الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي و البحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
كلية العلوم الدقيقة والتطبيقية
Faculté des sciences exactes et appliquées
قسم الإعلام الآلي
Computer Science Department

Thesis

For Obtaining a bachelor's degree
In

Mathematics & Informatics
Option: *Information System*

Presented By:

ASLAOUI MOHAMED EL HABIB DJAMEL
ABBAS IKRAM
Session 1 2023

THEME

IMPLEMENTATION OF KNOWLEDGE GRAPHS FOR MACHINE LEARNING IN CYBERSECURITY

Supervised by: Mostefa MOKADDEM
Karima MOKHTARI

.

Jury

Examiner:
S. BENABDALLAH

Bachelor Code: …. /2023

Promotion 2022/2023

Acknowledgments

In the Name of ***ALLAH,*** the Most Gracious and the Most Merciful.

We thank God for helping us accomplish this modest work.

We would like to take this opportunity to express our deepest thanks and gratitude to our supervisors Mr. Mokaddem and Mme. Mokhtari for their availability, advice, guidance, and their encouragement throughout our research.

We would like to express our sincere gratitude to the members of the jury who agreed to evaluate and examine this work, particularly to:

S. BENABDALLAH

Finally, our warmest thanks to everyone who has participated in any way in the success of this modest work.

Abstract

The world has never been more connected than it is today, becoming heavily reliant on the internet, whether it's for their job or personal needs. As a consequence of our growing dependency on these networks of digital systems, the variety and scale of cyber-attacks and threats has risen, making its security obligatory. Cybersecurity plays a major role in the safety of every system and application, striving to protect an individual's or a large organization's important and sensitive data, and ensuring their privacy. This field requires speed and an immense amount of variant data to produce reliable and performant anti-infiltration systems. The novel approach of cybersecurity knowledge graphs has shown a great potential in cybersecurity attack detection, due to their capabilities in knowledge management, aggregation, representation, and reasoning. In this project, we focus on one of the solutions that's still at the study stage in cybersecurity, which is the implementation of cybersecurity knowledge graphs to enhance attack detections. The main aim of this project is to set up a NoSQL database store that would stock cybersecurity knowledge graphs. Using the existing cybersecurity datasets, such as the NSL-KDD and DNS, they will be converted to RDF triples, and would later be used to build a grand knowledge graph for knowledge visualization in order to identify, find patterns, and track attack paths. This will be in preparation for later projects, where we will gradually add other related knowledge from other resources, to aid cybersecurity to build better and powerful cybersecurity knowledge graphs for intelligent vulnerability detection, that will support existing Machine Learning solutions for a stronger and reliable intrusion detection system.

KEYWORDS: Cybersecurity, NoSQL Store, Datacenter, RDF Graph, Knowledge Graphs., Intrusion Detection Systems, NSL-KDD.

Résumé

Le monde n'a jamais été aussi connecté qu'aujourd'hui, devenant fortement dépendant d'Internet, que ce soit pour leur travail ou leurs besoins personnels. En raison de notre dépendance croissante à l'égard de ces réseaux de systèmes numériques, la variété et l'ampleur des cyberattaques et des menaces ont augmenté, rendant sa sécurité obligatoire. La cybersécurité joue un rôle majeur dans la sécurité de chaque système et application, en s'efforçant de protéger les données importantes et sensibles d'un individu ou d'une grande organisation, et en garantissant leur confidentialité. Ce domaine nécessite de la vitesse et une immense quantité de données de variantes pour produire des systèmes anti-infiltration fiables et performants. La nouvelle approche des graphes de connaissances sur la cybersécurité a montré un grand potentiel dans la détection des attaques de cybersécurité, en raison de leurs capacités de gestion, d'agrégation, de représentation et de raisonnement des connaissances. Dans ce projet, nous nous concentrons sur l'une des solutions encore à l'étude en cybersécurité, qui est la mise en place de graphes de connaissances en cybersécurité pour améliorer les détections d'attaques. L'objectif principal de ce projet est de mettre en place un magasin de base de données NoSQL qui stockerait des graphes de connaissances en cybersécurité. En utilisant les ensembles de données de cybersécurité existants, tels que le NSL-KDD et le DNS, ils seront convertis en triplets RDF et seront ensuite utilisés pour créer un grand graphe de connaissances pour la visualisation des connaissances afin d'identifier, de trouver des modèles et de suivre les chemins d'attaque. Ce sera en préparation de projets ultérieurs, où nous ajouterons progressivement d'autres connaissances connexes à partir d'autres ressources, pour aider la cybersécurité à créer des graphiques de connaissances de cybersécurité meilleurs et puissants pour la détection intelligente des vulnérabilités, qui prendront en charge les solutions d'apprentissage automatique existantes pour un plus fort et système de détection d'intrusion fiable.

MOTS CLÉS : Cybersécurité, NoSQL store, datacenter, graphe RDF, graphes de connaissances, systèmes de détection d'intrusion, NSL-KDD.

الملخص

لم يكن العالم أكثر ارتباطًا مما هو عليه اليوم، حيث أصبح يعتمد بشدة على الإنترنت، سواء كان ذلك من أجل وظيفتهم أو احتياجاتهم الشخصية. نتيجة لاعتمادنا المتزايد على هذه الشبكات من الأنظمة الرقمية، ازداد تنوع وحجم الهجمات والتهديدات السيبرانية، مما يجعل أمنها أمرًا إلزاميًا. يلعب الأمن السيبراني دورًا رئيسيًا في سلامة كل نظام وتطبيق، ويسعى جاهداً لحماية البيانات المهمة والحساسة للفرد أو المؤسسة الكبيرة، وضمان خصوصيتهم. يتطلب هذا المجال سرعة وكمية هائلة من البيانات المتغيرة لإنتاج أنظمة موثوقة وفعالة لمكافحة التسلل. أظهر النهج الجديد للرسوم البيانية المعرفية للأمن السيبراني إمكانات كبيرة في اكتشاف هجمات الأمن السيبراني، نظرًا لقدراتها في إدارة المعرفة، والتجميع، والتمثيل، والاستدلال. في هذا المشروع، نركز على أحد الحلول التي لا تزال في مرحلة الدراسة في مجال الأمن السيبراني، وهو تنفيذ الرسوم البيانية للمعرفة بالأمن السيبراني لتعزيز عمليات اكتشاف الهجمات. الهدف الرئيسي من هذا المشروع هو إنشاء مخزن قاعدة بيانات NoSQL من شأنه أن يخزن الرسوم البيانية المعرفية للأمن السيبراني.
باستخدام مجموعات بيانات الأمن السيبراني الحالية، مثل NSL-KDD و DNS، سيتم تحويلها إلى ثلاثة أضعاف RDF، وسيتم استخدامها لاحقًا لإنشاء رسم بياني معرفي كبير لتصور المعرفة من أجل تحديد، والعثور على الأنماط، وتتبع مسارات الهجوم . سيكون هذا استعدادًا لمشاريع لاحقة، حيث سنضيف تدريجيًا المعرفة الأخرى ذات الصلة من الموارد الأخرى، لمساعدة الأمن السيبراني في بناء رسوم بيانية أفضل وأقوى للمعرفة في مجال الأمن السيبراني لاكتشاف الثغرات الأمنية الذكية، والتي ستدعم حلول التعلم الآلي الحالية للحصول على أقوى وأقوى. نظام موثوق للكشف عن التسلل.

<u>الكلمات الدالة:</u> الأمن السيبراني، NoSQL ، مركز البيانات ، الرسم البياني RDF ، الرسوم البيانية المعرفية ، أنظمة كشف التسلل ، NSL-KDD.

Table of Contents

INTRODUCTION

As the scale of the cyberspace is moderately increasing, it is gradually shifting from just a simple traditional internet to a more complex infrastructure, comprising interactions between network information systems such as smartphones and databases, human social systems such as healthcare, and industrial physical systems like aviation. Despite all the advantages and opportunities this provides for social development, it puts the cyberspace at a major disadvantage as it creates opportunities for attackers to find vulnerabilities and infiltrate systems, creating damages and discrepancies. Given that the data has, and is only continuing to, become more diverse, it had made it difficult for cybersecurity staffs to quickly track down the information they need. This creates a major problem as it will hinder and slow down effective data analysis, correlating data and identifying patters. Where instead, these elements are needed to be functional and quick to locate at all times, as they play a major role and are extremely necessary for attack detections (Llorens, 2021).

Among the cyberspace solutions was the cybersecurity assessment and analysis models, such as the attack graph and attack tree model. They were designed to be used to actively evaluate, analyze, and assess specific aspects of security in a network, such as vulnerabilities, risks, or system performance, consequently coming up with ways to reduce the risks depending on the results that were provided by the analysis. This however had a massive drawback, seeing how networks only continue to grow, resulting in data to become extremely vast, massive, and heterogenous. This has made the existing methods impossible to provide knowledge representations and reasoning (Zhang & Liu, 2020).

Another approach was incorporating datasets with machine learning in cybersecurity. A lot of work has been devoted to applying machine learning techniques in intrusion detection systems (IDS) in order to reduce malicious activity by improving the detection (Tsai, Hsu, Lin, & Lin, 2009; Mukkamala, Janoski, & Sung, 2002). The evolution of the IDS model implements Machine Learning (ML) and Data Mining (DM) techniques to classify the network traffic into malicious and healthy traffic flow. The feature of datasets helped the IDS classifier learn the basic traffic patterns in addition to the attack patterns through which the classifier can eventually classify the input data (Thakkar & Lohiya, 2020). In previous works, the AIR team proposed two Deep Learning methodologies. an Artificial Neural Network (ANN) and a Recurrent Neural Network (RNN) with different feature selection methods. These two methodologies both used the NSL-KDD dataset (Chaibi et al., 2020). Another Convolutional Neural Network (CNN) based Network Intrusion Detection System (NIDS) has been developed. The CNN was trained using the NSL-KDD dataset (Chaibi et al, 2021, 2022).

That is good for intrusion detection, however, IDS systems were found to occasionally generate a large number of false positive alerts. Furthermore, it doesn't help in anyway in restoring the full picture of attack behavior. It instead hides very useful expert knowledge relations in cybersecurity.

Cybersecurity defenses must be at all times tested to ensure protection. Scanning the network and locating vulnerabilities is not enough when it comes to complex environments. This is because complex cyber-attacks require an understanding of how the vulnerabilities interact with each other, their context, and their implications in order to choose the appropriate response strategy. Cybersecurity analysis research has shifted its focus towards being able to extract correlations and potential attacks from threat intelligence data. Technologies based on knowledge modeling, such as semantic reasoning, have become the novel solutions under the context of big data (Sikos et al., 2019). This proves that the problem is not the lack and shortage of available data for cybersecurity but instead, the issue lies on how to collect these diverse pieces of data and heterogeneous information scattered in various areas, and combine them in order for a better and a quicker decision making and possible solutions (Liu et al., 2022).

Cybersecurity knowledge graphs (CSKG) have proven to have potential to work well with complex environments, as they seem to provide a clear organization of these diverse, complex, and heterogeneous pieces of data. They also proved their ability to manage and reason with that knowledge presented (Zhang & Liu,2020).

Although the term of knowledge graphs (KGs) in Cybersecurity has captured the attention of many researchers and scholars, most of the research that has been done only revolves around how to construct a complete knowledge graph. There have been little to no attempts in putting the research into practice. It is therefore evident that there is a lack of CSKGs implementation in the cyberspace.

Currently, Cybersecurity information and data are represented in multiple different formats; since the information is heterogeneous, therefore it becomes difficult to integrate. So, the idea of using KGs in cybersecurity will open up opportunities to unify this knowledge into one common format, making it easy to detect intrusions, visually track down attack paths, and detect attack patterns. Therefore, it is necessary for cyberspace to enhance the use of CSKG in order to achieve quick and effective intrusion detection. The trivial approach is to use existing, precomputed datasets, since they have been proven to work well with ML and DM, and to convert these datasets to CSKGs.

Our contribution will focus on

- How we can visualize CSKGs obtained from several existing datasets,
- Demonstrate how information could be extracted from the CSKGs in an orderly manner, to efficiently identify and detect potential attack paths and cyber threats.
- Briefly touch on how the CSKGs produced will later be of use for intelligence vulnerability detection, that will support existing Machine Learning solutions.

The structure of this document:

Chapter 1: We are going to explore the basic concepts and principles of cybersecurity and KGs in the context of cyberattacks and big data for knowledge extraction. As well as the several tools to facilitate this purpose.

Chapter 2: We will begin by introducing the Proxmox Virtual Environment as a simple Cloud Knowledge Graph based on cybersecurity solutions and discuss the importance of virtual environment in cybersecurity to handle big data and KGs. Side by side, we will show an Oracle NoSQL store deployment to fit a distributed Knowledge Graph implementation with the Jena API in an apart gateway approach. Also, a Django solution is deployed as a Proxmox container to let using the API Rest Framework that Django provides for client applications.

Chapter 3: We will put altogether the previously discussed techniques and tools by providing a knowledge graph visualization in an aside client application. More details will be given related to our code implementation and execution.

Conclusion: an overall summary of the relevant obtained results will be presented.

Chapter 1

Context and Concepts

In order to better the use of Knowledge Graphs in Cybersecurity, the main concepts related to Cybersecurity and Knowledge Graphs should be well mastered. In the next sections, Cybersecurity concepts, as well as Knowledge Graphs concepts are explored. Finally, a combination of these concepts shows the survey of Knowledge Graphs in Cybersecurity.

1 Cybersecurity concepts

Computer systems are at constant risk of intrusion, theft and deletion. Cybersecurity strives to protection of internet-connected devices against all types of cyberattacks and threats which are mainly designed to access, destroy, and alter organizations/users' systems and sensitive data.

Cyber threats continue to grow at an exponential rate, with large numbers of data theft occurring yearly. Therefore, cybersecurity has only evolved in turn finding newer solutions to protect organizations and its employees from these sophisticated, complex, and malicious cyberattacks. The goal of a powerful cybersecurity strategy is to provide a functional barrier against these malicious attacks. We need cybersecurity to minimize crime and harm in the virtual world, and thus, cybersecurity is the practice used to defend computers, mobile devices, systems, servers, networks, and more importantly, data from harmful attacks.

1.1.1 Security attributes in information systems

The scope of cybersecurity evolves as fast as the capabilities of computing, it uses a range of techniques to protect the confidentiality, integrity, and availability of computer systems and data against cyber threats (Zhang, K., & Liu, J., 2020).

1.1.1.1 Confidentiality

Confidentiality is the art of allowing only authorized people to have the right to access specific computer systems and data. The attack comes in the form of targeting data confidentiality and privacy, such as data breaches by hackers, who take advantage of peoples' credit cards and reveal the information in it [1].

1.1.1.2 Integrity

Integrity refers to making sure that the data received is unaltered and reliable. It is therefore ensuring that only authorized people have the ability to use, modify, and alter data and systems in order for the data to be dependable, accurate, and legitimate [2]. An integrity attack will be equivalent to hackers sending emails masquerading as the owner due to them figuring out their password.

1.1.1.3 Availability

Availability is the regular accessibility of authorized users to their systems and data at any given time. A network may become inaccessible and unusable due to an attack on availability such as a Distributed Denial of Service (DDoS) attack, where hackers overload a certain website with fake requests to slow down the site and make it unreachable for users as its resources are depleted. This would be called an attack on the servers' availability [2].

Table 1 summarizes how the loss of any of these three essential elements of security in a system can have an enormous effect on the overall risk on the person/organization's system. Just because one of the risk events (loss of confidentiality) was recorded as 'High Risk' due to the likelihood of a potential threat event to be possible, and the impact of that threat event to being severe, it had resulted in the overall risk level for the system to become high, regardless of the other risk levels being rated fairly low [3].

System: IT Admin Laptop			
Threat Event	**Likelihood**	**Impact**	**Risk Level**
1. Loss of Confidentiality	Possible	Severe	**HIGH**
2. Loss of Integrity	Unlikely	Minor	**LOW**
3. Loss of Availability	Possible	Significant	**MODERATE**
		Overall Risk:	**HIGH**

Table 1: The measure of the overall risk of an IT Admin Laptop at the possibility of a threat event.

1.1.2 Securing computer systems

To achieve these security attributes (confidentiality, integrity, and availability) and ensure the security of a system, one example of the effective methods used by cybersecurity is the threat model. Threat modeling is a structured process enabling the identification of security requirements and locate possible vulnerabilities and security threats, quantify threat and prioritize reparation methods [4]. Threat models therefore allow preparation against specific threats; this is done by profiling attackers by acknowledging their capabilities, goals, and potential means of attack. How a system is protected and secured depends entirely on what exactly it is being secured against.

1.1.2.1 Authentication

There are multiple methods for protecting computer systems, networks and data. However, authentication is one of the many methods wildly used, as it limits access only to the appropriate person, and denies access to the wrong user.

Authentication is the process by which a computer is able to articulate who it's interacting with. Generally, there are three factors of authentication: knowledge factor (something you know), possession factor (something you have), and inherence factor (something you are) (William, K.& Aspen, O., 2017).

- **Something you know:** this type of authentication is based on the knowledge of information that should uniquely be known by the real user and the computer. This is the most used method of authentication as it’s the simplest to implement. The most popular example would be a password or a personal identification number (PIN).
- **Something you have:** this type of authentication is based on acquiring a secret token that only the real user possesses. Unlike in the previous example where password could be easily cracked, this type of authentication typically requires physical presence of the user, therefore it will make it much more difficult for remote attackers to gain access. An example of this would be a multi factor authentication (MFA) token or a smart card, such as a bank card.
- **Something you are:** this type of authentication is based on the user themselves. The user authenticates themselves to their computer device with the help of biometric authenticators, such as fingerprint readers or iris scanners.

Figure 1 below provides an overview of the multi factor authentication [5].

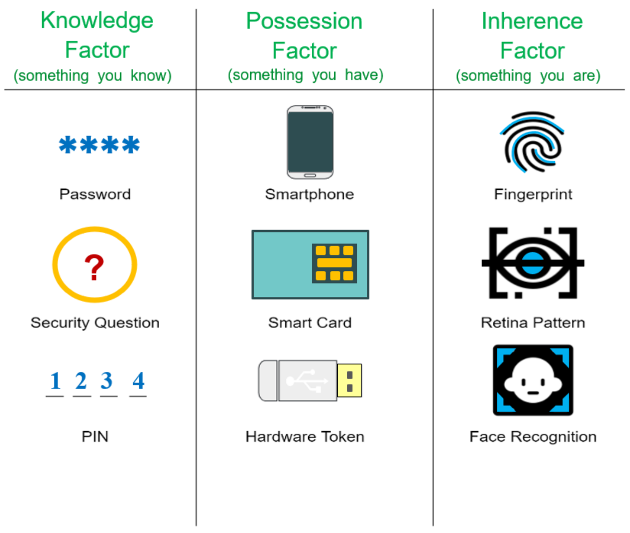


Figure 1: Multi factor authentication.

However, it is important to note that the knowledge factor and possession factor possess the property of being deterministic, meaning your input is either right or wrong with no in between, whereas the inherence factor possesses the property of being probabilistic, meaning there exist a chance the system may not recognize you.

It is encouraged by cybersecurity for the user to opt for two or more forms of authentication (known as two-factor or multi-factor authentication) when using important accounts that hold sensitive data. This way, the user can guarantee security, as an attacker is less likely to guess both authentications correctly.

1.1.2.2 Access control

Authentication is naturally followed by access control. Once a system identifies who the user is; the next step is to identify the assets that the user is limited to accessing, and for that, there is a specification on who should be able to see, modify and use what. This is possible through permissions or Access Control Lists (ACL) [6], which has the ability to describe what right each user has to perform on every file, folder and program on a computer by granting them permission.

1.1.2.3 Permissions

There are mainly three permissions known to be given to a user, which are: read, wri te and execute permissions. The read permission allows the user to see the contents of a file or a folder, while the write permission allows the user to modify the files contents. As for the execute permission, it allows the user to run a file, like a program for example.

Authentication and access control are the essential components to helping a computer determine who the user is and what the user should access. However, in the end it will all depend on being able to trust the hardware and software, that run the authentication and access the control programs, to function properly and accordingly. In many cases theses key components are disrupted when the hardware and software are exposed to cyber threats that gain control and target, specifically, the computer systems.

1.1.3 Types of cyber threats

There are three main types of threats, among others, widely known in cybersecurity:

1- **Cybercrime:** usually includes one or a group of actors targeting systems for financial gain or to cause disruption.

2- **Cyber-attack:** which is the act of gathering information with a malicious intent, it is usually politically motivated.

3- **Cyberterrorism:** it is usually intended to undermine systems to cause fear or panic.

The most common methods to gain control of computer systems and threaten cybersecurity are [7]:

1.1.3.1 Malware

Malware (or malicious software) is a software developed by a hacker to disrupt and damage a user's computer. This is done using deceit as this software is spread by an unsolicited email, or a downloadable link.

The different types of malwares include:

- **Virus:** which is a program able to self-replicate and attach itself to safe files and spread throughout a computer system infecting other files with malicious code.

- **Trojans:** This is a type of malware that is identical to a legitimate software which cybercriminals use to their advantage in order to trick a user into uploading Trojans into their computer. This enables hackers to collect data or cause permanent damage.
- **Ransomware:** this is a malware which blocks access to the user's files and data with the threat of publishing or erasing the victim's data unless a sum of money is paid.
- **Botnets:** these are networks of malware infected computers with bots which enables cybercriminals to perform tasks online without the user's permission. A good example of this is the Distributed Denial of Service (DDoS) attack.

1.1.3.2 Phishing
Phishing is used by criminals to target victims with attacks in a form of emails that appear to have come from a legitimate company, in an attempt to trick people into handing over their personal information such as their credit card data.

1.1.3.3 Distributed Denial of Service (DDoS) attacks
Its aim is to make online services unavailable by bombarding it with excessive traffic from many different locations and sources. Consequently, the website becomes slow in its response, depleting its resources and preventing access. Although DDoS attacks may not be considered a primary cybercrime, it aids in disruption as it acts as a distraction for crime to occur, such as fraud and other cyber intrusions.

1.1.4 Cybersecurity solutions
To achieve protection against these cyber threats, there exist multiple types of security which include: Network security, cloud security, mobile security, and many more.

1.1.4.1 Network security
The majority of attacks are facilitated by the network. Network security is therefore designed to acknowledge these attacks and identify them to block them. Some of the known solutions are data and access controls such as Identity Access Management (IAM), Next-Generation Firewall (NGFW), Network Access Control (NAC), and Data Loss Prevention (DLP). These application controls help to guarantee a safe web space.

1.1.4.2 Cloud security
As organizations continue to integrate cloud computing, a cloud security strategy ensures controls and services, such as data encryption, firewall protection, monitoring, and many more, which help provide protection to the entire organization cloud deployment including their data, applications, and infrastructure against potential attacks.

1.1.4.3 Mobile security
Mobile devices such as smart phones and tablets have access to sensitive corporate data, which puts businesses at risk in being exposed to malicious attacks in the form of phishing, zero-day, and Instant Messaging (IM) attacks. Mobile security included Mobile Device Management

(MDM) within devices as a solution to enable organizations to ensure that only valid and compliant devices have access to sensitive and corporate resources.

1.1.4.4 Knowledge graphs

One of the novel approaches is the use of CSKGs. CSKG, just like a KG, is made up of nodes and edges that capture the large-scale security semantic network. They showed the ability to provide an intuitive modeling method for multiple attacks and defense scenarios (Liu et al., 2022,2). As the amount of cybersecurity relevant data generated in the cyberspace increases, it makes it difficult to locate the information needed quickly. Therefore, where time plays an essential role in determining the effectiveness of locating cyberattacks, KGs is the solution to this problem in cybersecurity. They're used to manage, organize, and manipulate large amounts of heterogenous information in the cyberspace, and therefore making the tracking down of information much quicker.

KGs are efficient in providing a layout to the attack paths and identifying them as well as identifying their patterns, and therefore they facilitate in the representation of connection to differentiate between intrusions from normal connections, and thus detecting attacks and threats, such as the CyGraph (Zhang, K., *&* Liu, J., 2020).

1.2 Knowledge graph concepts

Introduction

Despite the massive progress made by researchers in modeling and analyzing network traffic, attackers continue to find new ways to launch attacks; therefore, a new approach to finding solutions must be paved. An enterprise for example possesses databases containing sensitive data and multiple services, such as internet access and business systems, therefore, it should create a safe environment where the availability of these network resources is augmented, yet at the same time, be able to protect the system's integrity from malicious attacks.

Although attacks can't always be detected or prevented, a resilient system should be built to gather as much knowledge and information about an enterprise's activity and monitor the amount of data flowing through the networks to be able to detect possible problems and the potential effects it can have on the organization to be able to figure out how and where to respond to these threats, in order to provide fast and optimal plan of action to best defend against these threats (Choudhury et al., 2018).

The concept of knowledge graphs (KGs) gained popularity when it was first launched by Google in 2012, to enhance the capabilities of its research engine. This aided analysts to gather knowledge and quickly react upon that knowledge, thus strengthening the users search quality (Liu et al., 2022). We'll later explore how the KG concept aided the concept of cybersecurity knowledge graphs (CSKG) to further strengthen the processes of monitoring, detecting, and locating security breaches and potential threats.

1.2.1 Knowledge graph overview

1.2.1.1 Definitions:

A KG is essentially a knowledge base, referred to as a semantic network with a directed graph structure used to describe concepts and their interrelationships in a symbolic form. It consists of nodes and edges; where the nodes in the graph represent entities or concepts, whereas the edges capture additional attributes (details) in the form of key-value pairs to represent the relationships between these entities and concepts. The triplet <entity, relationship, entity> equivalent to <concept, attribute, attribute value> (also referred to as <subject, predicate, object>) make up the overall composition of a directed multi-graph, where the concepts mainly include collections, object types, and categories, such as hosts and vulnerabilities, as represented in Figure 2, in the highlighted red box, with the entities labeled Attack01and host. Whereas the attribute value describes the characteristics that an object might have, like the host having an IP address of 192.168.1.100. The relationships connect entities in a meaningful way and form a graph structure; such as Attack01 **attacking** host relationship (Zhang, K., & Liu, J., 2020).

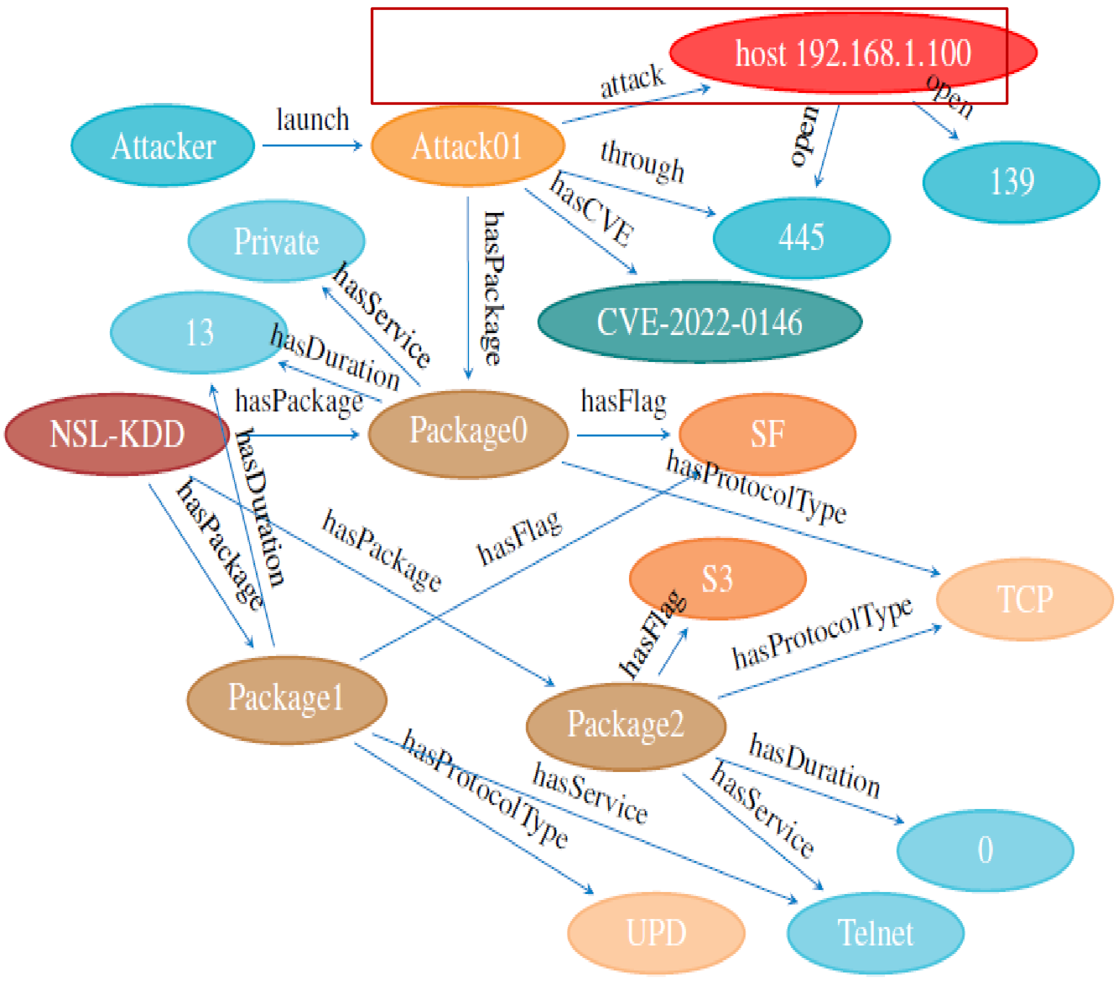


Figure 2: A typical snapshot of a Cybersecurity Knowledge Graph.

A KG supports heterogeneous computing environments such as cloud-based information technology infrastructures (Choudhury et al., 2018), some of the popular KGs are DBpedia (Auer, S., 2007), YAGO (Yet Another Great Ontology) (Suchanek et al., F., 2007), Google's Knowledge Graph (Singhal A, 2012), and YAGO2 (Hoffart, J. et al., 2013).

1.2.2 Knowledge graphs with RDFs

1.2.2.1 definitions

The Resource Description Framework (RDF) is a standard data model for representing and exchanging data. It provides a way to describe facts using <subject, predicate, object> triples, also known as RDF triples [8]. As shown in Figure 3, the building block of an RDF is the Unique Resource Identifier (URI), with the subject being represented by the URI (the red nodes) or a blank-node (the red empty node), while the object can be a URI, blank-node, or literal (e.g., a string or an integer, like the yellow nodes), and the predicate is a property defined in an ontology and it is strictly represented by a URI (the writing on top of the directed edges) [9]. The example below provides a clear visual of how an RDF triple will be used in a graph representation.

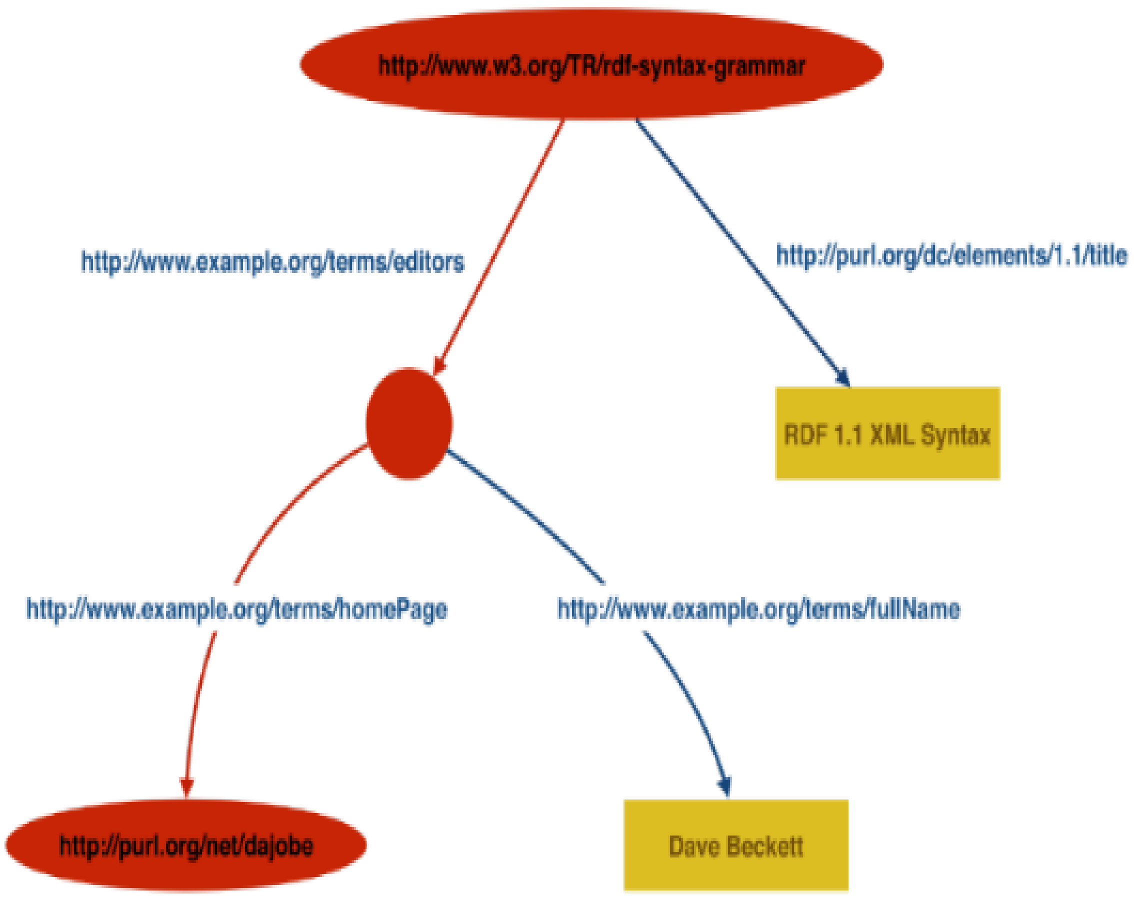

http://www.w3.org/TR/rdf-syntax-grammar
http://www.example.org/terms/editors
http://purl.org/dc/elements/1.1/title
RDF 1.1 XML Syntax
http://www.example.org/terms/homePage
http://www.example.org/terms/fullName
http://purl.org/net/dajobe
Dave Beckett

Figure 3: Graph for RDF example with a one-way path through the graph.

1.2.2.2 The advantage of RDF with knowledge graphs

Using RDF to represent knowledge helps you define your ontology and your properties and classes in a unique way with the help of URI's. This approach will provide the option to easily perform queries and link to internal and external background knowledge by extending SPARQL, as it will be suitable for our KG to query certain facts to exact values of a triple's subject, predicate, or object.

1.3 Cybersecurity ontology and knowledge graphs

Ontology is used to describe the relationships between concepts in multiple fields, including but not limited to semantic web services, image understanding, software engineering, cybersecurity, and so on. These relationships and concepts possess a clear, unique, and unambiguous definition that is collectively understood, therefore allowing humans and machines to essentially interact and communicate with each other. Representing knowledge on an ontology-based method is proved to effectively integrate different types of data from multiple sources within a particular domain. Similarly, by employing a comprehensive ontology language, it becomes possible to achieve knowledge reasoning, classification, and representation. (Zhang, K., *&* Liu, J., 2020).

1.3.1 Cybersecurity ontology

The cybersecurity ontology was developed to combine several different cybersecurity data, with the purpose to combine, organize, and use the cybersecurity domain knowledge in an efficient way, and provide support for the cybersecurity threat analysis and intrusion detection. Unified ontologies, such as STUCCO (Iannacone, M.,2015), and Unified Cybersecurity Ontology (UCO) developed by Syed et al. (2016), are used in order to integrate heterogeneous data and knowledge schemas from different cybersecurity systems. Researchers were able to develop multiple different ontologies for various specific application scenarios which include intrusion detection, cyberattack analysis, vulnerability analysis, malware categorization, etc. as represented in Figure 4 (Liu et al., 2022). It is important to build a generic network security ontology to stay up to date with the current complex cyber environment.

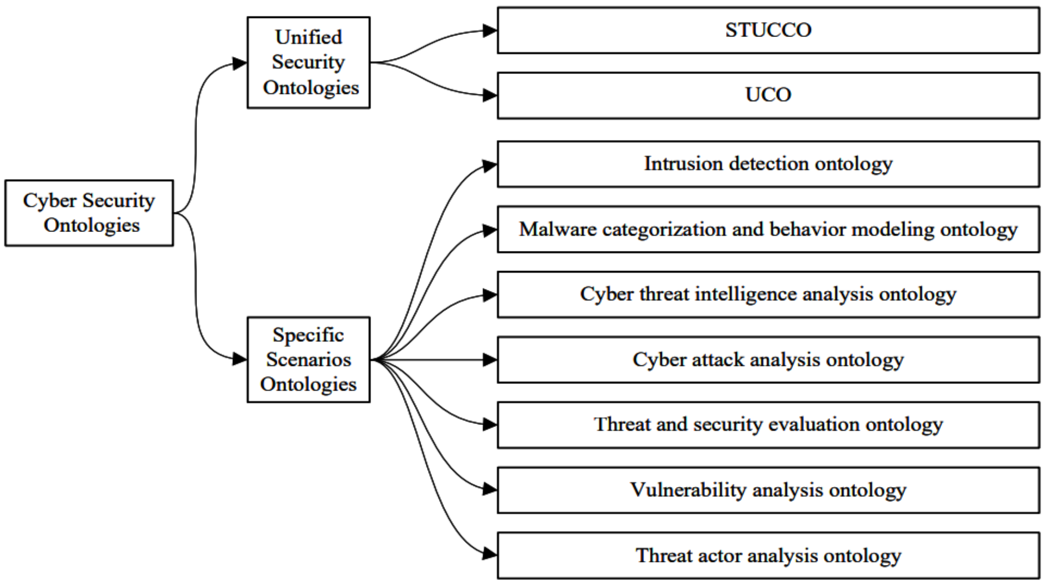

Cyber Security Ontologies
Unified Security Ontologies
Specific Scenarios Ontologies
STUCCO
UCO
Intrusion detection ontology
Malware categorization and behavior modeling ontology
Cyber threat intelligence analysis ontology
Cyber attack analysis ontology
Threat and security evaluation ontology
Vulnerability analysis ontology
Threat actor analysis ontology

Figure 4: Cybersecurity ontologies.

Ontology is the base of knowledge management in a knowledge graph, and it has the ability to provide a theoretical basis for knowledge graphs to manage entities, relationships and the relationships between objects like attributes and types. CSKG, therefore, is a way to represent and organize knowledge using graph theory and techniques to represent cybersecurity ontology.

1.3.2 CSKG construction overview

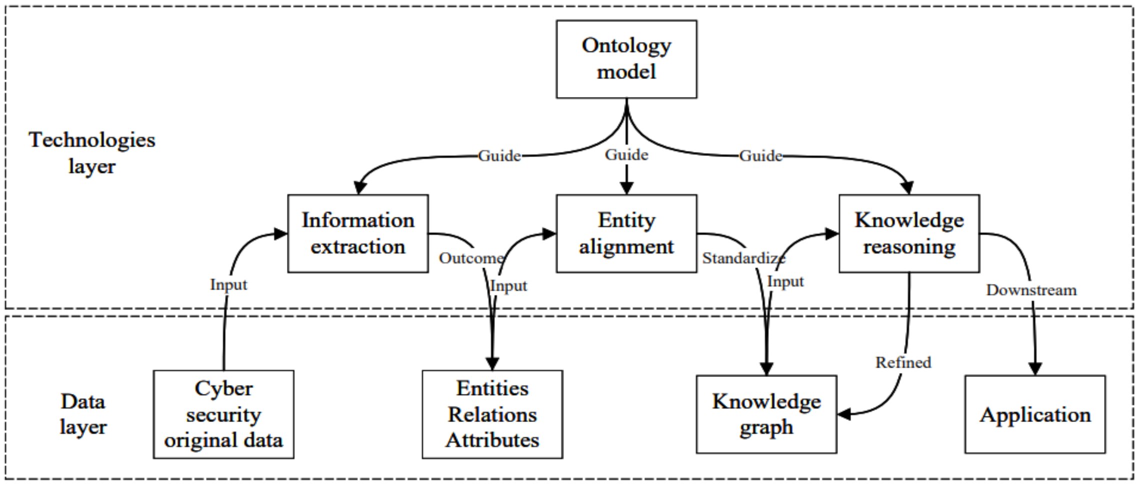


Figure 5: The construction framework of the cybersecurity knowledge graph.

The construction process of the CSKG is like that of a KG. The CSKG follows a top-down construction method, taking advantage of the mature and comprehensive knowledge data in the field of cybersecurity. This approach involves integrating fragmented domain data with the help of a framework or pre-designed cyber security ontology provided by domain experts.

To build the CSKG, information extraction and entity alignment technologies are utilized to extract entities and relationships from the original cyber security data. Additionally, knowledge reasoning technology is applied to generate new knowledge based on the existing KG. This new knowledge serves as support for prediction and inference tasks (Liu et al., 2022).

The construction framework of the CSKG is depicted in Figure 5 above.

1.3.3 Datasets

1.3.3.1 Definitions

Cybersecurity analysts constantly aim to secure systems though the collection of available knowledge, such as known and newly discovered weaknesses, threats, vulnerabilities, and attack patterns. This type of knowledge is gathered, structured, and published by research institutions, government agencies, and industry experts, such as Computer Emergency Response Teams (CERTs), MITRE (Liu et al., 2022) and the Canadian Institute for Cybersecurity (CIC).

1.3.3.2 Using datasets with CSKG

Multiple existing datasets can be used to populate a CSKG, provided with the enriched knowledge in the existing datasets, we can transform that knowledge into a structured graph that makes sense of that knowledge and makes it easy for analysts to visually spot attack paths. In our thesis, we'll be integrating datasets from the CIC.

The CIC offers multiple up to date dataset records, which are used around the world by universities, private industries, and independent researchers, as it aids users keep track of the datasets downloaded in every country, about the different attacks that occur regionally. The datasets available in CIC includes IoT Datasets, Malware, DNS Datasets, Dark Web, IDS Datasets, and the ISCX Datasets from 2009 to 2016 [10]. The two datasets that will be used throughout this project to transform into CSKG and track attack patterns, will be from the ISCX Datasets called NSL-KDD dataset, and from DNS (Domain Name System) Datasets, specifically the DNS over HTTPS (DoH) Dataset.

1.3.3.3 NSL-KDD dataset

The NSL-KDD dataset is an effective basis dataset to aid researchers in comparing different intrusion detection methods. This is because it contains plenty of records in the NSL-KDD train and test sets making them reasonable and reliable. Therefore, resulting in different research work to be consistent and comparable [11]. The NSL-KDD dataset consists of network traffic data captured from a simulated environment, including various types of intrusive activities. It contains several features extracted from network packets, such as connection attributes, protocol-related information, and host-based features.

1.3.3.4 DoH dataset

The DNS is a term well-known in network protocols and is notably one of the most vulnerable protocols that had multiple security vulnerabilities that were exploited by attackers over the years. One of the solutions that aimed to overcome these vulnerabilities was the DoH in RFC8484 protocol introduced by IETF. It's a protocol which fights eavesdropping and man-in-the-middle attacks by encrypting DNS queries, as a way to ensure privacy. The CIC research group were able to propose a systematic approach in order to generate datasets for analysis, test, and evaluate DoH traffic [12]. The DoH dataset contains the collection of data that captures various aspects of DoH usage, which includes information such as: the source IP addresses of DNS queries, the corresponding domain names being queried, timestamps of the queries, the DoH resolver used, and other relevant metadata.

Knowledge extraction technologies are a crucial part of generating a CSKG. Being provided with several quality datasets enhances the works for knowledge extraction, which will serve the purpose of training robust information extraction models in the future, such as the integration of Machine Learning.

1.3.4 Machine learning collaboration with knowledge graphs

In 2012, Garrido et al. was the first person to apply a machine learning method to KGs as a way of identifying unusual behaviors in the industrial automation systems which integrated both IT and OT components. This study used available graph ontology and built a KG using three key sources of knowledge: automation system information, application-level observations such as data access events, and network observations such as connections between hosts. This study adopts a graph embedding algorithm to estimate the likelihood of triple assertions appearing from observed security events. Analytically, this method produces intuitively interpretable alarms in a diversity of context. This alludes to the potential of relational machine learning benefits on KG for the delegated task of **intrusion detection.** Although the results are extracted on a small prototype, the present research progressively explores for the first time the collaboration of KG and industrial control systems.

In this project we want to allude to the fact that machine learning can be later used as a means to enhance the power of cybersecurity in intrusion detection when in collaboration with dataset populated KGs, which is by identifying similar pieces of data from a user and comparing it to one of the attackers (which will be already registered in the dataset) such as IP addresses that are generally used in DDoS attacks, and machine learning will be used to block that user.

1.3.5 Related work

There has been a few research conducted on the use of cybersecurity knowledge graphs in the cyberspace.

Choudhury et al. (2018) aimed to incorporate a set of methods for knowledge extraction, event detection, risk estimation and explanation, for incoming cyber-alerts as a method for securing an IT (information technology) system. These methods included the explanation of the relationships between entities given their context, by collecting different types of data such as event logs and netflows to construct a cyber knowledge graph. The aim of this was to unify this data and capture the association between three types of entities: user, machines and applications and their properties.

Jia Y., et al. research achieved the development of a cybersecurity knowledge base through machine learning techniques, enabling the extraction of entities and construction of an ontology. Deduction rules based on a quintuple model were established, facilitating logical reasoning within the knowledge base. By integrating knowledge graphs with cybersecurity, the study demonstrated the potential for effective threat detection and decision-making. The utilization of the Stanford Named Entity Recognizer (NER) improved information extraction, while providing insights for future work in the cybersecurity domain.

Zhang and Liu (2020) were able to arrive at the conclusion that ontology-based knowledge representation can officially accurately represent complex knowledge of heterogenous systems in cybersecurity. This was accomplished through researching the progress of ontology-based knowledge representation after weighing the pros and cons of the common cybersecurity assessment models, which included: attack graph model, attack tree model, game theory model, and petri net model.

Kurniawan K. et al. (2022) identified a key limitation to the various attack-graph methods being proposed recently and that is the different approaches that are being provided have a monolithic architecture and heterogenous in their internal models. They arrived at the conclusion that RDF based graphs are scalable and can efficiently support various threat detection techniques.

Liu K., et al. (2022) explored the existing application scenarios of CSKGs and related existing datasets found in practice, to motivate and provide an introduction to the application of KGs in cybersecurity by describing its origins, ideas and building methods. It explored the several existing datasets that are available for constructing CSKGs and the importance of the information extraction task.

Conclusion

In this chapter, we highlighted the main concepts for the progression of cybersecurity in the intrusion detection and threat prevention, which are the different types of security measures for protection, the cybersecurity threats, the cybersecurity solutions, cybersecurity knowledge graphs, datasets, and what machine learning could provide when in collaboration with knowledge graphs.

We also identified the tools used to maximize the efficiency of our knowledge graph and knowledge extraction using RDF, datasets, and SPARQL queries.

In the next chapter and based on the previous one, we will take a closer look and analyze the different tools used to construct, store, and manipulate knowledge graphs in the context of cybersecurity.

Chapter 2

Conception

Introduction

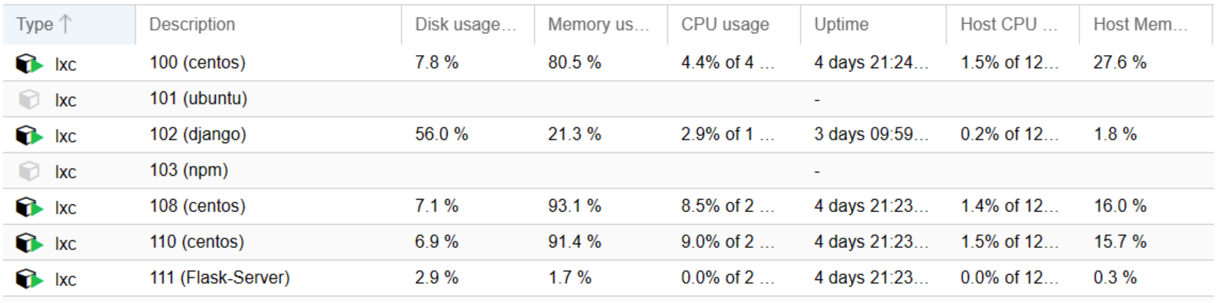
For an application to run successfully and function properly, an appropriate architecture and right tools should be carefully adapted. In this chapter, the Proxmox Virtual Environment that will enable to create and manage virtual machines, containers and virtual networks and bridges is first presented. Since the application is deployed on a Django Server that acts as an interface through which the administrator controls the application, its description is then fully detailed. Django server functionalities are commonly used throughout the Django REST Framework, enabling client remote access. This is also detailed next. Knowledge Graphs are implemented in java as Graph NoSQL based store accessed through a Py4j API based Gateway to enable a tunneling between python and java. The Knowledge Graph implementation and the Gateway tunneling is then described. Finally, sequence diagrams will be provided to explain the overall events flow and interaction between the application components, users and the different previous tools. A conclusion should summarize the overall conception details.

2 Proxmox Virtual Environment

Proxmox VE (Virtual Environment) is a complete open-source server virtualization management solution. It tightly integrates the Kernal-based Virtual Machine (KVM) hypervisor and Linux Containers (LXC), software-defined storage, and networking functionality on a single platform. Proxmox VE offers a web interface accessible after installation on your server which makes management easy, typically needing only a few clicks.

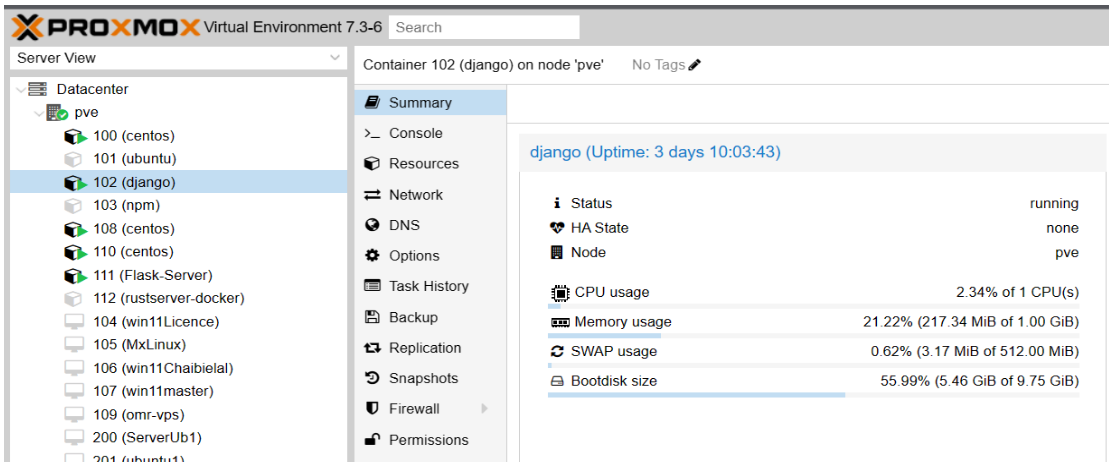
Proxmox VE was developed by Proxmox Server Solutions in Austria under the Internet Foundation of Austria and is released under the GNU General Public License. Since it’s an open-source solution it can be customized as per your requirements (Proxmox VE Administration Guide, 2023).

Figure 6 shows how we customized our Proxmox by creating three containers: the first container 102, contains our Django application, whereas containers 100, 108, and 110 represent the store, and are basically storage nodes. While Figure 7 present container 102 properties and capacities.



Type ↑		Description	Disk usage...	Memory us...	CPU usage	Uptime	Host CPU ...	Host Mem...
	lxc	100 (centos)	7.8 %	80.5 %	4.4% of 4 ...	4 days 21:24...	1.5% of 12...	27.6 %
	lxc	101 (ubuntu)				-		
	lxc	102 (django)	56.0 %	21.3 %	2.9% of 1 ...	3 days 09:59...	0.2% of 12...	1.8 %
	lxc	103 (npm)				-		
	lxc	108 (centos)	7.1 %	93.1 %	8.5% of 2 ...	4 days 21:23...	1.4% of 12...	16.0 %
	lxc	110 (centos)	6.9 %	91.4 %	9.0% of 2 ...	4 days 21:23...	1.5% of 12...	15.7 %
	lxc	111 (Flask-Server)	2.9 %	1.7 %	0.0% of 2 ...	4 days 21:23...	0.0% of 12...	0.3 %

Figure 6: Example of AIRLIO Proxmox VE.



Container 102 (django) on node 'pve'   No Tags

Datacenter
pve
100 (centos)
101 (ubuntu)
102 (django)
103 (npm)
108 (centos)
110 (centos)
111 (Flask-Server)
112 (rustserver-docker)
104 (win11Licence)
105 (MxLinux)
106 (win11Chaibielal)
107 (win11master)
109 (omr-vps)
200 (ServerUb1)

django (Uptime: 3 days 10:03:43)

Status	running
HA State	none
Node	pve
CPU usage	2.34% of 1 CPU(s)
Memory usage	21.22% (217.34 MiB of 1.00 GiB)
SWAP usage	0.62% (3.17 MiB of 512.00 MiB)
Bootdisk size	55.99% (5.46 GiB of 9.75 GiB)

Figure 7: Django CT details.

2.1 Architecture

The architecture of our program is quite simple. The administrator or cybersecurity expert launches his programs or scripts that call our app's APIs. The request sent should include the HTTP method, the graph, the triple elements, and the query type which can return JSON format, RDF turtle or a Boolean value. Any modification done to the store requires a piece of additional information which is the authentication token.

Django communicates with the Gateway using Python. The Gateway then sends and receives instructions from JENA using Java. Finally, it will communicate with the Oracle NoSQL store using SPARQL queries.

Figure 8 below demonstrates this process.

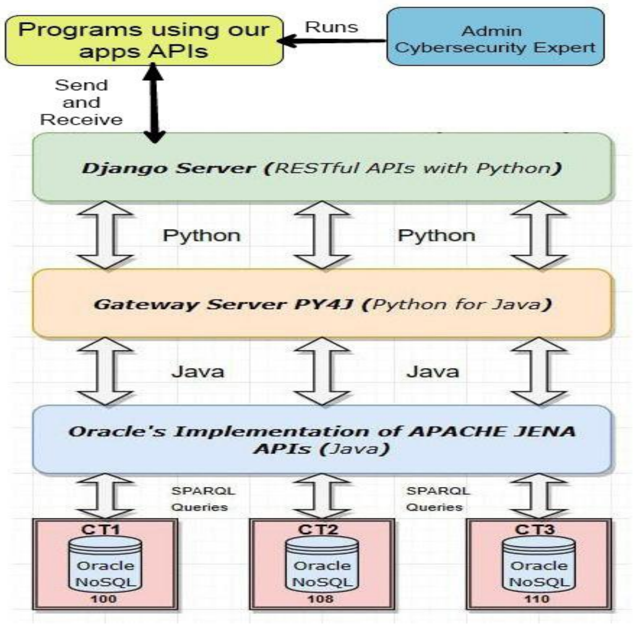

Programs using our apps APIs
Runs
Admin Cybersecurity Expert
Send and Receive
Django Server (RESTful APIs with Python)
Python
Python
Gateway Server PY4J (Python for Java)
Java
Java
Oracle's Implementation of APACHE JENA APIs (Java)
SPARQL Queries
SPARQL Queries
CT1
Oracle NoSQL
100
CT2
Oracle NoSQL
108
CT3
Oracle NoSQL
110

Figure 8: Our application's architecture.

2.2 NoSQL databases

2.2.1 Definitions

Originally named with reference to non-relational databases, nowadays people consider NoSQL as "not only SQL", emphasizing that a NoSQL database can perform as a SQL one with additional advanced functionalities that a traditional relational database does not have.

NoSQL databases are open-source, schema-less, horizontally scalable, and high-performance databases. These characteristics make them very different from relational databases, the traditional choice for spatial data [13].

2.2.2 The benefits of using a NoSQL database

Many devices and applications including mobiles and online activities require high performance, scalable, flexible, and highly functional databases for a good user experience. NoSQL provides all that:

- **Flexibility:** Flexible schemas in NoSQL databases allow quicker and an iterative development. This is because NoSQL databases has an adaptable data model which therefore supports semi-structured and unstructured data.
- **High performance:** NoSQL databases are well equipped to handle data models and access patterns which enable high performance. In addition, unlike relational databases, NoSQL is much more functional, efficient, and quicker when carrying out equivalent functions.
- **Highly Functional:** NoSQL databases provide functional APIs and data types which are unique to their individual data models.

- **Scalability:** NoSQL databases are built to expand by using distributed hardware clusters instead of scaling up by introducing and depending on more servers. Some cloud service providers carry out these tasks in the backend as a fully managed service.

2.2.2.1 Graph store

A graph database is designed to handle data with complex relationships and interconnections. In a graph database, data is stored as nodes and edges, where nodes represent entities and edges represent the relationships between those entities. They are particularly well-suited for applications that require deep and complex queries, such as social networks, recommendation engines, and fraud detection systems. Figure 9 illustrates an attack stored graph where the remote IP addresses are represented by IP nodes (178.1…) connected to the red event node 64 by ATTACKER_IP edges, while the local IP (192.1…) also uses this type of node to connect through the TARGET_IP edge. While the HAS_PID edge is used to aggregate events such as node 64 and 66 by PID (Hofer, D. et al., 2021).

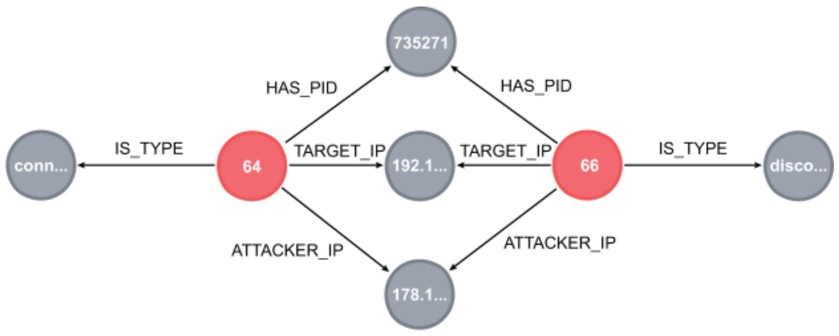


Figure 9: a property-based graph modeling the properties of an attacker.

NoSQL databases are well suited to handle typical challenges of big data, including volume, variety, and velocity. For these reasons, they are increasingly adopted by private industries and used in research. They have gained tremendous popularity in the last decade due to their ability to manage unstructured data.

2.3 Oracle NoSQL store

The KVStore, also known as Key/Value Store, is a datacenter consisting of storage nodes and replication nodes. The replication nodes hold distributed data. Each storage node is a physical or virtual machine with its own storage space, which may or may not be identical to other nodes in the store. This architecture, depicted in Figure 10, is commonly employed by applications utilizing Oracle NoSQL databases [14].

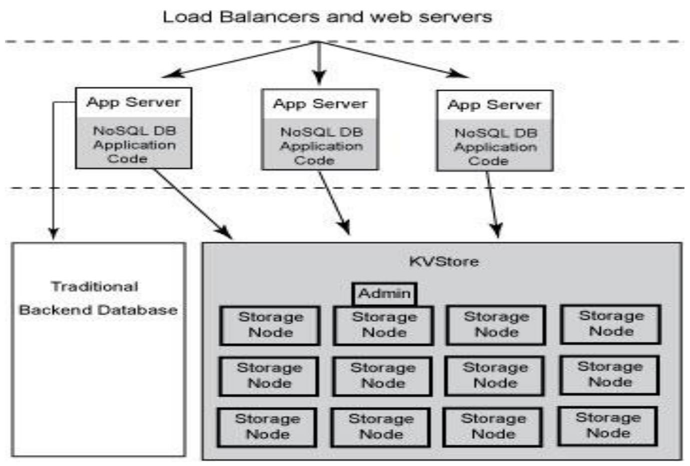

Load Balancers and web servers
App Server
NoSQL DB Application Code
App Server
NoSQL DB Application Code
App Server
NoSQL DB Application Code
Traditional Backend Database
KVStore
Admin
Storage Node

Figure 10: Store Architecture.

In a Key/Value Store, each Storage Node accommodates one or more Replication Nodes based on its capacity. The capacity of a Storage Node determines its hardware resources. A store can have Storage Nodes with different capacities, and Oracle NoSQL Database ensures that the workload assigned to a Storage Node aligns with its capacity. Replication Nodes consist of one or more partitions, and each Storage Node is equipped with monitoring software to ensure the health and running status of the Replication Nodes it hosts.

2.3.1 Replication nodes and shards

At a high level, a Replication Node in the Key/Value Store functions as a standalone database comprising key-value pairs. These nodes are organized into shards, where each shard contains a master node responsible for database writes and replicating them to other nodes within the shard. The remaining nodes in the shard serve read-only operations as replicas. While there can only be one master node at a time, any member of the shard can assume this role. Essentially, the use of single master or multiple replica strategies in each shard aims to enhance read throughput and availability.

The following Figure 11 shows how the KVStore of is divided up into shards:

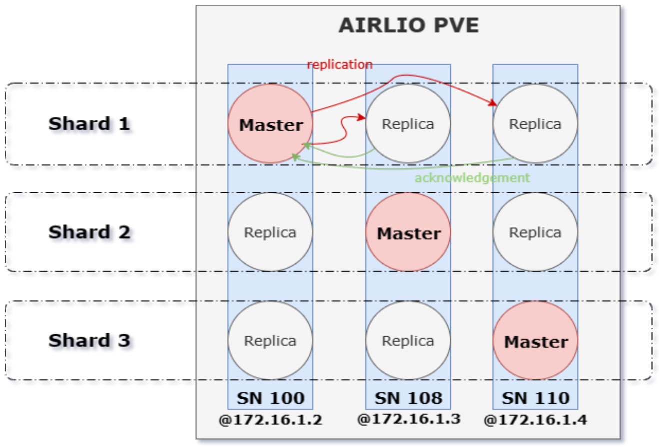


Figure 11: KVStore shards distribution.

In the event of a failure of the machine hosting the master node, an automatic failover occurs in which one of the replica nodes within the shard is promoted to become the new master node. This ensures continuous operation by transitioning the responsibilities to another available node within the shard.

In order to optimize write performance in Production KVStores, it is recommended to have multiple shards. During installation, you provide input to Oracle NoSQL Database, which helps determine the appropriate number of shards for the store. Having a greater number of shards enhances write performance as it increases the number of nodes responsible for handling write requests. In summary, more shards in the KVStore result in improved write performance due to a larger number of nodes dedicated to servicing write operations.

2.3.2 Replication factor

The Replication Factor refers to the number of nodes within a shard. A higher Replication Factor leads to faster read capacity because there are more machines available to handle read requests. However, it also results in slower write performance since wri tes need to be copied to multiple machines. When setting up the store, you specify the Replication Factor, and Oracle NoSQL Database automatically creates the required number of Replication Nodes for each shard in the store. In summary, adjusting the Repli cation Factor allows you to balance the trade-off between read throughput and write performance in your system.

2.3.3 Partitions

In a shard-based architecture, each shard consists of one or more partitions where key-value pairs are stored. Keys are assigned to a specific partition and cannot be moved to another. Oracle NoSQL Database automatically distributes keys evenly across all available partitions. When planning your system, you need to determine the number of partitions, which is not configurable once the store is installed.

Expanding or changing the number of Storage Nodes in use allows reconfiguration of the store by adding new shards. During this process, partitions are redistributed between new and existing shards to balance the load. To facilitate such reconfiguration, it is recommended to have a sufficient number of partitions for fine-grained control. However, having too many partitions incurs a minimal performance cost. A general guideline is to aim for 10 to 20 partitions per shard. Since the number of partitions cannot be altered after the initial deployment, it is crucial to consider the projected future size of the store when determining the number of partitions.

2.3.4 Topologies

A topology in a NoSQL DB store encompasses storage nodes, replication nodes, and administration services. It represents the state of the store at a specific time. During initial deployment, the topology is designed to minimize the risk of a single point of failure within a shard. This means that while a Storage Node may host multiple Replication Nodes, those Replication Nodes will not belong to the same shard. This setup improves the availability of shards for read and write operations even in the event of a hardware failure that affects the hosting machine.

Topologies can be modified to achieve different performance characteristics or to accommodate changes in the number or features of Storage Nodes. The process of changing and deploying a topology is iterative, involving updates and adjustments to the configuration.

2.4 SPARQL

2.4.1 Definitions

SPARQL is a query language and protocol for associated data. The purpose of SPARQL is to retrieve and manipulate data stored in RDF. It provides a plethora of expressivity for complex queries such as aggregation, negation, and subqueries. Moreover, it allows users to search for information from databases which can be matched to RDF. In the aspect of security, this is extremely beneficial as information is dispersed across various systems and networks.

2.4.2 SPARQL queries

SPARQL queries are a set of triple patterns. Each from a subject, predicate, and object can be a variable input in that triple pattern. The query is then matched to the triples in the database by looking for and matching patterns to it, therefore giving the solution to the variabl es.

SPARQL consist of four types of queries: ASK, SELECT, CONSTRUCT, and DESCRIBE. They can be used to [15]:

- **ASK:** used to question whether there is at least one possible match of the query according to the RDF graph
- **SELECT:** used to select all or just some of those matches in a form of a table. Selecting may include aggregation, sampling, and pagination through OFFSET or LIMIT.
- **CONSTRUCT:** is a query to create an RDF graph by inputting the variables in the matches in a set of triples.
- **DESCRIBE:** used to describe the matches found by creating an appropriate RDF graph.

The syntax of a SPARQL query is illustrated in Figure 12 below.

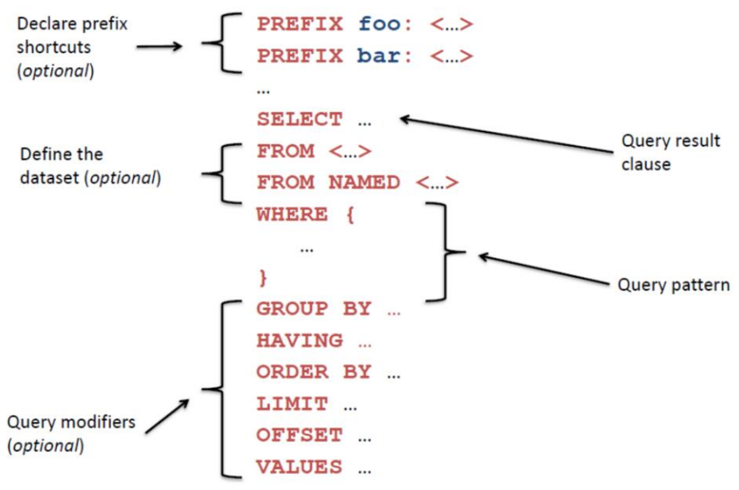


Figure 12: SPARQL syntax.

2.5 Django

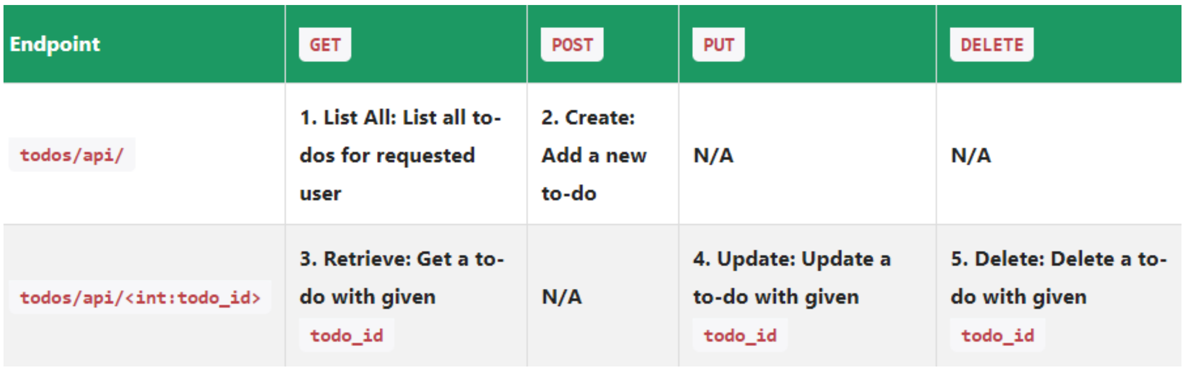
Django is a Python-based web framework that uses the Model-View-Template (MVT) architectural pattern. It possesses functional features for the user to use such as: login system, database connection, and CRUD operations (Create Read Update Delete) [16].

Django has a powerful and flexible toolkit called Django REST framework (DRF) and it's a useful tool for building Web APIs. It operates with class-based views, forms, model validators, QuerySet, and many more useful implementations. Its main functionality is making serialization much easier.

2.5.1 REST API

API or Application Programming Interface are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.

A REST API is used for systems to expose useful functions and data. REST stands for representational state transfer, which can be made up of one or more resources that can be accessed at a given URL and returned in various formats, like JSON, images, HTML, and more. Building a RESTful API is in most cases complicated, but with the help of Django and Django REST framework, this complexity is handled extremely well. In a RESTful API, endpoints define the structure and usage with the POST, GET, PUT, and DELETE (which correspond to Create, Read, Update, and Delete (or CRUD) operations respectively) HTTP methods, which must be organized logically [17]. As represented in Figure 13 below.



Endpoint	GET	POST	PUT	DELETE
`todos/api/`	1. List All: List all to-dos for requested user	2. Create: Add a new to-do	N/A	N/A
`todos/api/<int:todo_id>`	3. Retrieve: Get a to-do with given `todo_id`	N/A	4. Update: Update a to-do with given `todo_id`	5. Delete: Delete a to-do with given `todo_id`

Figure 13: API methods.

2.5.2 Models

Models in Django are classes containing the fields we want our database table to contain. After creating the model, you will need to migrate it to the database using the command lines:

python manage.py makemigrations

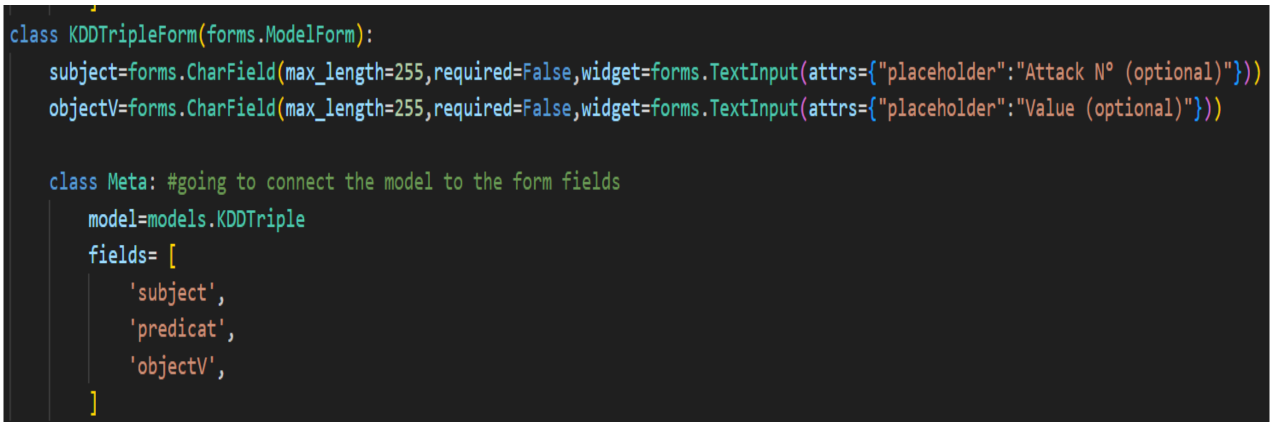
python manage.py migrate

However, it is important to note that we will not be using Django's traditional database system and would only use the models for the purpose of setting up the **forms**.

2.5.3 Forms

Django forms are a powerful feature of the Django web framework that allows developers to easily handle user input and validate data submitted through HTML forms. Forms in Django provide a convenient way to define and manage both the HTML rendering of the form and the validation of user-submitted data.

Django forms are created using Python classes that inherit from the *django.forms.Form* class or its subclasses like *django.forms.ModelForm*. The form class acts as a blueprint for rendering the HTML form and validating the submitted data (see Figure 14).



```python
class KDDTripleForm(forms.ModelForm):
    subject=forms.CharField(max_length=255,required=False,widget=forms.TextInput(attrs={"placeholder":"Attack N° (optional)"}))
    objectV=forms.CharField(max_length=255,required=False,widget=forms.TextInput(attrs={"placeholder":"Value (optional)"}))

    class Meta: #going to connect the model to the form fields
        model=models.KDDTriple
        fields= [
            'subject',
            'predicat',
            'objectV',
        ]
```

Figure 14: Example of a Django form.

2.6 Django server

Given that Django is a Python web framework, it needs a web server in order to operate properly. However, most web servers don’t natively communicate with Python, therefore, Django takes care of the struggle of web development for the client to focus solely in writing the code for the application. Django is a fully featured server-side web framework, written in Python, and which supports two interfaces which are WSGI and ASGI [18]:

- WSGI, or Web Server Gateway Interface, is the Python standard for web servers and applications as it allows communication between them, however, it only supports synchronous code.
- ASGI, or Asynchronous Server Gateway Interface, is the new, asynchronous-friendly standard that will allow your Django site to use asynchronous Python features and asynchronous Django features as they are developed.

In order to deploy with WSGI and ASGI, Django’s **startproject** management command will set up the default WSGI and ASGI configuration [19, 20].

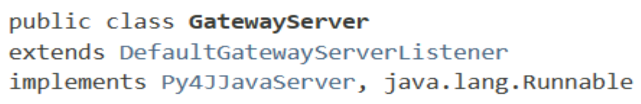
2.7 GatewayServer

A GatewayServer acts as an intermediary that handles the transaction between a client computer and another server. Without a GatewayServer, clients would be unable to access the remote server [21]. The class mentioned in Figure 15 below enables Python programs such as Django to access a Java program, and once a GatewayServer instance is started, Python programs can connect to the Java Virtual Machine (JVM) by calling the JavaGateway function:

gateway= JavaGateway()

After connecting to JVM, the entry passed to the GatewayServer can be accessed by the command below:

gateway.entry_point



```
public class GatewayServer
extends DefaultGatewayServerListener
implements Py4JJavaServer, java.lang.Runnable
```

Figure 15: Class syntax that enables Python to access Java programs.

A GatewayServer is only responsible for accepting connections, where each of the connections is taken care of by a GatewayConnection instance where the different states such as entryPoint and reference to returned objects are managed by a Gateway instance [22].

2.7.1 Py4j

Py4j is the tool that enables Python programs running in a Python interpreter to dynamically access Java objects and communicate with a JVM through a local network socket. This permits methods to be called as if the Java objects have become one with the Python interpreter as Java collections can be accessed through standard Python collection of methods. Py4j also enables Java programs to call back Python objects. The aim of this is to allow the use of Java libraries and frameworks (like we were able to use JENA API with this project) along with Python and its libraries [23].

2.8 UML

2.8.1 Sequence diagram

Figure 16 below represents the sequence diagram explaining our application functioning patterns.

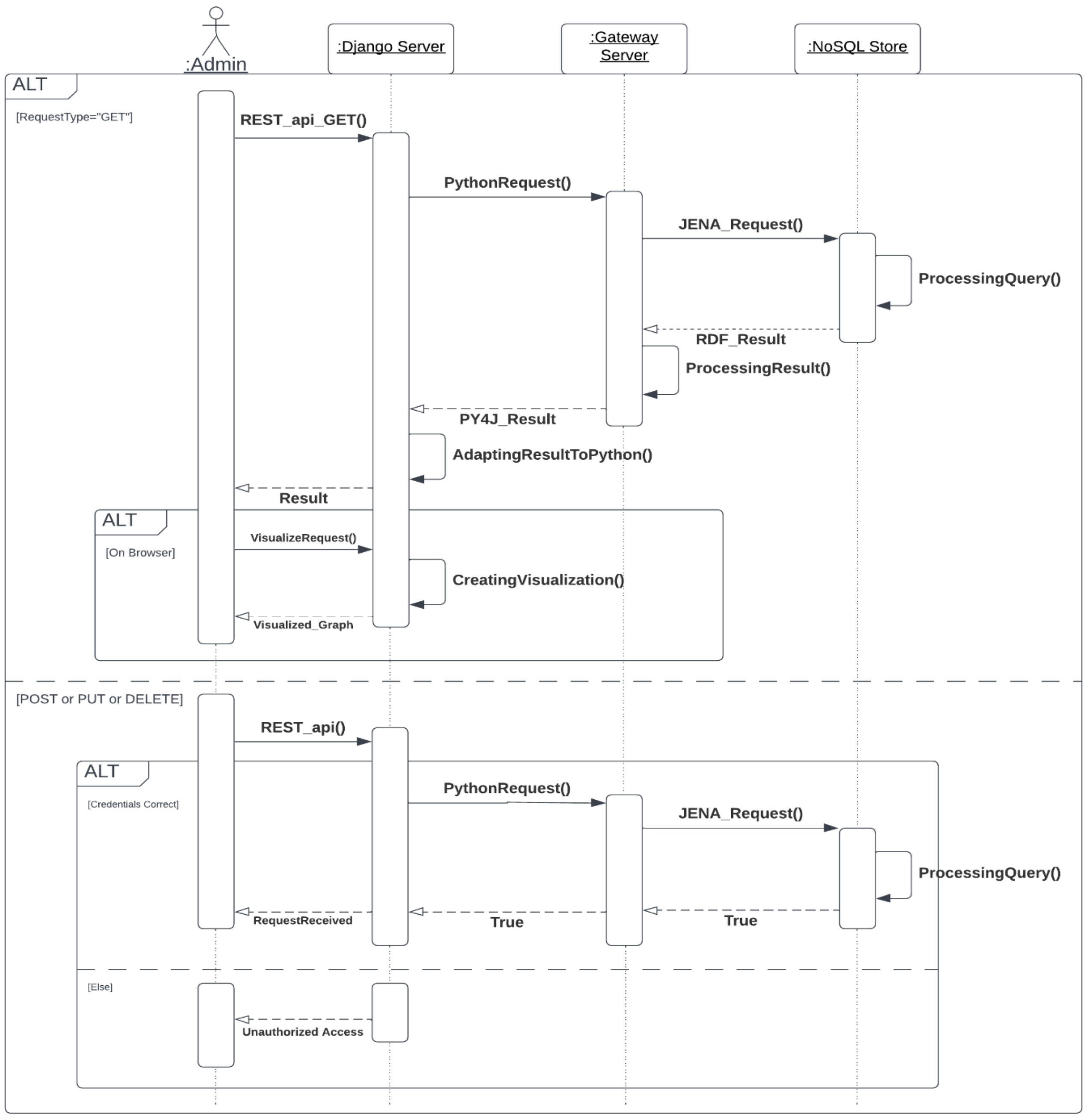


Figure 16: Diagram sequence.

2.8.2 Use-Case diagram

The Figure 17 below shows the use case diagram for our solution.

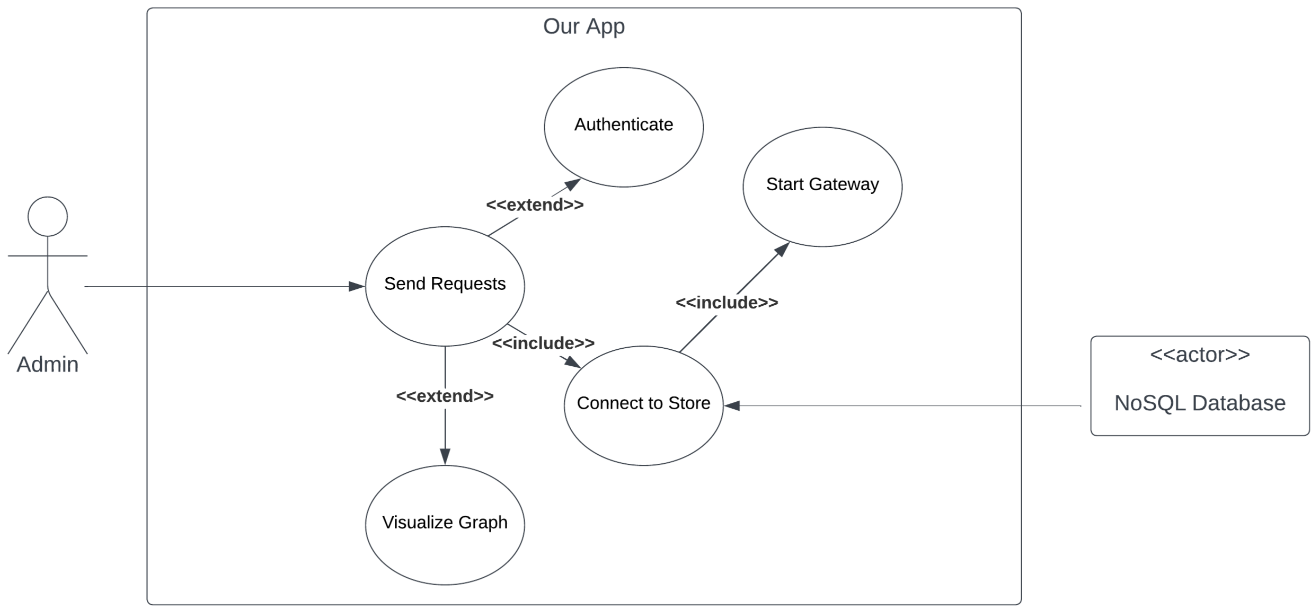


Figure 17: Use-Case diagram.

Conclusion

The conceptualization phase allowed for a clear visualization and served as a guide to carry out the implementation stage of the application. With the application architecture and the tools needed for the development of the application, in addition to the UML modelling, the implementation phase can now begin and will be discussed in the next chapter.

In the next chapter, the implementation phase will be carried out as the different components discussed previously will work together to be manipulated and executed, thus forming the overall product.

Chapter 3

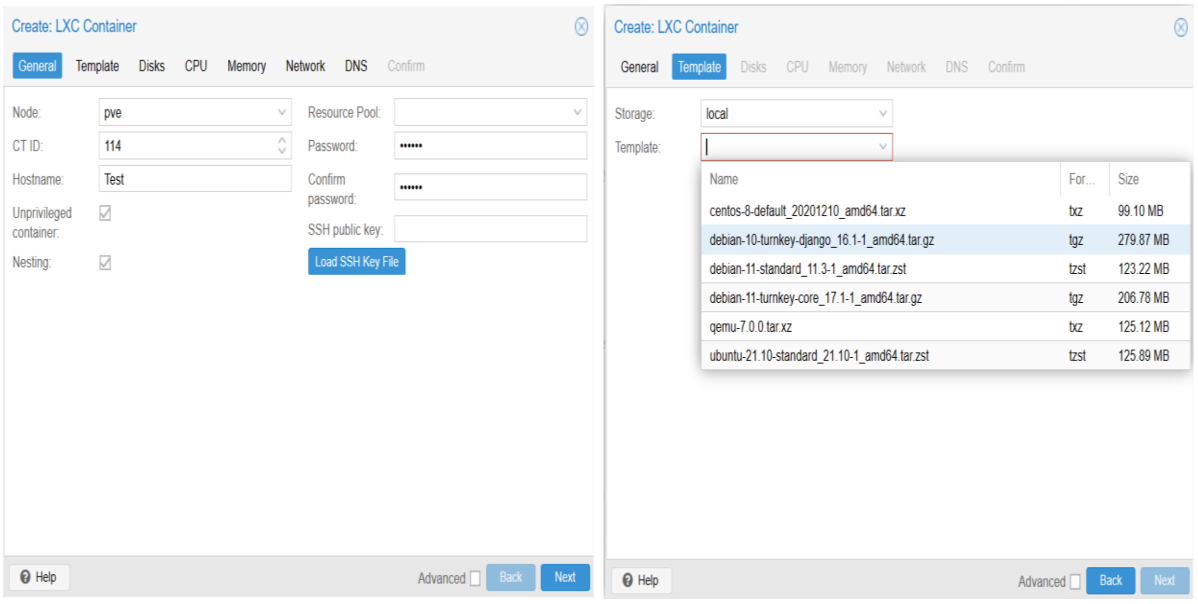
Implementation

This chapter aims to present the results we have achieved. We begin by briefly presenting the used tools, and the configurations required to make this process work. Next, we will present the main classes deployed in our solution, followed by their brief descriptions. Finally, a conclusion will synthesize the ideas developed.

3 Used tools

3.1.1 Setting up PROXMOX

After the PROXMOX VE setup, we now focus on creating the required containers acting as the store and as the Django server. Inside the PROXMOX VE, here is a container creation with the 'Create CT' button giving the following menu (Figure 18). A container name and password are required with a specific template chosen from a list. Here, a Debian based preinstalled Django server template is proposed that fit a Django Server container creation.



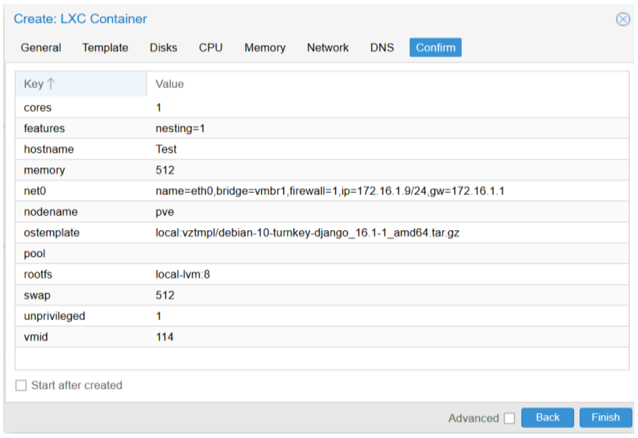
Node:	pve
CT ID:	114
Hostname:	Test
Unprivileged container:	✓
Nesting:	✓
Resource Pool:	
Password:	••••••
Confirm password:	••••••
SSH public key:	

Storage:	local
Template:	

Name	For...	Size
centos-8-default_20201210_amd64.tar.xz	txz	99.10 MB
debian-10-turnkey-django_16.1-1_amd64.tar.gz	tgz	279.87 MB
debian-11-standard_11.3-1_amd64.tar.zst	tzst	123.22 MB
debian-11-turnkey-core_17.1-1_amd64.tar.gz	tgz	206.78 MB
qemu-7.0.0.tar.xz	txz	125.12 MB
ubuntu-21.10-standard_21.10-1_amd64.tar.zst	tzst	125.89 MB

Figure 18: Container Creation page.

A disk storage amount, CPU cores, and memory size, as well as a static IP address and a gateway must be set up to finish the creation of the new CT. (Figure 19) summarizes this process.



Key	Value
cores	1
features	nesting=1
hostname	Test
memory	512
net0	name=eth0,bridge=vmbr1,firewall=1,ip=172.16.1.9/24,gw=172.16.1.1
nodename	pve
ostemplate	local:vztmpl/debian-10-turnkey-django_16.1-1_amd64.tar.gz
pool	
rootfs	local-lvm:8
swap	512
unprivileged	1
vmid	114

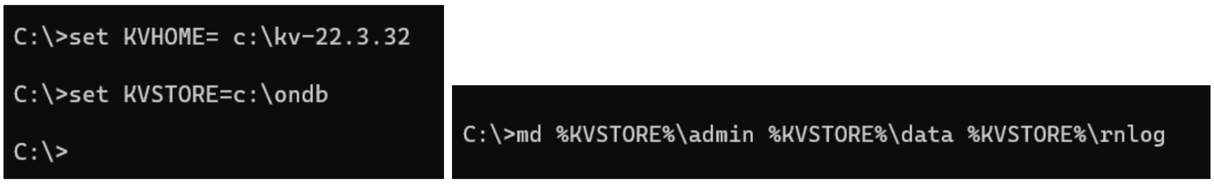
Figure 19: CT creation summary.

3.1.2 Creation of the datastore

3.1.2.1 Local

The datastore was created using the implicit commands in the KVSTORE tool offered by Oracle. This is done as follows along with Figures to demonstrate the process:

1. Creation of environment variables and required folders:

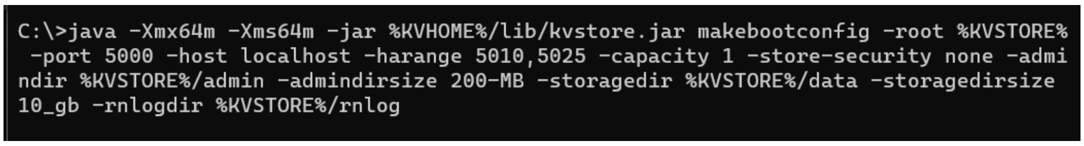


```
C:\>set KVHOME= c:\kv-22.3.32

C:\>set KVSTORE=c:\ondb

C:\>
```

```
C:\>md %KVSTORE%\admin %KVSTORE%\data %KVSTORE%\rnlog
```

Figure 20: Setting up environment variables and folders.

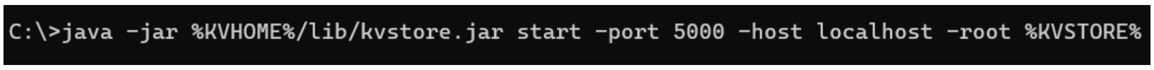
2. Creation of the database with *makebootconfig*:



```
C:\>java -Xmx64m -Xms64m -jar %KVHOME%/lib/kvstore.jar makebootconfig -root %KVSTORE%
 -port 5000 -host localhost -harange 5010,5025 -capacity 1 -store-security none -admi
ndir %KVSTORE%/admin -admindirsize 200-MB -storagedir %KVSTORE%/data -storagedirsize
10_gb -rnlogdir %KVSTORE%/rnlog
```

Figure 21: *makebootconfig* command.

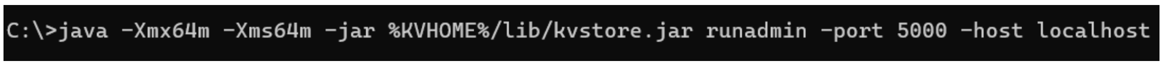
3. Launching the database



```
C:\>java -jar %KVHOME%/lib/kvstore.jar start -port 5000 -host localhost -root %KVSTORE%
```

Figure 22: Starting the store command.

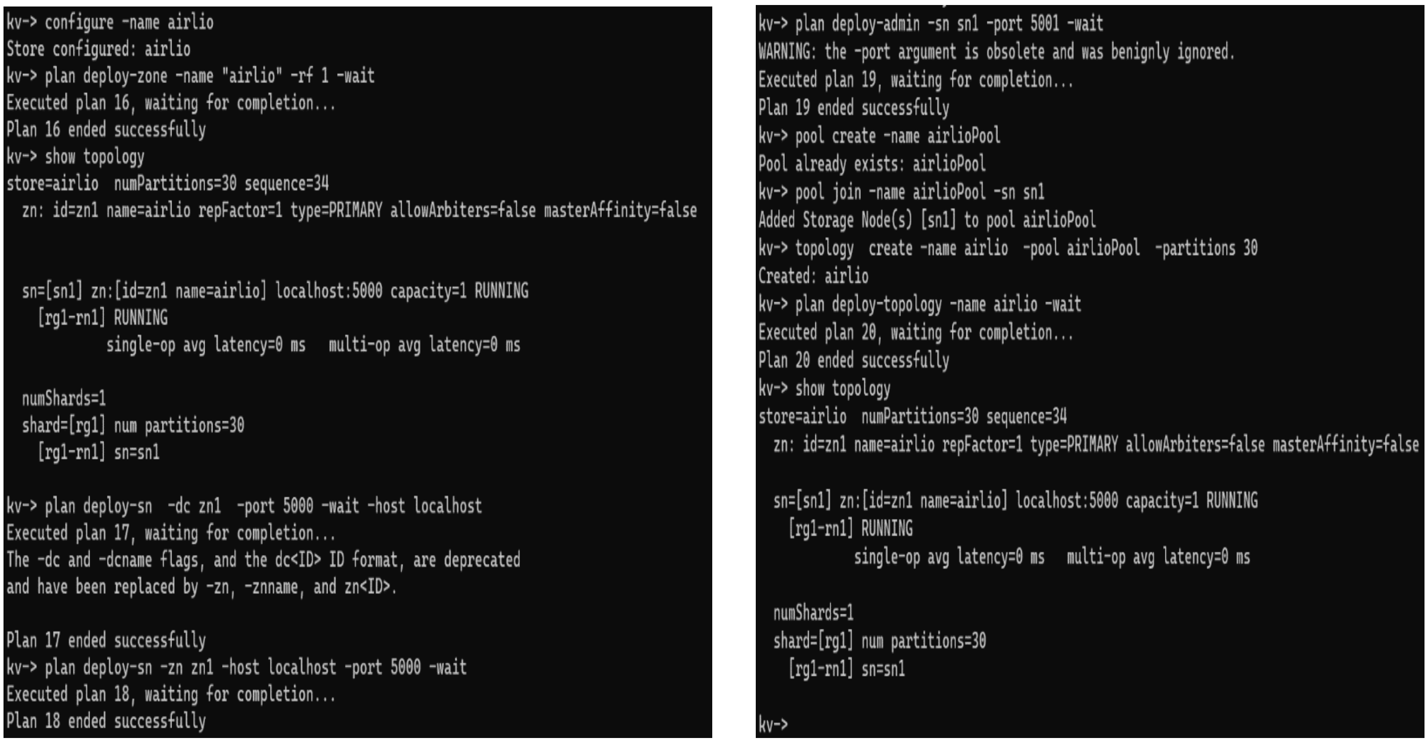
4. Configuring the store with *runadmin* (on a second cmd panel)



```
C:\>java -Xmx64m -Xms64m -jar %KVHOME%/lib/kvstore.jar runadmin -port 5000 -host localhost
```

Figure 23: the runadmin command.

5. Execution of the following commands



```
kv-> configure -name airlio
Store configured: airlio
kv-> plan deploy-zone -name "airlio" -rf 1 -wait
Executed plan 16, waiting for completion...
Plan 16 ended successfully
kv-> show topology
store=airlio  numPartitions=30 sequence=34
  zn: id=zn1 name=airlio repFactor=1 type=PRIMARY allowArbiters=false masterAffinity=false


  sn=[sn1] zn:[id=zn1 name=airlio] localhost:5000 capacity=1 RUNNING
    [rg1-rn1] RUNNING
          single-op avg latency=0 ms   multi-op avg latency=0 ms

  numShards=1
  shard=[rg1] num partitions=30
    [rg1-rn1] sn=sn1

kv-> plan deploy-sn  -dc zn1  -port 5000 -wait -host localhost
Executed plan 17, waiting for completion...
The -dc and -dcname flags, and the dc<ID> ID format, are deprecated
and have been replaced by -zn, -znname, and zn<ID>.

Plan 17 ended successfully
kv-> plan deploy-sn -zn zn1 -host localhost -port 5000 -wait
Executed plan 18, waiting for completion...
Plan 18 ended successfully
```

```
kv-> plan deploy-admin -sn sn1 -port 5001 -wait
WARNING: the -port argument is obsolete and was benignly ignored.
Executed plan 19, waiting for completion...
Plan 19 ended successfully
kv-> pool create -name airlioPool
Pool already exists: airlioPool
kv-> pool join -name airlioPool -sn sn1
Added Storage Node(s) [sn1] to pool airlioPool
kv-> topology  create -name airlio  -pool airlioPool  -partitions 30
Created: airlio
kv-> plan deploy-topology -name airlio -wait
Executed plan 20, waiting for completion...
Plan 20 ended successfully
kv-> show topology
store=airlio  numPartitions=30 sequence=34
  zn: id=zn1 name=airlio repFactor=1 type=PRIMARY allowArbiters=false masterAffinity=false

  sn=[sn1] zn:[id=zn1 name=airlio] localhost:5000 capacity=1 RUNNING
    [rg1-rn1] RUNNING
          single-op avg latency=0 ms   multi-op avg latency=0 ms

  numShards=1
  shard=[rg1] num partitions=30
    [rg1-rn1] sn=sn1

kv->
```

Figure 24: Setting up the store.

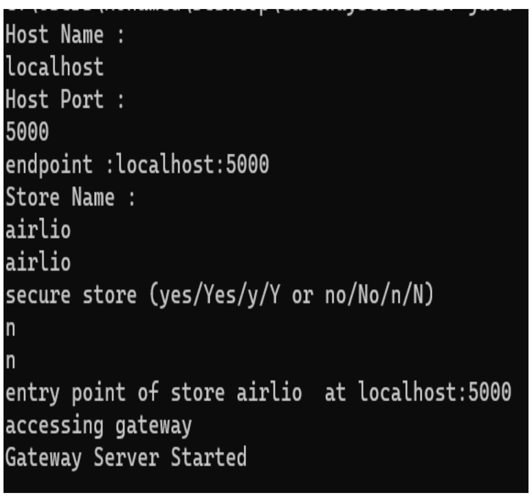
At this stage, the store was created locally with the name 'airlio'.

3.1.2.2 On our PROXMOX VE containers

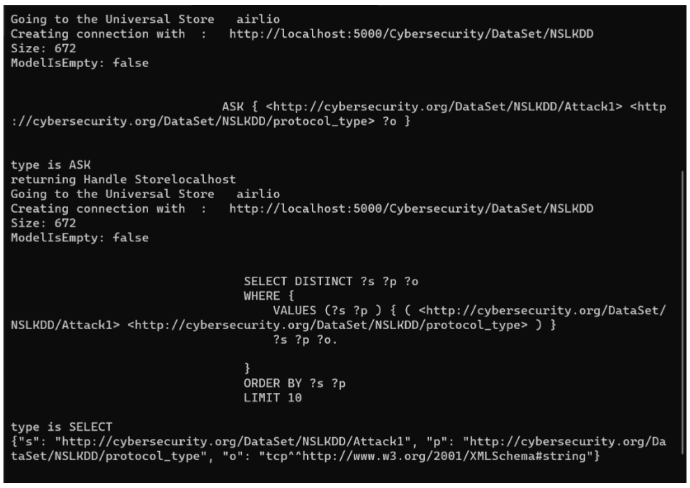
We used an adapted script called 'cluster_setup.sh'. In short, the script basically deploys the same configuration as the previous one but on 3 machines as shown in Figure 11 in the conception chapter.

3.2 Launching the GatewayServer

The command line '*java -jar GatewayServer.jar*', as shown in Figure 25, proceeds to start the GatewayServer so that it can be connected to the previously discussed running store. The GatewayServer is also responsible for logging the requests it receives, here's one example that will be further detailed in the Django section of this chapter.



```
Host Name :
localhost
Host Port :
5000
endpoint :localhost:5000
Store Name :
airlio
airlio
secure store (yes/Yes/y/Y or no/No/n/N)
n
n
entry point of store airlio  at localhost:5000
accessing gateway
Gateway Server Started
```



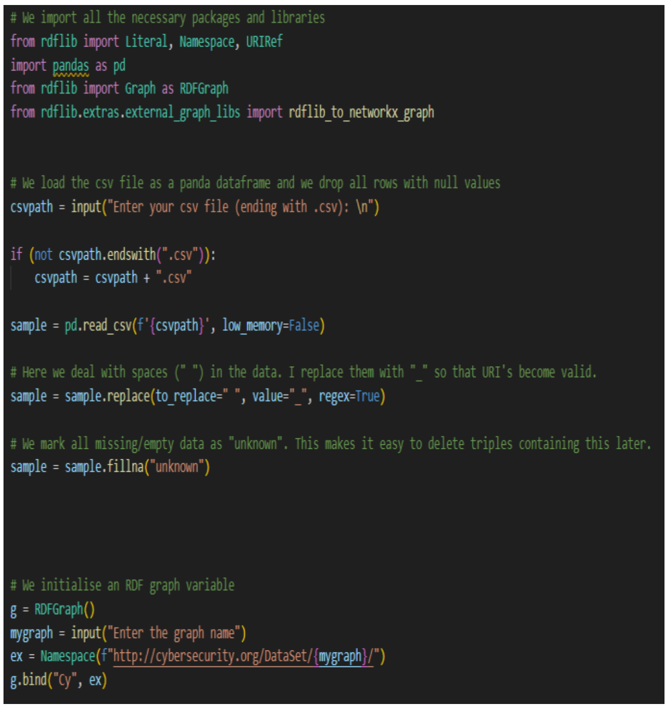
```
Going to the Universal Store   airlio
Creating connection with  :   http://localhost:5000/Cybersecurity/DataSet/NSLKDD
Size: 672
ModelIsEmpty: false


                              ASK { <http://cybersecurity.org/DataSet/NSLKDD/Attack1> <http
://cybersecurity.org/DataSet/NSLKDD/protocol_type> ?o }


type is ASK
returning Handle Storelocalhost
Going to the Universal Store   airlio
Creating connection with  :   http://localhost:5000/Cybersecurity/DataSet/NSLKDD
Size: 672
ModelIsEmpty: false


                                SELECT DISTINCT ?s ?p ?o
                                WHERE {
                                    VALUES (?s ?p ) { ( <http://cybersecurity.org/DataSet/
NSLKDD/Attack1> <http://cybersecurity.org/DataSet/NSLKDD/protocol_type> ) }
                                    ?s ?p ?o.

                                }
                                ORDER BY ?s ?p
                                LIMIT 10

type is SELECT
{"s": "http://cybersecurity.org/DataSet/NSLKDD/Attack1", "p": "http://cybersecurity.org/Da
taSet/NSLKDD/protocol_type", "o": "tcp^^http://www.w3.org/2001/XMLSchema#string"}
```

Figure 25: Launching the Gateway Server and Logs**.**

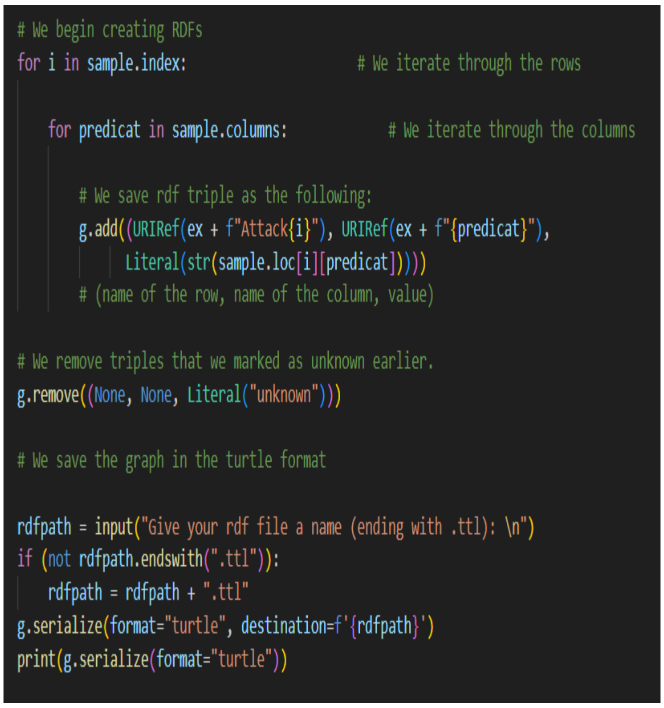
Note: '171.16.1.2' is the CT number 100 static IP address used as the Host Name.

3.3 Converting the dataset from CSV format to RDF

First, a conversion from CSV to RDF is required. Figure 26 shows its code 'anycsvtordf.py'.



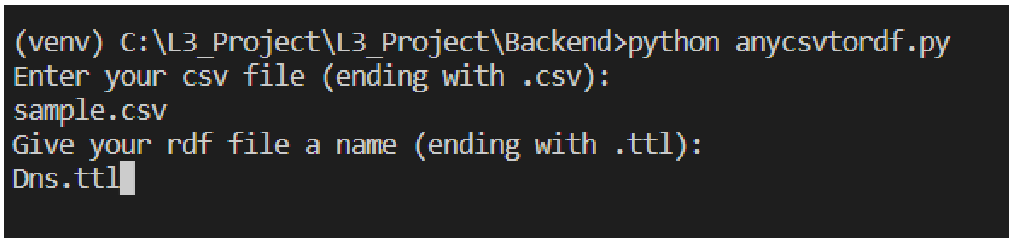
```python
# We import all the necessary packages and libraries
from rdflib import Literal, Namespace, URIRef
import pandas as pd
from rdflib import Graph as RDFGraph
from rdflib.extras.external_graph_libs import rdflib_to_networkx_graph


# We load the csv file as a panda dataframe and we drop all rows with null values
csvpath = input("Enter your csv file (ending with .csv): \n")

if (not csvpath.endswith(".csv")):
    csvpath = csvpath + ".csv"

sample = pd.read_csv(f'{csvpath}', low_memory=False)

# Here we deal with spaces (" ") in the data. I replace them with "_" so that URI's become valid.
sample = sample.replace(to_replace=" ", value="_", regex=True)

# We mark all missing/empty data as "unknown". This makes it easy to delete triples containing this later.
sample = sample.fillna("unknown")




# We initialise an RDF graph variable
g = RDFGraph()
mygraph = input("Enter the graph name")
ex = Namespace(f"http://cybersecurity.org/DataSet/{mygraph}/")
g.bind("Cy", ex)
```



```python
# We begin creating RDFs
for i in sample.index:                     # We iterate through the rows

    for predicat in sample.columns:              # We iterate through the columns

        # We save rdf triple as the following:
        g.add((URIRef(ex + f"Attack{i}"), URIRef(ex + f"{predicat}"),
              Literal(str(sample.loc[i][predicat]))))
        # (name of the row, name of the column, value)

# We remove triples that we marked as unknown earlier.
g.remove((None, None, Literal("unknown")))

# We save the graph in the turtle format

rdfpath = input("Give your rdf file a name (ending with .ttl): \n")
if (not rdfpath.endswith(".ttl")):
    rdfpath = rdfpath + ".ttl"
g.serialize(format="turtle", destination=f'{rdfpath}')
print(g.serialize(format="turtle"))
```

Figure 26: anycsvtordf.py .

Both the NSL-KDD and the DoH CSV datasets are loaded to produce the two RDF turtle files, ‘KDDTestRdf.ttl’ and ‘Dns.ttl’ respectively (Figure 27).

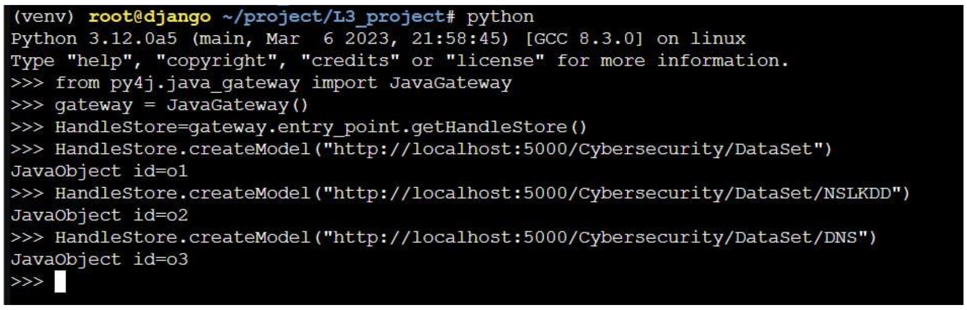


```
(venv) C:\L3_Project\L3_Project\Backend>python anycsvtordf.py
Enter your csv file (ending with .csv):
sample.csv
Give your rdf file a name (ending with .ttl):
Dns.ttl
```

Figure 27: Running ‘anycsvtordf.py’.

3.4 Creating the graphs inside our store

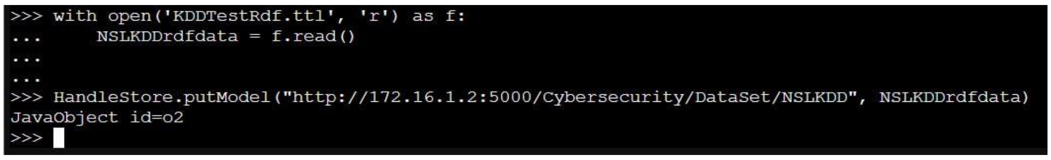
After Connecting to the store thanks to the Py4j API, we initiate graphs in our store (Figure 28).



```
(venv) root@django ~/project/L3_project# python
Python 3.12.0a5 (main, Mar  6 2023, 21:58:45) [GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from py4j.java_gateway import JavaGateway
>>> gateway = JavaGateway()
>>> HandleStore=gateway.entry_point.getHandleStore()
>>> HandleStore.createModel("http://localhost:5000/Cybersecurity/DataSet")
JavaObject id=o1
>>> HandleStore.createModel("http://localhost:5000/Cybersecurity/DataSet/NSLKDD")
JavaObject id=o2
>>> HandleStore.createModel("http://localhost:5000/Cybersecurity/DataSet/DNS")
JavaObject id=o3
>>> 
```

Figure 28: Initiating the models.

We then load our RDF graph into the store, here’s an example with the NSL-KDD graph:

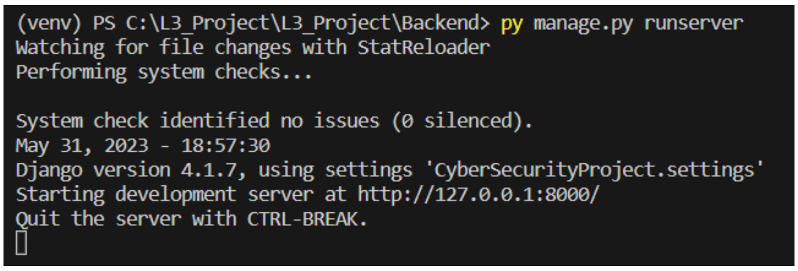


```
>>> with open('KDDTestRdf.ttl', 'r') as f:
...     NSLKDDrdfdata = f.read()
...
...
>>> HandleStore.putModel("http://172.16.1.2:5000/Cybersecurity/DataSet/NSLKDD", NSLKDDrdfdata)
JavaObject id=o2
>>> 
```

Figure 29: Loading our RDF graph into a model.

3.5 Django

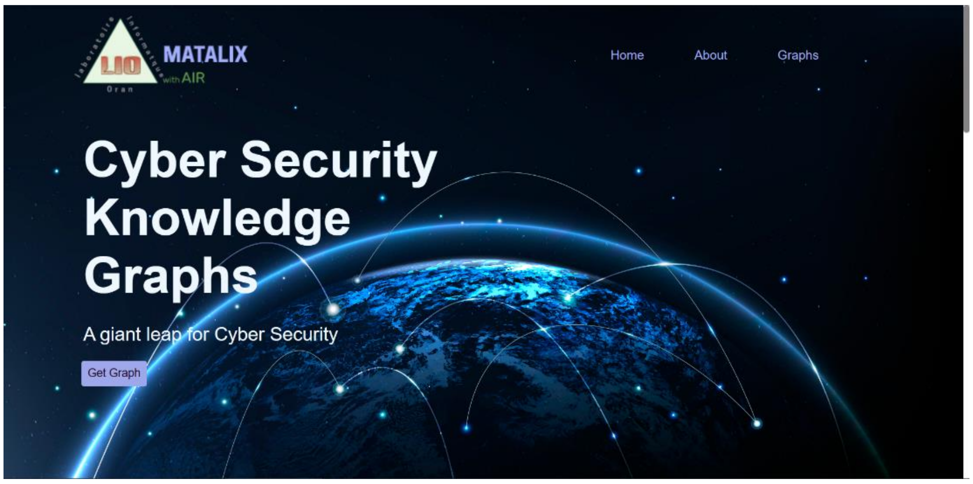
Once our store is up and running, we can then launch Django with runserver (Figure 30)



```
(venv) PS C:\L3_Project\L3_Project\Backend> py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 31, 2023 - 18:57:30
Django version 4.1.7, using settings 'CyberSecurityProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure 30: Launching Django.

3.5.1 Website home page

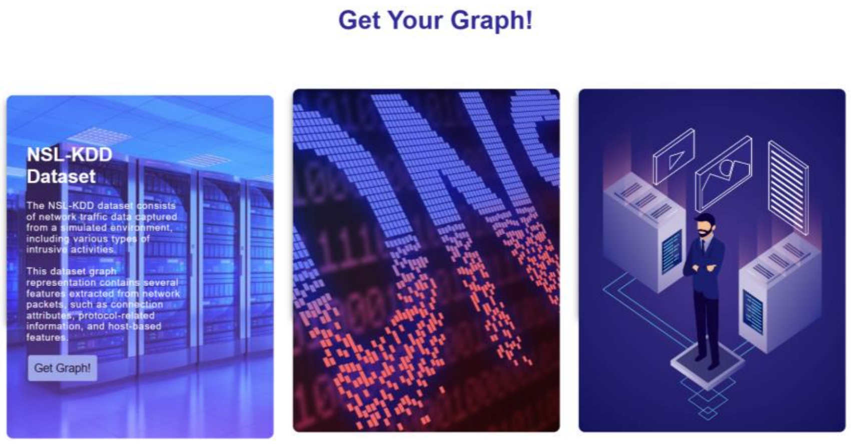


Cyber Security Knowledge Graphs

A giant leap for Cyber Security

Figure 31: Index Home Page ‘Index.html’.

3.5.2 The graph section



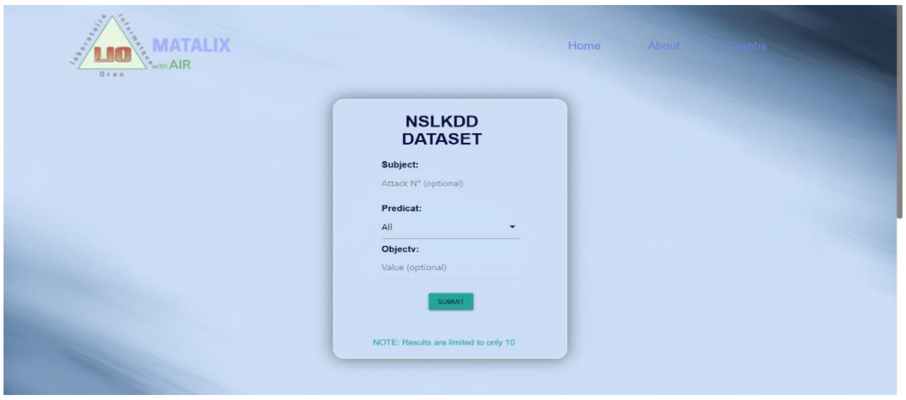
NSL-KDD Dataset

The NSL-KDD dataset consists of network traffic data captured from a simulated environment, including various types of intrusive activities.

This dataset graph representation contains several features extracted from network packets, such as connection attributes, protocol-related information, and host-based features.

Figure 32: The graph section on the home page.

3.5.3 The form page



NSLKDD DATASET

Subject:
Attack N° (optional)

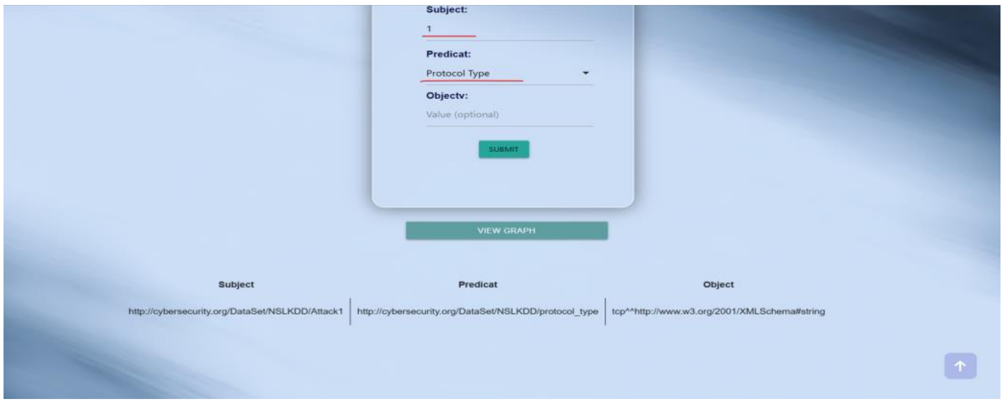
Predicat:
All

Objectv:
Value (optional)

NOTE: Results are limited to only 10

Figure 33: The form page.

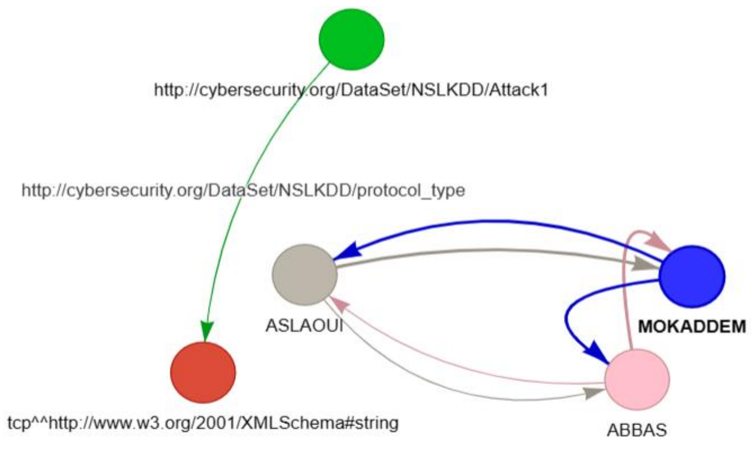
After querying for a specific result:



Subject	Predicat	Object
http://cybersecurity.org/DataSet/NSLKDD/Attack1	http://cybersecurity.org/DataSet/NSLKDD/protocol_type	tcp^^http://www.w3.org/2001/XMLSchema#string

Figure 34: Searching for specific attack information.

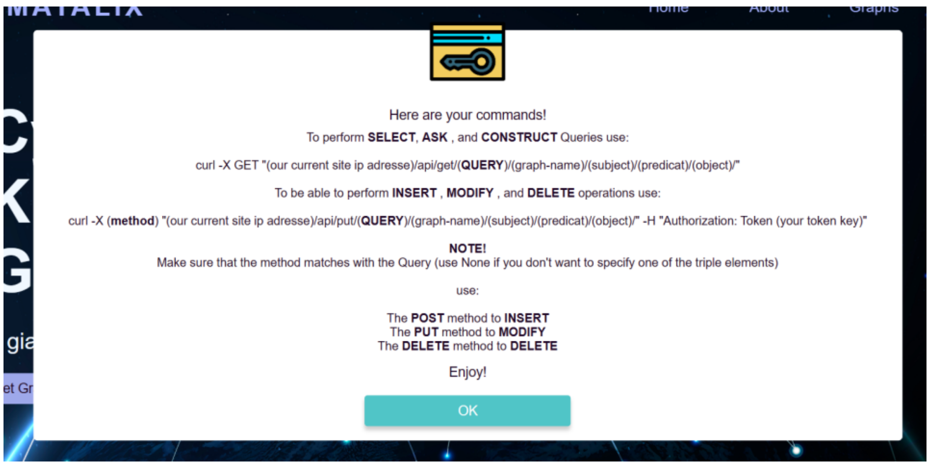
The corresponding visualized graph after clicking the 'View Graph' button (Figure 35):



http://cybersecurity.org/DataSet/NSLKDD/Attack1
http://cybersecurity.org/DataSet/NSLKDD/protocol_type
ASLAOUI
MOKADDEM
tcp^^http://www.w3.org/2001/XMLSchema#string
ABBAS

Figure 35: The visualized graph.

The panel that explains how to inspect or modify the graphs using APIs (Figure 36):



Here are your commands!

To perform **SELECT**, **ASK** , and **CONSTRUCT** Queries use:

```
curl -X GET "(our current site ip adresse)/api/get/(QUERY)/(graph-name)/(subject)/(predicat)/(object)/"
```

To be able to perform **INSERT** , **MODIFY** , and **DELETE** operations use:

```
curl -X (method) "(our current site ip adresse)/api/put/(QUERY)/(graph-name)/(subject)/(predicat)/(object)/" -H "Authorization: Token (your token key)"
```

NOTE!
Make sure that the method matches with the Query (use None if you don't want to specify one of the triple elements)

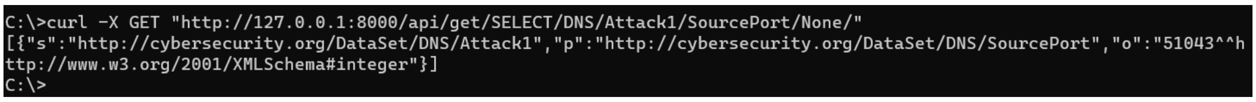
use:

The **POST** method to **INSERT**
The **PUT** method to **MODIFY**
The **DELETE** method to **DELETE**

Enjoy!

Figure 36: The API panel.

Finally, here’s an example of sending a request using CURL in the cmd (Figure 37):



```
C:\>curl -X GET "http://127.0.0.1:8000/api/get/SELECT/DNS/Attack1/SourcePort/None/"
[{"s":"http://cybersecurity.org/DataSet/DNS/Attack1","p":"http://cybersecurity.org/DataSet/DNS/SourcePort","o":"51043^^h
ttp://www.w3.org/2001/XMLSchema#integer"}]
C:\>
```

Figure 37: CURL command sending a get request while specifying the subject “s”, predicate “p”, and object “o” fields.

Conclusion

This chapter has illustrated the practical application of the proposed solution, by showing the various stages in the implementation of the design from the previous chapter.

We set up a data store. We developed a REST API using DJANGO Restful services to access the store via HTTP requests. All requests to Django are in JSON format.

Conclusion

Combining the concepts of knowledge graphs and cybersecurity ontology helped in developing a powerful knowledge representation of heterogenous data in cyberspace.

Datasets have provided an optimal approach to gathering known attacks across regions and organizing them into one document, where the fields categories can be converted into entities that contain attributes and attribute values. Thanks to RDF that helped convert these entities to triples, and consequently providing their storage and handling as graphs that act as knowledge bases, so called cybersecurity knowledge graphs.

The integration of current technologies such as NoSQL databases, Gateway Server (Py4j), and platforms such as Django REST Framework, and SPARQL, enhance the development of professional, functional, and reliable cybersecurity software.

Although setting up the Oracle NoSQL datastore was quite a difficult task due to its large parameters and data, it is the best and easy way to deal with Knowledge Graphs building as datastore, proving their scalability, high performance, and flexibility and serving as a huge advantage and ability to handle big data.

The cybersecurity knowledge graph visualization is based on the Django REST API used to send and retrieve requests to/from the store through HTTP requests. We made sure that the request sent to the store contained the HTTP method, the graph, the triple elements, and the query type. We ensured that all requests were returned to Django as JSON format, RDF turtle, or a Boolean value. We restricted any moderations done to the data in the store to authenticated users that possessed the authentication token. We developed a client in the form of a website that allows to view aspects of any desired CSKGs. A simple authenticated user, through a simple search using a simple form, can perform operations on the CSKG using specific commands.

The advanced facilities that the Gateway and JENA APIs offer allow java and python to easily communicate with the Oracle NoSQL store based on the advanced ability of SPARQL. This is typically a professional skill that we learned.

At this step, we think that we have successfully reached the project objectives and expectations, opening doors for future projects like integrating more and much larger datasets to the Oracle NoSQL store in the context of big data for a better, effective, and a more precise cybersecurity knowledge representation; for intelligent vulnerability detection that will support existing Machine Learning solutions. The integration of Machine Learning with cybersecurity knowledge graphs will enhance the works of Data Mining and Artificial Intelligence aiming to help produce powerful and intelligent intrusion detection systems.

References

- Liu, K., Wang, F., Ding, Z., Liang, S., Yu, Z., & Zhou, Y. (2022, April 10). A review of knowledge graph application scenarios in cyber security. *Cornell University*, 2. https://doi.org/10.48550/arXiv.2204.04769
- Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., & Lin, W.-Y. (2009, December). Intrusion detection by machine learning: A review. Expert Systems with Applications, 36(10), 11994-12000. https://doi.org/10.1016/j.eswa.2009.05.029
- Mukkamala, S., Janoski, G. and Sung, A. (2002) Intrusion Detection Using Neural Networks and Support Vector Machines. Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, 12-17 May 2002, 1702-1707.
- Thakkar, A., & Lohiya, R. (2020). A Review of the Advancement in Intrusion Detection Datasets. Procedia Computer Science, 167, 636-645. https://doi.org/10.1016/j.procs.2020.03.330
- Symantec, "Internet security threat report 2017," April, 7017 2017, vol. 22 Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf
- Breach_LeveL_Index. (2017, November). Data breach statistics. Available: http://breachlevelindex.com/
- Australian. (2017, November). Australian cyber security center threat report 2017. Available : https://www.acsc.gov.au/publications/ACSC_Threat_Report_2017.pdf
- Kai Zhang and Jingju Liu 2020 IOP Conf. Ser.: Mater. Sci. Eng. 768 052103 Review on the Application of Knowledge Graph in Cyber Security Assessment - IOPscience
- Thakkar, A., & Lohiya, R. (2020). A Review of the Advancement in Intrusion Detection Datasets. ScienceDirect. https://www.sciencedirect.com/science/article/pii/S1877050920307961
- Hofer, D., Jäger, M., Mohamed, A.K.Y.S. and Küng, J. (2021), "A study on time models in graph databases for security log analysis", International Journal of Web Information Systems, Vol. 17 No. 5, pp. 427-448. https://doi.org/10.1108/IJWIS-03-2021-0023

Krauss, C. (2021, June 7). Colonial Pipeline Paid Roughly $5 Million in Ransom to Hackers (Published 2021). The New York Times. Retrieved May 21, 2023, from https://www.nytimes.com/2021/05/13/us/politics/biden-colonial-pipeline-ransomware.html

- Llorens, Audrey. 5 Best Practices to Get More from Threat Intelligence, ThreatQuotient, Inc., 26 Oct. 2021, https://www.threatq.com/5-best-practices-more-threat-intelligence/.
- Sikos L F, Philp D, Howard C, et al. Knowledge representation of network semantics for reasoning-powered cyber-situational awareness[M]//AI in Cybersecurity. Springer, Cham, 2019: 19-45.
- Proxmox VE Administration Guide. (2023, March 22). Proxmox VE. Retrieved June 3, 2023, from https://pve.proxmox.com/pve-docs/pve-admin-guide.pdf
- Kennedy, William & Olmsted, Aspen. (2017). Three factor authentication. 212-213. 10.23919/ICITST.2017.8356384.
- Soren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. Springer, 2007.
- Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In Proceedings of the 16th international conference on World Wide Web, pages 697–706. ACM, 2007.
- Hoffart, J., Suchanek, F. M., Berberich, K., & Weikum, G. (2013, January). YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence, 194, 28-61. https://doi.org/10.1016/j.artint.2012.06.001

- Singhal, Amit (May 16, 2012). "Introducing the Knowledge Graph: Things, Not Strings". Google Official Blog. http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html
- Z. Syed, A. Padia, T. Finin, et al. (2016). UCO: A Unified Cybersecurity Ontology. https://www.researchgate.net/publication/287195565_UCO_A_Unified_Cybersecurity_Ontology
- Iannacone M, Bohn S, Nakamura G, et al. Developing an ontology for cyber security knowledge graphs[C]//Proceedings of the 10th Annual Cyber and Information Security Research Conference. 2015: 1-4.
- N. Chaibi, B. Atmani, and M. Mokaddem, Deep Learning Approaches to Intrusion Detection: A new Performance of ANN and RNN on NSL-KDD. In: ‘textitICPS Proc. Of the ISPR 2020: The ACM 1st int. conf. on Intelligent systems and Pattern recognition, Hamamet, Tunisia, Oct. 2020, pp. 45–49. https://doi.org/10.1145/3432867.3432889
- N. Chaibi, B. Atmani, and M. Mokaddem, A Convolutional Neural Network With Feature Selection Based Network Intrusion Detection. 1st National Conference on Applied Computing and Smart Technologies ACST’21, 2021, Sidi Belabes, Algeria, and in: *International Journal of Applied Evolutionary Computation*, 2022, Vol 13(1), pp. 1-21, https://doi.org/10.4018/IJAEC.302014
- N. Chaibi, M. Mokaddem and B. Atmani, A Random Forest Feature Selection Applied In A Deep Learning Based IDS. Third IEEE International Conference on Computer and Information Sciences 2021 (ICCIS 2021), 2021, Jouf University, Saudi Arabia.

Websites references

[1] *What is Confidentiality, Integrity and Availability - CIA Triad*. (2022). Creative Ground Technologies. Retrieved May 23, 2023, from https://creativegroundtech.com/what-is-confidentiality-integrity-and-availability/

[2] *What is CIA Triad & Why is it important?* (2022, August 23). Great Learning. Retrieved May 20, 2023, from https://www.mygreatlearning.com/blog/cia-triad/

[3] *System Risk Analysis | IT Security - The University of Iowa*. (2023). Information Security and Policy Office. Retrieved May 29, 2023, from https://itsecurity.uiowa.edu/awareness/data-types-and-regulations/system-risk-analysis

[4] *What Is Threat Modeling and How Does It Work?* (2023). Synopsys. Retrieved May 30, 2023, from https://www.synopsys.com/glossary/what-is-threat-modeling.html

[5] *What Are the Three Authentication Factors?* (2021, December 14). Rublon. Retrieved May 15, 2023, from https://rublon.com/blog/what-are-the-three-authentication-factors/

[6] *File system ACL*. (2021, March 2). IBM. Retrieved May 30, 2023, from https://www.ibm.com/docs/en/storage-scale/4.2.2?topic=STXKQY_4.2.2/com.ibm.spectrum.scale.v4r22.doc/bl1hlp_accessfilesystemacl.html

[7] *Know the types of cyber threats*. (2023). Mass.gov. Retrieved May 30, 2023, from https://www.mass.gov/service-details/know-the-types-of-cyber-threats

[8] *RDF - Semantic Web Standards*. (2014, February 25). W3C. Retrieved May 30, 2023, from https://www.w3.org/RDF/

[9] *RDF 1.1 XML Syntax*. (2014, February 25). W3C. Retrieved May 30, 2023, from https://www.w3.org/TR/rdf-syntax-grammar/#figure2

[10] *About the CIC | Canadian Institute for Cybersecurity | UNB*. (n.d.). University of New Brunswick. Retrieved June 1, 2023, from https://www.unb.ca/cic/about/index.html

[11] *NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. (n.d.). University of New Brunswick. Retrieved June 1, 2023, from https://www.unb.ca/cic/datasets/nsl.html

[12] *DoHBrw 2020 | Datasets | Research | Canadian Institute for Cybersecurity | UNB*. (n.d.). University of New Brunswick. Retrieved June 1, 2023, from https://www.unb.ca/cic/datasets/dohbrw-2020.html

[13] *What are NoSQL Databases?* (n.d.). IBM. Retrieved June 2, 2023, from https://www.ibm.com/topics/nosql-databases

[14] *Chapter 1. Introduction to Oracle NoSQL Database*. (2013). Retrieved June 2, 2023, from https://docs.oracle.com/cd/E26161_02/html/GettingStartedGuide/introduction.html

[15] *SPARQL 1.1 Query Language*. (2013, March 21). W3C. Retrieved June 2, 2023, from https://www.w3.org/TR/sparql11-query/

[16] *Introduction to Django*. (2023). W3Schools. Retrieved June 5, 2023, from https://www.w3schools.com/django/django_intro.php

[17] Singhal, G. (2022, March 3). *How to create a REST API with Django REST framework*. LogRocket Blog. Retrieved June 5, 2023, from https://blog.logrocket.com/django-rest-framework-create-api/

[18] *How to deploy Django*. (n.d.). Django documentation. Retrieved June 5, 2023, from https://docs.djangoproject.com/en/4.2/howto/deployment/

[19] *How to deploy with WSGI*. (n.d.). Django documentation. Retrieved June 5, 2023, from https://docs.djangoproject.com/en/4.2/howto/deployment/wsgi/

[20] *How to deploy with ASGI*. (n.d.). Django documentation. Retrieved June 6, 2023, from https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/

[21] Dagenais, B. (2022, August 12). *py4j · PyPI*. PyPI. Retrieved June 6, 2023, from https://pypi.org/project/py4j/

[22] *GatewayServer (py4j-java 0.10.9.3 API)*. (n.d.). Py4J. Retrieved June 6, 2023, from https://www.py4j.org/_static/javadoc/py4j/GatewayServer.html

[23] Rob. (2016, December 23). *How to run Java applications from other programming languages*. IDRSolutions. Retrieved June 7, 2023, from https://blog.idrsolutions.com/2-ways-we-could-make-our-htmlforms-conversion-easier-for-non-java-developers/