TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Kadir Burak Mavzer
177773

REPRESENTING CYBER THREAT INTELLIGENCE ON A KNOWLEDGE GRAPH

Bachelor’s Thesis

Supervisor: Toomas Lepikult
Tiago Nogueira

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Kadir Burak Mavzer
177773

KÜBEROHUTEADMUSE ESITAMINE TEADMUSGRAAFI ABIL

Bakalaureusetöö

Juhendaja: Toomas Lepikult
Tiago Nogueira

Tallinn 2020

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kadir Burak Mavzer

30.04.2020

Abstract

Cyber threats are increasing at a fast rate and security professionals are struggling to respond in time to protect the assets of their organizations. There are tools to ease the process such as threat feeds and threat intelligence sharing platforms. However, these mostly provide threat data in textual format, which is hard to navigate when there are thousands of threat attributes, emerging new threat actors and information about their complicated ways. Cyber threat data is highly interconnected, and this feature makes it a perfect match for graph databases. Graph databases are those that typically represent information in a directed graph with nodes and edges. For example, a threat actor, a malware and a target organization would be called nodes whereas the relations, such as in the sentence “threat actor *uses* malware” are called edges. Graph databases are inherently very intuitive as they use a similar structure to how we learn things around us. The importance of representing threat data in a manner that is meaningful to both humans and computers is that only the collaboration of both will give the maximum effect in fighting against emerging threats. There are complicated campaigns carried out by APTs (*Advanced Persistent Threat*) that take months or even years to complete. In order for security professionals to realize that there is even a possibility of such a campaign exists, it is necessary to represent the interconnectedness of threat data. This paper discusses the importance of representing data on a NoSQL (*Not Only Structured Query Language*) graph database and its benefits as well as a development process of a graph threat intelligence platform for project ECHO (*European Network of Cybersecurity Centres and Competence Hub for Innovation and Operations*). ECHO is a European Commission initiative under H2020 (*Horizon 2020*) that aims to bring together organizations from different sectors to collaborate on strengthening the cybersecurity of EU (*European Union*). One of the main outcomes of four years of effort from ECHO partners will be an Early Warning System that will help organizations from different sectors share threat information between each other. There will be plugins to help overcome some of the biggest issues within threat intelligence sharing and a knowledge graph will be one of them.

This thesis is written in English and is 39 pages long, including 7 chapters, 10 figures and 1 table.

Annotatsioon

Küberohuteadmuse esitamine teadmusgraafi abil

Küberohtude hulk laieneb tänapäeval väga kiiresti ning koos sellega ka küberspetsialistide tööpõld. On loodud spetsiaalsed vahendid turbeseisundi jälgimiseks, nende seas ka küberteadmuse jagamise platvormid. Paraku esitavad nimetatud platvormid infot vaid tekstina, milles on sageli raske orienteeruda kui turbeatribuute on sadu. Küberturbe-alased andmed on tihti tugevas seoses, mis teeb otstarbekaks graafi-põhiste andmebaaside kasutamise nende esitamisel. Näiteks ründaja, pahavara ja rünnatav organisatsioon võivad kujutuda graafi sõlmedena, ning ründetegevused graafi kaartena. Graafi-põhised andmebaasid on intuitiivselt lihtsasti hoomatavad, kuna kasutavad analoogset struktuuri inimese tajuga põhjuslikest seostest. Struktuuride kujutamine sellisel kujul, mis on hästi mõistetav nii inimesele kui ka arvutile, on äärmiselt oluline efektiivse kaitse loomiseks küberohtude vastu.

Käesolev bakalaureusetöö keskendub küberturbealaste andmete salvestamisele NoSQL-tüüpi andmebaasis, aga ka küberohuteadmuse platvormi arendamise projektile ECHO (*European Network of Cybersecurity Centres and Competence Hub for Innovation and Operations*). Üheks tähtsamaks ECHO eesmärgiks on varajase hoiatamise süsteem, mis aitab erinevatel organisatsioonidel jagada omavahel küberohtude-alast teavet. Projekti käigus luuakse erinevaid pistikuid küberohuteadmuse jagamiseks ning teadmusgraaf on üks neist.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 39 leheküljel, 7 peatükki, 10 joonist, 1 tabelit.

List of abbreviations and terms

DPI	*Dots per inch*
TUT	Tallinn University of Technology
APT	Advanced Persistent Threat
SQL	Structured Query Language
NoSQL	Not Only SQL
ECHO	European Network of Cybersecurity Centres and Competence Hub for Innovation and Operations
EU	European Union
US	United States
MISP	Malware Information Sharing Platform
ID	Identification
STIX	Structured Threat Information Expression
OWL	Web Ontology Language
H2020	Horizon 2020
SIEM	Security Information and Event Management
HTML	Hyper Text Markup Language
SVG	Scalable Vector Graphics
CSS	Cascading Style Sheets
CLI	Command Line Interface
JSON	JavaScript Object Notation
DTO	Data Transfer Object
IP	Internet Protocol
UML	Unified Modeling Language

Table of contents

List of figures

List of tables

1 Introduction

A great example of the importance of seeing the bigger picture when it comes to cyber threat intelligence is the attacks carried out against one of the biggest U.S retailers Target in 2013. According a report on the case [1], the attack could have been prevented if only Target security professionals could see where it was leading and treated it as a major threat. The graph representation could have been very helpful in this regard as it would connect all the dots for security professionals to easily make sense of what is happening. It is evident that threat intelligence sharing currently has lots of issues [2] and it has a long way to go. Improving the efficiency of threat intelligence sharing will only help in one of many issues that experts from different parts of the world try to solve.

When it comes to cybersecurity, data correlation and granularity are important terms. As threat actors and cyber threats created by them are getting more sophisticated, it becomes easier to get lost in hundreds of pages of threat reports and harder to make any sense of them. Using a knowledge graph, we can introduce new types of attacks, observables, threat actors etc. easily by just a few lines of simple queries and they will become immediately available as nodes. Then, we can also give attributes to our newly created node and create relations with other nodes.

One other very useful aspect of knowledge graphs is that the data is visualized in a structured way. This helps cybersecurity teams respond more quickly to threats as they can immediately see what the threat comprises of, what observables were seen etc. They can then run a quick scan on their network as in any other event to see whether their assets have been compromised.

Bigger organizations, such as MITRE [3] have recently started offering graph services for cyber threat intelligence paired with machine learning technologies to better fight against rapidly emerging threats. Chapter 2 will discuss these tools as well as the research conducted in this area and how this paper will bring added value to the topic at hand.

In chapter 3, an overview of the development of a cybersecurity ontology that will eventually be used as a data model for the graph, will be given. An ontology is a representation of knowledge in a certain domain and the relations between them.

Chapter 4 will be dealing with the project, within which the representation of threat intelligence on knowledge graphs will be used, namely ECHO and EWS (*Early Warning System*), as part of it. The aim is to develop a pluggable application to ECHO Early Warning System to help ease the process of identifying and analysing threats for security professionals within the ECHO consortium.

Chapter 5 will discuss the development methodology, used for the development of such application, including its technology stack, architecture, backend and frontend development. As the Early Warning System is still under development, main threat intelligence sharing platform of interest will be the commonly used Malware Information Sharing Platform. MISP [4] (*Malware Information Sharing Platform*) is a threat intelligence sharing platform project that is co-financed by the European Union. It aims to simplify the process of sharing threat intelligence between organizations through the use of certain standards. The application will be designed and developed in such a way that it can accept threat information from multiple feeds, creating a common graph for all threat intelligence sharing platforms. This will be helpful to identify false positives, which is another problem in threat intelligence sharing that is important to tackle.

Chapter 6 will identify work that could be done in order to further contribute to the solution to the issues mentioned. Finally, a summary of the work done during the writing of this thesis will be given in Chapter 7.

2 Related Work

In order to better understand what a graph database is and why it might be so much better to store cyber threat intelligence in a graph database rather than traditional relational database, it is a good idea to look at some formal definitions. In their book named “Graph Databases” [5] Ian R., Jim W. and Emil E. explain graphs as “just a collection of vertices and edges or … a set of nodes and the relationships that connect them”. The keyword here is edges or rather relationships. Graph databases take two arbitrary kinds of nodes – which are entities represented as nouns – and connect them through edges, which give the nodes a context and help us understand their meaning. This representation also makes modeling graph databases much easier. For comparison, let us try to model a small cybersecurity database using both relational and graph data modeling techniques. Data modeling is the structuring of data in a way that brings them together to represent a domain of interest.

Entities within this model will be Threat Actor, Organization and Malware. In order to properly model these three entities, some example queries that could be asked from the database should also be written. Examples can be seen below:

- Which *threat actor* targeted which *organization*?
- Which *malware* has been used by the *threat actor* to target the *organization*?
- Which *organization* is compromised by which *malware*?
- Which *organizations* did a *threat actor* target in the past?

First, these three entities will be modelled using a Unified Modeling Language diagram. The relational model of the data can be seen below:

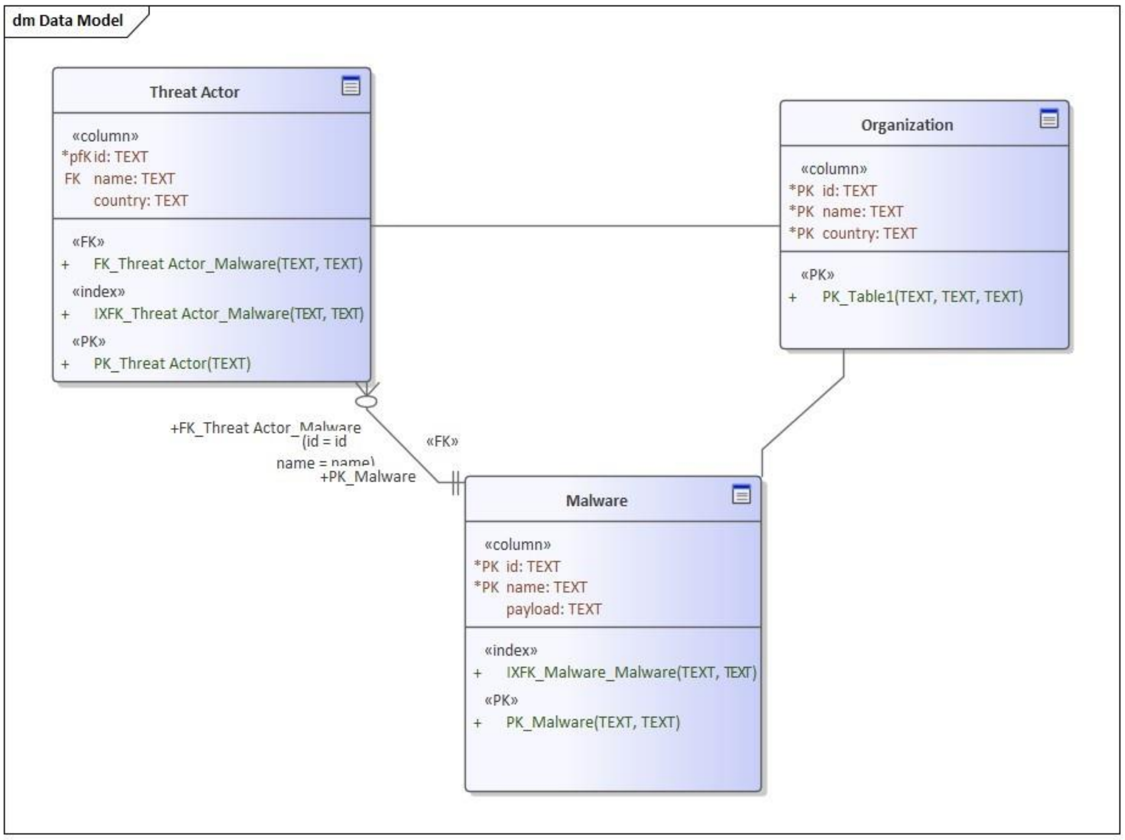


Figure 1: Relational Data Model Representation

As can be seen in the above figure, it is a simple data model containing 3 entities. However, what is missing from this diagram are the relations. Relational databases generally lack relationships unless they are specifically embedded into them. Even then, they require a lot more join operations to query, which are very expensive. Imagine a new entity comes into play that relates to all of the entities defined. This means a new table and new join operations and makes the database very unscalable. In today's cyberspace where everything changes very quickly, using a relational database for cyber threat intelligence could be cumbersome and inefficient at best if not impractical.

Now for the same model on a graph database, the only things needed are a few circles and the relationships between them as can be seen from the figure below:

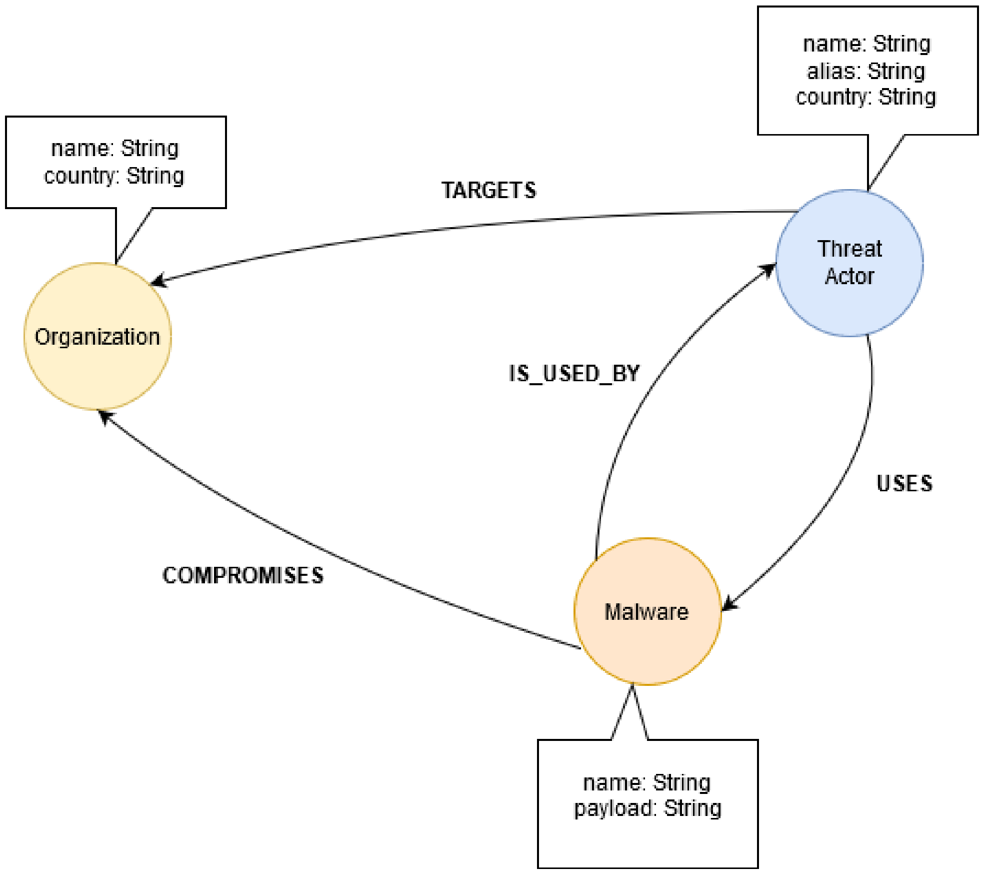


Figure 2: Graph Data Model of a Simple Cybersecurity Domain

The above figure not only looks better but also represents the relationships between entities quite clearly. That is because graph databases natively support relations. This figure already represents what the graph database would look like and in fact it is already a graph itself. This particular one is called a labeled property graph where there are nodes, relationships, properties and labels. In this case, the nodes are colored circles, relationships are the arrows coming into and out of the nodes, labels are the texts within circles and properties are those that are within the bubbles. Labeled property graphs are very intuitive and have proven to be one of the best ways to represent interconnected data. There is no overhead of joining tables and making sure everything fits perfectly. Some other properties of a labeled property graphs are given below:

- Nodes have properties such as IDs (*identifiers*), names etc. Properties are usually stored as key-value pairs.
- Relationships must have a direction, in other words a start and an end node. They are also given names for semantic clarity.
- Node labels are typically made of nouns, whereas relationships are verbs.

Moreover, the efficiency of graph databases is significantly higher than relational databases when it comes to interconnected data. A study conducted by Partner and Vukotic [6] shows that graph databases, Neo4j in this particular case, are notably faster for connected data than their relational counterparts. They have used a friends-of-friends social network database to demonstrate this. With a social network of one million people with about fifty friends each, the results are clearly better for graph databases as can be seen in the table below:

Depth	**Execution time – MySQL**	**Execution Time - Neo4j**
2	0.016	0.010
3	30.267	0.168
4	1.543.505	1.359
5	Not finished	2.132

Table 1: Neo4j vs. MySQL Comparison Over Interconnected Data

The reason graph databases perform so much faster compared to relational databases in terms of interconnected data lies in the fact that join operations in relational databases tend to be computationally expensive. Therefore, when a user tries to look up a friend-of-a-friend-of-a-friend-of-a-friend-of-a-friend query, relational database application has to do lots of index look-ups, hence it cannot finish the task in a meaningful time to be used in production. On the other hand, graph databases generally do not use indexes and instead all nodes in the database are linked to their neighbors directly.

The above analysis shows the main differences between relational and graph databases. For another example, graph databases will now be compared against another popular type of NoSQL database, called document databases (or document stores). The main difference between document stores and graph databases can be explained as follows:

- Graph databases are better at traversing highly interconnected information, that does not necessarily have to be structured beforehand. Examples of this can be:
    - Social relationships (as demonstrated by Facebook’s Open Graph [7])
    - Roadmaps
    - Cyber threat intelligence, where connectedness plays an important role
- Document stores perform much better when a user wants to store information that is initially structured.

In the end, there is no one-size-fits-all solution for databases and it highly depends on how the data is structured, if at all. In case of cybersecurity related threats, the data seems to be a perfect fit for graph databases as connectedness plays a very important role.

Using graphs as a means to increase the efficiency of threat intelligence sharing, hence of security professionals, has been attempted by multiple significant organizations. For example, U.S.-based MITRE Corporation has done some wonderful work in this area, mainly on their proprietary cyber threat intelligence tool CyGraph [3]. CyGraph is a cybersecurity focused graph analytics and visualization tool that aims to reduce the response times of security professionals by giving them a visual representation of both actual and potential threats as well as tools to easily identify and analyze them. In the related research paper [8], it is stated that CyGraph can also build a predictive model of potential attack paths and that it provides decision support and situational awareness. As stated in MITRE website, it is currently only a prototype tool. In his CyGraph paper, S. Noel mentions the problem with threat intelligence sharing is not the lack of available information but the ability to assemble seemingly irrelevant information into a meaningful picture. As it seems, this is exactly what CyGraph is about.

STIX (*Structured Threat Information Expression*) [9] is a standardization attempt for a structured language for threat intelligence sharing. It makes contributing to and sharing cyber threat intelligence much easier and worry-free. At the time of writing, the latest version of STIX is STIX 2.1 and it identifies 18 objects to represent threat data, these are:

- Attack Pattern: the ways in which the threat actors try to compromise their targets
- Campaign: a set of malicious activities that happen over a period
- Course of Action: what the organizations should do against such an attack
- Grouping: asserts that referenced STIX objects share a context
- Identity: represents individuals, organizations, threat actors as well as their classifications
- Indicator: is a pattern that leads to a malicious cyber activity
- Intrusion set: the behaviors and resources of an adversarial organization

- Location
- Malware
- Malware Analysis
- Note: denotes a note about the object in question
- Observed Data: represents information about the attributes of a cyber event
- Opinion: is an assessment of the information to provide higher quality CTI (*Cyber Threat Intelligence*)
- Report
- Threat Actor
- Tool: represents the tools that can be used in a cyber attack.
- Vulnerability

A cyber-attack can be represented as a STIX bundle, meaning that all the information relating to the attack is represented as one of the STIX objects. STIX can be stored as JSON (*JavaScript Object Notation*) or visualized for an analyst to better understand what a single attack comprises of. An example STIX visualization, that is made through OASIS STIX visualizer [10] can be found below:

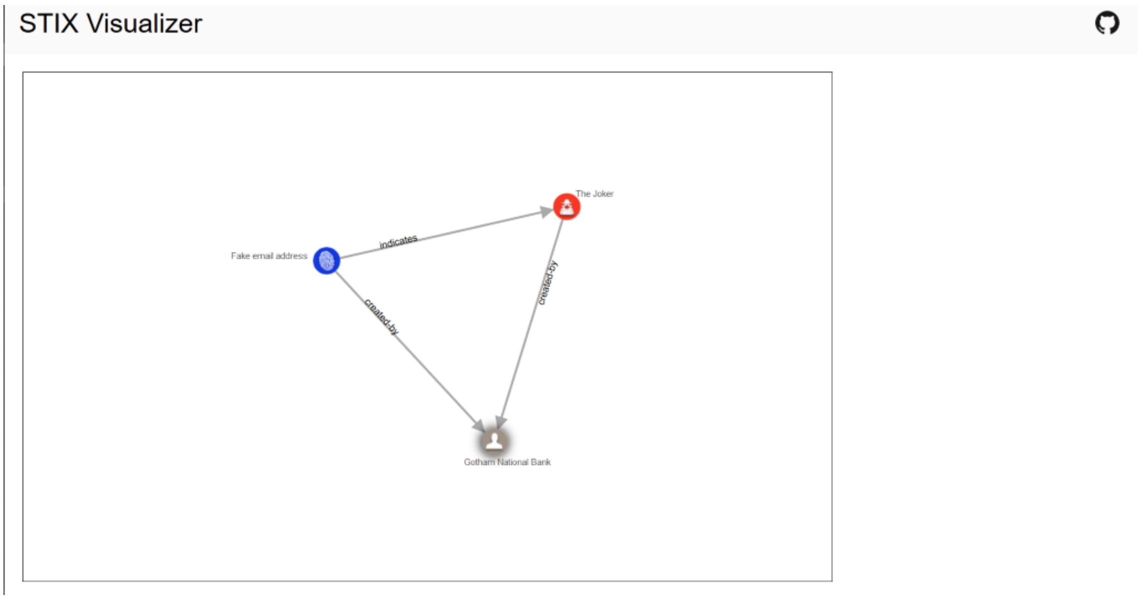


Figure 3: STIX Visualization Example

The difference between visualizing STIX and cyber knowledge graph will be that STIX does not keep a database of events. The visualizer simply parses the JSON given and comes up with a very basic visual representation of it. Users cannot query or correlate any of the data.

3 Ontology Development

An ontology is a representation of knowledge in a specific domain using domain-specific terms and the relations between those terms. For example, in case of cybersecurity this could be as depicted in the following:

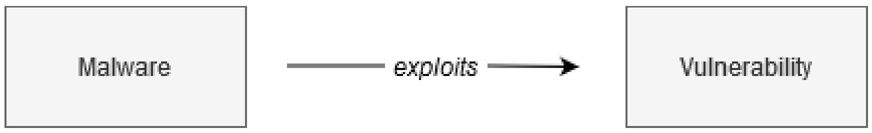


Figure 4: An Example of a Semantic Triple Related to Cybersecurity

In ontological terms, this would be called a “semantic triple”, which as indicated means three entities that are respectively called subject, predicate and object. Subjects and objects can also have instances as in “Microsoft Windows 10” can be an instance of “Operating System” whereas relations cannot. In order to create an ontology, one must first select a domain. In this case, the domain that will be represented using semantic triples is cybersecurity. But as cybersecurity comprises of vast amount of knowledge, it is best to split it into sub-domains, such as offensive security, digital forensics etc. To represent threat intelligence data meaningfully, a comprehensive data model is needed. Therefore, during the development phase, an ontology that represents sub-domains of cybersecurity will be used as a data model. As for the ontology language that is comprehensible by both humans and computers, OWL (*Web Ontology Language*) will be used. According to World Wide Web Consortium Semantic Page wiki [11], OWL has been developed by OWL Working Group [12] and it is “a semantic web language designed to represent rich and complex knowledge about things, groups of things, and relations between things.” OWL can also be supported by data and relationship properties, it is very suitable for the schema of a graph database as they follow a similar approach. Although a language would be enough, it is better to use a supporting product that leverages the capabilities presented by OWL. For this purpose, protégé [13] will be used. Protégé is a suite of tools that helps its users build semantic applications and domain models. It was originally developed by Stanford Center for Biomedical Informatics Research [14] and has a growing user community. Although its original purpose was to help in bioinformatics research, it grew big enough to support any type of domain modeling.

In order to develop an ontology that will help generalize the cybersecurity domain and be used in a production-ready solution for a graph database threat intelligence platform, many data sources have been used. These include large taxonomies published by multiple cybersecurity-focused organizations, data models of existing tools, as well as earlier ontology development attempts. Some of the sources include:

- MISP taxonomies (includes broad range of taxonomies) [15]
- Structured Threat Information Expression (STIX) data model
- Unified Cybersecurity Ontology by University of Maryland, Baltimore County [16]

The list is not exhaustive. After relevant information and terms were gathered, duplications were removed. The remaining terms were grouped into sub-domains and relations between them were created. A concise language was used in order not to cause any confusion. The current version of the cybersecurity ontology consists of more than 400 nodes and around 500 relations. The following shows a visual representation of a small portion of the cybersecurity ontology using the online WebVOWL [17] tool:

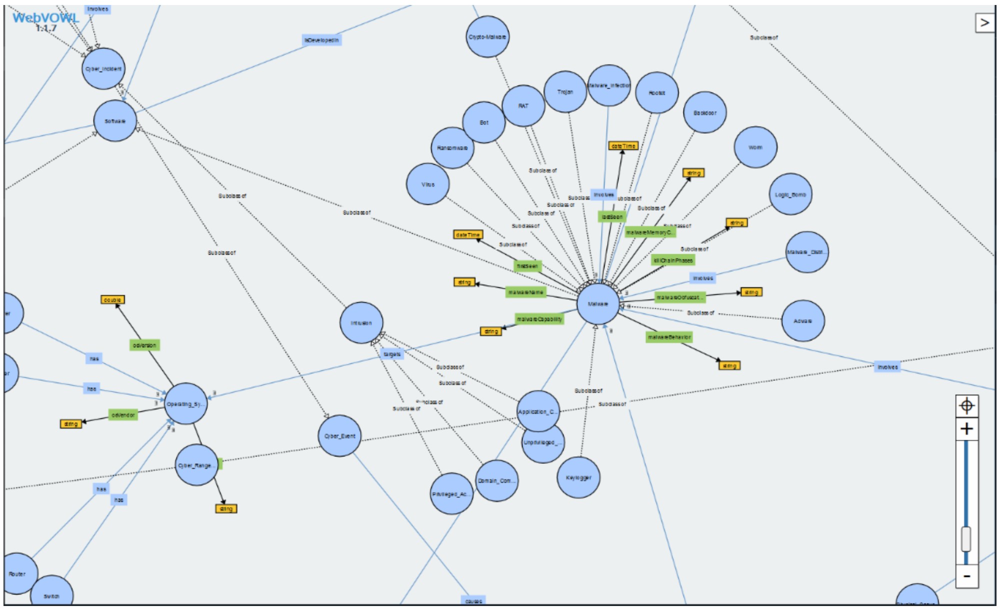


Figure 5: Cybersecurity Ontology Visualization

The way the ontology will be used in the application that will be developed as part of the whole project is that it will serve as an extension of domain models that are already implemented (namely, EWS and MISP data models) to enrich the information that is contained in the database. In a way, it will be a superset of these data models. For example, a MISP entry can have an IP (*Internet Protocol*) address as an attribute of an event. The application will dissect the information contained within this attribute and through analysis, it will enrich this information in a way that makes it more valuable to the user while maintaining relevancy. One of the key points in this project is that the information must be rich as well as relevant to the event in question. As stated in a research report published by ESG [18], 72% of cybersecurity professional think that half of the intelligence within threat feeds is irrelevant while 74% believe that it is difficult to determine their quality. Therefore, the thin balance between quantity and quality must be kept.

Later on, this ontology will be converted into Java objects and used as a common domain model for all threat feeds that the user wants to enable. Using an ontology helps in creating a broad knowledge base that encapsulates all data related to threat intelligence ranging from malware types to finer details such as the likely geolocation of the threat actor that uses the malware.

4 ECHO / H2020 Introduction

The application that will eventually be developed will be utilized as a plugin to the Early Warning System that is being developed as part of the ECHO project. European Network of Cybersecurity Centres and Competence Hub for Innovation and Operations [19] is a European Commission initiative under H2020 [20], an EU Framework Programme for Research and Innovation, that comprises of more than 40 organizations from different sectors, in which Tallinn University of Technology also takes part. ECHO aims to strengthen the cybersecurity stance of EU organizations with state-of-the-art products such as EWS. It mainly focuses on the following sectors:

- Energy
- Defense
- Maritime
- Healthcare

Throughout the lifetime of the project as well as until the consortium is fully established, there will be new partners joining. Therefore, new sectors will be added to the list, eventually covering all key sectors.

As mentioned above, the software that is mentioned in this paper will be used as an Early Warning System plugin. EWS is a tool that helps security professionals within organizations manage cyber events relating to their organizational network as well as share them within their constituencies of trusted partners. There are many tools that help organizations manage this type of information, typically called SIEMs (*Security Incident and Event Management*). However, the power of EWS will come from its architecture that allows organizations employ plugins that will improve the efficiency of their security operations. The plugins will be enabled or disabled by users' choice. Initial plan is that when the graph plugin is enabled, users will have the ability to view a representation of the textual threat information as well as to query the underlying graph database for more information on what the threat comprises of. The goal is to eventually allow users, according to their permissions, query the EWS database along with MISP database and a database that has events from different threat feeds. This way, users will be able to see the threat landscape much better and distinguish false positives easier.

The visualization component will be highly intuitive as it will show how the threat came to be in simple terms with a colorful force-directed graph. Another goal of the project is to show how the threat evolved by means of a temporal graph representation. This will be helpful in pointing out the crucial points of the attack.

5 Development Methodology

This section will elaborate on the development stage of the application, main goal of which is to represent threat data on a knowledge graph and help security professionals analyze and respond to those threats quickly. With this goal in mind, the next steps are to define the requirements, after which an architecture as well as the user stories will be drawn up before the actual development starts.

5.1 Software Requirements

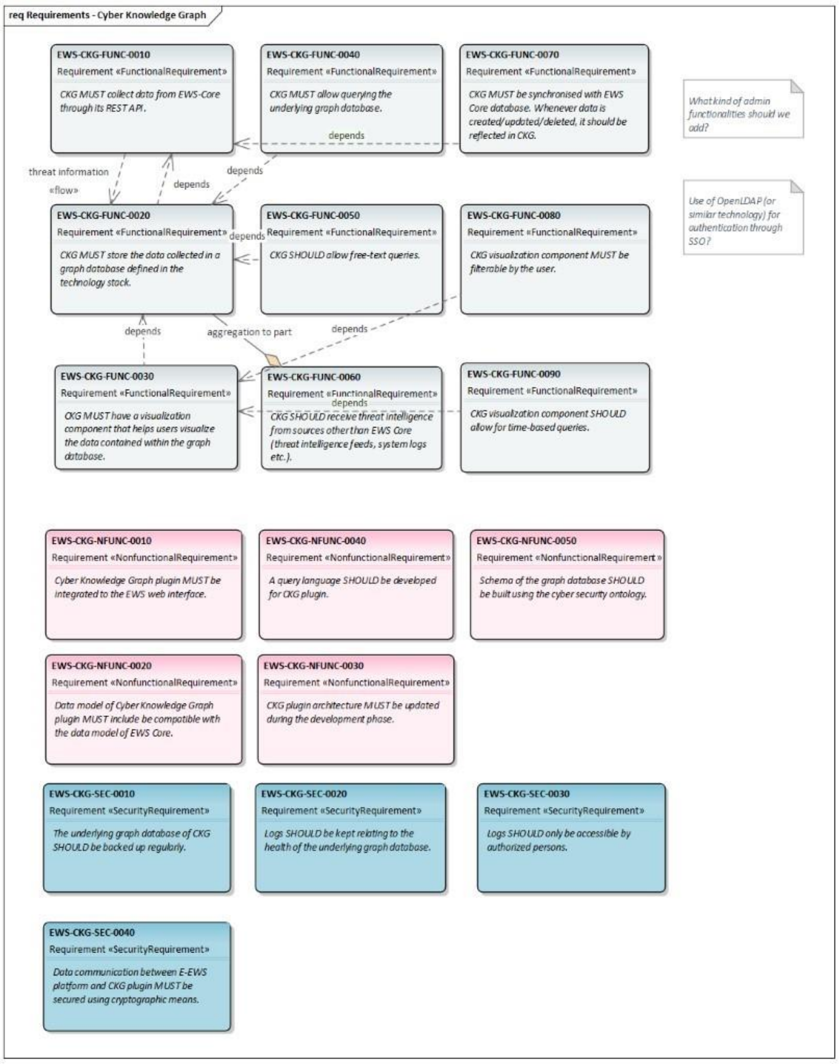
Developing a robust, enterprise-ready application requires well-defined software requirements. And in order to derive those requirements, first step is to define the capabilities of the software that is under development. The capabilities of the cyber knowledge graph shall be:

- Visualization of Cyber Threat Intelligence: visualization will help the users of the application have a better, quicker overview of what the threat comprises of
- Granularity: the aim is to use a data model as granular as possible while maintaining relevancy
- Data correlation: simple graph traversal queries will help users correlate data and act on the available data faster, reducing response time to threats
- Behavior analysis: the application will be constructed in such a way that, as the more data goes into it, the more apparent behaviors of threat actors will be.

In light of all of these required capabilities, software requirements were categorized into two:

- Functional Requirements: the requirements that relate to the functionalities of the application such as its database querying feature
- Non-functional requirements: the requirement that relate to the user experience and ease of operations including security

This categorization helps in establishing borders and order while developing. Also, it is useful when defining user stories. User stories are definitions derived from requirements that developers can use to implement that requirement. The following diagram shows the requirements defined for the Cyber Knowledge Graph in UML (*Unified Modeling Language*):



EWS-CKG-FUNC-0010
Requirement «FunctionalRequirement»
CKG MUST collect data from EWS-Core through its REST API.

EWS-CKG-FUNC-0040
Requirement «FunctionalRequirement»
CKG MUST allow querying the underlying graph database.

EWS-CKG-FUNC-0070
Requirement «FunctionalRequirement»
CKG MUST be synchronised with EWS Core database. Whenever data is created/updated/deleted, it should be reflected in CKG.

What kind of admin functionalities should we add?

EWS-CKG-FUNC-0020
Requirement «FunctionalRequirement»
CKG MUST store the data collected in a graph database defined in the technology stack.

EWS-CKG-FUNC-0050
Requirement «FunctionalRequirement»
CKG SHOULD allow free-text queries.

EWS-CKG-FUNC-0080
Requirement «FunctionalRequirement»
CKG visualization component MUST be filterable by the user.

Use of OpenLDAP (or similar technology) for authentication through SSO?

EWS-CKG-FUNC-0030
Requirement «FunctionalRequirement»
CKG MUST have a visualization component that helps users visualize the data contained within the graph database.

EWS-CKG-FUNC-0060
Requirement «FunctionalRequirement»
CKG SHOULD receive threat intelligence from sources other than EWS Core (threat intelligence feeds, system logs etc.).

EWS-CKG-FUNC-0090
Requirement «FunctionalRequirement»
CKG visualization component SHOULD allow for time-based queries.

EWS-CKG-NFUNC-0010
Requirement «NonfunctionalRequirement»
Cyber Knowledge Graph plugin MUST be integrated to the EWS web interface.

EWS-CKG-NFUNC-0040
Requirement «NonfunctionalRequirement»
A query language SHOULD be developed for CKG plugin.

EWS-CKG-NFUNC-0050
Requirement «NonfunctionalRequirement»
Schema of the graph database SHOULD be built using the cyber security ontology.

EWS-CKG-NFUNC-0020
Requirement «NonfunctionalRequirement»
Data model of Cyber Knowledge Graph plugin MUST include be compatible with the data model of EWS Core.

EWS-CKG-NFUNC-0030
Requirement «NonfunctionalRequirement»
CKG plugin architecture MUST be updated during the development phase.

EWS-CKG-SEC-0010
Requirement «SecurityRequirement»
The underlying graph database of CKG SHOULD be backed up regularly.

EWS-CKG-SEC-0020
Requirement «SecurityRequirement»
Logs SHOULD be kept relating to the health of the underlying graph database.

EWS-CKG-SEC-0030
Requirement «SecurityRequirement»
Logs SHOULD only be accessible by authorized persons.

EWS-CKG-SEC-0040
Requirement «SecurityRequirement»
Data communication between E-EWS platform and CKG plugin MUST be secured using cryptographic means.

Figure 6: Software Requirements for Cyber Knowledge Graph

These requirements were then used to derive user stories that show the basic functionalities of the application as well as how the user will interact with the application.

5.2 User Stories

User stories in Agile software development methodology are short and simple descriptions of a feature that the users of a system will use in order to perform an operation. They are generally written as “*As <user type>, I want to/I can <feature> so* *that <purpose>*". As explained above, the requirements were used to derive user stories. As an example, two of the user stories can be seen below with their short overviews:

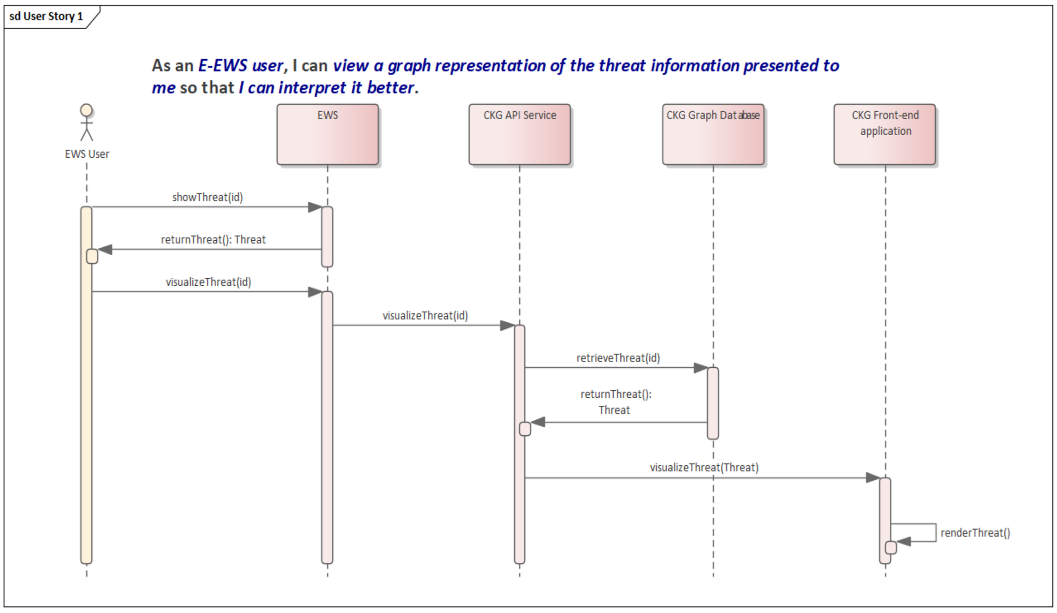


Figure 7: First User Story Example

This is a very simple core functionality that is relatively simple to implement. In this case, the EWS user wants to be able to see the graphical representation of a threat that he/she found through EWS interface. The components that will be involved in as well as their communication pattern for this task can also be seen in the above sequence diagram.

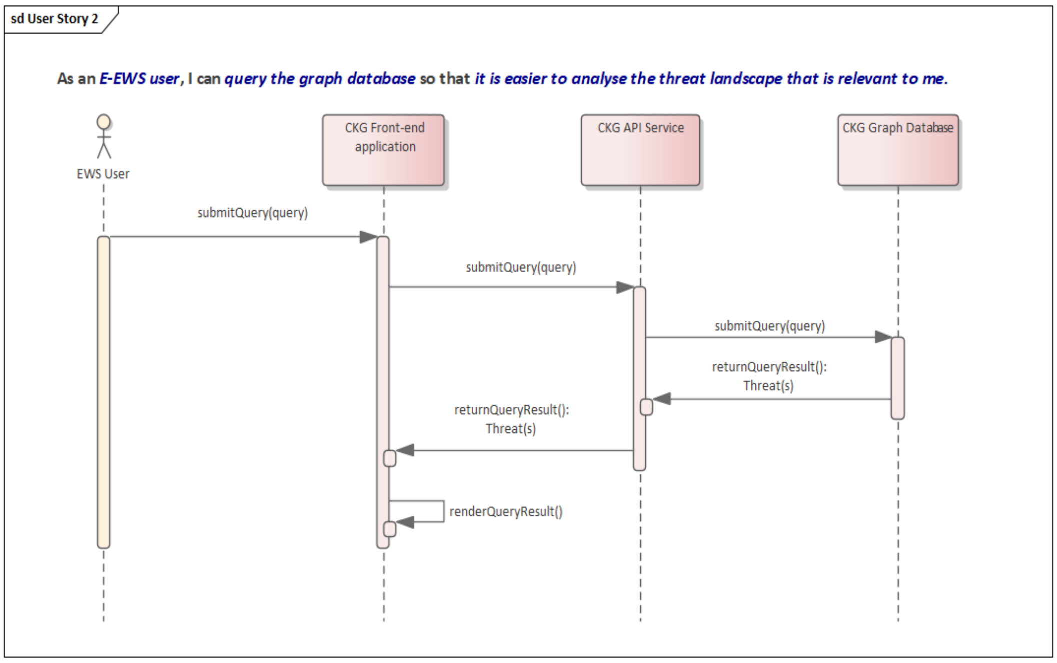

sd User Story 2
As an E-EWS user, I can query the graph database so that it is easier to analyse the threat landscape that is relevant to me.
EWS User
CKG Front-end application
CKG API Service
CKG Graph Database
submitQuery(query)
submitQuery(query)
submitQuery(query)
returnQueryResult(): Threat(s)
returnQueryResult(): Threat(s)
renderQueryResult()

Figure 8: Second User Story Example

The above user story is about querying the underlying graph database, which is another core functionality. The components involved and their foreseen communication can be seen in the figure.

5.3 Software Architecture

The architecture of the application needs careful work, evolvability and scalability. After thorough research on how to architecture the whole application, it was decided that the database be abstracted from all the other components as to avoid vendor lock-in. When it comes to graph databases, there are many choices within the industry and all of them offer different appealing features. However, in order to avoid licensing issues when it comes to deployment, the choice must be done carefully. Therefore, an abstraction layer is implemented with a TinkerPop Server. Apache TinkerPop [21] is an open source project licensed under Apache2 License. According to their documentation [22], TinkerPop is a graph abstraction layer over different graph databases and different graph processors. It allows developers to change the underlying graph database without much refactoring to their code base. This is one of the major features in the architecture of Cyber Knowledge Graph. As the cyber threats are changing and evolving faster, the goal is to have an application that can evolve and scale fast enough to keep up the pace.

The following UML diagram shows the container architecture of Cyber Knowledge Graph:

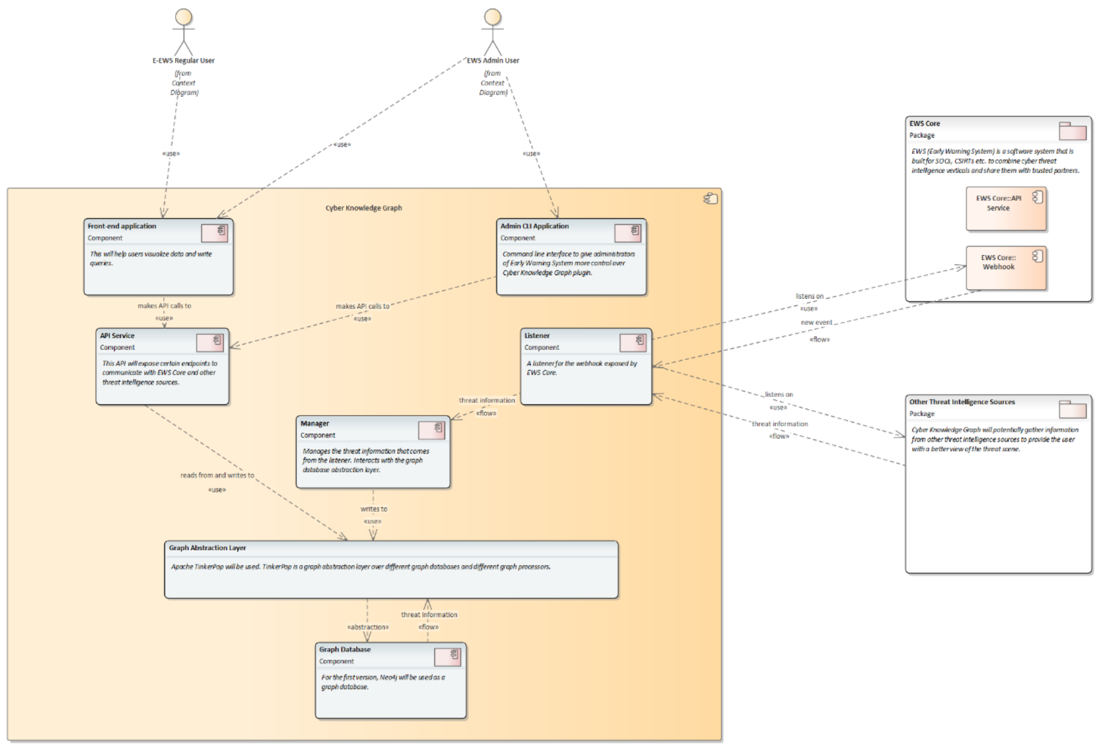

E-EWS Regular User
(from Context Diagram)
EWS Admin User
(from Context Diagram)
«use»
«use»
«use»
Cyber Knowledge Graph
Front-end application
Component
This will help users visualize data and write queries.
Admin CLI Application
Component
Command line interface to give administrators of Early Warning System more control over Cyber Knowledge Graph plugin.
makes API calls to
«use»
makes API calls to
«use»
API Service
Component
This API will expose certain endpoints to communicate with EWS Core and other threat intelligence sources.
Listener
Component
A listener for the webhook exposed by EWS Core.
threat information
«flow»
Manager
Component
Manages the threat information that comes from the listener. Interacts with the graph database abstraction layer.
reads from and writes to
«use»
writes to
«use»
Graph Abstraction Layer
Apache TinkerPop will be used. TinkerPop is a graph abstraction layer over different graph databases and different graph processors.
«abstraction»
threat information
«flow»
Graph Database
Component
For the first version, Neo4j will be used as a graph database.
EWS Core
Package
EWS (Early Warning System) is a software system that is built for SOCs, CSIRTs etc. to combine cyber threat intelligence verticals and share them with trusted partners.
EWS Core::API Service
EWS Core:: Webhook
listens on
«use»
new event
«flow»
listens on
«use»
threat information
«flow»
Other Threat Intelligence Sources
Package
Cyber Knowledge Graph will potentially gather information from other threat intelligence sources to provide the user with a better view of the threat scene.

Figure 9: Container Architecture Diagram for Cyber Knowledge Graph

A brief explanation regarding all the components within the application is given below:

- Frontend application: This is one of the most important components in the whole application as this is what the user will be interacting with for the whole duration they are using the application. Also, this is where the visualization will take place. For visualization, d3.js library [23] will be used. According to their website, d3.js is a JavaScript library that helps its users visualize data using HTML (*Hyper Text Markup Language*), SVG (*Scalable Vector Graphics*) and CSS (*Cascading Style Sheets*). D3.js includes an extension to represent labeled property graphs in a collapsible force-directed format. What this means is that the user will be able to collapse the relationships of a node and open them back again. This way, the user will have the ability to see the most relevant relationships between entities. One other use of this feature is that they can start with a simple query and take it from there to find their own way to meaningful information contained within the database. Force-directed means that users can change the view as they like by clicking and dragging certain nodes. This feature could help users in grouping certain nodes together to form a better picture. The nodes will also be color-coded. Certain types of nodes will be colored with the same color so as to allow users to differentiate between entities better. Nodes and relationship will be labeled and bi-directional relationship will be represented using curved links.

- Administrator CLI (*Command-Line Interface*) application: A CLI application will be developed for administrators of the cyber knowledge graph as the application will not be hosted on a central server, complying to what EWS will implement. EWS will eventually be a peer-to-peer network of instances that each organization will host on their premises. The deployment of each plugin will be optional. Users can decide whether or not they want to enable or disable a certain plugin. Therefore, an administrator application will be helpful in managing the database. It is understandable that this might not be very appealing to organizations as they would have to deal with two databases. However, the value this plugin will provide greatly surpasses the overhead it will cause. Administrator CLI application will support Create, Read, Update, Delete operations in the database and help administrators enable or disable certain feeds, e.g. MISP, and configure them accordingly with specific URLs (*Uniform Resource Locator*) the API (*Application Programming Interface*) calls should be made and their respective API keys.
- API Service: This service was designed to allow users to send their queries to underlying server. There will be an endpoint exposed to handle incoming queries and send a response back to the application in JSON format that will be rendered in the frontend accordingly.
- Listener: As one of the key components within the whole application, a listener will be the one to send 3rd party API calls to the feeds the user has subscribed to. For example, in case of MISP, this will be sending API calls at certain points during the day and passing the response, if any, from the call to the respective parser. In case of EWS, the architecture includes a webhook which all plugins can interface with using the SDK (*Software Development Kit*) provided. Therefore, it is fairly easier and not very expensive as whenever there is new information, EWS will trigger a new event for the plugin, which in turn will fetch the information and parse it accordingly. There will be multiple listeners

and parsers to handle the incoming information as all 3rd party APIs provide information in a different manner.

- Graph Abstraction Layer: An abstraction layer is deemed necessary to avoid vendor lock, meaning that it will be possible to change the underlying database without much refactoring to the code base. As mentioned above, Apache TinkerPop graph computing framework is used for this purpose. This not only prevents a lot of unnecessary work, when and if the database is changed, but also provides a common way to write queries with Gremlin query language that comes pre-packaged with TinkerPop.
- Graph Database: The database chosen for this application is Neo4j [24]. Neo4j was chosen as the underlying graph database as it is already TinkerPop-enabled meaning that it natively supports TinkerPop and the Gremlin server that is used to implement the graph abstraction layer already includes Neo4j configuration files, therefore it is very easy to get started. Moreover, Neo4j is one of the oldest and most robust graph databases in the industry. It has a great community and a selection of tools to complement it.
- 3rd Party Threat Feeds. As the aim is to bring as many threats together as possible, and MISP includes more than 60 threat feeds in itself [25], it is a great choice to begin with, alongside EWS.

5.4 Technology Stack

The technologies to be used for this project needed to be robust, compatible and also open for improvement so as to allow for the application to evolve in different ways. Also, there needs to be a well-established community using these technologies so that the users of the application will not have to learn much from scratch. Therefore, the selected technologies are the following:

- Underlying graph database: Neo4j
- Graph Abstraction layer: Apache TinkerPop Gremlin server
- Back-end development framework: Java Spring Boot [26]
- Front-end development framework: d3.js

All of these technologies have great support for developers and are commonly used for this kind of graph database applications.

5.5 Backend Development

As mentioned above, the backend development for Cyber Knowledge Graph is being carried out with Java Spring Boot 5 framework. For the development process, agile software development methodology was used. What this means is that the development process was split into certain timeframes, called sprints, all of which had a certain goal that needs to be achieved. The goals were pre-defined at the beginning of each sprint week. Since EWS is still in development phase as well, the focus for the first sprints was more on MISP rather than EWS. Some of the sprint goals can be found below:

- Goal: Implement graph abstraction layer on top of underlying graph database
    - The goal here was to avoid vendor-lock by using a certain graph database application, which will definitely be helpful in the future when, for any reason, the database needs to change. For this goal, a Gremlin server with embedded Neo4j database was used. The server runs when the application starts through a Docker container image.
- Goal: Fully synchronize the underlying graph database with MISP database when the application starts
    - Focus on this goal was on how the data, that comes through MISP REST (*Representational State Transfer*) API, will be requested. The solution was to develop a component that would send requests to certain MISP instance REST endpoints continuously until all the data that is in that certain MISP instance was also in the graph database. However, the data was put directly into the database, hence there were no relations between any of the data points.
- Goal: Keep the databases synchronized throughout the lifecycle of the application
    - This goal was selected solely based on the fact that there was a need to update the database whenever there was new data within the MISP instance. This was planned to be achieved with a scheduler task and continuous API

calls as MISP does not provide users with a webhook mechanism that would trigger an event when subscribed as EWS will.

- Goal: Convert EWS relational data model to graph data model
  - Considering the fact that EWS has been built on a relational data model, there was a need to convert this data model into a labeled property graph model in order to efficiently use it within cyber knowledge graph before the ontology was integrated.
- Goal: Create a parser to convert EWS DTOs (*Data Transfer Object*) to graph data model representation
  - This goal is the code implementation of Sprint 4 goal where the EWS relational data model was converted to a graph data model.

5.6 Frontend Development

Currently, frontend development for the cyber knowledge graph has not started. The initial idea is to start from an open source visualizer for Gremlin databases, such as the GraphExp project Benjamin Ricaud [27] that can be found on GitHub software development platform. Another nice option is Gremlin Visualizer by Umesh Prabushitha Jayasinghe [28], which is a project to visualize a graph network corresponding to a Gremlin query.

For now, Neo4j graph visualizer server the purposes as visualizer is the next step in development. As mentioned above, the ECHO project lasts about 4 years and not even the 1st version of EWS has been released at the time of this writing. An example Neo4j representation of a cyber threat can be seen in the following figure:

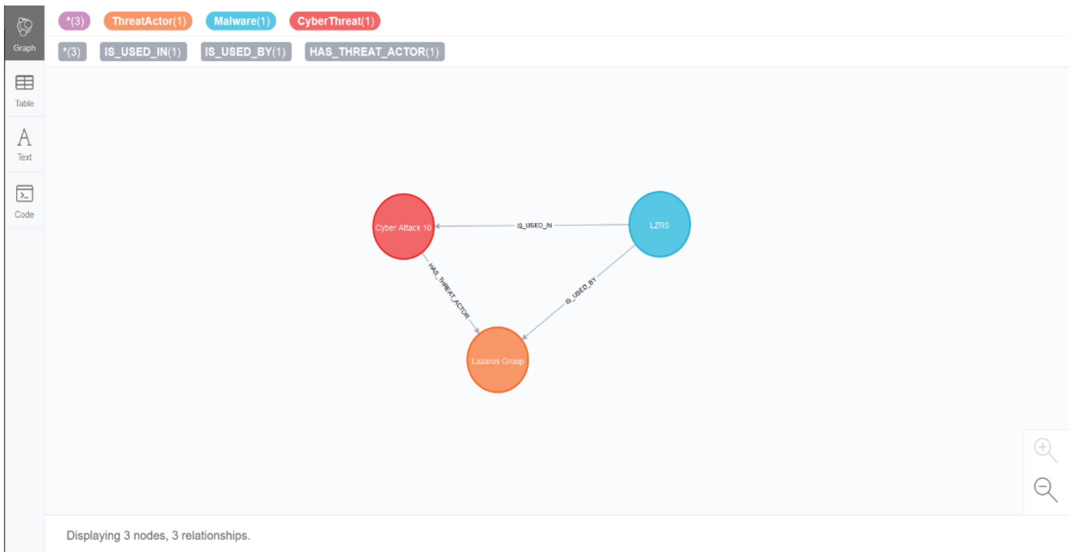


Figure 10: Representation of a Cyber Threat on Neo4j Graph Visualizer

The above figure is a representation of an example 3-node cyber threat with the relations in between. This is a limited view as the end goal is to represent the strength of a connection in between the nodes with scores. However, all of these foreseen features will be addressed in the next section.

6 Further Work

The idea could be improved in many ways, mainly relating to the analysis of the information. One way to do this is to include temporal information within all the vertices and show different types of graphs based on, for example, when a malware was first seen and how long it has been in use etc. This could prove helpful in the case that if the threat has been seen before many times in other sources, there is most likely a solution to it. Since security professionals will not waste time on looking for a novel solution, this could reduce their response times dramatically.

The ontology that has been created to serve as the data model of the underlying graph database must be improved in order to keep the balance between quantity and quality. The work on this is already ongoing.

As most of the threat intelligence feeds or applications, such as EWS or MISP, is using a relational data model, they mostly lack the analytical and visualization features of graph databases. Therefore, the data models and the databases of these applications must be duplicated. However, this creates many security loopholes as the applications have different architectures and might be working on different principles when it comes to user authentication and authorization. In order to avoid the security issues, such as a user querying data through Cyber Knowledge Graph that he/she is not authorized to see, the necessary precautions must be taken and tested carefully.

7 Summary

This paper has analysed a new way of representing cyber threat intelligence using graph databases. Threat intelligence data is highly interconnected, which makes it a perfect match for graph databases. Graph databases has many advantages compared to the other database solutions when the initial data is unstructured. Whenever a new concept appears or an existing one dramatically changes, it can be represented easily in graph databases without much overhead. Also, they perform much better when the data is interconnected as shown in the analysis part of this paper.

A cybersecurity ontology has been created to serve as a data model for graph database applications. It currently includes 450 nodes and around 500 relationships between those nodes. It needs to be improved over time. Also, the development of a graph database application that will eventually be used as an Early Warning System plugin for ECHO Project has started and well ongoing at the time of this writing. It will serve as both an analysis and a visualization tool for threats shared between EWS partners and from other feeds, such as MISP.

References

1. Shu, Xiaokui & Tian, Ke & Ciambrone, Andrew & Danfeng, & Yao,. (2017). Breaking the Target: An Analysis of Target Data Breach and Lessons Learned.

2. Abu, Md & Rahayu, S. & Ariffin (DrAA), Dr Aswami & Robiah, Y.. (2018). Cyber threat intelligence – Issue and challenges. Indonesian Journal of Electrical Engineering and Computer Science. 10. 371-379. 10.11591/ijeecs.v10.i1.pp371-379.

3. MITRE Organization, 2018, MITRE Corp., accessed 21.02.2020, <https://www.mitre.org/research/technology-transfer/technology-licensing/cygraph>

4. MISP Project, 2020, MISP Project, accessed 10.03.2020, https://www.misp-project.org/

5. Ian Robinson, Jim Webber, and Emil Eifrem. 2015. Graph Databases: New Opportunities for Connected Data (2nd. ed.). O'Reilly Media, Inc.

6. Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox, and Jonas Partner. 2014. Neo4j in Action (1st. ed.). Manning Publications Co., USA.

7. Facebook, 2020, Open Graph Protocol, accessed 13.05.2020, <https://ogp.me/>

8. Noel, Steven & Harley, E. & Tam, K.H. & Limiero, M. & Share, M.. (2016). CyGraph: Graph-Based Analytics and Visualization for Cybersecurity. 10.1016/bs.host.2016.07.001.

9. OASIS Cyber Threat Intelligence (CTI) TC, 2020, OASIS, accessed 04.04.2020, <https://docs.oasis-open.org/cti/stix/v2.1/csprd01/stix-v2.1-csprd01.html>

10. OASIS Cyber Threat Intelligence (CTI) TC, 2020, OASIS, accessed 04.04.2020, <https://oasis-open.github.io/cti-stix-visualization/>

11. World Wide Web Consortium, 2019, World Wide Web Consortium, accessed 07.04.2020, <https://www.w3.org/2001/sw/wiki/Main_Page>

12. OWL Working Group, 2012, World Wide Web Consortium, accessed 07.04.2020, <https://www.w3.org/2007/OWL/wiki/OWL_Working_Group>

13. Musen, Mark, 2015, The protégé project, AI Matters, 1, 4-12, 10.1145/2757001.2757003.

14. Stanford Medicine, 2020, Stanford Center for Biomedical Informatics Research, accessed 07.04.2020, https://bmir.stanford.edu/

15. MISP, 2020, MISP Taxonomies, v1.0, accessed 10.04.2020, <https://github.com/MISP/misp-taxonomies>

16. Ebiquity, 6 May 2019, Unified Cybersecurity Ontology, v1.0, accessed 10.04.2020, https://github.com/Ebiquity/Unified-Cybersecurity-Ontology

17. Lohmann, S.; Link, V.; Marbach, E.; Negru, S.: WebVOWL: Web-Based Visualization of Ontologies. Proceedings of EKAW 2014 Satellite Events, LNAI 8982, pp. 154-158, Springer, 2015

18. ESG Research Group, 2015, ESG Research Report: Threat Intelligence and Its Role Within Enterprise Cybersecurity Practices, ESG Research

19. ECHO Project, 2020, ECHO, accessed 15.04.2020, <https://echonetwork.eu/>

20. European Commission, 2020, Horizon 2020 | The EU Framework Programme for Research and Innovation, accessed 15.04.2020, <https://ec.europa.eu/programmes/horizon2020/en>

21. Apache Foundation, 2020, Apache TinkerPop, accessed 16.04.2020, <https://tinkerpop.apache.org/>

22. Apache Foundation, 2020, Apache TinkerPop documentation, accessed 16.04.2020, <https://tinkerpop.apache.org/docs/current/>

23. Mike Bostock, 2020, d3.js, accessed 16.04.2020, <https://d3js.org/>

24. Neo4j, Inc., 2020, Neo4j Graph Platform, accessed 16.04.2020, <https://neo4j.com/>

25. MISP Project, 2020, MISP Default Feeds, accessed 16.04.2020, <https://www.misp-project.org/feeds/

26. VMware Inc, 2020, Spring Boot, accessed 16.04.2020, <https://spring.io/projects/spring-boot>

27. Benjamin Ricaud, 15 April 2020, GraphExp, accessed 13.05.2020, <https://github.com/bricaud/graphexp>

28. Umesh Prabushitha Jayasinghe, 7 January 2020, Gremlin Visualizer, accessed 13.05.2020, <https://github.com/prabushitha/gremlin-visualizer>