known as The Vocabulary for Event Recording and Incident Sharing (VERIS) in JSON data format. It was primarily developed for representing CTI information such as Incidents, Victim Demographics, Impact Assessment, Discovery and Tracking in structured manner for collaboration and exchange. These concepts are devised as "metrics" to express security information in a common and structured manner for information exchange. It models the assets and the adversary behaviour, similar to the MITRE ATT&CK framework. However, the VERIS Data Model is not complete. It can only used for

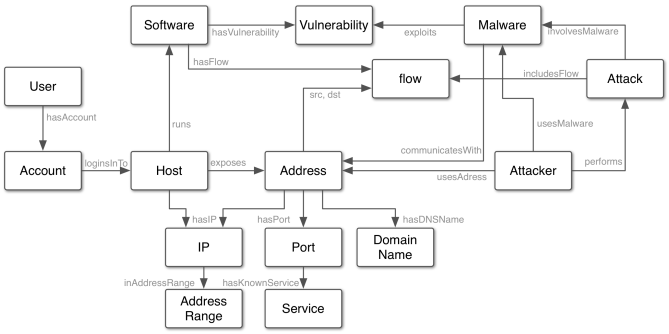


Figure 3.6: The STUCCO data model represented as a knowledge graph, Source: [12]

strategic post-incident risk-based analysis, rather than proactive analysis, since it focuses on the Incident and what led to this incident. The STIX 2.x Data Model however captures also the Indicators of Compromise (IoC) which enables the STIX Data Model to be more proactive and identify the threats and assess their impact beforehand. VERIS is a risk-based model with a narrow scope of capturing cyber-security incidents in a structured manner for information exchange. Nevertheless, VERIS offers a rich and thorough Incident concept that covers all aspects of security incidents. The STIX Data Model even reused this Incident concept of VERIS for their own. The major drawback of the VERIS Data Model, however is its inability to represent relationships amongst its different concepts.

3.1.4 The IODEF Data Model

The Incident Object Description Exchange Format (IODEF v1)[1] is another incident-focused data model developed by the Internet Engineering Task Force (IETF)to represent cyber security incident information. This model was influenced by the The Intrusion Detection Message Exchange Format (IDMEF) Data Model that concentrates on the Alert data in a cyber-security incident. The problem with these two data models is that they tend to focus on a single aspect of CTI, namely the alerts and incidents which are predominantly used for post-incident analysis. Moreover, they do not capture other important concepts such as Vulnerability, Indicators, Attack , Network Traffic and so on. The IDMEF Data Model was used to capture the Ping of Death attack only since it involved the specific Alert and Heartbeat concepts required for expressing this Ping of Death attack[2].

Nevertheless, the IODEF v2 Data Model extends the IODEF v1 by introducing the Indicator, Campaign, Threat Actor and other concepts just like the STIX v2 Data Model. It is different from STIX v2 only in terms of the data format used for creating and manipulating this data model - the XML data format. The eXtensible Markup Language (XML) is one of the most widely used Data Formats explained in the section 3.2.1 below. The major drawback of this model is its weak documentation and human readability of the XML data format, which is eliminated by the STIX v2 Data Model as the JSON data format is known for its human readability and its rich documentation and community support. The author of [19] also argues that the STIX is the most widely adopted data model for expressing and exchanging CTI information.

[1] https://tools.ietf.org/html/rfc5070#section-1.3
[2] https://en.wikipedia.org/wiki/Intrusion_Detection_Message_Exchange_Format

3.1.5 The UCO Data Model

The Unified Cyber-security Ontology (UCO) is a framework written in the RDF data format for unifying and exchanging CTI data. The authors of the UCO Data Model [34] attempt to create a standard representation of CTI information by inculcating other standards such as STIX, CyBox, STUCCO and so on. It is an open source project, but has not been part of any practical application so far. It is important to mention this ontology since it is a vital contribution to the cyber-security community.

3.1.6 The AnzoGraph Framework

The Anzo Graph by Cambridge Semantics [30] uses the RDF data format for creating knowledge graphs. Cambridge Semantics do not specialize in security, rather they provide a solution to build enterprise knowledge graphs that can serve any domain. The AnzoGraph is a solution for enterprises that wish to create a knowledge graph out of their data [30]. Using this platform, enterprises can aggregate data from their various data stores, create a data model using RDF data format, validate their actual data against the created data model and finally build and store their knowledge graph. The figure 3.7 below shows the architecture of the AnzoGraph solution. Their solution was inspired by Google's knowledge graph, which was first introduced in 2012. The AnzoGraph is a proprietary solution and thus cannot be leveraged by this research to build a unified data model.

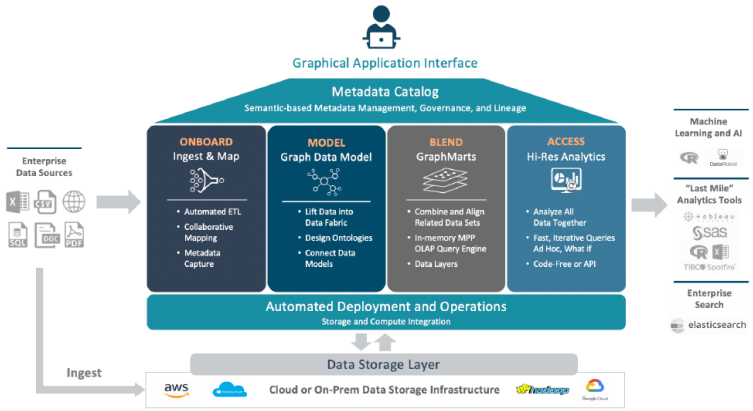


Figure 3.7: The AnzoGraph solution's approach to building a knowledge graph for enterprises, Source: [30]

3.1.7 Other Data Models

Splunk, which is a leader in Enterprise Security solutions, also uses the JSON data model to express its security concepts such as Vulnerabilities, Alerts, Network logs and so on. These concepts are expressed as JSON elements and the mappings between them are specified on the fly by the users of Splunk [13]. Splunk provides a Common Information Model (CIM) that has a pre-defined security schema that can be extended by users. They also offer a REST API to access the schema, create mappings and validate the datasets against the JSON data model.

There are several other data models such as [8, 24, 25] which are interrelated. They all try to build a three layered unified data model by dividing the security concepts among the high-level

domain independent ontologies spanning multiple knowledge bases, middle level less abstract ontologies and low level domain-specific ontologies that have significant detail. Representing security ontologies in this way can help model detailed concepts and complex relationships. However, representing knowledge in this way can become complicated for querying if there are a lot of concepts, that more or less mean the same, with multiple relationships between the different layers of the ontology. Thus, this research will focus more on the works of [15] and [23] to build the knowledge graph data model.

3.1.8 Qualitative Comparison of Data Models

The literature [19] is a survey of the various data models in the cyber-security domain, many of which are explained above. The literature compares the STIX, IODEF and VERIS Data Models against each other, the summary of which is given in table 3.8 below. As can be observed from the table, both the versions of STIX and IODEF have also been compared and contrasted in the survey. Clearly, the STIX v2 Data Model outperforms the rest by satisfying most of the criteria with a greater "degree of fulfillment" than its peers. This was attributed to its choice of data format (JSON), documentation and community support and low ambiguity which was not provided by its close competitor IODEF v2. These strong features of the STIX v2 Data Model allows it to outshine among the other data models mentioned above.

Table 3 – Analysis of incident reporting formats.

	STIX	STIX2	IODEF	IODEF2	VERIS	X-ARF
Indicator	●	●	○	●	◢	○
Attacker	●	●	○	◖	●	◖
- Attributes	●	●	○	●	●	◔
- Objectives	●	●	○	○	●	○
Attack	◔	●	●	●	◖	◢
- Event	●	●	●	●	◖	◢
- Action	●	●	●	●	◖	◢
- Target	●	●	●	●	◖	◢
- Methods and tools	●	●	●	●	◖	◢
- Vulnerability	●	●	●	●	◖	◢
- Result	●	●	●	●	◔	◢
Defender	●	●	○	◖	●	○
- Reaction	●	●	○	◖	●	○
Contentual coverage	●	●	◖	●	◖	◢
Machine-readability	●	●	◖	◔	◖	◢
Human-readability	◢	◢	◖	◢	◖	●
Unambiguous semantics	●	●	◖	●	◔	●
Interoperability	◔	●	●	●	◖	○
Extensibility	◢	◔	◔	◔	◔	◖
Aggregability	●	●	●	●	○	○
Practical application	●	○	◔	○	◖	●
External dependencies	●	●	○	◔	○	○
Licensing terms	●	●	●	●	◖	●
Maintenance efforts	◢	●	◢	●	◔	◖
Documentation	●	●	◖	◖	●	◢

Figure 3.8: Qualitative Comparison of Data Models in Cyber-Security, Source: [19]

3.2 Implementation and Storage of the Data Model

3.2.1 The Data Format - Implementation

Expressing the data model in a machine understandable format invokes the need to choose an appropriate data format that captures the data model accurately. The following subsections

discuss some of the most popular data formats used for expressing data schema, followed by a qualitative comparison between them to determine the best suited data format for this research.

The JSON Data Format

The JavaScript Object Notation (JSON) data model represents data objects as a key-value pair, where the keys are attributes of a particular data element. The attributes can be of data type string, char, boolean and so on. JSON data elements can also have nested elements inside them to create a hierarchy or sub-class instances. Each data element can be visualized as a class that can be instantiated. It is one of the most simplest and easiest form of defining a database schema.

The Resource Description Framework Data Format

The JSON data format is one of the most widely accepted language in the security industry, as can be seen from the above examples. However, there is another standard data format used for expressing security concepts in an ontological manner - the Resource Description Framework (RDF) data format. The RDF data format models concepts as triples, consisting of the subject, object and the predicate. The subject is the concept such as alert or vulnerability and the object is an instance of the subject. The predicate is the relationship between the subject and the object. This RDF data format is an umbrella of various types of data formats such as Turtle, Trig, Web Ontology Language (OWL) and so on. These languages were used to model the World Wide Web (WWW) as a semantic data model [3].

The authors of [32] use the OWL data format to express cybersecurity concepts such as assets, vulnerabilities, etc. along with their relationships. They chose the OWL format since they believed that the knowledge base in this format is easy to port and share. However, [12] argues that the JSON format is better than the OWL, since JSON is more compatible with the GraphSON format and can validate incoming data against their security schema better than OWL. Also, the authors of [28] claim that the OWL data format does not support nested objects or complex relationships.

Protocol Buffers

Protocol Buffers, also known as Protobuf commonly, is a serialization technology developed by Google [10]. It is a binary data format, which means that data is sent and stored as binary zeros and ones. Protobuf data format resembles the XML data format, and can be used to define data schemas. It is supposedly said to be better than JSON since it is faster and much more than JSON in terms of features. The author of [16] says that while JSON is just a message format, Protobuf is more than a message format since it also includes rules for data exchange and definition. However, the popular opinion according to several people [16][4] is that Protobuf has some serious disadvantages such as lack of community support and documentation, which makes it less adoptable by enterprises. Since JSON is simple, human-readable and widely adopted, it could be a better option than Protobuf for designing a unified data model for this research.

Qualitative Comparison of Data Formats

The comparison between the above-mentioned data formats considers the following metrics to evaluate the best suited choice to express the data model for this research.

1. **Backwards Compatibility:** To see if newer versions are compatible with older version so that there are no conflicts/loss of data.

2. **Polyglot:** To check if the format can be translated into multiple programming languages.
3. **Strict Scheme:** Allow for increased discipline and accurate data descriptions.
4. **Nested Objects:** To allow for complex data objects and relationships between them.
5. **Values human readability:** Implementation and structure should be understandable and simple. Not too complicated and messy
6. **Complexity:** Enough community support, complete documentation and whether training/onboarding is required to understand the data model format.

These metrics were devised by Forescout Technologies for the comparison of data formats. The following table 3.1 summarizes the comparison of the data formats against these metrics.

Table 3.1: Qualitative Comparison of Data Formats for implementing the Data Model

	JSON	**RDF**	PROTOBUF	AVRO	THRIFT
Backward Compatibility	YES	YES	YES	YES	YES
Polyglot	YES	NO	YES	YES	YES
Strict Schema	YES	YES	YES	YES	YES
Nested Objects	YES	YES	YES	YES	NO
Values Human Readability	YES	YES	PARTIAL	PARTIAL	YES
Complexity	LOW	MEDIUM	MEDIUM	MEDIUM	MEDIUM
Total Score (avg)	2.16	1.83	1.75	1.75	1.83

3.2.2 Databases - Storage

The choice of a data format to express the data model was the first step towards building the data model. The second step is to choose an appropriate database technology that utilizes this model to validate the OT data against the defined data model and store them in a manner suitable for querying and visualization. This section discusses some of the most prominent databases and compares them against a set of metrics that were derived from previous literature [1, 2, 33, 17, 18, 11], as explained below.

There are two broad categories of databases predominantly used by industries - relational and Not Only SQL (NOSQL) databases. Relational databases store data as tables and have the schema-on-read policy which requires the database schema to be loaded before loading the actual data. NOSQL databases on the other hand store data in an unstructured format, such as a graph (Neo4j), inverted index (Elasticsearch) or as structured documents (MongoDB). These databases are schema optional, which means that they do not require a database schema before loading the data. The schema is sometimes dynamically defined when data is written to the database, in some of the NOSQL databases. These two categories are discussed in detail below.

Relational Databases

Relational databases are one of the oldest forms of storing data on disk. The idea of relational databases was first introduced by Edgar Codd in 1970 [6]. These databases store data which is normalized, which means that there is no data redundancy, across several tables. These tables will then be joined using their foreign keys to produce the results of a user's query. However, these joins can become highly expensive when dealing with enterprise data. Nevertheless, relational databases do have some advantages such as structured data storage and compliance to ACID (Atomicity, Consistency, Isolation and Durability) properties of transactions in a database.

PostgreSQL Database - PostgreSQL database[3] was developed at the University of California, Berkeley and became one of the most advanced relational databases. PostgreSQL has a huge community sponsoring and supporting its development and deployment. It was acclaimed as having the world's best documentation support, which is open-source. PostgreSQL unlike other relational databases support data types that include documents, JSON, composite data types and have advanced indexing capabilities, such as bloom filters. It offers rich security features such as multi-factor authentication using certificates for enterprises where access control is necessary. PostgreSQL does not store data as a graph and thus cannot be used for this research. However, it can still be used to store the raw incoming data from the sensors and devices from the various Purdue Levels in a normalized structure before they can be validated against the data model. The features of PostgreSQL are summarized in table 3.2.

NOSQL Databases

NOSQL databases are an extension of the relational databases, with high scalability and support for storage of data in an unstructured fashion. NOSQL databases are of different types, such as document databases, graph databases, column-store databases, distributed hash tables and so on. Each type of NOSQL database have their own query language unlike relational databases that have the common Structured Query Language (SQL) to query the databases. Some of the prominent NOSQL databases are Neo4j, MongoDB, ElasticSearch, Apache HBase and Apache Cassandra.

Neo4j - Neo4j[4], is a native graph data store with graph processing engines for all the Create, Read, Update and Delete (CRUD) operations. Neo4j was designed with an intent to enable enterprises gain a useful insight into their data by allowing them to view and query relationships between the data. Graph databases can be used for applications such as fraud detection, real-time recommendations such as Google's search engine which uses a knowledge graph data model for their data. Since the aim of this research was to build something similar for the OT environment, Neo4j could be a promising candidate as a database choice to query and store the knowledge graph. The features and advantages of using Neo4j are manifold. Neo4j is probably one of the few graph databases with ACID compliance. The most alluring feature of Neo4j is that it supports a dynamic schema, that is created automatically as data is fed into the database. Neo4j has a huge community and features to inter-operate with a variety of other technologies such as Elasticsearch, MongoDB and so on. These features are summarized in table 3.2 below.

MongoDB - MongoDB[5] is the second most popular NOSQL database after Neo4j. It stores data as JSON documents that can consist of nested documents. They offer cloud based storage for

[3] https://www.postgresql.org/about/
[4] https://neo4j.com/
[5] https://www.mongodb.com/

enterprises and work well with large volumes of data. The feature that gives Neo4j an edge over MongoDB is that Neo4j is a native graph database that offers unparalleled graphical visualization and analytics than any NOSQL database. MongoDB on the other hand stores data as disparate documents that are aggregated when a query is executed and the results are displayed as charts with aggregate statistics. They do not help enterprises explore relationships between data like Neo4j. MongoDB can however be used to obtain aggregate statistics of data and store data in a structured format other than the relational model's structure. It is however more advantageous than PostgreSQL since PostgreSQL, like other relational databases, requires a database schema to be specified beforehand. Moreoever, MongoDB has high availability and scalability than PostgreSQL due to its distributed style of architecture. These features are summarized in table 3.2 below.

Elasticsearch - Elasticsearch[6], like MongoDB stores data that it obtains from different sources in JSON documents. Each keyword in these documents are indexed and an inverted index structure is created by Elasticsearch. In this structure, every unique word across every JSON document is present with references to all the documents they appear in. This is the reason behind the extremely fast full-text-search query response times in Elasticsearch. It is composed of different elements such as Kibana for querying and visualization and Logstash for data aggregation and processing. This unique indexing feature of Elaticsearch makes it stand out among other NOSQL databases, since indexes were originally used only in relational databases. Elasticsearch can also be used in the security domain as it offers endpoint security and SIEM capabilities that are updated to use MITRE ATT&CK framework for detection of security incidents and enhanced host monitoring functionalities. Since Elasticsearch offers rich security analytics features much more than storage, it is possible to use Neo4j as a database model and the Elasticsearch engine top of Neo4j to augment faster full-text search.

Qualitative Comparison of Databases

The above-mentioned databases each offer rich features to support enterprise data analytics. However, the goal of this research is focused on OT security enterprises and their knowledge graph data model. The authors of [1, 33, 11, 18, 17] use metrics to compare the different type of databases such as SQL vs. NOSQL, the different graph databases and the different NOSQL databases. They used metrics such as query expressiveness, structure of data storage, query response times, ACID compliance, usage of indexes, community support and documentation, extensibility, schema strictness and so on. These metrics were refined to compare the databases, explained in the previous sub-sections, for this research and are explained in detail below.

1. **Type:** To understand how data is represented (graph/document/tables)
2. **Schema-full/schema-less:** Schema defines how the database is structured, the entities and the relationships between them. Having a database schema is essential in understand the data in the database.
3. **Native Technologies:** The query languages used for querying the database
4. **ACID compliance:** To ensure integrity and consistency of data across the entire DB during concurrent reads/writes.
5. Full Text Search: To know if the database supports full text searches (ex: search for an IP address returns all entries/records containing the IP address from the entire database).

[6] https://www.elastic.co/what-is/elasticsearch

6. **Relationships Discovery:** The ability to view and retrieve relationships between data objects using match queries and so on.
7. **Extensibility:** Allow addition of new features/custom code without much hassle.
8. **Complexity:** Query expressiveness, query length, easy to learn and operate
9. **Community Support:** Is there enough documentation, new releases, active sponsors and likewise.

The overview of the comparison of the database technologies is shown in the table 3.2 below. The scoring metrics are also provided alongside the table for reference. It is evident from the table 3.2 that Neo4j would be good choice for a database backend, due to its native graph support and rich interoperability features.

Table 3.2: Qualitative Comparison of Databases for storing the Data Model

Databases	Neo4j	Postgres	Elasticsearch	MongoDB	Apache Cassandra	Apache HBase
Type	Graph	Relational	Inverted Index	Document	Column Store	Column Store
Strict Schema	NO	YES	NO	NO	NO	NO
Full Text Search	YES	YES	NO	NO	NO	NO
ACID compliant	YES	YES	YES	NO	NO	NO
Relationships Discovery	EASY	HARD	MEDIUM	MEDIUM	EASY	HARD
Extensibility	YES	YES	YES	-	YES	NO
Complexity	MEDIUM	LOW	MEDIUM	LOW	HIGH	MEDIUM
Community Support	HIGH	HIGH	HIGH	HIGH	HIGH	LOW
Total Score (avg, -excluding Type)	2.28	2	1.71	1.57	1.85	1.28

Scores:
YES – 1
NO – 2
YES – 2
EASY – 3
HARD – 1
NO – 1
LOW – 3
MEDIUM – 2
HIGH – 1
HIGH – 3
LOW – 1

3.3 Summary

The tables 3.1 and 3.2 give a brief overview of the various technologies for building the data model and storing it. From table 3.1, it can be seen that the JSON data format can be a good choice for expressing the data model. The reason for choosing this data format can be further augmented by the fact that MITRE, VERIS and Splunk use the JSON format to express their data model. This research aims to extend MITRE's STIX 2.0[7] by adding new concepts that correspond to the OT environment, such as Links, Network Logs, Network Operations, Protocols, Host Change Logs and so on. The latest STIX framework [7] uses the JSON data format to express all the concepts and relationships. They were previously using XML, but switched to JSON since its much simpler than XML and human-readable. The work done by [15, 23] so far can also be used to define and identify concepts and their relationships in the security domain.

In order to store and analyze the knowledge graph of an OT security enterprise, Neo4j might be the best candidate for storing and visualizing it. With the Neo4j database, it is possible to click

[7] https://stix2.readthedocs.io/en/latest/index.html

on node and view all its relationships. In this way, relationship discovery is very intuitive and interactive, which makes it easier for user to explore the nodes and edges in a visually appealing and interactive manner. Moreover, the native graph engine allows for faster data processing and querying. Moreover, the Elasticsearch can be built on top of Neo4j for faster querying and advanced full text searches [20]. The following chapter 5 elaborates on the proposed solution and how the existing [26] framework can be extended to build the unified data model for the OT security domain.

Chapter 4

Overview of The STIX Data Model

This chapter provides an overview of the STIX data model, introduced in the previous chapter, that can be used to build the knowledge graph with concepts from the OT domain for threat intelligence purposes and information exchange.

The STIX data model has been developed by MITRE Corporation for representing CTI information in a structured manner [7]. The new STIX 2.x standard is completely written in JSON, which is claimed to be more "lightweight" and "developer-friendly" than XML [7]. The list of concepts and relationships available in the latest STIX data model are explained in the following subsection.

A detailed study of the STIX 2.x standard and their Python APIs[1] was undertaken to understand the STIX standard. The various STIX domain objects were studied to understand how new concepts can be created in STIX 2.x standard and how their attributes can be defined. Similarly, the cyber-observable objects explained the creation of parent classes, with subclass instances to create a hierarchy of concepts.

4.1 STIX Domain Objects

The STIX data model has 18 different data concepts with various attributes [26]. The concepts with their attributes and relationships are summarized in the list 4.1 below. The STIX 2.x data objects have their own JSON schemas that control the data type of their attributes and which attributes are required when instantiating a particular data object. The STIX data objects can be represented as the nodes of a graph and the relationship between them as the edges in the graph. Essentially, any STIX document can be represented as a graph. This visualization can be done using their own Visualizer module[2], using their open-source API. It is also possible to visualize the graph using other means, since the STIX document is written in JSON and can be transpiled to any programming language of our choice.

1. **Attack Pattern** - This concept defines an attacker's Techniques, Tactics and Procedures (TTP), usually containing external references to taxonomies such as CAPEC[3].

[1] https://github.com/oasis-open/cti-python-stix2
[2] https://github.com/oasis-open/cti-stix-visualization
[3] http://capec.mitre.org/

Example: Spear-phishing is an attack pattern where an attacker attempts to compromise a victim via a specially crafted malicious email.

2. **Campaign** - This concept describes an attacker's behavior aggregated over a period of time. It usually enlists the objectives of an attacker in the Intrusion Set concept explained below.

   Example: Campaigns can be used to describe the attack on ACME Bank during the summer of 2016 that aimed to gain secret information about its merger with another bank.

3. **Course-of-Action** - This concept describes the mitigation strategies for an attack. The definition is not yet complete in the data model.

4. **Grouping** - This object defines a logical grouping of security incidents observed in the network. It references other STIX objects that have a shared context

   Example: A Malware and Indicator object can belong to a grouping, if the Indicator of Compromise (IoC) signature matches a Malware's signature.

5. **Identity** - This object can be used to represent any entity/asset in the data model. It can describe an individual or an organization, their roles and so on

   Example: An Identity object can be used to represent victims of an attack

6. **Indicator** - This is the Indicator of Compromise (IoC) which is very important for threat intelligence. It can be used to identify malicious urls, files or blacklisted IP addresses in the system, that would be leveraged in a cyber attack.

   Example: An IoC can be a malicious file, identified by its hash, a URL, or a packet from a blacklisted IP address.

7. **Intrusion Set** - An Intrusion Set is a collection of an attacker's behaviour and his resources used for orchestrating one or more Campaigns.

8. **Malware** - Malware is the malicious software TTP used by an attacker to hack/disrupt an organization. It can be a trojan horse, worm, backdoor, etc.

   Example: Poison Ivy

9. **Malware Analysis** - This concept records the results of a static or dynamic analysis performed on a malware instance or family. **Example:** The analysis of an Unknown malware (file/tool) can result in it being labelled "malicious".

10. **Infrastructure** - This concept represents the physical and virtual resources or the TTP used by an attacker in an attack.

    Example: The command and control server communicated to by a malware/compromised asset.

11. **Location** - This object represents the geographic location of an entity.

    Example: The country, region or the latitude and longitude.

12. **Note** - This concept is to add extra metadata/context around another object, if needed.

13. **Observed Data** - This concept captures a wide range of cyber security artifacts/observables in the network and the system.

    Example: IP address, file, user-account, URL, file, network traffic and so on.

14. **Opinion** - This concept is used to record an analyst's views on a particular object in the system after analysis.

    Example: The listing 4.1 below shows an example of an analyst's opinion on a particular relationship.

Listing 4.1: An Opinion object in the STIX data model [26]

```
{
    "type": "opinion",
    "spec_version": "2.1",
    "id": "opinion--b01efc25-77b4-4003-b18b-f6e24b5cd9f7",
    "created_by_ref": "identity--f431f809-377b-45e0-aa1c-6a4751cae5ff",
    "created": "2016-05-12T08:17:27.000Z",
    "modified": "2016-05-12T08:17:27.000Z",
    "object_refs": ["relationship--16d2358f-3b0d-4c88-b047-0da2f7ed4471"],
    "opinion": "strongly-disagree",
    "explanation": "This doesn't seem like it is feasible. We've seen how
        PandaCat has attacked Spanish infrastructure over the last 3 years,
         so this change in targeting seems too great to be viable. The
        methods used are more commonly associated with the FlameDragonCrew
        ."
}
```

15. **Report** - This concept is used to aggregate and present a collection of STIX concepts as a CTI report of a particular focus area.

    Example: A story of an attack (Campaign) with attackers (Threat Actors), their TTPs (Attack Patterns, Infrastructure) and the victims(Identity) involved in the attack.

16. **Threat Actor** - The concept used to model an attacker, their goals, roles and sophistication levels.

    Example: An APT group such as "Equation Group" behind the Stuxnet attack.

17. **Tool** - A legitimate software resource used by an attacker for carrying out their attack.

    Example: An Nmap scanner, Metasploit, etc.

18. **Vulnerability** - The weakness exposed by an Identity in the system, which can possibly be leveraged in an attack against the Identity.

    Example: Contains Common Vulnerability Exposures (CVE) references with links to the National Vulnerability Database(NVD) and its details such as the Common Vulnerability Scoring System (CVSS), vendor, summary, etc.

19. **Relationship** - The relationship linking two concepts is expressed as a concept in the STIX data model, thus decoupling the concept and its relationship dependency. In this way, the data model allows for dynamic creation and revocation of relationships between concepts.

    Example: The following example 4.2 shows the relationship between an Identity and its Vulnerability. It lists the source, target and the name of the relationship.

Listing 4.2: A Relationship object in the STIX data model

```
{
   "type": "relationship",
   "spec_version": "2.1",
   "id": "relationship--d861f5af-0f25-4fc6-95ae-ea7c844f5e13",
   "created": "2020-06-18T17:35:48.137Z",
   "modified": "2020-06-18T17:35:48.137Z",
   "relationship_type": "exhibits",
   "source_ref": "identity--5316a6e2-358f-428d-b8fa-7dc2699989a8",
   "target_ref": "vulnerability--73231d13-7612-4c9a-a2fc-6594a20cf494"
}
```

4.1.1 Attributes and Relationships

As can be seen from the above list 4.1, all concepts have the `type, id, created, modified` attributes created by default which are the metadata describing the concept. The Relationship is a separate concept in STIX and it can be created using the `id` attribute of the source and target concepts. This makes it easy for us to de-reference the relationships, thus allowing relationships to be dynamic in the data model.

Considering this new STIX 2.x data model, it can be extended with concepts and relationships from the OT security setting. This research aims to extend the work of [29] for the OT domain as a whole that contains vulnerabilities, alerts, network logs, devices, protocols, links and so on. For this purpose, the data provided by Forescout Technologies was used to test and validate the novel data model and the resulting knowledge graph.

4.2 Integrating Raw Data with the STIX Data Model

The STIX data model provides a way for security experts to represent CTI data in a standard format for storage and exchange. It does not however offer a method to integrate raw cyber threat data from different data sources in an enterprise. In order to use this data model, it is essential to devise a methodology to integrate and validate raw cyber threat data against the data model and provide a complete STIX document that can then be shared across the enterprise, or stored in a unified data store for analysis and visualization.

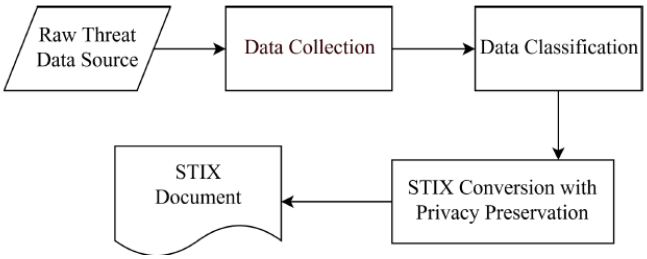


Figure 4.1: The STIX Automation Workflow of [29]

The authors of [29] have devised a novel methodology to incorporate privacy in STIX, in addition to achieving the above-mentioned goal for network traffic data. Their workflow is illustrated in figure 4.1 above. A Data Collector module gathers data from different sources and

format and stores them in a buffer, while assigning a tag to it, that would later be used for classifying the data. The Data Classification module processes this tagged data to match the different elements of the data to their corresponding STIX Data - Cyber Observable Objects (COO) in their case. Finally, the STIX Conversion module organizes these mapped elements as a STIX document which is standard and suitable for sharing information. The authors of [29] refer to this process as STIX automation - the automatic process of generating standard STIX Documents from raw cyber threat data.

However, the authors of [29] do not validate the STIX data objects to check if it has valid relationships and attributes. The architecture in the next chapter 5 explains in detail the flow of data from the raw data sources in an enterprise to the final unified data store, which is the Neo4j graph database. The architecture of the UDM prototype also includes how data is transformed at various stages in the pipeline.

Chapter 5

The Unified Data Model for CTI in OT Networks

The previous chapters focused on comparing the existing data models for cyber-security and contrasted them against one another. This chapter elaborates on how the proposed Unified Data Model (UDM) can be leveraged to address the research question mentioned in 1. The quality of a data model can have a major impact on the quality of investigation of security incidents by the security analyst. Given a data dump from the IDS, the UDM maps all the information from this huge dump to the different concepts in the UDM, and creates relationships between them accordingly to generate a "knowledge graph", like [23] does. This highlights the expressiveness of the data model from the ideal features of the UDM prototype in section 5.1. The standardization of the data emphasizes the consistency of the UDM, as envisioned in section 5.1.

In particular, the STIX Data Model, explained in the previous chapter, has been extended with concepts such as Devices, Alerts, Host Change Logs and so on to accommodate OT-IDS specific entities and express them efficiently. The following sections explain how this purpose can be achieved. This extended data model may also be used to map concepts and relationships of different IDSs, which can then be used to identify and mitigate potential threats in different organizations and domains. Thus, it can allow for proactive monitoring of risks and the development of defensive strategies to face the threat.

5.1 Ideal UDM Prototype

This section discusses the ideal features of the UDM prototype, which will serve as the criteria to assess the quality of the data model in chapter 6. Some of the main criteria the UDM must achieve are expressiveness, consistency, performance and usability, which are explained in detail below.

1. **Expressiveness -** Unify and represent the IT and OT concepts such as devices, engineering workstations, users, network logs, host change logs, links, protocols and so on
    - For this we would like our data model to be able to categorize the raw data in OT enterprises to appropriate concepts and relationships.
    - Example: There is an "Operations" concept that captures the network operations in the organization. This concept will classify and capture the type of network operation as a file read, file write, authentication operation, file access error and so on to

represent the raw network operation information with structure and semantics. The "file-read" will also include the file that was read, the timestamp and other necessary data. It will be linked to the "User" who read the file and is responsible for this network operation. In case of any discrepancies later, this user shall be investigated. Thus, quite simply, this data model is meant to capture the concepts and link them together in a structured manner from raw data sources that can be either structured (like SQL tables) or unstructured (live streams).

2. The second use case will be its ability to express threats and perform risk assessment according to ISO 27001 or NIST 800-53 cyber-security standards[1]. These standards define frameworks for performing risk assessment in an enterprise, evaluate the risk and implement policies to mitigate and reduce the impact of the risk.

   Threats are a part of every organization. This data model will govern the structure of the enterprise's data. Thus, from this data, an analyst will be able to explore the vulnerable devices, their risks and impact on the organization. Additionally, they can assess and investigate how a vulnerable device can affect the normal operation of other devices. In other words, they can view the path from one device to another via a vulnerability, risk, alert and so on. This is useful because it encourages proactive threat analysis and not strategic post-threat analysis, like the VERIS Data Model, that can only be used for forensics. The UDM should be able to relate concepts that are not explicitly linked, such as devices with the same vulnerability, alerts raised by devices, the type and impact of risk a vulnerability can have on the organization and the devices with these vulnerabilities. The path from a compromised device to a normal operation device (one with no vulnerabilities) can also be analyzed if the data model captures these concepts and relationships.

3. **Consistency -** Standardize the structure of CTI data for information exchange and knowledge discovery.

   Contribute to the CTI community by extending the data model to support OT concepts for cyber-security. Currently, all existing data models only support the IT domain, and the convergence of IT and OT infrastructures raises the need for a unified standard data model to support both these infrastructures. Moreover, this data model should ensure data follows a certain schema, which has well-defined data types and relationships across the different concepts in the data model.

4. **Performance -** The data model must be rich and powerful such that it can reduce the query complexity and response times for a security analyst querying the graph database.

   Example: In Forescout's IDS, they did not have a data model, which introduced a lot of performance issues. For instance, Alerts were not linked to the Devices involved. Rather, the Alerts were linked to an IP Address concept which was then linked to the Device with that IP Address. Thus the path from a device to an alert was `Device-->IPAddress-->Alert`. This increased the query response times in datasets with more than 100,000 alerts. A device has a single IP Address and thus the IP Address could have been embedded as an attribute of the device, thereby linking devices and alerts directly, instead of looking up the IP Address of the source and destination device for each and every alert.

5. **Usability -** The data model should not be IDS specific or technology-specific (IT/OT), rather it should be designed to be used with any IDS irrespective of the IT/OT data

[1] `https://www.nist.gov/cyberframework`

collected by the IDS. Overall, the data model should contribute to a bigger CTI community and not be bound to any product.

Additionally, a minor improvement from the usability point of view is to have the concept and relationship names and attributes in lower-case, since most query languages are case-sensitive and in this way, it is easy for the security analyst to query without having to worry about case sensitivity of the underlying data.

5.2 Methodology

The STIX data model explained in chapter 4 introduces the different concepts in it and an example workflow to integrate raw data and transform them as STIX objects for information exchange within an organization. This section introduces the methodology adopted for creating the proposed UDM by extending the STIX data model with new concepts to support OT-IDS data and support threat intelligence in OT networks. Prior to the transformation of raw data, the mapping between the OT security concepts and the existing concepts in the STIX data model must be determined. This is done to avoid re-inventing the wheel and reusing the existing STIX data model as much as possible. The following figure 5.1 highlights the high-level methodology used to define and create the Unified Data Model (UDM). Each step of the methodology is explained in detail in the sections below.

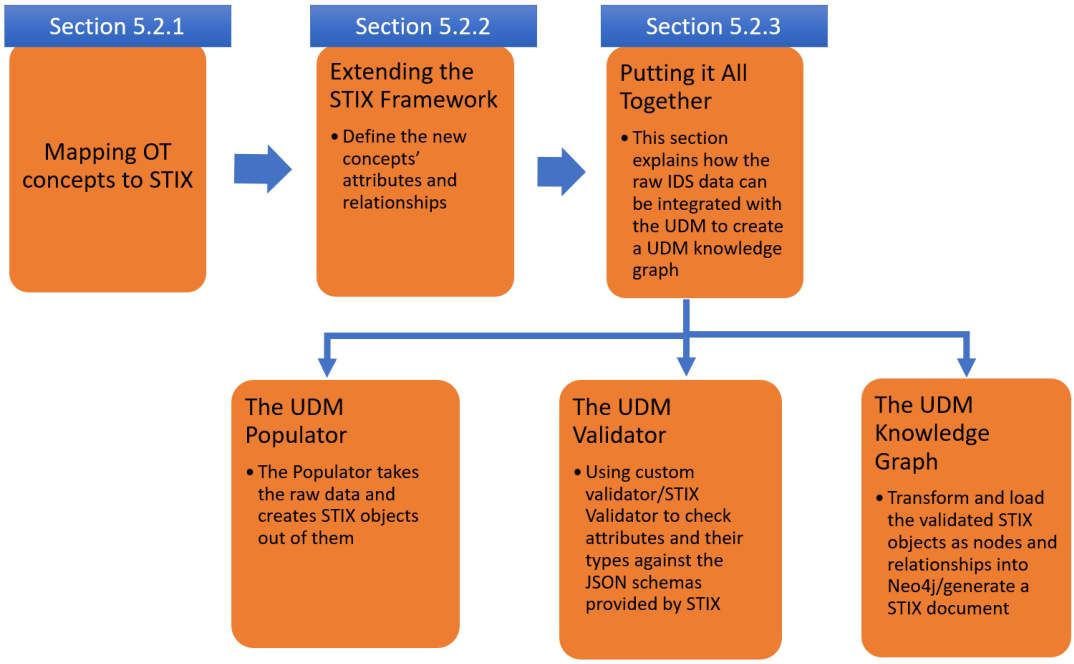


Figure 5.1: The high-level methodology for building the Unified Data Model

5.2.1 Mapping OT Concepts to STIX

This was the first step towards developing the UDM for OT incident investigation purposes. This research has been done in collaboration with Forescout Technologies. Forescout provided all the

necessary infrastructure and support for creating this unified data model and a platform to test the data model with real-time OT security data in order to validate it.

The different concepts present in an OT enterprise can be summarized in the following table 5.1. These concepts were derived from the data present in Forescout's IDS and inspired from other data models that were reviewed in the literature. The "Device", "Alert", "Network Operation", "Host Change Log" and "Risk" were some important concepts that were not present in most of the data models described in the literature in chapter 3. Nevertheless, these concepts are paramount in an OT domain, since it consists of several physical and logical workstations, controllers and other components that are vital to the normal functioning of an OT industry. These industrial devices and the operations they perform are captured as network logs in the Network Level and Change Logs in the Device Level. This is done to give a complete picture of a device, its role in the network and the activities that it performs. If any of these activities are identified as suspicious in the network, an alert is generated that indicates the type of suspicious activity and the devices involved in it. Thus, with this information, the following table 5.1 presents the different concepts that will be part of the proposed UDM.

Table 5.1: Concepts and Relationships in an OT environment

Concepts	**Attributes**	**Out-Bound Relationships**	**Purpose**
Device	ip_address, mac_address, mac_vendor, name, id, labels, vendor, firmware_version, os_version, purdue_level, role	functions-as ("Role"), uses-client-protocol ("L7 Protocol"), uses-server-protocol ("L7 Protocol"), exhibits ("Vulnerability"), exposes ("Risk")	The device object represents the items in the enterprise's asset inventory.
HostChangeLog	id, old_value, new_value, information_source, event, name, timestamp	affects ("Device") originated-from ("User")	This object captures the event where a device changes its IP Address
L7Protocol	id, name	Nil	This object captures the application layer protocols such as HTTP, DNS (IT) and MODBUS, STEP7, and other proprietary OT protocols

Continued on next page

Table 5.1 – continued from previous page

Concepts	Attributes	Out-Bound Relationships	Purpose
Link	first_seen, last_seen, rx_bytes, id, ports, tx_bytes, proto	from-device, to-device ("Device"), uses-protocol ("L7 protocol")	This object captures the network flow information between two devices in the internal network. It captures the source & destination ports, bytes exchanged and the cross-network flow between devices in different Purdue Levels.
Risk	name, id, risk_value (enum: Low, Medium, High, Critical), risk_label, likelihood, impact, description	Nil	The Risk object represents the operational risk that a device poses in the enterprise. It is calculated using the vulnerabilities device criticality and other attributes of the device that can contribute to the risk.
Role	id, name	Nil	The role that a device plays in the system - web-server, database-server, etc.
Vulnerability	name, id, vendor, published_datetime, cvss	Nil	The vulnerability information for a device
Alert	severity, event, name, src_port, dst_port, id, status	uses-protocol ("L7 Protocol"), from-device, to-device ("Device") exploits ("Vulnerability")	The alert raised by the SIEM tool in the enterprise. This alert will be investigated by the SOC analyst to determine if it is a false positive or not.

Continued on next page

Table 5.1 – continued from previous page

Concepts	Attributes	Out-Bound Relationships	Purpose
Issue	id, name, description, object_refs	comprises-of (“Event (alert/networklog/ hostchangelog)”)	This a grouping of events such as change logs, alerts and network operations. Similar events are linked together as a single issue. and
User	user_id, id	logged-in, associated-to (“Device”) located-in (“Location”)	The username of the user of a particular device in the enterprise.
Network Operation	id, name = (Device Operation/File Operation/ Authentication/ Encryption/Name Resolution), count severity, objects, object_refs, dst_port	from-device, to-device (“Device”) uses-protocol (“L7 Protocol”) originated-from (“User”)	This concept captures the network operations of each device such as file creation deletion, device reprogramming name resolution, authentication success or failure.
Indicator	pattern, pattern_type (stix2 patterns, yara rules, snort rules), id indicator_type	triggers (“Alert”) triggered-by (“Device”)	The Indicator of Compromise (IoC) represents the malicious urls, files or blacklisted IP addresses in the system, that would be leveraged in a cyber attack.
Location	region, country, id	Nil	The location object represents the geographic location of a user, if present in the data captured by the SIEM tool.
Attack Pattern	kill_chain_name, id, kill_chain_phase (Tactic), url, external_id, name (Technique), description (Procedure)	detects (“Alert”) used-by (“Device”)	This concept defines an attacker’s Techniques, Tactics and Procedures (TTP), usually containing external references to taxonomies such as CAPEC[2]

Some of these concepts can already be mapped to the existing STIX objects, such as Vul-

nerability, Attack Pattern, Protocols, etc. The Vulnerability STIX object can be extended by adding new attributes to include those mentioned in the above table 5.1. The Identity object in STIX can represent an individuals, organizations, institutions and so on. It can be modified to include the L7 Protocol, which is an application layer protocol, and Role concepts with the identity_class attribute set to protocol , role respectively.

The complete mapping between the OT security concepts and the existing STIX objects are given in table 5.2 below.

Table 5.2: Mapping OT concepts to STIX objects

STIX Objects	**OT Objects**	**New Attributes to support OT Objects**
Vulnerability	Vulnerability	cvss, published_datetime, vendor
Identity	Role	identity_class=”role”
Identity	L7 Protocol	identity_class=”protocol”
Network Traffic	Link	used as it is
Grouping	Issue	used as it is
Indicator	Indicator	used as it is
Location	Location	used as it is
Attack Pattern	Attack Pattern	used as it is
User Account	User	used as it is

5.2.2 Extending the STIX Framework

The rest of the concepts from the above table 5.1 can be created as new STIX objects by extending the STIX 2.x framework. These new concepts can be created using the Python programming language, since STIX offers Python APIs with extensive documentation to understand and operate with their data model. A script was developed to automate the generation of new STIX concepts with a minimal configuration needed.

The listing in 5.1 shows how the device concept can be instantiated in Python and the output produced by the `stix2` library. The `stix2` library outputs the newly instantiated “device” concept in the JSON format.

Listing 5.1: Creating an instance of the device class in stix2

```
device = Device(name="Device#1", ip_address = "0.0.0.0", mac_address="
    00:00:00:00:00:00", purdue_level=0, vendor="Cisco", mac_vendor="unknown",
    firmware_version="0xc", os_version="Windows XP")

print(device)

{
    "type": "device",
    "spec_version": "2.1",
    "id": "device--492778e7-c5a1-4d0c-a72d-f9f0fcc05c9c",
    "created": "2020-03-20T12:34:56.926Z",
    "modified": "2020-03-20T12:34:56.926Z",
    "name": "Device#1",
```

```
    "firmware_version": "0.0",
    "vendor": "Cisco",
    "os_version": "",
    "mac_vendor": "Unknown",
    "ip_address": "0.0.0.0",
    "mac_address": "00:00:00:00:00:00",
    "purdue_level": 0
}
```


In this manner, all the OT specific concepts mentioned in table 5.1 were added to the STIX Data Model. The UDM prototype was generated as a JSON file containing all the concepts and their relationships together as one JSON Bundle. A JSON Bundle is a STIX Bundle which wraps the different concepts and their relationships as a single bundle[3]. The following picture 5.2 shows the Unified Data Model visualized using the STIX Visualizer.

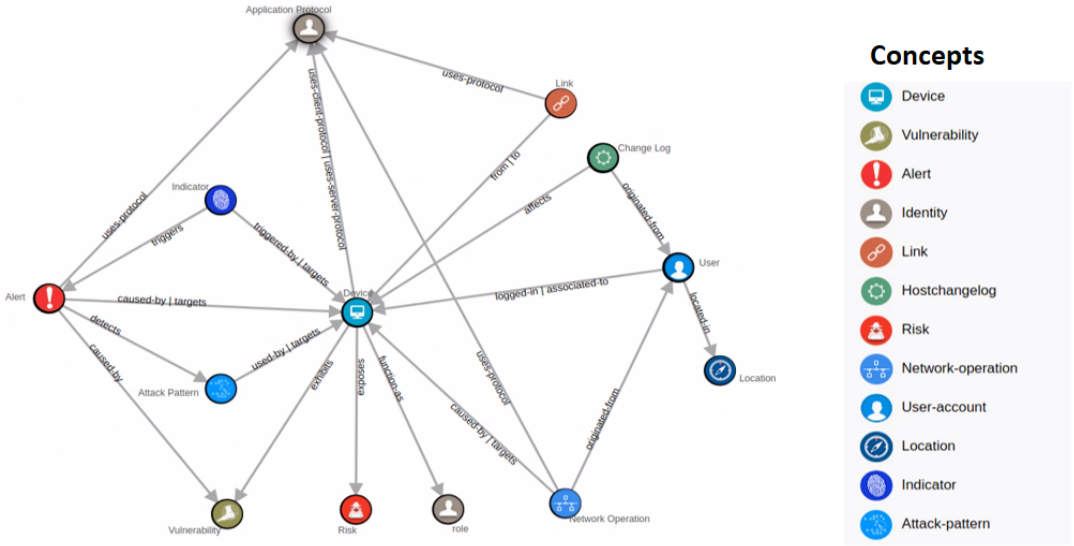


Figure 5.2: The Unified Data Model as visualized using the STIX Visualizer

5.2.3 Putting it All Together

This section explains the architecture used for integration raw data from the IDS such as logs, asset inventory, vulnerability databases and so on, with the proposed UDM prototype in figure 5.2 to create a complete knowledge graph. The following sub-sections explain each module in the diagram 5.3 in detail.

[3] https://stix2.readthedocs.io/en/latest/guide/creating.html#Creating-Bundles

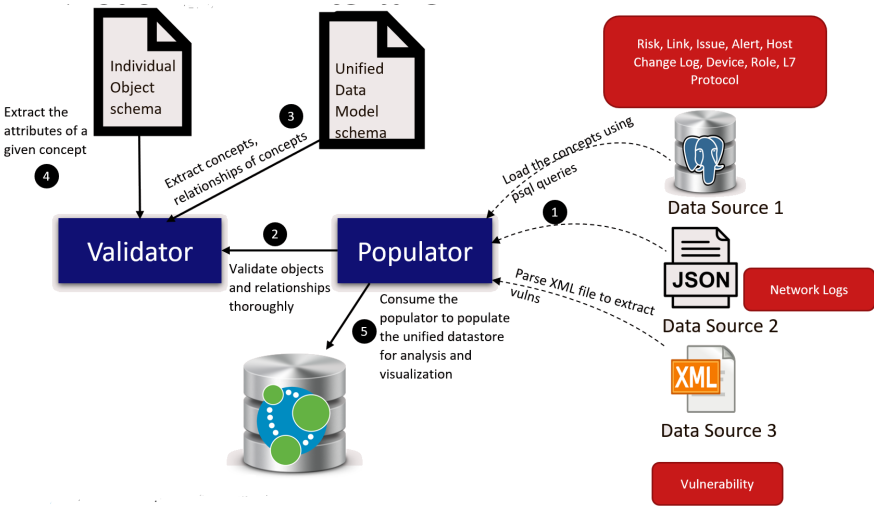


Figure 5.3: The architecture pipeline of the Populator module for populating the various STIX concepts as per the UDM

The UDM Populator

The UDM Populator module handles the creation of STIX objects from raw data from the different data sources. The Population Pipeline is shown in the figure 5.3 above. As shown in the figure, the populator ingests data from three data sources - SQL Databasess, JSON File and XML Files on disk. These data sources contain different concepts collected from Forescout's IDS. They are stored in a raw format aimed at high performance and low readability. The concepts from these data sources are fetched, populated as appropriate STIX concepts using the STIX Python API's and other handlers for interacting with data from SQL and XML. All these entities, relationships and their attributes are validated in the next step as explained below in the next subsection.

The UDM Validator

The UDM Validator is a custom module for validating the UDM concepts and their relationships defined above. There are JSON schemas dictating the list of attributes and their datatypes that a concept must have in the data model. While creating these concepts, STIX internally validates the concept's attributes and their types. However, STIX 2.x does not validate the relationships between the concepts. The present UDM Validator module validates all concepts, their attributes and relationships using the `jsonschema` library in Python.

A parent JSON Schema called `unified-data-model-schema.json` was developed that contained the list of allowed concepts and relationships between those concepts. The following figure 5.4 shows that this schema contains the concepts as a list and for each concept, the list of concepts it can be related to, along with the name of the relationship. As shown in the example, a "Device" concept can be related to a "Risk" concept via the label "exposes". Thus, all these attributes of the relationship between concepts are also validated in this module. If a relationship does not exist in this parent schema, then it is not possible to create it. Thus, the data is strictly governed using this schema, which can then be extended to accommodate data from different IDS'.

Figure 5.4: UDM Schema containing concepts and relationships between them

Using these methods, the STIX concepts were validated against the UDM schema and thus were ready to be generated as a STIX document for sharing CTI information in the enterprise or generate a report of security incidents. These objects were also transformed and loaded into the Neo4j database for interactive querying and visualization for analyzing security incidents.

The UDM Knowledge Graph

The UDM Knowledge graph is expressed in the graph data format in Neo4j as nodes and edges as shown below. The following picture 5.5 below shows the database's schema visualized in Neo4j. It can be observed that this schema matches the UDM illustrated in 5.2 above. Thus, the data in the unified data store - Neo4j, adheres to the STIX standard (as validated by the UDM Validator) and contains real-time data from an OT network.

Figure 5.5: The UDM Knowledge Graph schema as viewed in Neo4j representing 11 concepts and 21 different relationships in the Stuxnet Dataset.

Exploring this knowledge graph becomes easier with Neo4j, as we can go from one concept to another and explore the different relationships between concepts to understand data connectivity. The semantics of the data can be inferred by expanding each node to view its attributes. The

use of this knowledge graph for satisfying our use-cases mentioned in section 5.1 will be discussed in detail in the upcoming chapters.

5.3 Summary

This chapter explained the creation of the data model, focusing on its implementational and technical aspects. Moreover, the data-driven methodology used for creating this unified data model was explained in detail, highlighting the contribution of the data model to the CTI community. The UDM is designed to unify the IT and OT data of an enterprise by extending the STIX data model, which was more IT-oriented. Extending this pre-existing data model ensures that this new UDM conforms to industry and community standards, while enhancing the features that the pre-existing model offers, satisfying the Consistency criteria mentioned in 5.1. Furthermore, the goal is not to re-invent the wheel, rather focus on improving community standards for wider coverage as it is easier to learn and adopt by enterprises.

Chapter 6

Application of the Unified Data Model

This chapter discusses the case studies developed for evaluating the quality of this UDM and reviews the results of the same. Moreover, this chapter also discusses how the UDM can be used with another IDS to organize its IT data as a knowledge graph. Finally, this chapter presents the results of evaluating the UDM Populator module mentioned in section 5.2.3 detailed in the previous chapter.

6.1 Case Study on the Stuxnet Malware

This section explains how the UDM can be used to organize the Stuxnet incident data and describes the different concepts and relationships present in Stuxnet. Using this knowledge graph, generated by validating the raw Stuxnet data against the UDM, the analyst can perform the security incident investigation.

This case study is an attempt to evaluate the quality of the UDM and review the advantages and limitations of the data model for future improvements.

6.1.1 About Stuxnet

The following figure 6.1 shows a schematic illustration of the Stuxnet Malware Attack, which is explained in detail below.

Stuxnet is a highly complex and advanced computer worm that was spread via rogue USB drives connected to Windows Machines in the Nuclear Plant in Iran. It required almost zero user interaction for it to begin exploitation. It exploited some zero-day vulnerabilities in specific Windows XP workstations in the network. Only if the configuration specified in the Stuxnet Malware matched that of the target host, it infected them. Otherwise, it was dormant and silently scanned the network for potential targets. Once it found a target, it downloaded a copy of itself and updated its signature to masquerade as a legitimate Windows executable program. It then spread from this machine and moved laterally across the network infecting other workstations. All the infected workstations were part of a Peer-to-Peer (P2P) Malware Network, communicating with each other to update the latest version of the malware amongst themselves offline. Thus, there was no central Command and Control (C&C), which if taken down could have purged the malware communication in the network. Each infected device acted

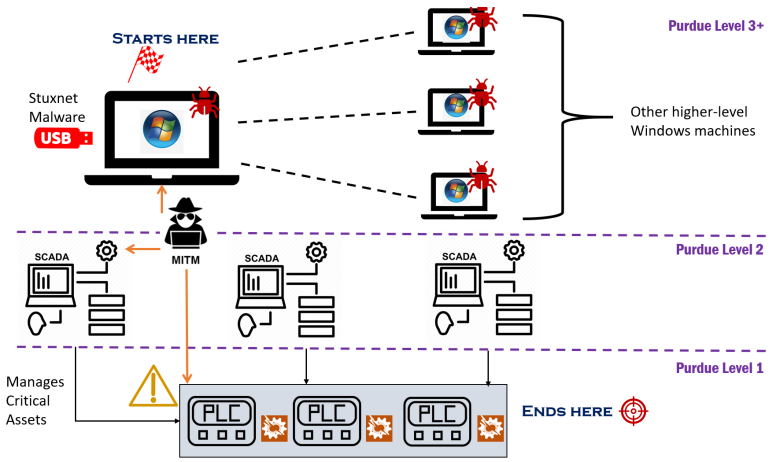


Figure 6.1: The Stuxnet Malware Attack

as it own C&C and thus it was difficult to shut them down completely. Once the P2P network was complete, they went on to infect the critical assets – the low-level slave PLCs that were responsible for physical tasks such as centrifugation, cooling the combustion plant and so on. These were critical assets because of the operations they performed, and the risk factor associated with it. Any change in their operational parameters such as fan speed, temperature can lead to a catastrophic explosion in the ICS plant. The Stuxnet Malware targeted these PLCs and reduced their frequency by a few Hertz for a while, then reverted it back to normal and remained dormant for a couple of weeks only to repeat the whole process again. It falsified the information sent by the PLCs to their master SCADA systems via a simple Man-in-The-Middle (MiTM) attack [36]. Thus, the master SCADA systems were unaware of these abnormal reconfigurations to the PLCs. Stuxnet Malware caused a very slow, yet powerful degradation to the Nuclear Plant in Iran and was a revolutionary large-scale ICS attack that inspired several such attacks even till date [38].

6.1.2 Technologies Used

- Neo4j Graph Database for querying and visualization
- Python 3.7 for creating STIX objects out of raw data from different data sources
- JSON data sources for Network Logs, IoCs, Vulnerabilities and so on.
- SQL Database containing Stuxnet devices information, alerts and so on.

6.1.3 The Stuxnet Data Model

The various concepts available in the Stuxnet Dataset are:

- Network Operation
- User
- Indicator of Compromise

- Link
- Device
- Alert
- Host Change Log
- Vulnerability
- Role
- Protocol (Application Layer)
- Attack Pattern

These 10 concepts are linked to each other as specified in the following figure 6.2. The task is to now classify the data from the Stuxnet Attack into the above stated concepts and highlight the usefulness of this structured data in analyzing the Stuxnet Incident. The Data Model captures and relates concepts as STIX objects expressed in the JSON format. These STIX objects can also be collected in a JSON document that can then be shared across the enterprise/organization. Moreover, this data model extends STIX framework by introducing new concepts for alerts, devices, logs, protocols and so on, in order to express low level details for efficient analysis by a security incident investigator.

Preparation

This preparation phase refers to the population of the Stuxnet data using the UDM. The following figure 6.2 summarizes the steps involved in the preparation phase, which are detailed below.

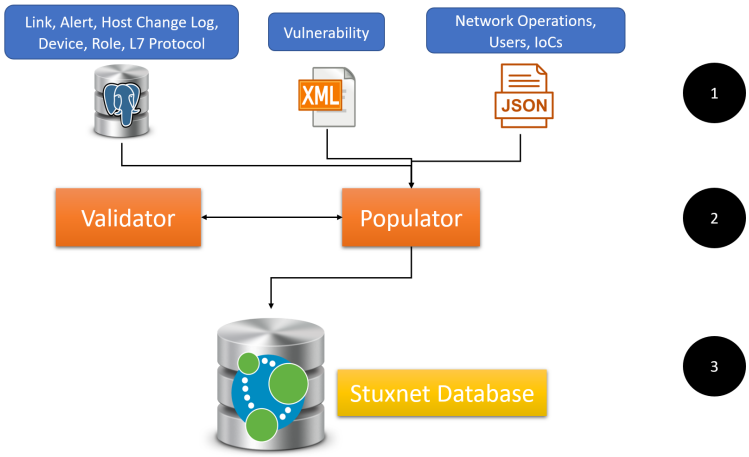


Figure 6.2: Populating the Stuxnet Data for Incident Investigation

The first step is to integrate and categorize the raw data from the different data sources as STIX objects using the Python APIs. The UDM Populator module mentioned in chapter 5 performs this task. Once the STIX objects are created, they are validated against their JSON schemas to check a concept's allowed relationships and its attributes. Later, these validated

objects are pushed to the Neo4j graph database as nodes and edges to visualize the entire Stuxnet incident as a graph with a schema in figure 6.3 as follows. It can be observed that this above schema is consistent with the UDM and thus we can say that the raw unstructured data has been standardized and structured for investigation.

Figure 6.3: Stuxnet Data Model

6.1.4 The Investigation of Stuxnet

The investigation is the second most important phase, which illustrates the effectiveness of the data model. In this phase, we analyze the above data to identify infected assets, the attack paths (if any), the point of infection and so on.

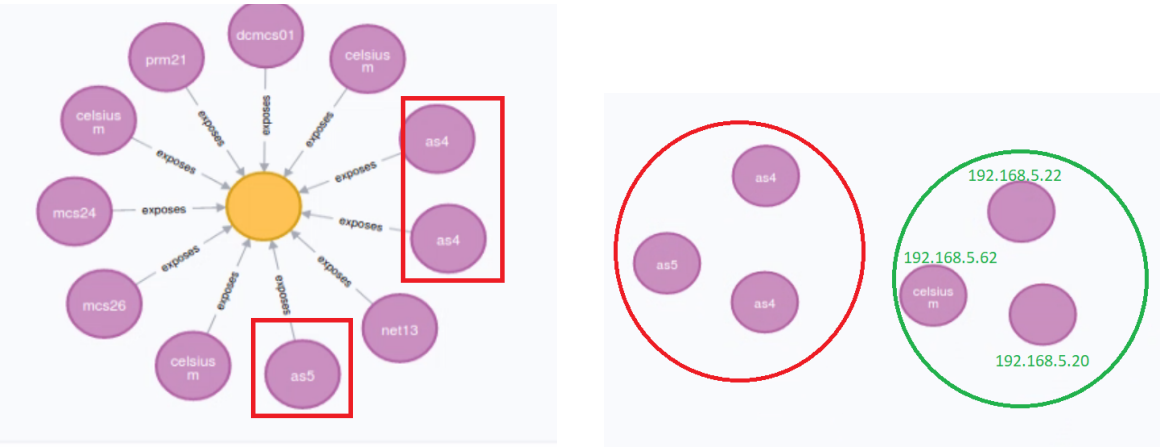
In order to evaluate the UDM prototype we choose to focus on 3 important aspects of the Stuxnet Malware data:

1. Investigating the critical assets in the network
2. Investigating the Point of Malware Infection
3. The Man-in-The-Middle Attack

Investigating the critical assets in the network

The analysis began with identifying the high-risk assets in the system. These assets ("devices") are critical to the organization and have to be protected from malicious activity. As seen from figure 6.4a, some of these devices were identified as PLCs (highlighted in red in the figure) in Purdue Level 1, which are not connected to the internet and perform tasks such as cooling and centrifugation in an ICS Plant.

The high-risk PLCs were also found to have certain vulnerabilities, shown above in 6.4b, with CVEs from the year 2018, which indicates that they might not have exhibited these vulnerabilities during the Stuxnet Attack. The PLCs were slaves reporting their operation progress to the Purdue Level 2 master SCADA systems. The PLCs were Siemens products running the Siemens STEP7 protocol which was the target of the Stuxnet Malware. The six devices (3 masters and 3



(a) Devices with High Risk Factor

(b) Vulnerable Devices in the Network

Figure 6.4: Devices with High Risk and Vulnerable Devices in Stuxnet

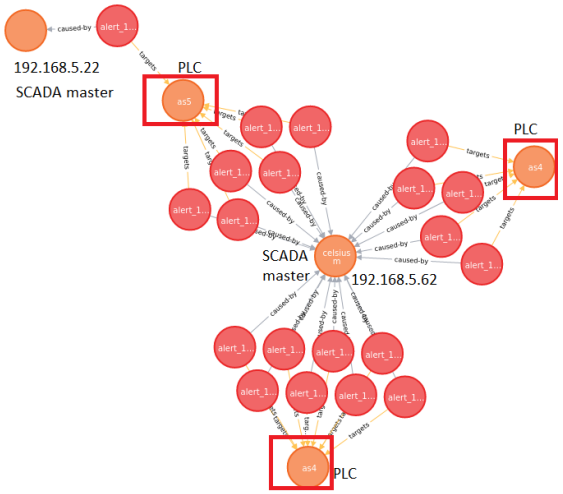


Figure 6.5: PLCs performing dangerous operations commanded by Master SCADA systems

slaves), in figure 6.4b were all observed to have vulnerabilities and involved in several dangerous device reconfiguration operations. This is shown in figure 6.5 below.

This confirms the Modus Operandi (MO) of the Stuxnet Malware that regularly disrupted the normal functioning of the critical PLCs. However, these device reconfiguration commands seemed to be issued by the Master SCADA devices themselves. This can be attributed to the successful MiTM attack performed by the Stuxnet Malware to masquerade as master to the slave and vice versa [36]. Thus, the Stuxnet Malware had been disrupting the normal operation of the critical assets in the network. This has been confirmed in our analysis using the data model.

The goal of this analysis was to start from the high-risk critical assets and trace the path to the Point of Infection (POI) of the Stuxnet Malware. The next step of the analysis was to identify the malware infected assets and discern the POI amongst them.

Investigating the Point of Malware Infection

The following figure 6.6 shows the malware-infected devices in the network, and highlights two important devices that were prominent in the network. Prominence here refers to the number of connections these devices had with other entities in the network.

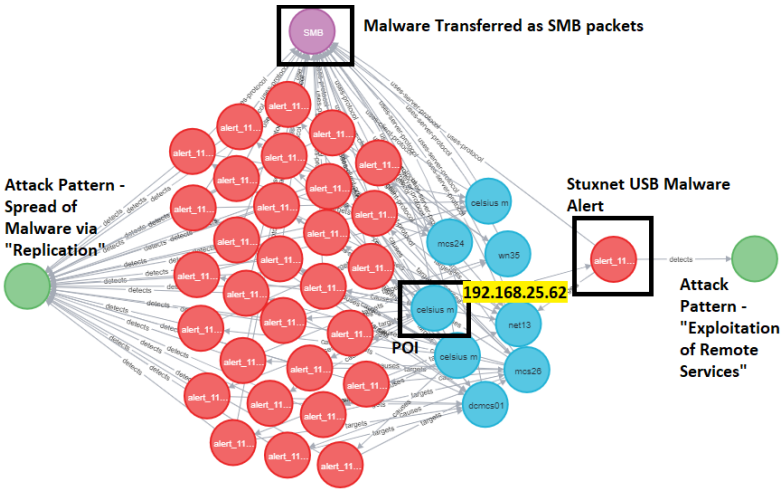


Figure 6.6: Point of Malware Infection in Stuxnet

The POI was a Windows XP machine in the Purdue Level 3. This device was the one in which the Stuxnet Malware was injected via a malicious USB drive. It was identified using the Alert event in the network, which recognized this device as trying to add another device as malware pawn. From these two devices, the malware recursively scanned and spread across the whole network infecting all Windows XP machines, via the DCOM (Remote Procedure Call for Windows) and SMB payloads, installing and updating itself, thereby creating a full-fledged malware P2P network. These infected devices, shown in figure 6.6, communicate and collaborate with each other to keep themselves updated with the latest malware signatures offline.

Tracing the Path from the Malware to the Critical Assets

As shown in figure 6.7 below, a series of links led us from `192.168.25.104` a malware infected device to `192.168.5.62` that initiated dangerous device reprogram operations. The malware-infected `192.168.25.104` seemed to have been targeted by an IoC, which is also linked to two devices in the `192.168.5.1/24` subnet. Further inspection into this IoC indicates that it was triggered by `192.168.25.62` and `192.168.5.162` and targeted `192.168.25.104`, `192.168.5.105` respectively.

The `192.168.25.104` seems to have been linked to `192.168.5.162` and `192.158.5.105` via links and an attack pattern that indicates this there might be an attempt to communicate to the command and control device. This is precisely what happened. `192.168.5.162` has been trying to communicate to the same internet blacklisted domain name (flagged as an IoC) as `192.168.25.62` and has also been in contact with the infected `192.168.25.104`. Thus, it can be assumed that both `192.168.5.162` and `192.168.5.105` are also infected with the Stuxnet malware.

One of these devices mentioned above might have impersonated as the master to send rogue commands to the PLCs. This is investigated in detail below.

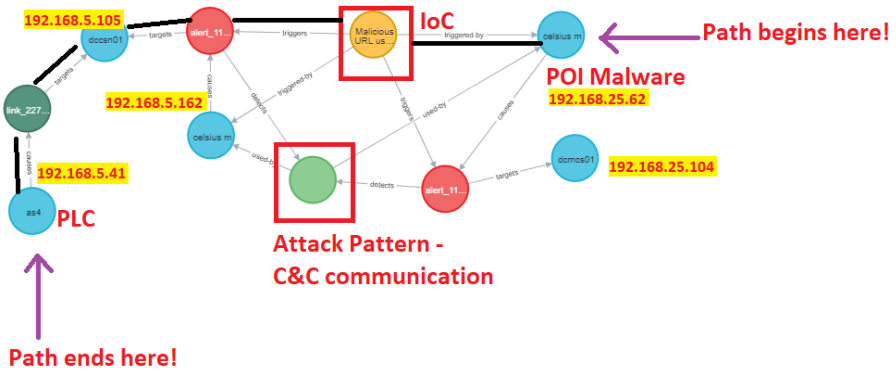


Figure 6.7: Path from Malware-Infected Devices to the PLCs

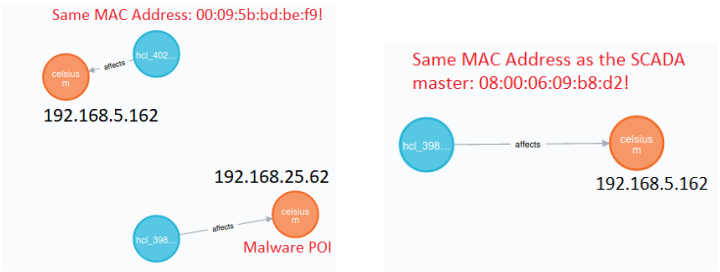
The Man-in-The-Middle Attack

The POI was also found to have several change logs pertaining to changing its MAC address and names to impersonate other devices in the network indicating possible MiTM attacks. This is shown in figure 6.8a below.

Moreover, when `192.168.5.62`'s (the device that allegedly sent dangerous reprogram commands to the PLCs) MAC Address was investigated using full-text search in Neo4j, cross references to `192.168.5.162` were found. This device impersonated as the master (`192.168.5.62`) and sent malicious device reprogram commands to the PLCs, shown in figures 6.8b and 6.8c. This device also triggered an IoC to be raised for trying to access/send a malicious URL to another device, shown in 6.7. The DNS request was observed to have originated from the POI device for the same domain. It is most likely that these two devices were trying to access some remote unknown C&C. It can be said with confidence thus that the `192.168.5.162` device might also be a malware-infected device sending out rogue device reconfiguration commands to the PLCs via `192.168.5.105` (also a Windows XP machine) and as an MiTM attacker.

6.1.5 Summary

Thus, this concludes the analysis of the Stuxnet Attack using the Unified Data Model. The analysis successfully confirmed the facts of the original attack on the Iran ICS Plant. This implies that the UDM did indeed capture the concepts of the attack well and express it a comprehensive and detailed manner for meaningful investigation of security incidents.



(a) (b)

(c)

Figure 6.8: Malware POI Impersonating a SCADA Master Device

6.2 Population Performance of the UDM

This section presents the time taken by the UDM Populator mentioned in chapter 5 for different datasets provided by Forescout. The time taken to populate each concept in the UDM is also presented to understand which concepts hamper the overall performance.

The following table 6.1 shows the time taken to populate each dataset as a knowledge graph satifying the UDM with all its concepts and relationships. A figure 6.9 summarizing this table is also given below. In the figure, the size of the circles correspond to the number of alerts in the dataset, since alerts dominate in size and importance in every IDS. The range of color represents the time taken to populate the knowledge graph, with the lightest circle corresponding to the dataset that takes the least amount of time and vice versa. Naturally, the bigger circles (one with the greatest number of alerts) are darker, implying that the alerts take the longest to populate.

This is confirmed in the following figure 6.10 that shows the time taken to populate each and every concept in the UDM for different datasets. It can be observed that the concepts that take the longest to populate, in order are Alerts, Network Logs, Indicators, Links, Host Change Logs and Devices. Therefore, the concepts that involve a source and destination relationship with a Device take the longest time to populate. This can be attributed to a large number of disk reads to retrieve the devices and populate the relationships.

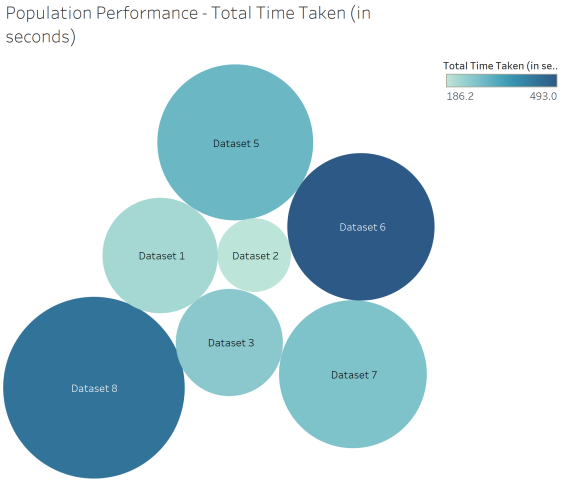


Figure 6.9: Bubble Chart representing the total time taken to populate the different knowledge graphs

Table 6.1: Table representing the total time taken to populate different UDM knowledge graphs

Datasets	Total Time Taken (in secs)	# Alerts	# Devices	# Network Logs	# Change Logs	# Links
Dataset 1	214.235	1239	24	24	69	43
Dataset 2	186.236	503	59	798	560	380
Dataset 3	251.241	1068	57	798	547	375
Dataset 4	31683.343	341166	346	0	3935	1440
Dataset 5	292.897	2263	53	224	1077	635
Dataset 6	492.987	2020	116	2689	931	364
Dataset 7	267.251	2035	195	0	401	608
Dataset 8	438.265	3070	55	200	691	666

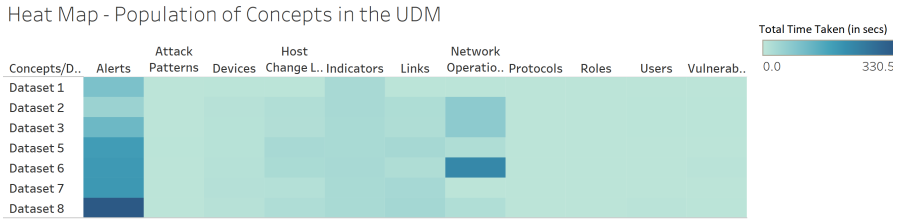


Figure 6.10: Heatmap showing Population Time per Concept

6.3 Generalization of the UDM for Other IDS

The UDM developed in this thesis is not meant to be used for a particular IDS alone, rather it is an abstract representation that is meant to be used across different IDSs and organizations. This section discusses how the same UDM could be used for a different IDS, namely the University's (TU/e) SOC Environment to capture the different types of data present in the IDS and connect

them using meaningful relationships for efficient incident investigation purposes.

6.3.1 Raw Data from TU/e's IDS

This data is a sample that was used for representational purposes. This section shows that the data model can be used universally across any IDS, if re-factored to fit that IDS's data.

Figure 6.11: The JSON data from TU/e SOC's IDS

The data from the SOC in TU/e was exported as a JSON document containing alerts and network operations. "Alerts" were identified as events that did not have the `bro_` prefix in their event type attributes. The latter were categorized as "Network Operations", since these "bro_" events represented a device's activity in the network in detail. Each source and destination IP Address involved in these events were categorized as "Devices". The geographic attributes of the source and destination form the "Location" concept and are linked to Devices via a relationship. Moreover, the SNORT rules are represented in the "Indicator" concept and linked to the Alert it triggers and the Device that triggered it.

6.3.2 Concepts in the IDS

The various concepts that were identified in the IDS data were:

1. Alert
2. Network Operation
3. Indicator
4. Device
5. Protocol
6. Location

The above concepts were fitted against the data from TU/e's SOC Environment to obtain a subset of the UDM in the following figure 6.12.

Attributes and Relationships of the Concepts

The following table 6.2 shows the different attributes and relationships that a concept has in the data model. As can be seen from the following table, the events such as "Alerts" and "Network Operations" have attributes to identify the type of event and are linked to the source and destination devices.

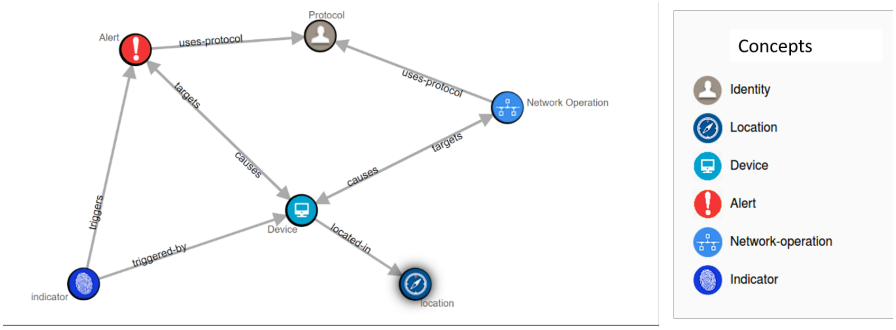


Figure 6.12: The Data Model showing the entities and relationships for the data from TU/e's IDS

Table 6.2: Attributes and Relationships of Concepts

Concept	**Attributes**	**Out-Bound-Relationships**
Alert	event_name, src_port, dst_port, event_type_id, timestamp, labels, priority, soc_id, type, id, name, description	uses-protocol ("Protocol") targets ("Device")
Device	ip, type, id, name	causes ("Alert", "Network Operation") located-in ("Location")
Network Operation	description, event_type_id, event_name, labels, timestamp, dst_port, src_port, type, id, name, soc_id	uses-protocol ("Protocol") targets ("Device")
Indicator	indicator_types, pattern, pattern_type, valid_from, id, type, name	triggered-by ("Device") triggers ("Alert")
Protocol	id, name, type	Nil
Location	id, type, country, region, latitude, longitude	Nil

6.3.3 The Sample Knowledge Graph

The following figure shows how the sample data from subsection 6.3.1 was fitted against the UDM and validated to generate a knowledge graph as shown in figure 6.13 below. This knowledge graph represents only a subset of the data for readability purposes. This can be extended with more concepts and relationships in the future, as data flows into the IDS. Thus, this proves that the UDM prototype is highly extensible and usable across different organizations with some initial refactoring.

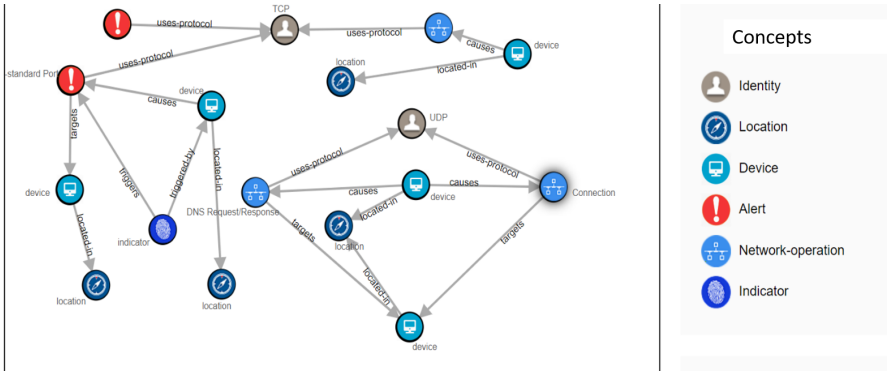


Figure 6.13: Knowledge Graph of TU/e SOC data

Thus, from this sample, it can be deduced that the Unified Data Model prototype can indeed be used on any IDS, even for "IT" networks like the one above, with some refactoring. The original vision of developing this prototype as a contribution to the community has been fulfilled, as stated in sectionn 5.1. This organization and holistic representation of the IDS data could enable analysts investigating security incidents to identify all the alerts/network operations caused by a device or illustrate all the alerts a specific SNORT rule triggered and so on. Thus, this data modelling provides a contextually rich and holistic view of IDS data for efficient incident investigation in IT and OT networks.

6.4 Summary

Thus, this chapter described how the UDM could be used to evaluate an OT incident such as Stuxnet and derive usable knowledge from the graph generated by the UDM Populator. This prototype can however also be used for proactive threat investigation in the network, if integrated with the IDS, allowing analysts to view and query the entire knowledge graph or a subset of it. Thus, if installed properly in an IDS, this data model can categorize and populate concepts present in any IDS in real-time. The population of Alerts and Network Operations however have to be improved in order to match real-time data flow in the IDS.

Chapter 7

Validating the UDM Prototype

The previous chapter elaborated on the case study of the Stuxnet Malware Attack that was investigated using the Unified Data Model. The chapter also shows how the Data Model is not IDS-specific and can be used across organizations. This chapter discusses the results of conducting a validation experiment with some experts in Forescout regarding the quality, completeness and coverage of the data model.

Two types of validation were performed - one where experts were asked questions to measure the completeness (missing attributes and relationships) and coverage (missing concepts)of the unified data model. This was done to identify if the data model prototype missed any important concept and judge the limitations of the data model which could be a scope for future work. The following sections discuss in detail the two different methods of validation with their results.

7.1 Validation Interview - Sanity Check

The goal of this interview is to evaluate the UDM prototype developed in this research with experts in the OT domain. The experts were presented with a JSON file containing the UDM prototype. This JSON document could be visualized as a graph using the STIX Visualizer. This graph visualization helped the experts understand better the different concepts, their attributes and relationships. With this visualization at hand, the experts were asked to fill an Excel sheet, as shown in figure 7.1 below, with their responses to measure the two metrics - Completeness and Coverage of the data model. The experts were asked to express their opinion on the different concepts in the data model in terms of:

1. **Completeness** – The measure of how complete (w.r.t relationships and attributes) a concept is. For eg. Attack Pattern models adversary TTPs and Kill Chain Phases of the Cyber Kill Chain. Experts might think of separating the TTP and the Kill Chain Phases and link them as `Attack Pattern --belongs-to--> Kill Chain Phase`
2. **Coverage** – The average number of concepts covered by the data model against the total available concepts in Forescout's IDS.

7.1.1 Responses

The two metrics completeness and coverage were measured from the experts responses. For the completeness metric, the experts could choose between "Agree" (Score 3), "Partially Agree" (Score 2) and "Disagree" (Score 1) for each concept in the UDM prototype. It was measured

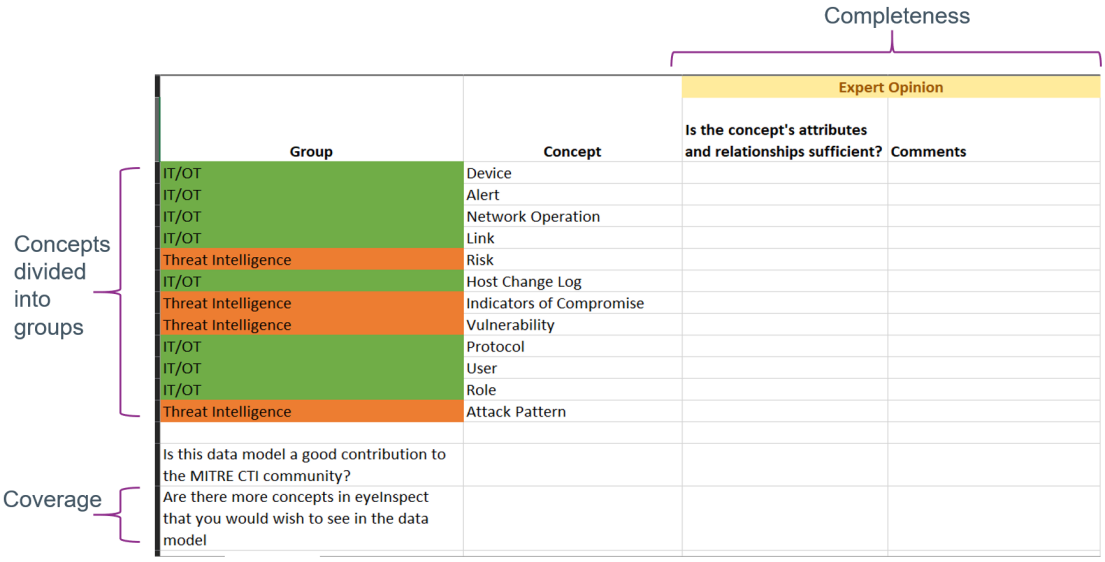
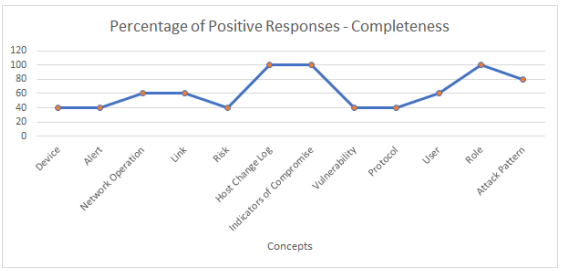


Figure 7.1: The Worksheet given to Experts

across experts as well as across the different concepts to judge which concepts needed modifications. The following figures summarize the two flavors of the same metric.

(a) Completeness measured across concepts



(b) Completeness measured across experts

Figure 7.2: Graphs showing the Completeness Metric as measured from their Responses

From the above figures 7.2a, 7.2b, it can be observed that the 5 experts scored a median score of 2.5 (which is slightly more than "Partially Agree"), indicating that they agreed with most concepts in the prototype. The figure 7.2b shows that the concepts they did not completely agree on were "Device", "Risk", "Alert", "Vulnerability" and "Protocol". Their general comments on the prototype are given below.

- The Protocol concept can be extended to support transport, data link layer and other protocols. Moreover, the type of protocol (if they are IT or OT protocols) must be specified.
- The Risk as a concept does not leverage the graph representation. It must be embedded as an attribute of a Device since it is out of scope for other concepts.

- The events in the data model (alerts, network operations, change logs) must follow the same set of attributes for consistency.
- Common Vulnerability Scoring System (CVSS) for vulnerabilities is not enough to assess the criticality of the vulnerable devices. The "matching confidence" of the vulnerability for the given device must also be included.
- The Devices must also include the serial number, hardware version and labels for additional information on the Device.
- A new concept for Malware should be introduced and linked to Alert, Attack Pattern and Device concepts.

With regard to the Coverage metric, it can be derived from the above responses that the data model was missing the other Protocol concepts and the Malware concept which plays a vital role for investigating malware incidents. Thus, the protocol concept had to be extended to accommodate the Transport and Link Layer protocols, and new concept for Malware had to be introduced in the prototype and linked to the other concepts that were already present. These comments were addressed immediately and the final data model looked like the one shown in figure 7.3 below. The Risk concept became obsolete and was embedded as "security_risk" and "operational_risk" attributes in a Device.

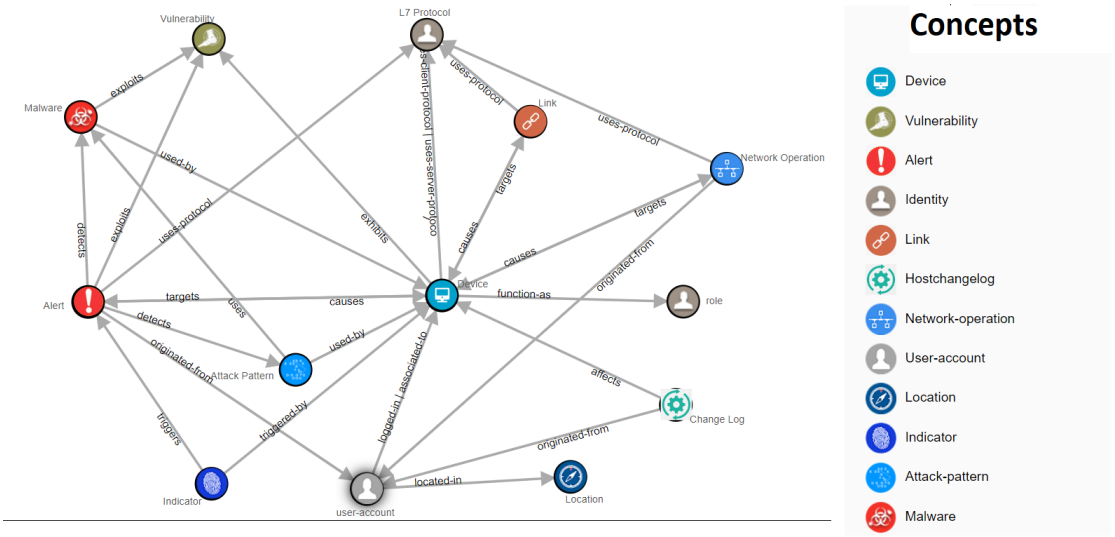


Figure 7.3: The UDM Prototype refined after incorporating expert comments

Changes to the UDM Prototype

Thus, taking the above comments into consideration, the following table 7.1 summarizes the main changes applied to the different concepts in the Unified Data Model prototype.

7.1.2 Summary

Thus, with respect to Coverage, the UDM prototype was an acceptable solution since only one concept was seen to have been missed in the data model. Moreover, with respect to Completeness,

Table 7.1: Changes to the UDM Prototype

Concept	Attribute	Out-Bound-Relationship
Device	serial_number, hardware_version, labels, operational_risk, security_risk, risk_factor	
Alert	payload, severity (is now a string “high”, “low”..)	detects (“Malware”) exploits (“Vulnerability”) originated-from (“User”)
Vulnerability	matching_confidence, impact	
Attack Pattern	kill_chain_phase –>tactic, name –>technique,	uses (“Malware”)
Risk	removed	removed
Malware (new concept)	name, aliases, is_family, capabilities, description, malware_types	exploits (“Vulnerability”) used-by (“Device”)

the average response of the experts was that they agreed with the prototype and believed it to be sufficient for investigating security incidents in the OT network.

7.2 Validation Experiment

This is an experiment with experts in Forescout to observe their incident investigation capabilities using their IDS without the unified data model and compare it against their investigation activities with the unified data model. The following section highlights some use cases for analyzing the Stuxnet Incident and observes the experts' steps taken to analyze these use-cases using their IDS with and without the unified data model. The Stuxnet Dataset was provided to all experts and they were requested investigate for the following use cases in Stuxnet, since the Stuxnet incident captures a lot of OT and IT concepts and the baseline has already been established in the previous chapter in section 6.1.

7.2.1 Technologies Provided

- A VM containing an instance of the IDS with all the Stuxnet incident data for investigation
- A fully populated Neo4j Bloom instance with the Stuxnet Knowledge Graph satisfying the UDM prototype for investigation.

7.2.2 Outline of the Experiment

In the first part of the experiment, the experts were provided with a fully-populated IDS containing the Stuxnet Incident data and they investigate the following use-cases in the incident. The steps taken by them during their investigation were documented. In the second part of the experiment, they were given access to a fully populated Neo4j database with the Stuxnet Incident. The experts investigated the same use cases and documented their steps and findings.

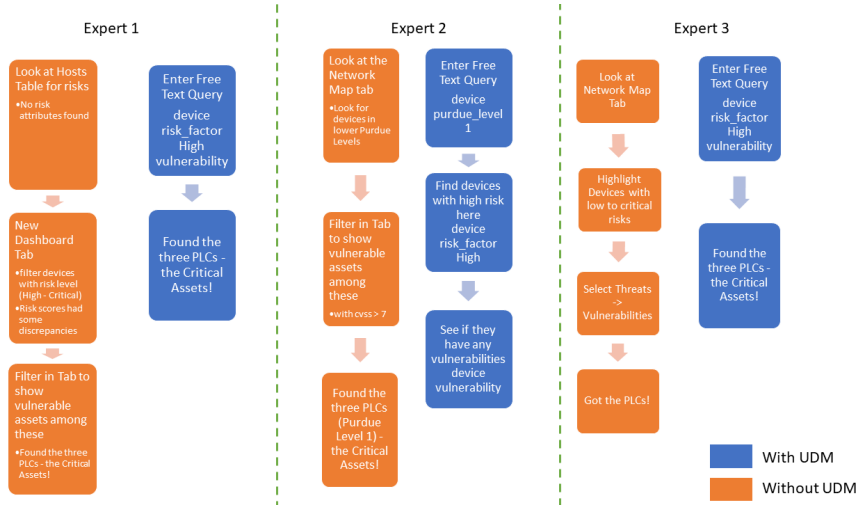
At the end of the experiment, the expert was asked which method they found easier to explore and analyze the incident and whether they wish to have this kind of interface for their incident investigation activities. The expert findings were collected, and a flowchart was prepared from their responses. These flowcharts were compared to evaluate their ease and quality of investigation with the data model (part 2 of the experiment) and without the data model (part 1 of the experiment). Their responses were judged by the number of steps they took to arrive at an answer to each use-case and the quality of the results they obtained for each use-case. The answers to the question determine the powerfulness of the data model in organizing the IDS data for efficient incident investigation.

7.2.3 Use-Cases

The following four use cases were selected for the validation session with experts, since they were straightforward and captured the crux of the Stuxnet Incident.

Identify all the critical assets

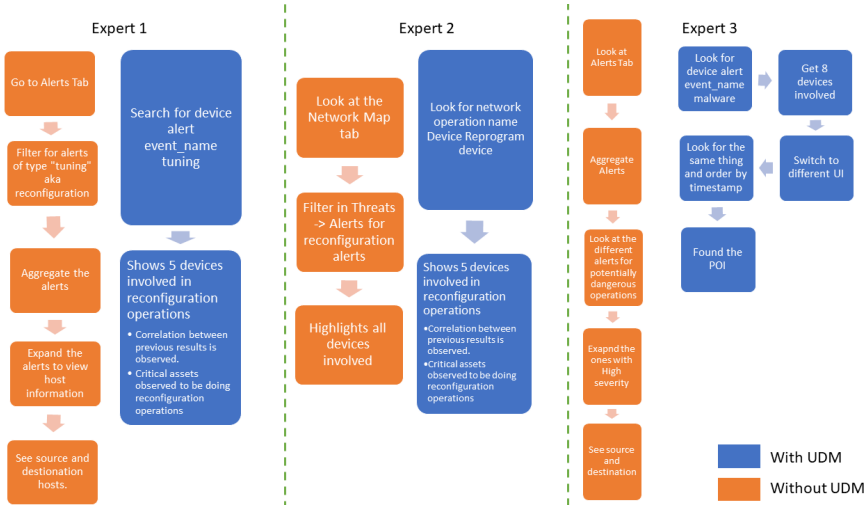
This tests whether the experts are able to identify the devices with high risk_factor) and critical vulnerabilities. The following figures highlight the responses of the different experts as a flowchart. The flowchart in orange color 7.4a shows the steps taken by the expert to investigate this use-case in Forescout's IDS and the one in blue color shows the investigation steps taken for analyzing the Stuxnet Knowledge Graph in Neo4j.



(a) Critical Assets

Find devices performing dangerous OT operations

This tests whether the experts were able to identify the devices performing device reconfiguration operations. The following figures highlight the responses of the different experts as a flowchart. The flowchart in orange color 7.4a shows the steps taken by the expert to investigate this use-case in Forescout's IDS and the one in blue color shows the investigation steps taken for analyzing the Stuxnet Knowledge Graph in Neo4j.

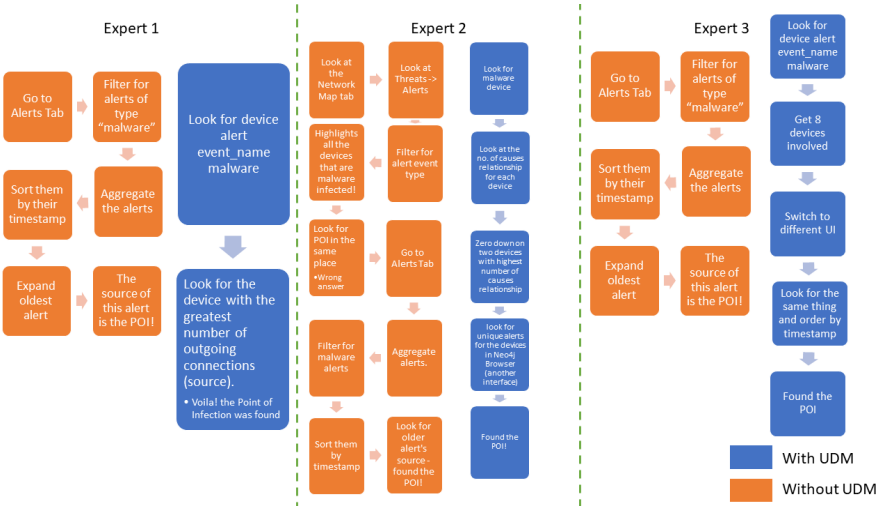


(a) Dangerous Operations

Investigate the POI of Stuxnet Malware

This tests whether the experts were able to identify the devices infected with the Stuxnet Malware the Point of Infection from where the Malware spread across the network. The following figures highlight the responses of the different experts as a flowchart. The flowchart in orange color 7.4a

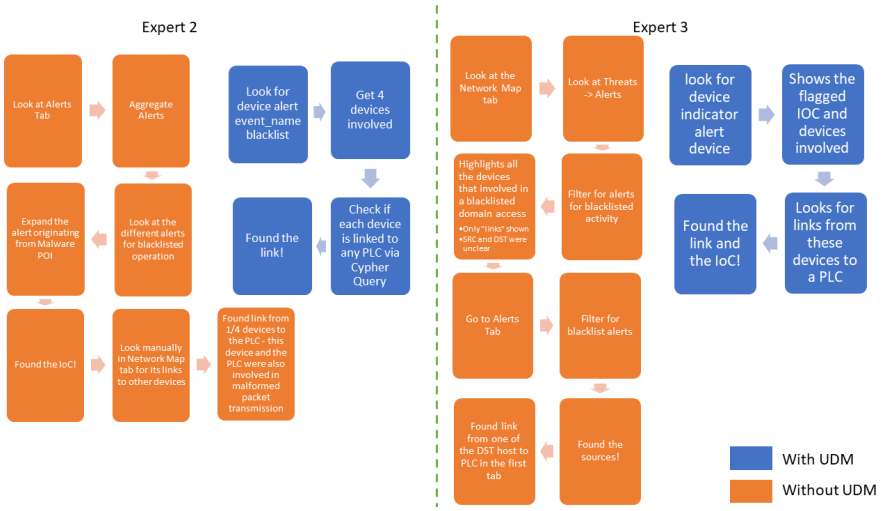
shows the steps taken by the expert to investigate this use-case in Forescout's IDS and the one in blue color shows the investigation steps taken for analyzing the Stuxnet Knowledge Graph in Neo4j.



(a) Malware Devices and POI of Stuxnet Malware

Tracing the path from the Malware to the PLCs

This use-case was not present for the Validation with Expert 1. This tests whether the experts were able to trace the path from the Stuxnet Malware infected devices to the Critical Assets in the network. The following figures highlight the responses of the two experts as a flowchart. The flowchart in orange color 7.4a shows the steps taken by the expert to investigate this use-case in Forescout's IDS and the one in blue color shows the investigation steps taken for analyzing the Stuxnet Knowledge Graph in Neo4j.



(a) Path from the Malware Infected Devices to the PLCs

7.2.4 Results

From the above figures, it can be observed that the experts were indeed found to have taken more number of steps to arrive at the results when investigating without the UDM than with the UDM. These experts believed that if they had this holistic view of data with rich context, their investigation would have been more powerful in the IDS itself. The following highlights some of the remarks expressed by the experts.

Positive Remarks from Experts

- Experts found it easy to correlate the results from different use cases, while investigating with the UDM. For example, they were able to understand that the critical assets were indeed involved in device reconfiguration operations.
- "I like the graph, I want it", said one expert who liked the visual representation of concepts as nodes and most importantly the relationships linking them.
- One of the experts also mentioned that they liked the User Interface of the IDS better, but would have preferred to use the data model inside the IDS itself for better data organization. The expert was keen on integrating this prototype with the product.
- One of the experts also mentions that data model goes beyond their company's IDS and can be used with any IDS, which makes it "a very good contribution to a wider community".

Other Remarks from Experts

- The experts raised the concern that alerts were not aggregated, and that a sea of alerts were thrust upon them when querying the data model in Neo4j. It was later clarified that the "Issue" concept in the UDM prototype addresses precisely this concern and that events of the same genre could be grouped together as a single concept.
- The experts mention that the IDS and Neo4j interfaces are completely different and thus the validation experiment may have been a little biased. Due to time constraints, a proper interface for exploring the data model could not be created. "The powerfulness of the data model was not very clear from this validation", said one of the experts. However, the expert also agreed that this was more a limitation of the interfaces themselves and not the data model. If the data model were integrated with the IDS, investigation would have been seamless and unbiased.

Chapter 8

Conclusion and Future Work

This research introduced a unified data model to integrate raw IDS data and organize them into a self-explanatory, contextually-rich knowledge graph for investigating security incidents in OT and IT environments. The raw IDS data from different sources were encapsulated as concepts of the data model, with relationships linking these concepts. This data model was successfully tested on the Stuxnet Malware incident (an ICS attack) and the resulting knowledge graph was investigated for several use cases such as tracing the Point of Stuxnet Infection to understanding the big picture as to how the Stuxnet incident actually happened. Additionally, this data model was also used to model the data from an IT-IDS at Eindhoven University of Technology and was successfully able to capture and represent some of the concepts in the proposed prototype with added semantics and relationships. This data model prototype was then validated by experts in Forescout who had extensive knowledge on OT-IDSs and the type of data present in them. Finally, a few improvements were suggested by these experts which was incorporated into the data model prototype. The final refined UDM prototype was successfully able to achieve most of the criteria specified in the section 5.1, such as Consistency (the usage of STIX standard), Expressiveness (data as a knowledge graph), Usability (model data from different IDSs) and Threat Intelligence purposes. Moreover, the research question(s), in section 1.1.1 were also satisfied by this prototype, to model data in an OT domain for Threat Intelligence information exchange and incident investigation purposes.

In the following sections, some concluding remarks on the advantages and limitations of the unified data model prototype and its scope for future improvement are discussed.

8.1 Merits of the Data Model

This section discusses the advantages of using this UDM prototype for structuring the OT IDS data. The data model prototype offers a(n)...

- Structured, holistic view of the network for investigative purposes - This feature of the UDM was also confirmed by the experts during the validation sessions as discussed in the previous chapter. This was the research question that this thesis aimed to solve, and this UDM prototype fulfills the same.
- Well-connected knowledge graph with consistency (standardized STIX objects)
- Representation of IT (User, Alerts,..), OT(Devices, Protocols, Change logs, operational risk...) and Threat Intelligence concepts(Vulnerability, IoC, Attack Pattern,..) for an

effective incident response in both IT and OT networks.

- Abstract representation with multiple database backends – can be used with Neo4j as a knowledge graph for incident investigation or exported as a JSON document for information exchange across the organization.
- Highly extensible and customizable interface to add new concepts, attributes and relationships that fit the IDS in any organization. This was confirmed by applying the data model to fit entities from Forescout's IDS and the IDS at TU/e.

8.2 Limitations of the Data Model

This section discusses the shortcomings of the UDM. The UDM prototype is not a plug-and-play module. The installation of the prototype for an IDS is expensive. This is because it has to be refactored to fit data for different SOC environments and IDSs. However once refactored, it can be used indefinitely for the SOC environment, either directly or out-of-the-box depending on the SOC Environment.

8.3 Future Work

The future work for this thesis would be to extend it with more concepts to make it more expressive and generic across different IDS' in different organizations. Some refinements to improve the performance of the population of the UDM knowledge graph is essential to work with the data model in real-time. In order to achieve this, a well-defined caching mechanism can be devised to reduce the number of disk-reads to load the "Device" concept from raw data sources. A possible Graphical User Interface (GUI) module could also be developed to view the UDM knowledge graph better in real-time, that can also be used universally across different IDS.

Bibliography

[1] Renzo Angles. A comparison of current graph database models. In *2012 IEEE 28th International Conference on Data Engineering Workshops*, pages 171–177. IEEE, 2012. 18, 20

[2] Shalini Batra and Charu Tyagi. Comparative analysis of relational and graph databases. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2):509–512, 2012. 18

[3] Sean Bechhofer, Frank Van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, Lynn Andrea Stein, et al. OWL web ontology language reference. *W3C recommendation*, 10(02), 2004. 17

[4] Sakshi Chahal. JSON vs Protocol Buffer Simplified. https://medium.com/@sakshichahal53/json-vs-protocol-buffer-simplified-dbd6b69ca528, 2019. [last accessed: 2020-02-25]. 17

[5] Sudip Chowdhury. Knowledge Graph: The Perfect Complement to Machine Learning. https://towardsdatascience.com/knowledge-graph-bb78055a7884, 2019. [last accessed: 2020-02-10]. 7

[6] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, 13(6):377–387, 1970. 19

[7] Cyber Threat Intelligence Technical Committee. Introduction to STIX. https://oasis-open.github.io/cti-documentation/stix/intro, 2020. [last accessed: 2020-02-24]. 21, 23

[8] Stefan Fenz and Andreas Ekelhart. Formalizing Information Security Knowledge. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, page 183–194, New York, NY, USA, 2009. Association for Computing Machinery. 15

[9] José Manuél Gómez-Pérez, Jeff Z. Pan, Guido Vetere, and Honghan Wu. Chapter 1 Enterprise Knowledge Graph : An Introduction. 2017. 6

[10] Google. Protcol Buffers. https://developers.google.com/protocol-buffers/docs/overview, 2019. [last accessed: 2020-02-25]. 17

[11] Jing Han, E Haihong, Guan Le, and Jian Du. Survey on NoSQL database. In *2011 6th international conference on pervasive computing and applications*, pages 363–366. IEEE, 2011. 18, 20

[12] Michael Iannacone, Shawn Bohn, Grant Nakamura, John Gerth, Kelly Huffer, Robert Bridges, Erik Ferragut, and John Goodall. Developing an ontology for cyber security knowledge graphs. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, pages 1–4, 2015. 13, 14, 17

[13] Splunk Inc. Splunk® Common Information Model Add-on Common Information Model Add-on Manual 4.15.0. https://surfdrive.surf.nl/files/index.php/s/NDeDSvnOMpLyvuz, 2020. [last accessed: 2020-02-18]. 15

[14] Yanko Ivanov. What is an Enterprise Knowledge Graph and Why Do I Want One? https://enterprise-knowledge.com/what-is-an-enterprise-knowledge-graph-and-why-do-i-want-one/, 2018. [last accessed: 2020-03-10]. 6, 7

[15] Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Ekaputra. The SEPSES Knowledge Graph: An Integrated Resource for Cybersecurity. In *International Semantic Web Conference*, pages 198–214. Springer, 2019. 12, 13, 16, 21

[16] Bruno Krebs. Beating JSON performance with Protobuf. https://auth0.com/blog/beating-json-performance-with-protobuf/, 2017. [last accessed: 2020-02-25]. 17

[17] Neal Leavitt. Will NoSQL databases live up to their promise? *Computer*, 43(2):12–14, 2010. 18, 20

[18] Yishan Li and Sathiamoorthy Manoharan. A performance comparison of SQL and NoSQL databases. In *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 15–19. IEEE, 2013. 18, 20

[19] Florian Menges and Günther Pernul. A comparative analysis of incident reporting formats. *Computers & Security*, 73:87–101, 2018. 14, 16

[20] Luanne Misquitta and Alessandro Negro. Knowledge Graph Search with Elasticsearch and Neo4j. https://neo4j.com/blog/knowledge-graph-search-elasticsearch-neo4j/, 2020. [last accessed: 2020-03-02]. 22

[21] Glenn Murray, Michael N Johnstone, and Craig Valli. The convergence of IT and OT in critical infrastructure. 2017. 1

[22] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015. 7

[23] Steven Noel, Eric Harley, Kam Him Tam, Michael Limiero, and Matthew Share. Cygraph: graph-based analytics and visualization for cybersecurity. In *Handbook of Statistics*, volume 35, pages 117–167. Elsevier, 2016. viii, 11, 12, 13, 16, 21, 28

[24] Leo Obrst, Penny Chase, and Richard Markeloff. Developing an Ontology of the Cyber Security Domain. In *STIDS*, pages 49–56, 2012. 15

[25] Alessandro Oltramari, Lorrie Faith Cranor, Robert J Walls, and Patrick D McDaniel. Building an Ontology of Cyber Security. In *STIDS*, pages 54–61. Citeseer, 2014. 15

[26] Rich Piazza, Trey Darley, and Bret Jordan. STIX Version 2.0. Part 2: STIX Objects. https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.html, 2017. [last accessed: 2020-02-21]. viii, 2, 10, 11, 22, 23, 25

[27] Thomas Roccia. Triton Malware Spearheads Latest Attacks on Industrial Systems. https://www.mcafee.com/blogs/other-blogs/mcafee-labs/triton-malware-spearheads-latest-generation-of-attacks-on-industrial-systems/, 2011. [last accessed: 2020-04-15]. 3

[28] Luis M Álvarez Sabucedo, Luis E Anido Rifón, Flavio Corradini, Alberto Polzonetti, and Barbara Re. Knowledge-based platform for eGovernment agents: A Web-based solution using semantic technologies. *Expert Systems with Applications*, 37(5):3647–3656, 2010. 17

[29] Farhan Sadique, Sui Cheung, Iman Vakilinia, Shahriar Badsha, and Shamik Sengupta. Automated structured threat information expression (STIX) document generation with privacy preservation. In *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 847–853. IEEE, 2018. v, 26, 27

[30] Cambridge Semantics. Start Building Your Enterprise Knowledge Graph. https://info.cambridgesemantics.com/build-your-enterprise-knowledge-graph, 2018. [last accessed: 2020-02-24]. 7, 15

[31] Amit Singhal. Introducing the knowledge graph: things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/, 2012. [last accessed: 2020-02-24]. 6

[32] Anoop Singhal and Duminda Wijesekera. Ontologies for modeling enterprise level security metrics. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, pages 1–3, 2010. 17

[33] Florian Stegmaier, Udo Gröbner, Mario Doeller, Harald Kosch, and Gero Baese. Evaluation of Current RDF Database Solutions. 12 2009. 18, 20

[34] Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews, and Anupam Joshi. UCO: A unified cybersecurity ontology. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 15

[35] VERIS. VERIS: The vocabulary for event recording and incident sharing . http://veriscommunity.net/, 2017. [last accessed: 2020-04-08]. 2, 13

[36] Wikipedia. Stuxnet. https://en.wikipedia.org/wiki/Stuxnet#Operation, 2011. [last accessed: 2020-06-30]. 40, 43

[37] Wikipedia. Purdue enterprise reference architecture. https://en.wikipedia.org/wiki/Purdue_Enterprise_Reference_Architecture, 2019. [last accessed: 2020-02-24]. 5

[38] Kim Zetter. How Digital Detectives Deciphered Stuxnet, the Most Menacing Malware in History. https://www.wired.com/2011/07/how-digital-detectives-deciphered-stuxnet/?utm_campaign=Previous&utm_medium=RelatedLinks&utm_source=Contextly, 2011. [last accessed: 2020-04-15]. 1, 40