ISCTE IUL

Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Industrial IT Security Management supported by an Asset Management Database

André Filipe Tarrucha Narciso

A Dissertation presented in partial fulfillment of the Requirements for the Degree of

Master in Computer Engineering

Supervisor

Doctor, Maria Cabral Diogo Pinto Albuquerque, Assistant Professor

ISCTE-IUL

Co-Supervisor

Master, Tiange Zhao, Cyber Security Consultant

Siemens Corporate Technology

October, 2019

Acknowledgements

I would like to acknowledge how grateful I am for having the opportunity to be supervised by Dr. Maria Albuquerque. I also would like to thank my co-supervisor Tiange Zhao, for always being available to listen to my doubts, and answer my questions in a clear and professional manner. I will always be delighted to have had the opportunity to work in collaboration with her, and her commitment will be forever an inspiration to me.

I would like to express my gratitude to Dirk Kroeselberg for his advice, guidance and leadership throughout the project, which always had a compliment that maintained my motivation positive. To Tiago Gasiba, I would like to highlight all the help, concern and guidance that he offered me, for being the person responsible for this adventure in my life, because without him all my adaptation process would have been more complicated. To Siemens, I am thankful for giving me the possibility to perform my master's thesis in a professional context, with skilled and dedicated professionals, and for making available all the tools I needed to develop my work.

To my friend and university colleague Rafael Soares, thankful for having embarked on this great experience with me, and for all the adventures we lived together, and with the other Siemens' students. To my friends in my hometown, thank you for encouraging me to accept this opportunity and for all the support messages over the six months away from home.

Finally, and with the greatest importance of all, I am completely grateful to my parents who have always believed in my capabilities, for all the support they have given me, not only at this stage of my life, but always. Without your support, I wouldn't be who I am today.

Once again, thank you all.

Resumo

Gerir a segurança dos sistemas e activos de Tecnologias da Informação (TI) ao longo do seu ciclo de vida é uma tarefa muito complexa e importante para as organizações, onde o número de ameaças externas e de vulnerabilidades nos sistemas industriais continua a aumentar. Gerir a complexidade e ter uma visão geral clara de todos os ativos dentro das infra-estruturas industriais são desafios chave. Além disso, não há uma estrutura de dados bem definida para organizar todos os dados sobre ativos de TI de diferentes fontes industriais. Este estudo descreve uma solução para auxiliar a gestão da segurança de ativos de TI industriais, desenvolvida em colaboração com o departamento de Security Life-Cycle da Siemens CT. A ferramenta desenvolvida tem como objetivo integrar dados de ativos presentes na ferramenta de engenharia Component Object Server (COMOS) e no software de segurança Siemens Extensible Security Testing Appliance (SiESTA), utilizando uma base de dados Neo4j para armazenar e visualizar as relações entre os ativos, bem como os seus atributos relevantes para segurança. Foi realizada uma comparação entre cinco diferentes modelos de bases de dados com o objetivo de avaliar qual o mais adequado para os requisitos de base de dados definidos. Foi definido um modelo de dados, baseado na National Institute Standards Technology (NIST) Asset Identification Specification 1.1. Os objetos e relações do modelo de dados foram determinados para dar suporte aos casos de uso: Descoberta de *Hosts* , Análise de Portas e Gestão de Vulnerabilidades. Ao usar como input um ficheiro com a configuração de redes exportado do software COMOS, a ferramenta de suporte à base de dados permite importar e exportar dados da base de dados, e automatizar a criação de ficheiros para o SiESTA realizar testes de segurança em redes industriais. A solução proposta foi validada através de um questionário conduzido aos consultores de Segurança de TI, obtendo opiniões positivas sobre a utilidade da ferramenta na gestão de ativos em ambientes industriais.

Palavras-Chave: Gestão de Ativos, Modelo de Dados, Segurança de Rede, Redes Industriais, Base de Dados de Gráficos, Descoberta de *Hosts*, Análise de Portas, Gestão de Vulnerabilidades

Abstract

Managing the security of Information Technology (IT) systems and assets throughout their lifecycle is a very complex and important task for organisations, where the number of external threats and vulnerabilities in industrial systems continues to grow. Managing the complexity and having a clear system overview of all assets within industrial infrastructures are key challenges. Beyond that, there is no well-defined data structure to organize all data about IT assets from different industrial sources. This study describes a solution to support security management of industrial IT assets, developed in collaboration with the Siemens Corporate Technology Security Life-Cycle department. The database support tool aims to integrate asset data present on Component Object Server (COMOS) engineering tool and the Siemens Extensible Security Testing Appliance (SiESTA) security scanner software, using a Neo4j graph database to store and visualize the relationships between the assets, as well as their relevant security attributes. It was performed a comparison between five different database models, to assess which database model was more appropriate for the defined database requirements. It was defined a data model, based on National Institute Standards Technology (NIST) Asset Identification Specification 1.1. The objects and relationships of the data model, were determined to support the following use cases: Host Discovery, Port Scanning and Vulnerability Management. Using as input a network blueprint exported from COMOS software, the database support tool enabled to import and export data from the database, and to automate the creation of input files to enable SiESTA to perform scan tests on industrial networks. The proposed solution was validated through a questionnaire conducted to IT Security consultants, obtaining positive feedback on the tool usefulness in managing assets on industrial environments.

Keywords: Asset Management, Data Model, Network Security, Industrial Networks, Graph Database, Host Discovery, Port Scanning, Vulnerability Management

Contents

Acknowledgements **iii**

Resumo **v**

Abstract **vii**

Abbreviations **xiii**

List of Figures **xv**

List of Tables **xvii**

1 Introduction **1**
- 1.1 Context . . . 2
- 1.2 Motivation . . . 2
- 1.3 Research Questions . . . 3
- 1.4 Goals . . . 4
- 1.5 Outline . . . 5

2 Background **7**
- 2.1 Industrial IT Security . . . 7
  - 2.1.1 SCADA Systems . . . 8
  - 2.1.2 Purdue Enterprise Reference Architecture . . . 10
- 2.2 Asset Management . . . 12
  - 2.2.1 Industrial Asset Management . . . 13
- 2.3 Siemens ITAM Status . . . 15
  - 2.3.1 Host Discovery, Port Scan and Vulnerability Management . . 16
- 2.4 Related Standards and Specifications . . . 20
  - 2.4.1 ISO/IEC 19770 . . . 20
  - 2.4.2 NIST Publication for IT Asset Management . . . 21
  - 2.4.3 NIST Specification for Asset Identification 1.1 . . . 24

3 Literature Review **27**
- 3.1 Database Comparison . . . 27
  - 3.1.1 Database Models . . . 27
    - 3.1.1.1 Relational Database . . . 29

3.1.1.2 Graph Database 30
3.1.1.3 Document Stores 31
3.1.1.4 Column Stores 32
3.1.1.5 Key Value Stores 33
3.1.2 Database Requirements 33
3.1.3 Database Evaluation 35
3.1.4 Neo4j Graph Database 38
3.2 Security Software Tools 40

4 Proposed Approach and Solution 43
4.1 Proposed Approach 44
4.1.1 Proposed Architecture 44
4.1.2 Asset Data Model 45
4.2 Prototype Development 48
4.2.1 Tool Dependencies 48
4.2.2 Folder Structure 48
4.2.2.1 Excel Folder 49
4.2.2.2 JSON Folder 50
4.2.2.3 Scripts Folder 50
4.2.3 Dataflow 52
4.2.4 User Interface 57
4.2.4.1 Connection to Database 57
4.2.4.2 Extract and Import 58
4.2.4.3 Import and Export 59
4.2.4.4 SiESTA Template Writer 59
4.2.4.5 Manual Update 60
4.2.4.6 Update Database 63

5 Evaluation 65
5.1 Evaluation Method 65
5.2 Database Evaluation 71
5.3 Data Model Evaluation 76

6 Conclusions 79
6.1 Contributions 80
6.2 Future Work 81

Bibliography 83

Appendices 95

Appendix A - Questionnaire: Data Model Evaluation 95

.0.1 Personal Info 97
.0.2 Closed-ended Questions 97
.0.3 Open-ended Questions 98

Abbreviations

ABB	**A**sea **B**rown **B**overi
AM	**A**sset **M**anagement
ACID	**A**tomicity **C**onsistency **I**solation and **D**urability
API	**A**pplication **P**rogramming **I**nterface
APOC	**A**wesome **P**rocedures **o**n **C**ypher
COMOS	**Com**ponent **O**bject **S**erver
CRUD	**Cr**eate **U**pdate and **D**elete
CSV	**C**omma **S**eparated **V**alues
CVE	**C**ommon **V**ulnerabilities and **E**xposures
CVSS	**C**ommon **V**ulnerability **S**coring **S**ystem
DBMS	**D**ata**b**ase **M**anagement **S**ystem
DS	**D**esign **S**cience
GDB	**G**raph **D**atabase **B**enchmark
HMI	**H**uman **M**achine **I**nterface
IEC	**I**nternational **E**lectrotechnical **C**ommission
IP	**I**nternet **P**rotocol
ISO	**I**nternational **O**rganization **S**tandardization
IT	**I**nformation **T**echnology
ITAM	**I**nformation **T**echnology **A**sset **M**anagement
ICS	**I**ndustrial **C**ontrol **S**ystem
JSON	**J**ava **S**cript **O**bject **N**otation
NIST	**N**ational **I**nstitute **S**tandards **T**echnology
NVD	**N**ational **V**ulnerability **D**atabase
OCS	**O**pen **C**omputer and **S**oftware

OT	**O**perational **T**echnology
PCS	**P**rocess **C**ontrol **S**ystem
PDF	**P**ortable **D**ocument **F**ormat
PERA	**P**urdue **E**nterprise **R**eference **A**rchitecture
PLC	**P**rogrammable **L**ogic **C**ontroller
RDBMS	**R**elational **D**ata**b**ase **M**anagement **S**ystem
RTU	**R**emote **T**erminal **U**nit
SCADA	**S**upervisory **C**ontrol **a**nd **D**ata **A**quisition
SiESTA	**S**iemens **E**xtensible **S**ecurity **T**esting **A**ppliance
SQL	**S**tructured **Q**uery **L**anguage
SWID	**S**oft**w**are **Id**entification
XML	E**x**tensible **M**arkup **L**anguage

List of Figures

1.1 Design Science Research Methodology Process Model 4

2.1 Industrial Control System generic operation flow. 8
2.2 SCADA system general layout. 9
2.3 PERA reference model 10
2.4 ITAM Reference Functionality. 23
2.5 Typical Asset Lifecycle. 24

4.1 High Level Project Workflow 44
4.2 The process of extracting the asset data from COMOS source system. 52
4.3 Data flow of importing asset data to the Neo4j graph database. . . 53
4.4 Data flow of exporting the asset data from the Neo4j graph database. 54
4.5 Data flow of writing SiESTA scan input to detect vulnerabilities on network's hardware and devices. 54
4.6 The process of update the graph database with the SiESTA scan results. 55
4.7 User interface for "Connect to Database" functionality. 57
4.8 User interface for "Extract Data and Import" functionality. 58
4.9 User interface for "Import and Export" functionality. 59
4.10 User interface for "SiESTA Template Writer" functionality. 60
4.11 User interface for "Manual Update" functionality. 61
4.12 User interface for "Update database" functionality. 63

List of Tables

2.1 Summary Table of most relevant aspects of Standards and Specifications addressed in this study. 25

3.1 Quantitive evaluation of the database model requirements 36

3.2 Security software tools classification 41

4.1 Data model asset types 46

4.2 Data model relationships 47

5.1 Software versions used during the evaluation of the developed solution. 69

5.2 Hardware specification of the machines used during the evaluation. 69

Chapter 1

Introduction

This thesis arises in the context of an ISCTE-IUL partnership with Siemens Corporate Technology located in Munich, Germany.

This study was conducted over six months (from February to July 2019), where the author worked in collaboration with a project team from the Security Life-Cycle department.

1.1 Context

Nowadays there is a big hype and trend towards security, particularly Information Technology (IT) security. This knowledge area includes control and prevention of unauthorised access to organisational assets such as networks and data. As attackers are getting smarter and the tools available easier to use, the need to protect assets increased. Provide strong security to all organisation assets can be expensive, but it is essential that IT security teams should define response plans to current cyber threats. Any information breach can jeopardise the confidentiality, integrity and availability of industrial data and devices, thus threatening human safety, causing loss of intellectual property, reputation, ending up disrupting customers business processes.

Besides that, industries are under increasing pressure to reduce costs, meet tougher performance and production targets in order to maximize return on investment, while seeking to comply with regulatory requirements. In determining the continued operational performance and profitability of organisations, effective physical and IT Asset Management (ITAM) in all stages of its life cycle (e.g. acquisition, maintenance and renewal) plays an important key role.

Consequently, the current needs of organisations end up encouraging the search for opportunities that enable decision making and the gain of competitive advantages, improving the performance of assets, extending their life to the maximum while reducing maintenance costs.

1.2 Motivation

Industries such as chemical, power generation, oil processing and telecommunications, use industrial IT systems to manage their business and manufacturing processes. These are different from standard IT systems, and can range from few mounted controllers to complex and interconnected systems with thousand of networks and devices types.

In fact, managing the complexity and having a clear system overview are important challenges. The increase in complexity is likely to increase external threats, as the system will have more points likely to fail or be exploited by attackers. Clear overview should be understood as the ability to know at any time and with certainty, how many devices are on the network, what their current operating system configurations are, what software is installed as well as its versions. It is also important to know which ports are open, the list of known vulnerabilities for hardware and software on the device or which services are available through its interfaces. This lack of clear overview is ultimately the consequence of the absence of a well-defined data structure to organize all data about IT assets from different industrial sources (e.g. information of embedded devices, network devices, motors, office PC with remote access), essencial to perform ITAM with required quality. The motivation lies on the possibility of providing the industry a solution that ensures life cycle asset management, with a data structure that allows all the essential elements of the network architecture to be represented as well as its attributes.

1.3 Research Questions

As a research question, this study is intended to investigate a solution to describe and organize the industrial networks asset data set throughout its life-cycle.

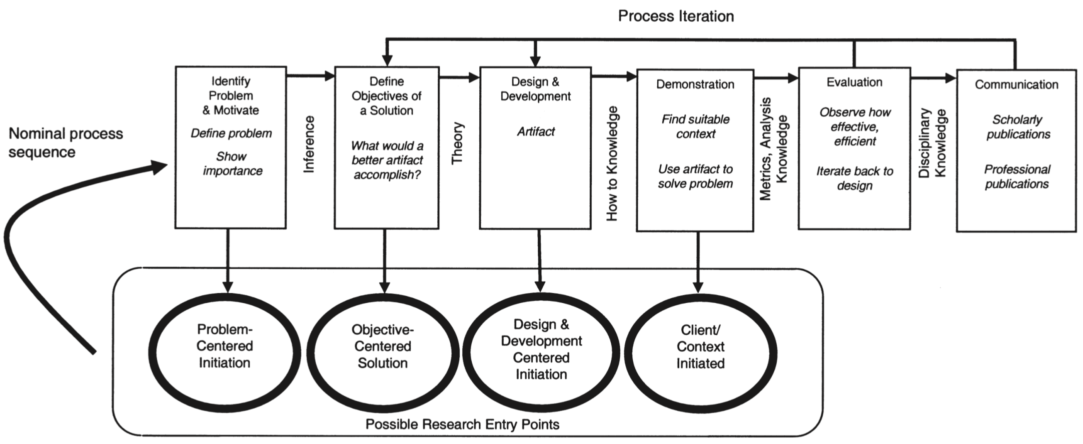
It is intended to research which asset types and relationships should be included in a data model to be defined, which must be representative of the actual network scenarios and suitable for the defined use cases. Beyond that, the research aims to select of a database type appropriate to the defined requirements.

Both the data model and the database must pursue a solution to the problem stated, of not having a well-defined structure to organize data about IT assets coming from different industrial sources and devices.

1.4 Goals

The main goal is to build a tool, that by using the chosen data model and database, enables support for asset management in industrial networks, answering to the challenges raised by motivation and research questions. This will be achieved by implementing a proof of concept prototype, using real data from a business unit and test it according to previously defined use cases. The study's evaluation will also involve assessing the specified data model and database chosen against the requirements initially defined.

As research methodology and to achieve the objectives of this study, it is intended to follow the work of Peffers et al [1], that presents and demonstrates in use, and evaluates a methodology for conducting design science (DS) research in information systems (IS). The proposed method incorporates principles and practices required to carry out DS research, describing a DS process with six steps: problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication. The figure 1.1 is a representation of this iterative process illustrating the different phases previously described through a model.


Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Source: A design science research methodology for information systems research [1]
FIGURE 1.1: Design Science Research Methodology Process Model

1.5 Outline

Chapter 2 describes the main concepts and challenges on the topics inherent to the study subject, such as the importance of industrial asset management and how industrial control systems security are key to services maintenance and delivery to today's societies.

Chapter 3 starts with a comparison between current databases types, explaining the basic concept, main advantages and disadvantages, highlighting the most popular implementations for each type. Afterwards, this chapter aims to define the necessary database requirements and assess which database type is most suitable to the problem raised by this study. This chapter also describes the database solution adopted, Neo4J Graph Database. The chapter proceeds with a research of relevant software tools in areas such as Asset Management, Security Scanning and Engineering Configuration, developed by representative companies in different industries.

Chapter 4 describes the proposed approach of develop a solution regarding the IT Security Asset Management problem in consideration. It details the proposed solution as well as systems, software and information involved, ending with a description of assets types and relationships within the data model. The second part of the chapter explains the prototype developed as proof of concept, in particular it explains the data flow, describing how each file is generated and used by different the systems. It is also presented the structure and external dependencies, ending with the description of different user interfaces and functionalities that are part of the prototype.

Chapter 5 explains the methodology used to evaluate the solution developed and describe the approach to evaluate the chosen database type and the data model, within the scope of the study conducted. The evaluation results are then analysed and discussed.

Finally, Chapter 6 concludes the study by showing the work contributions made to the research area and what aspects could be added or improved in the future.

This thesis is jointly carried out with support and supervision from Siemens, which is an industrial manufacturing company, and the ISCTE - Instituto Universitário de Lisboa. Some of the data is sensitive and will not be released according to the agreement made between the author and Siemens.

Chapter 2

Background

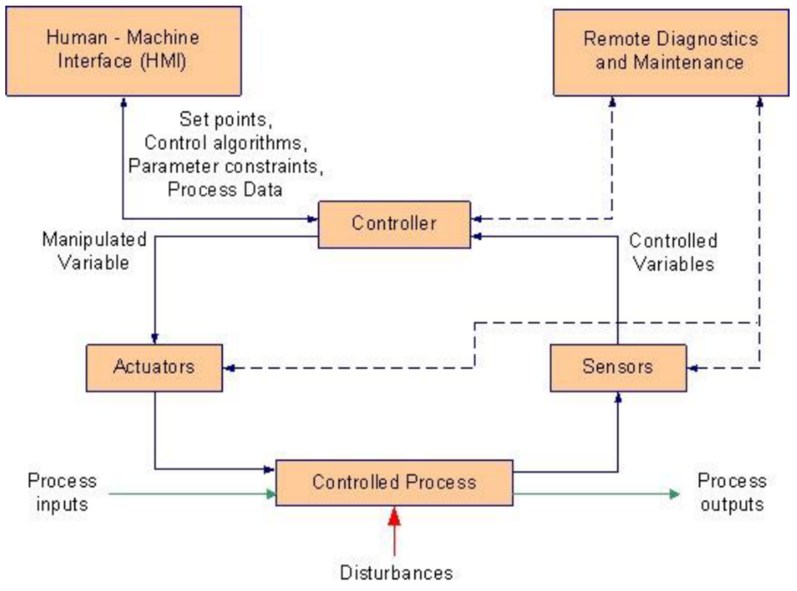
This section describes the topics and concepts covered throughout this thesis. The purpose is to describe the state of the art of asset management in organisations and industrial facilities, while highlighting the major challenges of asset management with focus on security related aspects in industrial solutions. Subsequently, topics such as the role of industrial control systems, the hardware components that generally compose them and the security challenges they present, are addressed in the context of Industrial IT Security. Finally, the last section addresses and highlights the importance of some current industrial standards and specifications, that present guidelines related to IT asset management within organisations.

2.1 Industrial IT Security

Industrial Control System (ICS) is a general term used to represent a combination of control systems and components, often found in industrial sectors and critical infrastructures, that act together to achieve an objective such as product manufacturing, matter or energy production. Its ability to control industrial processes makes them typically used in electrical, water, oil and chemical industries, among others [2].

Over the years, many ICS evolved with the insertion of IT capabilities in existing industrial systems, often replacing physical control mechanisms with embedded digital controls. This increased the connectivity and importance of these systems, but also created the need to improve their safety, security and resilience to threats.

As described by Stouffer et al [2], an ICS contains key components like control loops, human interfaces and diagnostics tools built on a layered network architecture. Figure 2.1 shows how a control loop uses devices known as sensors to produce a measurement of some physical property and send the collected information to a controller.


Human - Machine Interface (HMI)
Remote Diagnostics and Maintenance
Set points, Control algorithms, Parameter constraints, Process Data
Controller
Manipulated Variable
Controlled Variables
Actuators
Sensors
Process inputs
Controlled Process
Process outputs
Disturbances

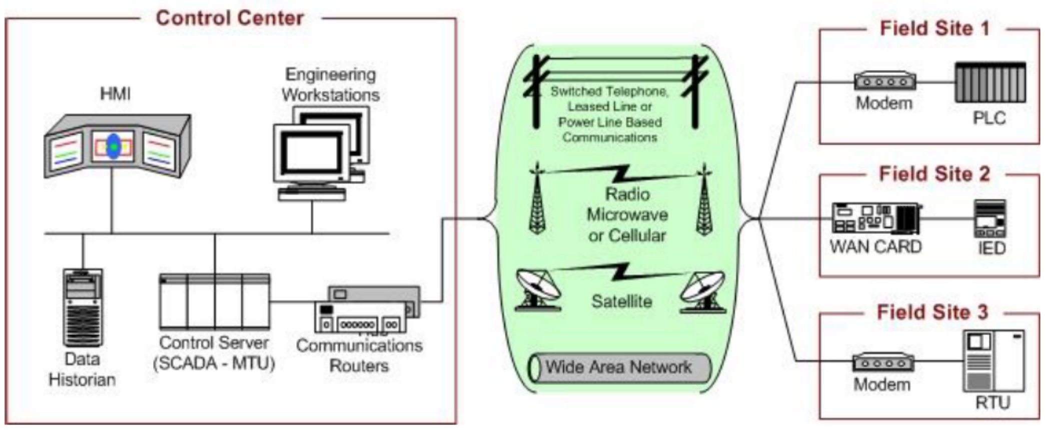
Source: NIST Guide to Industrial Control Systems [2]
FIGURE 2.1: Industrial Control System generic operation flow.

The controller interprets the signal received and generates corresponding command to an actuator, based on a control algorithm and a predefined target set of points. Actuators such as valves, switches and motors are then used to directly manipulate the controlled process based on the commands received.

2.1.1 SCADA Systems

Supervisory Control and Data Acquisition (SCADA) architecture is an ICS architecture type where the main objective is to control dispersed assets, with centralised data acquisition being as important as control. This system type integrates data acquisition systems with data transmission systems and Human-Machine Interface (HMI) software, providing centralised monitoring and control system for the inputs and outputs of industrial processes, by displaying graphic or text information to human operators in real time.

The figure 2.2 shows that the hardware include a control center (processes and stores information), communications equipment such as radio or cable (allowing the information transfer) and field sites composed by Remote Terminal Units (RTU) and Programmable Logic Controllers (PLC) controlling the local process (sending commands to actuators and monitoring sensors) [2].

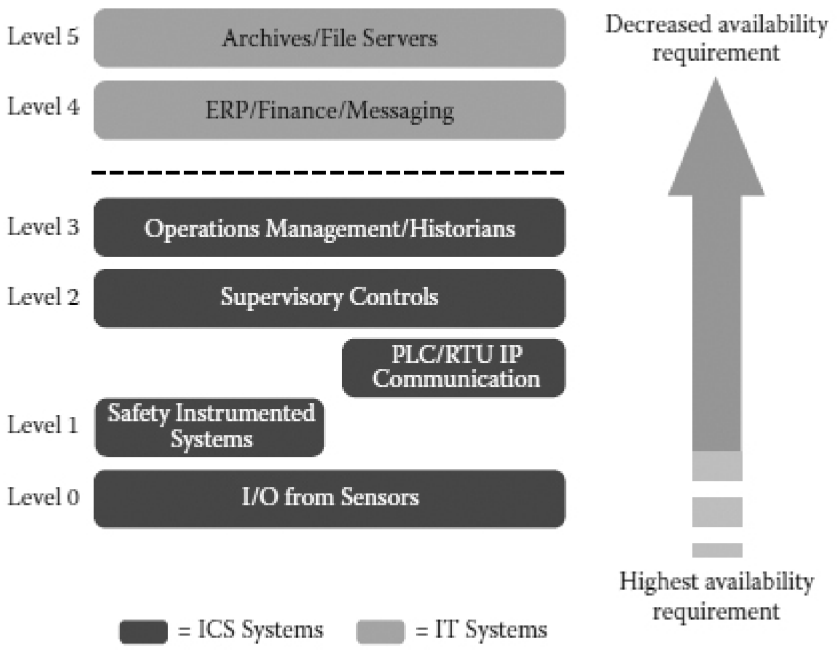


Source: NIST Guide to Industrial Control Systems [2]
FIGURE 2.2: SCADA system general layout.

It is important to know the generic architecture of these systems, because is heavily related to where the system to be developed will operate. Knowing the hardware device types also allow to gain knowledge of what information needs to be stored for each one and how they relate to one another.

2.1.2 Purdue Enterprise Reference Architecture

The Purdue Enterprise Reference Architecture, also known as PERA, is a useful tool to understand the distinction between ICS and IT systems, providing the capability for modelling the human component as well as the manufacturing component of any enterprise in addition to the information and control system component [3]. It is used for representing and designing the interface between ICS and IT, where production information is meant to be shared with management systems. It was written especially for IT departments so that they could understand why and how the control system networks needed to be segmented and what expectations each layer had for responsiveness. It also introduces network segmentation as a means for keeping traffic in a control network at deterministic levels or to specify test rules for the same network type.



Source: Cybersecurity for Industrial Control Systems [4]
FIGURE 2.3: PERA reference model

PERA recognises that a network architecture must manage assets (e.g. PLC's, RTU's and control servers) at different levels of speed to achieve adequate response and reliability. The architecture has 6 levels, from 0 to 5, where each level manage different systems, processes and asset types.

In lower levels of PERA, where sensors and actuators are located and basic control systems manipulate the physical process, high asset availability is the most relevant requirement. Level 2, known as Supervisory Control Level, include functions such as monitoring and supervisory control of the physical process, where devices related to SCADA systems can be found. In higher levels (management level) where IT-like assets can be found (e.g. Windows servers or desktop computers as HMIs), high confidentiality of information is valued at the expense of availability [4].

2.2 Asset Management

Organisations are aimed to provide services to people and to another organisations. The provision of these services is only possible after the assets acquisition by the organisation. The asset definition may vary regarding the organisation or entity, but is commonly accepted to be a resource with economic value that an individual or corporation holds with the expectation of providing a future benefit [5]. Therefore, assets can be evaluated as anything which has value and where organisations should be responsible by its management.

Assets might have different types, including financial assets (e.g. cash, stocks), physical assets (e.g. land, buildings), human resources, and IT assets (e.g. hardware, software, networks, intellectual property or data). IT assets are the most relevant for this study due to its presence in industrial networks. For medium and large sized companies, cost and management of these assets can be complex and should not be maintained manually. For IT assets, there are tools available that allow devices discovery and monitoring based on a given network or location. Data about IT assets is then usually stored within an asset inventory, commonly represented by a database. This data is then managed and updated using an asset management tool. The increasing number of tools available in the market allow organisations to carry out these processes in a more efficient and agile way.

Asset management comprises all of the infrastructure and processes necessary to effectively control and protect assets throughout their life-cycle, in order to get the most out of them and enable organisations to return on their investment [6]. The development of a single integrated view of organisation assets through the implementation of ITAM allow multiple benefits, enabling organisations to continually improve their informed decision-making and risk mitigation capabilities. The ITAM value is demonstrated in [7], through benefits such as compliance audit, risk's limitation, security, monitoring and detection, standardisation and cost savings.

2.2.1 Industrial Asset Management

Assets ownership, namely IT assets, present organisations with several challenges that require careful management. When attempting to compile an asset inventory, security engineers confessed [8] being challenged with problems related to asset identification: knowing asset operational functions, its exact location and lack of specific information such as status and current configurations. The different types of hardware (e.g. servers, workstations, and network devices), software (e.g. operating systems, applications, and files) make assets hard to track, thus creating lack of centralised control. Very often it does not exist a coherent definition of asset identification, or what asset attributes should be considered when creating an asset inventory. Beyond this, Gelle E. et al [9] states that the way an organisation is divided, increase the complexity and makes it harder to evaluate risk, assess vulnerabilities and respond quickly to external threats.

For industrial systems, IT infrastructures are becoming more complex and time consuming which end up requiring in-depth knowledge of IT protocols, interfaces, and compliance standards to manage such infrastructures. Therefore, the number of attacks on ICS's increases concerns about security issues in industrial assets and networks. The progress and development of new ideas is slow and difficulties when developing security measures lie above all in complexity and financial factors. These are systems that require high priority availability, being hard to perform maintenance and updates and to be able to find solutions, for instance, to manage vulnerabilities that exist in used machines. This makes industrial network security time-consuming and requires effort from all organisation elements.

Industries used to adopt "security through obscurity", but nowadays it's too difficult to manage physical facilities as if they were completely separated from external networks. The industrial systems are thus exposed to outside threats and vulnerabilities that can be soon exploited by attackers [10]. It is known that the perfect solution for secure environment doesn't exist, so [10] defines the mission as finding "a viable approach to maintain the stability within a given environment so that the production can be carried on." For that to be achieved, is suggested that one of the mains tasks should be "to systematically find all the vulnerabilities of the assets so that the risks could be investigated and foreseen".

2.3 Siemens ITAM Status

The solution being developed is aimed to be used by a Siemens business unit where the current status shows no well-defined data structure to organize all data set from different sources, as required for ITAM. This project aims to enable a network security scan and management, where there is a need to compare the industrial blueprint specification with the actual configuration status of a given industrial network and its devices. It also aims to help overcome the isolation that exists regarding asset information, spread across different systems.

The following tools are developed by Siemens to support asset management within an industrial environment. These are software and hardware tools defined to have a role on the Siemens project that serves as the backbone for this study.

COMOS - acronym for Component Object Server, it is an integrated plant engineering management software developed by Siemens, used on different industries to manage the entire project life-cycle, from engineering blueprint to operation [11]. Each project's phase has different tasks and each task presents challenges to the engineers. The software is designed to help solve these challenges ensuring an information flow without data loss in plant's life-cycle phases, making project integration, planning and documentation easier. The industry advantages of using COMOS are shorter project runtimes through faster and more efficient engineering, collaboration of all project stakeholders on a global database. It also enables cost savings through an always up-to-date and consistent data and fast commissioning through reduction of errors. This information is only available on Siemens intranet.

SIMCO - a software product developed by Siemens , also known as SIMATIC Management Console [12], that provides functions for determining inventory data and for software administration of a Process Control System (PCS) plant, showing current status of installed hardware and available software components. The use of this product on industrial environments allows to reduce costs with administration, due a more efficient workflow and fast analysis of installed software and hardware versions, enabling precise updates planning.

SiESTA - a hardware and software product introduced by Siemens in 2014, which stands for Siemens Extensible Security Testing Appliance, to check vulnerabilities on its products at an early development stage, thus closing security gaps [13]. Once configured, it runs automatic test programs based on various existing security software tools (e.g. Nmap, Nessus, Kali). SiESTA is also offered as a service to end customers, enabling the identification of very well known vulnerabilities such as Heartbleed (a bug that allows an attacker to read the memory of a server or client) and Wannacry (a crypto-ransomware that affects Windows operating system diffused through phishing techniques). It also enables the discovery of new vulnerabilities sending invalid inputs to systems, verifying the device's behaviour. On an industrial environment, SiESTA is also used to do network security scans determining which devices exist within the network, which ports are open and which services are running, making sure that they are properly configured.

2.3.1 Host Discovery, Port Scan and Vulnerability Management

The tool to be developed aims to provide support to asset management and intend to be used in particular with use cases involving host discovery, port scanning and vulnerability management. These use cases were chosen by project managers at Siemens as the ideal ones to start a prototype, and then make it possible to add other cases of higher complexity. Therefore these are important topics for the study performed.

The purpose of host discovery is to detect whether the host is alive and reachable and then to further find out any possible host information, such as operating system or running applications [14]. Host detection is usually done in a scenario where one attempts to determine the accessible hosts on a network [15] and thus get a network hosts map.

Network scans can follow two different approaches: a passive or an active scanning. One of the main interests in using a passive approach is that the information gathering process has no impact on the bandwidth or on the monitored assets, allowing near real-time awareness of the security posture of ever-changing networks, and thus helping network administrators remain in control and anticipate upcoming security problems. This is in contrast with active scanning techniques that are often noisy and intrusive [16], taking up a lot of bandwidth.

A port scan is a method to determine which services are available on a host or a network by sending a message to ports and listening for an answer, where the response can indicate port status and other informations about the host [17]. Scanning ports can be a useful and important technique to gather technical information to network and security administrators, because it characterises the amount of hosts and networks exposed to potential external attacks [18], but also can be used by attackers searching for weak points to gain access to network devices. A survey performed by Bhuyan et al [17], includes a discussion of common port scan attacks while provides a port scan methods comparison based on type, detection mode and mechanism used for detection. Another study [19] presents a port scan detector that is effective against stealthy scans and show different approaches for port scans.

Port scanning can also have negative effects like, congesting the network, enabling future attacks, and if repetitive port scans are made, it can create a denial of service [20]. Services can use two port types for communication: Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). Port scanning usually happens on ports that use a connection-oriented protocol, like TCP, while UDP ports can be easily blocked by network defenders [17].

The universe of tools available have common functionalities while some of them can be used for specific purposes. Among the existing ones, popularity of Nmap [21], Nessus [22] and Netcat [23] can be highlighted.

Vulnerabilities are security implications, which hackers can use directly to access protected data, which provide confidencial information or capabilities in a direct access. An explosion in the different ways to access and use these systems accompanies this increased visibility, giving access to an unauthorised vulnerabilities that can expose private information about customers and employees. Determining the vulnerabilities and exposures embedded in commercial and open source software systems and networks is a critical first step where a simple patch, upgrade, or configuration change could be sufficient to eliminate even the most serious vulnerability. [24].

Ten et al [25] published a study on the impact of a cyber attacks on SCADA systems. The authors affirm that is known the relationship between the vulnerabilities increase with the connection of these systems to external networks such as the Internet. The authors recommend the development within a test bed, as an effective way to identify vulnerabilities in an industrial infrastructure.

The existence of different names by different organisations for the same vulnerability made it difficult to identify which vulnerabilities should or should not be sought by security tools. In 1999, MITRE Corporation [26] began designing a method to sort through the confusion, critical for the information-security community, creating a reference list of unique vulnerability and exposure names and mapping them to appropriate items in each tool and database, named Common Vulnerabilities and Exposures (CVE). This is cross-industry effort to create and maintain a standard list of vulnerabilities and exposures, where its adoption enabled a more systematic and predictable handling of security incidents [24]. It is also noteworthy the National Vulnerability Database (NVD) developed by NIST [27], as a repository of standards based on vulnerability management data, security measurement, and compliance. NVD staff are tasked with analysis of CVEs by aggregating data points from the description, references supplied and any supplemental data that can be found publicly at the time. This analysis results in association impact metric known as Common Vulnerability Scoring System (CVSS). This score classifies the threat level placed by each of these system vulnerabilities, publicly alerting system manufacturers of which vulnerabilities present the greatest threat and consequently the highest priority to be fixed. Another important system vulnerability repository is the Vulnerability Notes Database (VND) managed by Carnegie Mellon University. The Vulnerability Notes Database provides information about software vulnerabilities that can include summaries, technical details, and lists of affected vendors [28].

2.4 Related Standards and Specifications

The use of standards is something that has existed for thousands of years in societies, for instance, for measure distances or weights. With this in mind, standards are what allows us to make sure that we all use the same degree of comparison, as if it were a baseline. Standards promote production efficiency, and they increase consumer confidence by ensuring consistent product quality. Standards also mean more safety from a legal perspective – in meeting standards manufacturers achieve a high level of legal security [29].

A specification is often a type of technical standard which is often referenced by a document providing necessary details for a set of requirements to be satisfied by a design, product, or service. Specifications are often provided by government agencies, standards organisations, such as National Institute of Standards and Technology (NIST), International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC).

2.4.1 ISO/IEC 19770

The family of standards ISO/IEC 19770 define processes and technology needed for managing software and related IT assets. This international standard specifies requirements for an ITAM system within the context of an organisation and can be applied to all types of IT assets by all organisations types and sizes. It was created to manage IT assets but can be applied to other asset types as well [30]. The ISO 19770 is divided in five major parts, where the one more related to this study, provide an ITAM data standard for Software Identification Tags (SWID) [31]. A SWID tag is a standardised data structure, which contains information about a software product. Detailed SWID information essentially provides price-efficient management assistance and automation possibilities. Security-wise SWID tags provide software management assisting data which may help to identify vulnerabilities and mitigate them. Through the use of SWID tags several other benefits can be achieved in software maintenance, such as: metadata for consistent and authorised software identification, updates, issues or vulnerabilities that can be related to automatically installed software and automated license handling [32].

It is also worth mentioning the IEC 62443 standard, known as series of international industrial security standards that include a set of recommendations for defending networks against threads, helping industries to be prepared for the rising cyber attacks, thus preventing equipment damage [33]. IEC 62443 standard defines five security levels: from level 0, to systems where security don't exist at all, up to level 4, focused in systems that should be resistant against nation-state attacks).

2.4.2 NIST Publication for IT Asset Management

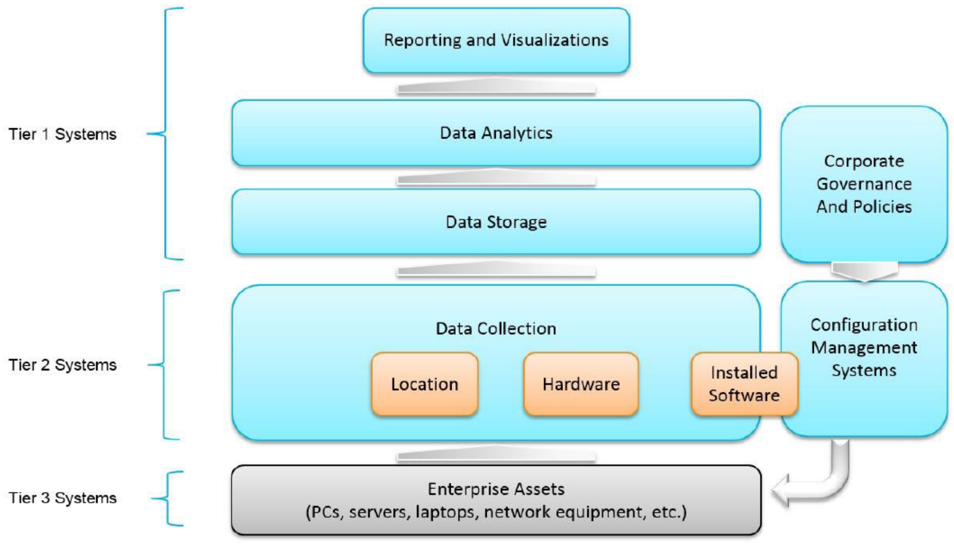
In this guide [8], NIST shows a proof-of-concept solution using available technologies that can be implemented to track the location and configuration of network devices and software within an enterprise, helping them align more easily with relevant standards and best practices. It addresses a common solution for traditional physical asset tracking and security, IT asset information, vulnerabilities and compliance information where users can query a system to gain insight from IT assets.

Different roles in an organisation can find usefulness within this guide, such as chief security and technology officers that will be interested to know what challenges the organisations face when implementing an ITAM system. Technology or security managers will be concerned with how to understand, identify, assess and mitigate risk. IT professionals will find the guide useful because it presents the configuration steps for a practical approach.

ITAM is defined on the guide, as a set of policies and procedures that an organisation could use to track, audit and monitor the state of IT Assets and their system configurations. For NIST, an effective ITAM solution should complement existing asset management, security, and network systems, providing an Application Programming Interface (API) to communicate with other security devices such as firewalls or intrusion detection systems. It should be able to know and control which assets are connected to the enterprise network, automatically detect and alert when unauthorised devices attempt to access the network. From the administrators point of view it should be able to define and control the hardware and software that can be connected to the corporate environment, enforcing software restriction policies to what it's allowed to run in the corporate environment. The system should be enable audit and monitor changes in assets state, thus the solution must be integrated with log analysis tools to collect and store audited information [8]. When fully implemented, the ITAM solution provides to an organisation fast responses to security alerts by showing updated asset information and latest pertinent errors. It also increases cyber security resilience and reduces the attack surface of machines by ensuring that software is correctly updated, helping analysts focusing on the most critical assets. It also improves and reduces reporting time for management and auditing, improving the quality of administrative decision making [8].

NIST claims that the ITAM processes can be supported by the reference architecture presented on Figure 2.4. It has three different tiers and shows how data flows through the ITAM architecture. Each tier is composed by different components. Tier 3 is made up of all assets being tracked including hardware, software, and virtual machines. Tier 2 includes the sensors and independent systems that feed data into the enterprise ITAM system. Tier 1 comprises the enterprise ITAM components that provide data aggregation from all Tier 2 systems into business and security intelligence.

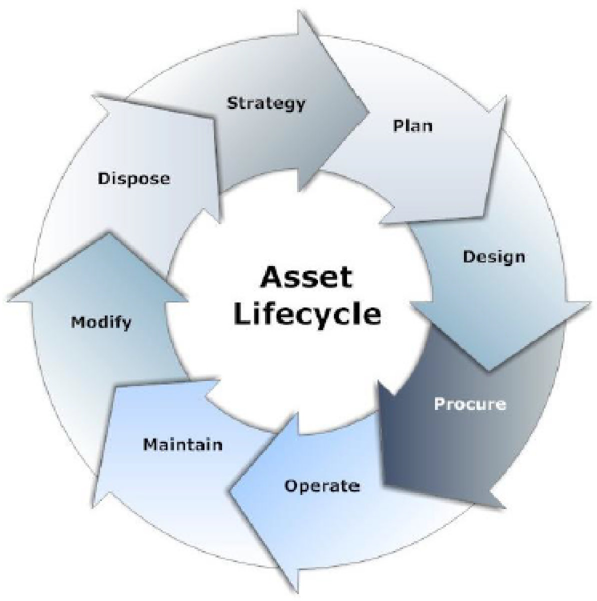
Data Collection process enumerates the unique software and system configuration of each Enterprise Asset and sends the information to Data Storage. Data Storage process, after receiving this data, organizes the data in the desired format and stores it in a storage system. Data Analytics process performs analytic functions on data made available by the Data Storage process. Corporate Governance and Policies stores the rules defined for IT assets, like networks and websites that employees can visit or which software can be installed. Configuration Management


Reporting and Visualizations
Tier 1 Systems
Data Analytics
Data Storage
Corporate Governance And Policies
Tier 2 Systems
Data Collection
Location
Hardware
Installed Software
Configuration Management Systems
Tier 3 Systems
Enterprise Assets
(PCs, servers, laptops, network equipment, etc.)

Source: NIST Publication IT Asset Management [8]
FIGURE 2.4: ITAM Reference Functionality.

Systems make use of Corporate Governance and Policies through actions like software patches and updates, blacklisted software and automation for configuration updates. Finally, the Reporting and Visualisation process generates deliverable documents with numerical tables of information to support decision making.

NIST guide also describes, in Figure 2.5, that the typical asset life-cycle within an organisation can go through the following main phases: Enrolment (Plan, Design, Procure), Operation (Operate and Maintain, Modify) and End-of-Life (Dispose and Strategy). Enrolment phase involves initial activities performed by IT staff like assigning new asset and its attributes into a database. In Operation phase, the asset can go through multiple changes, such as introduction of new or unauthorised software, hardware upgrades, where all changes need to be tracked and recorded. Asset changes are monitored using agents installed on the asset, as well as network-based monitoring systems that capture network traffic, sending periodic reports to the analytics engine. When the asset reaches the end of its life in the organisation, it goes through activities that include the configuration, data and registration removal from the asset database.



Source: NIST Publication IT Asset Management [8]
FIGURE 2.5: Typical Asset Lifecycle.

2.4.3 NIST Specification for Asset Identification 1.1

The NIST Specification for Asset Identification document describes the approach to asset identification with the objective to uniquely identify assets based on known identifiers and information related to them. It features an asset identification data model with different asset types and its attributes, methods for identifying assets and conformance requirements to comply with the specification [34] .

The specification defines eleven asset types and definitions of how these asset types may be identified using a set of literal attributes and relationships to other assets. It also specifies that two element can be connected to each other by a defined relationship. This connection between the elements could also have attribute information to enhance the relationship context. There are several extension points where the data model allow for the inclusion of asset types beyond what is included as well attributes and relationships.

All asset types are derived from two abstract elements: Asset and IT Asset. Some of the elements presented on the specification will be used as a baseline for the study to be conducted. The relevant types and attribute information present on the asset identification data model for the thesis' subject are Software, Network, Service and Computing Device and all of them are considered Concrete Elements.

This specification is therefore regarded as relevant for this thesis topic because allows to have a guideline perspective of an asset data model and how the relationships work between them. However, this specification does not fit completely well for the project's scope, because it describes asset types not relevant (e.g. People, Organisation, Website) and lacks of other needed asset types (e.g. Interface, Vulnerability). There is also lack of detail in the attributes defined for each asset type. Even so, the specification will be used as a baseline for the study to be conducted.

The following Table 2.1 shows a summary of the most relevant aspects of each standard discussed throughout this chapter, highlighting its relevance to the current thesis.

Publication	**Relevance**
ISO/IEC 19770	Specify the requirements for an ITAM system, addressing both the processes and technology for managing software assets and related IT assets within an organisation scope.
NIST Publication for ITAM	Shows a proof-of-concept solution to track the location and configuration of network devices and software in an organisation, while describes the challenges and benefits of ITAM. It presents an ITAM architecture divided by three tiers, each one composed by different device types, explaining how data flows through the system architecture. Propose an example of typical asset life-cycle, describing what tasks should be done on each phase.
NIST Specification for Asset Identification 1.1	Describes an approach throughout a specification to asset identification, proposing a baseline data model with defined asset types, attributes, relationships and compliance requirements, that can be used as a guideline.

Table 2.1: Summary table of most relevant aspects of standards and specifications addressed in this study.

Chapter 3

Literature Review

3.1 Database Comparison

In this section it is intended to do a research about the database state of art, starting with a concept description of each database model, its features highlighting the differences between models, technical advantages and disadvantages as well as some purposes for which they are used and which business problems they help to solve. After considering the options currently available, the database requirements will be defined and described. Subsequently, the intention is to evaluate how the different databases models fit the defined requirements and choosing the best one to store the asset data.

3.1.1 Database Models

The need for humans to store data is not new, and from an early age has been important to store paper records in institutions such as government offices, hospitals, libraries and private companies, to allow business progress monitoring and all assets identification. A database can be understood as an organised collection of stored data to serve some predefined purpose. Access to this data is often provided by a Database Management System (DBMS) software that allows user to interact with it. The database universe can be currently divided between relational databases and non relational database models [35].

The survey elaborated in [36] define database model as a collection of conceptual tools used to model representations of real-world entities and the relationships among them and is also often used to refer to a collection of data structure types. In 1970, E.F.Codd published a paper [37] that changed the way people thought about databases, proposing the fundamental concepts for the relational database model, which eventually became the standard for database systems. In 1987, ISO 9075 is published [38], which determines Structured Query Language (SQL) as a standard query language for information processing systems. This was also a consequence of the commercial popularity of relational databases regarding other available models, such as network, object-oriented and hierarchical database models. The arrival of the Internet to most consumers in early 2000s, has also expanded database applications for new purposes and brought new approaches to database models.

Google Big Table [39], that started to be developed since 2004, is usually accepted as the first major commercial non-relational database system. Their success, together with Amazon Dynamo [40], brought to the surface many other open source non-relational database development projects. These databases have seen their popularity grow due to betting on features such as ease of access, speed and scalability [41].

In the same study conducted above, authors argue, based on results of tests performed, that contrary to what is generally accepted as true, not all NoSQL databases perform better than SQL databases. Each database performance varies with each operation, concluding that there is little correlation between performance and the data model each database uses. This allows us to say that there is no "one size fits all", or better model of database, thus the best choice should always take into account the purpose for which it will be used.

There are more than the database models represented here. However, the five present here represent the most popular, widely used [42] and thus obvious choices for most purposes. It is not intended to make an exhaustive evaluation of each model, and the facts shown may always be debatable regarding on several variables under study.

3.1.1.1 Relational Database

A relational database store the data in a tabular form of columns and rows. For each table, a column represents an attribute, each row a record, and each field within the table represents a data value. This is the most popular, well accepted and widely used database model benefiting from many years of improvement and good customer support from large companies that develop relational database software, such as Oracle and Microsoft. The use of SQL as standard query language make it easier to learn and interpret because its syntax is simple. Other advantages of relational databases are evidenced in Atomicity, Consistency, Isolation and Durability (ACID) compliance as a model for storing data. These are good choice when the data structure is very predictable, since unstructured or semi-structured data will not work well due the schema and type constraints.

According to some of the surveys carried out [43], relational databases seem to have limitations when used in large scale and high-concurrency applications such as slow reading and writing, being prone to bring deadlocks and other concurrency issues. Relational Database Management Systems (RDBMS) are also poor at handling data relationships because they do not store relationships between data elements as explicitly as the non-relational databases [44]. When it's time to write some queries, it might be necessary to do joins with different tables to obtain the relationships between data, which may increase the complexity and computing resource consumption.

There are several applications of relational databases in our daily lives, however, it should be highlighted its use in banking, industrial and retail systems all over the world. Benefiting from many years of improvement and stable customer support, the relational databases are then usually linked to legacy systems with decades of existence where the main criteria is continuous system stability and availability. Most popular implementations for relational databases include Oracle [45] and MySQL [46].

3.1.1.2 Graph Database

It's a non relational model that uses graph structures such as nodes, edges (or relationships) and properties to represent and store data, where the data management is ensured by the possibility of executing Create, Update and Delete (CRUD) operations. A graph is built and based on graph theory where nodes (or vertices) represent entities or classes. Edges (or lines) are connections between different nodes representing relationships and can be defined either as uni-directional or bi-directional, depending on the purpose and meaning needed to be represented. Properties are detailed information about nodes and relationships commonly used to filter and select data based on values. Retrieving data from a graph database requires a query language different from SQL. Currently, the most commonly used are SPARQL, GraphQL, Cypher and Gremlin [47], but no single graph query language has been universally adopted yet.

Today's organisations need to manage large volumes of data and extract insights from it. Current applications for graph databases are helping overcome today's data challenges. Most common use cases [48] are Fraud Detection (where real-time analysis of data relationships helps finding scams), Knowledge Graphs (building better digital asset management powered by graph-based search tools), Network and Infrastructure monitoring (helping ensuring systems cybersecurity) and Recommendation Engines (improving companies services customising products and content in real time). Graph databases are a good answer to these use cases because relationships between nodes are more important to generate knowledge than the nodes itself.

Thus, graph databases are the technology that handles and stores relationship information very well, as a first-class entity. Most popular implementations for graph databases include Neo4j [49] and OrientDB [50].

3.1.1.3 Document Stores

Document database is another popular non relational model presenting a schema-free data organisation that handles semi-structured and unstructured data. All document information is saved in a single database instance representing the object, its attributes and relationships to other data elements, where data can be queried based on multiple attribute value constraints.

In relational databases, two of same type records will have the same data fields and unused ones are kept empty, while in document stores, two of same type records may have dissimilar data fields, displaying data schema flexibility. This database is also known for storing data in form of documents encoded in different formats including Extensible Markup Language (XML), Portable Document Format (PDF) and JavaScript Object Notation (JSON).

As described in [35], document stores systems generally do not provide explicit locks, have weaker concurrency and do not provide ACID transactional properties in comparison with traditional ACID-compliant databases. Although, comparison made in [51] states that document oriented databases should be used for applications in which data don't need to be stored in a table with uniform sized fields, offering great performance and horizontal scalability options, but should be avoided if the database has too many relationships.

The study performed by Zhang [52], shows a software development for managing project documents in an Internet-based collaborative environment, using a document-oriented database technology. The author justify their choice saying that this database model has the advantages of efficient storage of big data files and data replication over many servers, while relational databases work poorly in performance, scalability and reliability when processing big amounts of data spread out across many servers. Nevertheless, the author argue the fact that non relational databases do not have a universal query language, weaker authentication and security management compared with relational databases act like a disadvantage. Most popular implementations for document store databases include

MongoDB [53] and Couchbase [54].

3.1.1.4 Column Stores

Column Store is a data model very suitable for semi-structured data, and not too complex queries, that enables storing data using column families in massively distributed architectures [51]. Each database table column is stored separately, with attribute values belonging to the same column stored contiguously, compressed, and densely packed, as opposed to traditional database systems that store entire records (rows) one after another [55]. It is known that column store can use traditional database query languages like SQL, to load data and perform queries.

Also mentioned in [55], data compression is described as one of the advantages using this database type because compression algorithms seems to perform better on data with low information entropy. Although, write operations and tuple construction are outlined as the most-often cited disadvantages. Column database models are generally aimed at higher throughput, and may provide stronger concurrency guarantees, at the cost of slightly more complexity than the document stores [35].

Study performed in [51] also refer this database model as suitable for big data, data mining and analytic applications, offering high scalability in data storage. Amazon describe it as optimised for fast retrieval of data columns [56], typically in analytical applications, data warehousing processing, adding that it plays an important factor in analytic query performance because it drastically reduces the overall disk input and output requirements, as well as the amount of data needed to be loaded from disk. Most popular implementations for column store databases include Cassandra [57] and HBase [58].

3.1.1.5 Key Value Stores

This is one of the most simple, but efficient and powerful database model. Suitable for structured data and no complex queries, only stores key-value pairs. Data retrieval could be achieved using a key to get the actual data referred as value. This model is known for emphasise high scalability over consistency, where high concurrency, fast look-ups and mass storage [51] are recognised as the main features. During this research project it was not possible to find a default query language for this database model, as it seems that most DBMS only provide simple operations like get, put, and delete.

Key-value stores are generally good solutions, for simple applications with only one kind of object, or where it is only needed to look up objects based on one attribute [35]. It is not an adequate and efficient solution for complex applications, when data is relational or when it is required to query for a given key, or search for some specific detail within the value. In [51] is also mentioned that the lack of rigid data schema might make the data customs views task much more difficult.

Storing user profiles, sessions information or even Internet Protocol (IP) forwarding tables are also often purposes for this database model. Most popular implementations for key-value databases include Redis [59] and Amazon DynamoDB [60].

3.1.2 Database Requirements

This section specifies a set of requirements considered relevant to evaluate different databases models previously described. The goal is to detail each requirement, and explain its importance in choosing the most suitable database model to be used in this study.

Scalability - this concept could be understood as the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and to be susceptible to enlargement. A system that does not scale well adds labour costs or harms the quality of service [61]. The database to be used should be able to process a few hundred items, maintaining the same performance over time. This can be a simple task for the vast majority of existing database types, given that most were built to deal effectively with thousands or millions of records.

Data Schema - evaluates the database structure that defines the objects and their constraints within the database. It is intended that the database should allow irregular data creation to be stored and manipulated. It should be possible to force and manage the assignment of values to certain object properties through constraints definition. It must be flexible enough to allow the creation of new object types or attributes, without compromising the existing schema stability.

Data Visualisation - it is considered the ability to visualize data and communicate information clearly and efficiently, using tools like statistical graphics, plots or information graphics. It is intended that the database allows easy and simple data visualisation like item attributes and relationships between the different stored items.

Query Language - it refers to any computer programming language used to request and retrieve data from a database. A query could be defined as a user structured command sent to the database system, expecting the respective output. Query language semantics will depend on database model and implementation. NoSQL implementations don't use SQL like relational databases do. Also, NoSQL database types does not have a standard query language. The database to be used should have a simple and easy to learn query language, to allow a fast user learning curve on data request and modification.

Import and Export Data - the chosen database type should ease import and export of data from the database to most common data file formats, such as

Comma Separated Values (CSV) and JSON, in order to allow data and system integration.

Backup and Snapshot - enables the creation of a duplicate instance of the database at any point in time, being a way of protect a given schema state or stored data. A backup also allows a database restore in case of the primary instance crashes, is corrupted or being lost. It is intended that the database to be used allows the backups creation, as well as comparison between two snapshots. This feature would allow to highlight the differences between two different snapshots: current and intended state of the objects, their attributes and relationships.

3.1.3 Database Evaluation

A quantitative evaluation was performed for the different database models for which research has been carried out.

A scale between 1 and 3 has been defined for each of the requirements mentioned above. A value of 1 is assigned when the database model does not meet the requirement as required or when its performance for the requirement is not as desired. A value of 2 means that the database model partially meets the requirement, but some adjustments are needed to achieve the desired goal. Finally, a value of 3 means that the database model fully meets the requirement, and is up to the required performance for the database intended to be used, for the system to be developed.

Table 3.1 presents the quantitive evaluation for the requirements' compliance of each database models and some considerations regarding the ratings assigned. The classification values assigned to each database model were defined according to the research done on the features of the main commercial solutions available for each model.

Within the scope of the scalability requirement all the models covered were given the maximum rating of 3, because it was considered that all of them would

Model / Requirements	Relational	Graph	Document	Column	Key-Value
Scalability	3	3	3	3	3
Data Schema	2	3	3	2	1
Data Visualization	2	3	2	1	1
Query Language	3	2	2	3	1
Import and Export	3	3	3	3	3
Backup and Snapshot	2	2	2	2	2

TABLE 3.1: Quantitive evaluation of the database model requirements

be able to accommodate a few hundred records while keeping a constant and acceptable performance.

For the data schema requirement, graph, document and column models were given the maximum score, for being able to handle semi-structured or unstructured data, enabling records of the same type to be created with different property sets. For the relational model was assigned a value of 2, since it is a model that stores data in a tabular way, imposing a set of properties for all records in the same table, with less flexible structure and less easy to update the schema if changes are needed. The minimum classification of 1 was given to the key-value model because it seems more suitable for structured data, applications where there is only one object type, where queries are made based on an attribute or where there are no relationships between the data. All these factors are against the intended solution to be developed.

The key factor for the classification given for the data visualisation requirement was the possibility of easily analyse the relationships between the database records of each model. Making use of the graphical and analytical capabilities of graphs, the graph model is undoubtedly the best choice among all, since with the nodes and edges it is possible to easily identify which records are linked and through which relationships. The document model was classified with 2, because it is possible to store the information about the relationship with other documents within the document, but the existence of many relationships and the absence of a graphic component might make the analysis more complex. The relational model was also classified with a value of 2, by the fact that through the table query it is possible to analyse the relations between different records, however it can become confusing when there are many relations from a single object or when the database contains more than a few dozen of objects. The column and key-value models received a classification of 1 because they do not contain default characteristics to enable a relationships analysis between records and would need external solutions for data visualisation purposes.

The assessment performed for the query language follows the guidelines defined for the requirement, where the model is valued if it has a simple and quick to learn language, facilitating the data management. With this in mind, database models such as relational or column models, that use SQL, have an advantage because they use a simple and well-know language available since a long time ago. The document and graph models obtained a classification of 2 because to interact with these database types it is necessary to dedicate some time to learn a new query language, such as Cypher or JavaScript. The key-value model received a minimum classification of 1 because most of this model implementations have no SQL-style query language, using only simple key operations to reference data [62].

Within the scope of Import and Export Data requirement, the research focused more on verifying whether the commercial implementations for each database model allowed importing and exporting data to the desired formats. This approach was adopted because this requirement is not strictly related to the database model, but rather how each software company develops the commercial implementation of a database system. Consequently, research revealed that, at least one implementation for each database model, meets the requirement of importing and exporting data in CSV and JSON formats, thus so all models were given a score of 3.

Finally, the Backup and Snapshot requirement is also heavily dependent on each commercial implementation of database system. The research showed that the most popular implementations of each database model provide backup features, but the functionality of comparing different backups and highlighting the differences is something usually not done by default. There would be a need to develop additional tools to make this requirement possible. For this reason, a score of 2 was assigned to all models evaluated.

Concluding, the choice for most suitable model seems to lie within the graph model. In addition to partially or fully respecting all requirements, it is the database model where data schema and visualisation are a key component. Graph model allows data visualisation through graphs, easing the relationship analysis between the different nodes. This is a fundamental characteristic for the observations to be made for the industrial networks monitored within this study's scope. Beyond that, graph model data schema allows the creation of new node types, or new nodes of already existing types with dissimilar number of attributes, without need for data schema update or compromising the existing schema stability.

3.1.4 Neo4j Graph Database

There are some graph database implementation projects, and it was considered to investigate the related literature to evaluate which tool has the best overall performance in the graph database market.

S. Jouili et al [63] present a distributed graph database comparison framework to compare four important players in the graph databases market: Neo4j [64], OrientDB [50], Titan [65] and DEX [66]. The main purpose of Graph Database Benchmark (GDB) is to objectively compare graph databases using usual graph operations, like exploring a node neighborhood, finding the shortest path between two nodes, or simply getting all vertices that share a specific property are frequent when working with graph databases. Based on their measure, the database that obtained the best results with traversal workloads was definitely Neo4j: it outperforms all the other candidates, regardless the workload or the parameters used. Concerning read-only intensive workloads, Neo4j, DEX, Titan and Orient achieved similar performances [63]. D. Dominguez-Sal et al [67] also present a performance evaluation of four of the most scalable native graph database projects: Neo4j, Jena [68], HypergraphDB [69] and DEX. They tested the performance of each database for different typical graph operations and graph sizes, concluding that only DEX and Neo4j were able to load the largest benchmark sizes given the hardware setup. DEX was the fastest database loading the graph data and performing the operations that scan all the edges of the graph, with Neo4j as the second best. Neo4j also obtained a good throughput for some operations, despite scalability problems for some operations on large data volumes.

With this being said, Neo4j seems to be a suitable graph database choice and ended up being chosen for the context of this study. It is freely available to download, it also has a large user support community with lots of examples and documentation helping developers find solutions to their problems, a very good graph visualisation, an easy-to-learn query language (Cypher) and it is fairly easy to import and export data from the database.

3.2 Security Software Tools

This section aims to show and categorize some of the current available tool solutions that support asset management within an industrial environment. Some of these tools are proprietary and others open-source and free to use, but all of them have been helping organisations, offering solutions to manage industrial network assets.

The goal is to describe the purpose for which each of them is used, highlighting the advantages of its use in supporting the security management of industrial assets. Table 3.2 show the surveyed tools, which include Siemens business tools shown in section Siemens ITAM Status. The tools were classified according to its applicability to the following functionalities: Asset Management (central repository for asset and workflow related data), Security Scanning (gathering asset data through scans within a given network) and Engineering Configuration (system design, handling of engineering and configuration data).

Claroty - a set of cybersecurity software products developed to ensure the safety and reliability of industrial control networks, enabling cyber threat detection (discovering and eliminating vulnerabilities, misconfigurations and insecure connections), secure remote access management (enforcing access policies and recording sessions) and risk assessment for industrial networks. Claroty and Siemens provide to industrial operators and cybersecurity teams a real-time solution [70] for Operational Technology (OT) network visibility, with emphasis on fast deployment, high scalability, enhanced performance and reliability, thus extending the life-cycle existing hardware and network infrastructure.

Open Computer and Software (OCS) Inventory - an open source software for asset management and deployment [71], enabling creation of IT asset inventories. The OCS Inventory Agent attempt to collect asset information about the hardware and software present on a given network, doing an IP discover scan. The information gathered is later displayed on a web interface available to allow users analysis.

Asea Brown Boveri (ABB) Cyber Security Asset Inventory - a tool developed by ABB [72], that provide a solution to automatically identify and monitor IT assets on a network (gathering information related to ports, services and software), keeping an up-to-date asset database on controlled networks, thus helping in asset life-cycle management and decision-making issues. Asset scanning can be done through two different ways: network data collection, where the tool continuously monitors traffic from the networks and updates the database with new and changed assets; or through device data collection, where users can manually trigger scans to specific assets, in order to gather more detailed information. The database can keep device details such as operating system versions, open ports, device model, serial number, known vulnerabilities, among others. Data collected using this tool can also be printed or exported to different formats (PDF or CSV) supporting compliance activities such as regulatory software audit and ongoing risk assessments. Thus, this tool proved to significantly reduce costs and shorten incident response time, helping in troubleshoot network issues and overall security of controlled industrial network assets.

Table 3.2 shows how the above mentioned software tools and Siemens tools described on section Siemens ITAM Status fall into the following functionalities: Asset Management, Network Scanning and Engineering Configuration.

Category / Tool	Asset Management	Security Scanning	Engineering Configuration
COMOS	x		x
SIMCO	x	x	x
SiESTA		x	
Claroty		x	
OCS Inventory	x	x	
ABB Asset Inventory	x	x	

Table 3.2: Security software tools classification

This research provided an insight into different software tools categories and roles, used to address security assets challenges and involved in industrial control systems presented, like power grid, nuclear power plants and others. Software tools such as COMOS and SiESTA, are part of the systems scope used within the study's project, and consequently its roles will be detailed in the next chapter.

Chapter 4

Proposed Approach and Solution

This chapter aims to describe the prototype details, built to address the initially stated problem. First of all, the proposed approach will be presented, where a high level system architecture is displayed, the role of each Siemens software tool within the system architecture is described, as well as the assets and relationships types present on the asset data model defined.

The second part analysis the database support tool prototype implementation details regarding this study scope. The detailed project data flow shows a more complete view of each prototype sub-task, where it is explained how each file is generated or used by different systems, during the workflow execution. The project's folder structure is then described and the Python [73] package dependencies are explained.

Finally, the different system user interfaces are presented and the features developed for each interface are explained.

4.1 Proposed Approach

This section explains the proposed approach adopted to develop an industrial security asset management solution, regarding the conducted background research and literature review, where a database model was chosen based on the defined requirements, and the software tools were analysed and categorised.

4.1.1 Proposed Architecture

For this study scope, two Siemens software will be used as information source, and tool to scan device configurations present on industrial networks. Figure 4.1 represents the high level system architecture and the phases (arrows) defined for the project's scope.

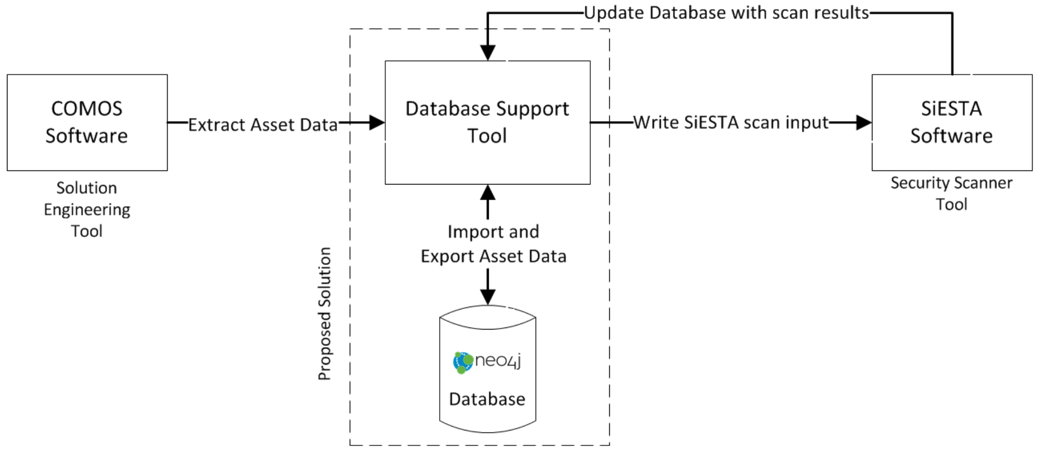


FIGURE 4.1: High Level Project Workflow

The proposed solution (dashed square) goal is to achieve system integration between COMOS and SiESTA software. To this end, it is intended to use the asset data blueprint present on COMOS software engineering tool to enable the scan tests execution on the security scanner SiESTA software, thus enabling the database to be updated later with the tests results. The database support tool will have the role of extracting the asset data from COMOS blueprint and perform its import into the Neo4J graph database, like shown in Figure 4.1.

Once the data has been imported, the graph database allows the visualisation and analysis of nodes and relationships, through the use of its graphical capabilities. The database support tool will give the possibility of exporting the data from Neo4j graph database, outlining the necessary input for SiESTA software to successfully run the network security scanning tests, represented by "Write SiESTA scan input" in Figure 4.1.

After SiESTA software completes the set of security scanning tests regarding the network device configurations, the database support tool will have the role of comparing results with the blueprint originally specified in COMOS software and currently present within the graph database, represented by "Update Database with scan results" in Figure 4.1. According to the scan test results, the database support tool will give to the user the possibility to update or maintain current device configurations within the graph database, thus keeping the consistency of data present on the industrial network.After the database is updated, the user can again check the nodes and relationships, as well as export the data for future scanning tests, subsequently performing an iterative cycle, until the desired results for the industrial network devices configuration are achieved.

4.1.2 Asset Data Model

The data extracted from the systems used in this project, regarding devices within the industrial networks, can then be represented by an object set within a data model. A data model should be an abstract model that shows how the different objects and its attributes are organised, how they represent the entities in real world, and through which relationships they connect to one another.

To design the data model, research was performed, in order to verify the existence of specifications for the asset identification on an industrial or organisational context. Consequently, the data model follows the guidelines for asset identification specified in [34], mentioned on the background section of this study.

Some of the asset types specified in [34] were reused on the data model defined (e.g. Software, Network, Service and ComputingDevice). The data model was then extended with more asset types to meet the requirements of previously defined use cases: host discovery, port scanning and vulnerability management. The ComputingDevice, Network, Software, TestCase and SecurityZone asset types were added to meet the host discovery use case requirements, allowing to have a baseline of the network configuration and devices present. For the port scanning use case, were added to the data model the Interface and Service asset types, which allows to know which services are available on the distinct interfaces and which ports they use to connect. The Vulnerability and VulnerabilityScanRule were added as part of the use case of vulnerability management, listing the vulnerabilities of each device and allowing to identify which of them need to be fixed and updated.

Table 4.1 shows the asset types defined for the data model and a brief description of what they represent.

Asset Type	**Description**
ComputingDevice	Electronic devices such as a PC, embedded or a network device, which process inputs and returns an output.
Network	Web of connections between computing devices to exchange data.
Interface	A software point of interconnection between a computer device and a network.
Service	An application running in or provided by some device or via certain interface.
Software	Computer program to perform some task, which may be an internal or 3rd party software.
TestCase	Test types currently supported by SiESTA or other known sources to run on computing devices.
SecurityZone	Logical way to group networks based on a similar security requirement.
VulnerabilityScanRule	Policy for which criteria should be applied between a target host and vulnerability.
Vulnerability	A known security weakness present in a computing system or software.
Tag	Some note with specific information about an asset.

Table 4.1: Data model asset types

Part of the relationships set specified on the publication, such as "installedOnDevice" and "connectedToNetwork" were reused with some changes. Other ones were defined according to the use case requirements for the prototype. Part of the attributes defined for some of the asset types and relationships specified in the specification were also reused.

Table 4.2 contains the relationships defined within the data model, showing the asset types involved in each relationship, as well as a short description of how the relationship should be understood.

Relationship	**From**	**To**	**Description**
connectedBy	ComputingDevice	Interface	A computing device is connected by an interface.
connectedTo	Interface	Network	An interface is connected to a network.
groupTo	Network	SecurityZone	A network is part of a group of a security zone.
installedOn	Software	ComputingDevice	A software is installed on computing device.
provides	Interface	Service	An interface provides a service.
run	ComputingDevice	TestCase	A computing device runs a test case.
taggedWith	All Asset types	Tag	An asset type is tagged with a tag.
hasVulnerability	Vulnerability	ComputingDevice, Network, Interface	A vulnerability scan is assigned to a computing device, network or interface.
communicateWith	Interface	Interface	An interface communicates with another interface.

TABLE 4.2: Data model relationships

4.2 Prototype Development

This section presents the database support tool prototype development in a more in-depth view. Firstly, Python package dependencies are reported, and then project's folder structure and input files role are explained.

Then the section on Dataflow shows a more detailed view of each prototype sub-component, describing how it works using sequence of steps, and explaining how each data file is generated or used by different systems considered within the projects' scope.

Finally, the last part of this section displays the different user interfaces developed for database support tool prototype, explaining the features present on each of them, how to use them, and how they relate to the workflow defined for the project.

4.2.1 Tool Dependencies

For software development, Python language was used for being versatile, for containing extensive support libraries that can be used to enhanced its features, and for existing support through plugin with the DBMS chosen (Neo4j).

The libraries used include OpenPyXL [74] and XLWings [75] to read and write data Excel files, JSON python module [76] to read and write JSON files, Py2Neo toolkit [77] was used to import, export and retrieve data from the Neo4j graph database and PyQT5 [78] was used to design the tool's graphical user interface. Most of these libraries were installed using the "pip install" command, which is available through Package Installer for Python [79].

4.2.2 Folder Structure

The main project directory is based on three sub-folders: the Excel folder is where the tool input files are located. The JSON folder is where the Test Cases and Vulnerabilities Scan Rules files are stored, containing data about these nodes and relationship types. Lastly, Scripts folder is where python scripts, developed to perform the different tool functions, can be found.

4.2.2.1 Excel Folder

Communication Matrix File - a manually written document and main data source, holding the information about how the system's network should be arranged. It shows how devices are expected to be connected to each other, in which networks they should be contained and which details should be configured for their services. It will be useful for relationship creation between the assets within the graph database.

COMOS File - a plant management software used to manage life-cycle processes, where its exported file is meant to be used by the tool to extract detailed information from the devices contained on the monitored networks. This file contains details regarding device type, manufacturer, operating system version installed, as well as IP and Media Access Control (MAC) address values, among other device properties.

Software File - contains data about the software installed (internal and third party software) on the hosts presented on the monitored networks, such as software name, release version, and installation dates. This information will be used to create the Software nodes and their relationships (installedOn) with hosts within the database.

SiESTA Tests Template File - SiESTA is a Siemens software used to do network scans and check system vulnerabilities. This Excel template is used to write the expected configurations and security policies to be applied to the devices scanned during the tests, known as scan rules. There are two spreadsheet types: the specification and the evaluation one. Each specification spreadsheet type should contain the validation input to be performed on hosts, ports, and vulnerabilities. After the template is filled, the goal is to run the test scan and check which settings are not as expected, which devices should or should not be on the network. The scan results will be then written by SiESTA on the evaluation spreadsheets, and will be used to decide which changes should be made to the database, according to some predefined business rules.

4.2.2.2 JSON Folder

Test Cases File - a manually written file that contains information about the different Test Cases that can be performed on the device types present on monitored networks. This data will be used to create the nodes and relationships between the Test Cases and other nodes within the database.

Vulnerabilities Scan Rules File - a manually written file that contains information regarding the different Vulnerability Scan Rules to be verified on the device types present on monitored networks. This data will be used to create the Vulnerability Scan Rules nodes within the database.

4.2.2.3 Scripts Folder

excel_reader.py - a script responsible for reading and extracting data from the multiple input files (Communication Matrix, COMOS file and Software file, etc) using the OpenPyXL Python package [74], as well as for creating the node structures and relationships based on the information contained in these files. The purpose is to store the node structures and relationships on a JSON format file. This script may receive different input combinations that will impact the data present on the generated JSON file. It is also responsible to read the evaluation spreadsheets from Excel scan results file, after SiESTA scan tests execution.

db_data_importer.py - a script responsible for establishing the connection to the graph database using the Py2Neo toolkit, importing data from a JSON file and also for exporting current database data to a JSON file, as a backup. Additionally, it is responsible for performing CRUD operations commands on the database, received throughtout the Graphical User Interface (GUI). It is also responsible for handling the update database logic, creating new nodes and relationships, after SiESTA scan tests execution.

excel_writer.py - a script responsible for filling the spreadsheets from the SiESTA Template file, using the XLWings Python package [75], based on the data present in a JSON file used as input. The system is prepared to handle "S_hosts", "S_ports" and "S_vulnerabilities" spreadsheets, when enough data is provided.

gui.py - a script responsible for designing and displaying the GUI, using PyQT5 package. It contains the logic defined for the interactive elements for each tool feature that can be found in GUI's functionalities. It calls the other scripts methods described above, delegating tasks responsibility. The user interface is divided in 7 different functionalities: "Connection to Database", "Extract and Import", "Import and Export", "SiESTA Template Writer", "Manual Update", "Update Database" and "About".

config.py - a script that stores the tool settings and parameters as well as the so-called "magic numbers" to enhance tool readability and maintenance for near future. Values or constants related to the graphical interface or other scripts can be edited quickly and changes will be propagated.

launch.bat - this batch file was created with the purpose of simplifying the tool execution without the user being required to open a command line terminal. By double-clicking, it will invoke the tool's GUI.

4.2.3 Dataflow

This section on Dataflow shows a more detailed view of each functionality, describing how it works explained in a sequence of steps.

The Figures 4.2, 4.3, 4.4, 4.5, 4.6 describe the data flow for each tool functionality, explaining how each data file is generated or used by different systems considered within the projects' scope.

The diagrams and their descriptions have been simplified: algorithms or logic are not described in detail, since they are confidential information of Siemens and to facilitate the understanding of the main flow of each process.

Extract Asset Data

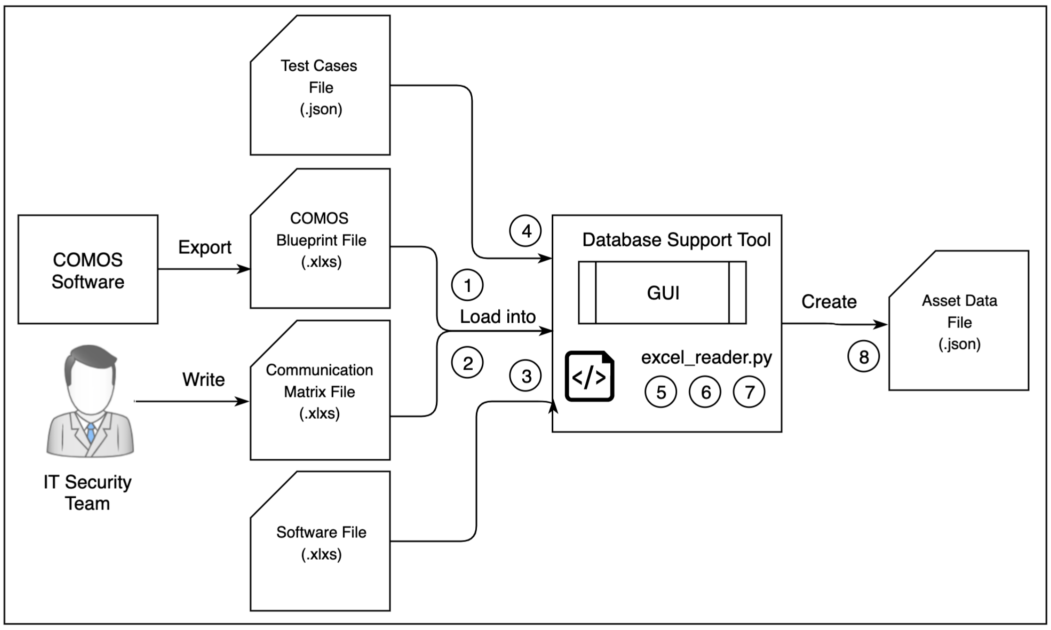


FIGURE 4.2: The process of extracting the asset data from COMOS source system.

1. Read the Communication Matrix excel file and its different spreadsheet contents and store data in memory data structures.

2. Read the COMOS blueprint exported file and store data from each asset in memory data structures.

3. Read the Software file and store software information on data structures.
4. Read the Test Cases JSON file and store data on data structures.
5. Cleans and merge all the data from the same host data in the COMOS file based on predefined logic.
6. Create Computer Devices, Interface, Network, Service and Software nodes.
7. Create different relationships JSON structures such as "connectedTo", "connectedBy", "provides" and "installedOn", relating to information present on the Communication Matrix and other input files.
8. Writes the nodes and relationships created for a JSON file.

Import Asset Data

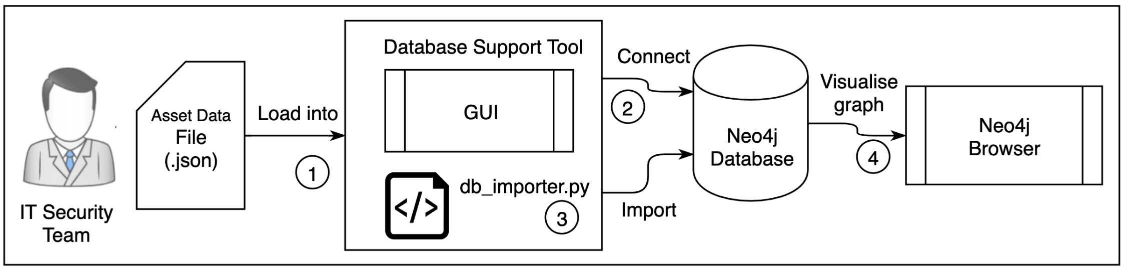


FIGURE 4.3: Data flow of importing asset data to the Neo4j graph database.

1. The user invokes the action of importing asset data using a JSON file on the tool's GUI.
2. The tool establishes a connection to the graph database given user authentication provided.
3. To import the asset data the tool reads the input JSON file and then run the queries to create the nodes and relationships regarding the file provided.
4. After the import is completed, the user can visualise the generated graph using the Neo4J Browser.

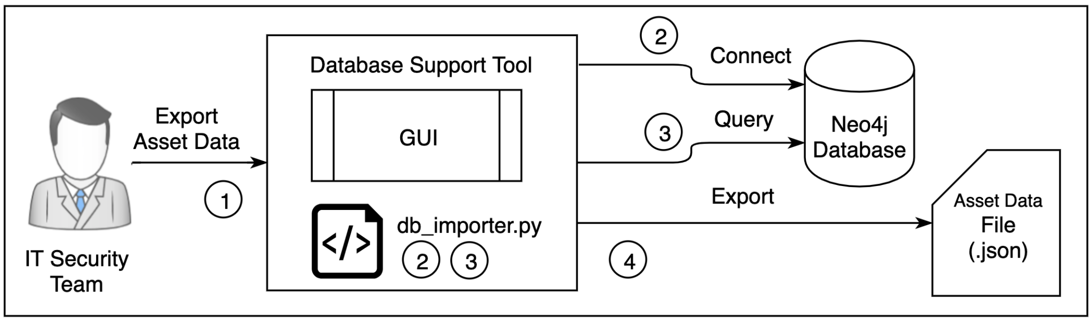


FIGURE 4.4: Data flow of exporting the asset data from the Neo4j graph database.

Export Asset Data

1. The user invokes the action of export asset data to a JSON file using the tool's GUI.
2. The tool establishes a connection to the graph database given user authentication provided.
3. Run a query to select all the nodes and relationships present in the database.
4. Save the data selected exporting a JSON file with the name previously chosen by the user.

Write SiESTA scan input

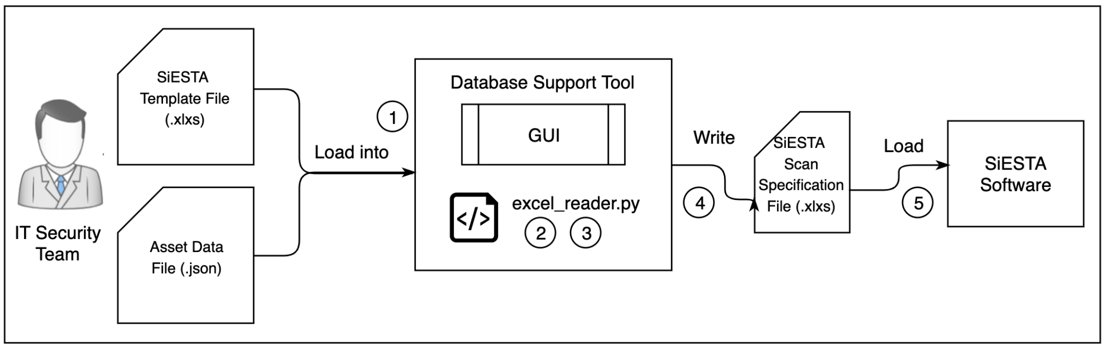


FIGURE 4.5: Data flow of writing SiESTA scan input to detect vulnerabilities on network's hardware and devices.

1. The user invokes the action of writing the SiESTA scan input using the tool's GUI, providing an empty SiESTA template file and a JSON file with asset data.
2. Reading the JSON input file, the tool store the content on data structures.
3. Using the relationships of the data provided, the tool applies the business requirements previously defined in discussions with project members, selecting the desired hosts, ports and vulnerabilities data needed to define the scan policies.
4. The selected data is then written into the corresponding spreadsheet (hosts, ports and vulnerabilities) on the SiESTA template file, and saved with the name previously chosen by the user.
5. The file is then ready to be loaded to the SiESTA software.

Update Database with scan results

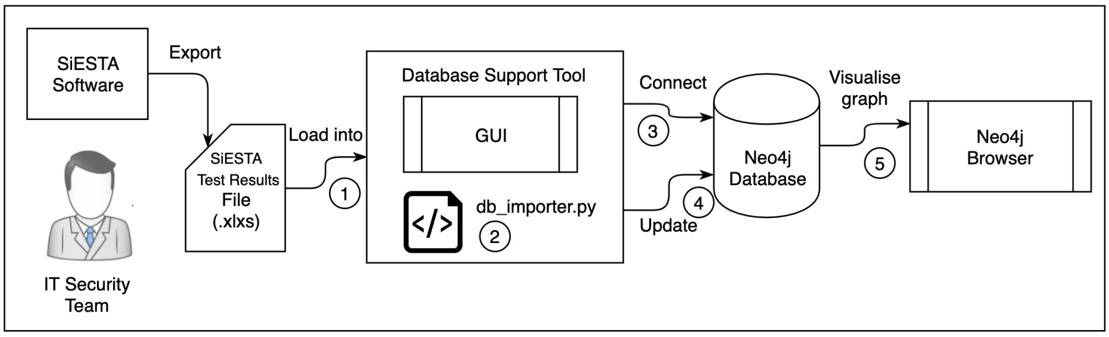


FIGURE 4.6: The process of update the graph database with the SiESTA scan results.

1. The user load into the tool the Excel file with results from tests performed by SiESTA software.
2. Using the excel_reader.py script, the tool parses the data present on "E_Hosts", "E_Ports" and "E_Vulnerabilities" spreadsheets and, based on business requirements, puts the data into different data structures.

3. The tool establishes a connection to the graph database given user authentication provided.

4. Then the tool uses the previously built data structures to update the nodes and relationships present within the database. On "E_Hosts" it is evaluated if new Interface nodes must be created, in such cases new "connectedTo" relationships can be generated between Interface and a Network nodes, as well as "connectedBy" relationships between existing ComputingDevice and Interface nodes. "E_Ports" spreadsheet data can lead to the creation of Service nodes, and of new "provides" relationships, between created and existing Interface and Services nodes.

   After extracting the results information within "E_Vulnerabilities", new Vulnerability nodes and "hasVulnerability" relationships will be created, thus identifying on the database to which ComputingDevices nodes belong these new vulnerabilities.

5. All the nodes and relationships created can then be visualised using the Neo4J Browser, through the analysis of the updated graph.

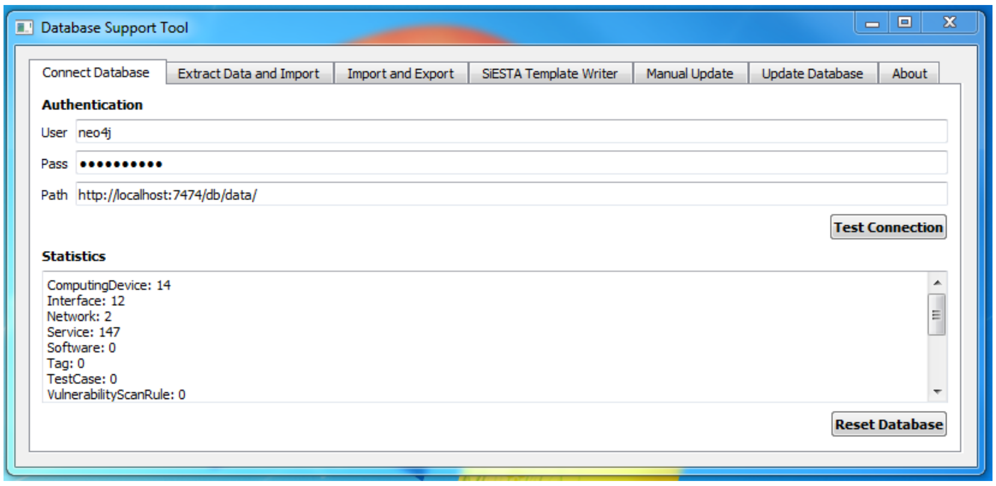
4.2.4 User Interface

This section describes the user interfaces, how users can interact with the database support tool features, while explaining the design decisions made during the development process.

The user interface is structured into seven functionalities like mentioned before, each one associated with a different feature or group of features. This separation allows each functionality to be associated with a different phase of the overall process, assigning the features to its own space, thus increasing the tool usability. All functionalities interface have been designed using a form design, where the user fills in the fields starting at the top and ending at the form's bottom, where the operation buttons are placed.

4.2.4.1 Connection to Database

This functionality allows to enter the login parameters for the database created in Neo4j. After that, the user will be able to test the connection success, as well as delete all the database content. Figure 4.7 shows a screenshot of the user interface for this functionality.



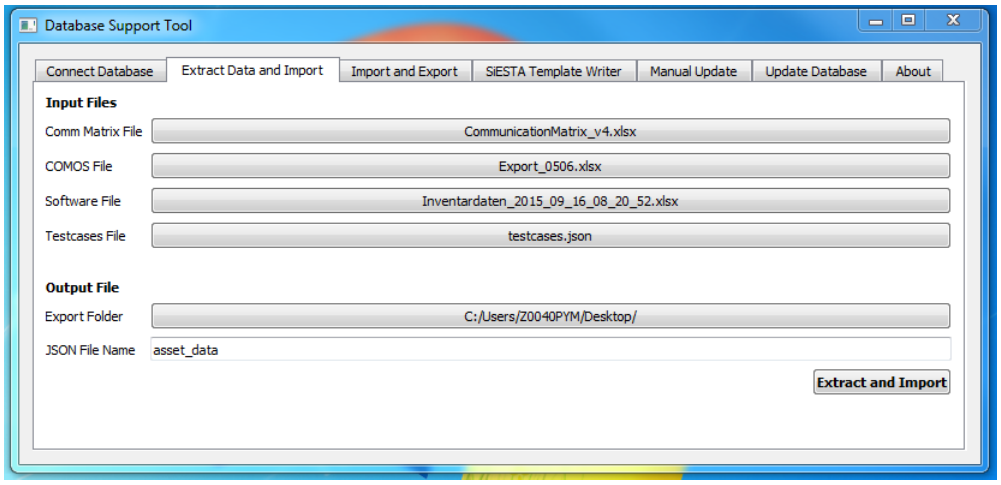
User neo4j
Pass ••••••••••
Path http://localhost:7474/db/data/

```
ComputingDevice: 14
Interface: 12
Network: 2
Service: 147
Software: 0
Tag: 0
TestCase: 0
VulnerabilityScanRule: 0
```

FIGURE 4.7: User interface for "Connect to Database" functionality.

4.2.4.2 Extract and Import

This functionality allows to generate a JSON file with node and relationships data based on the different input files selected. The generated file can have a name defined by the user and be exported to a chosen folder. Afterwards, these nodes and relationships on the generated JSON file are imported into the database. Figure 4.8 shows a screenshot of the user interface for this functionality, where it is possible to observe two areas, Input files and Output files.



Input Files

Comm Matrix File	CommunicationMatrix_v4.xlsx
COMOS File	Export_0506.xlsx
Software File	Inventardaten_2015_09_16_08_20_52.xlsx
Testcases File	testcases.json

Output File

Export Folder	C:/Users/Z0040PYM/Desktop/
JSON File Name	asset_data

FIGURE 4.8: User interface for "Extract Data and Import" functionality.

The following combinations of input files are valid:

1. Communication Matrix and COMOS files

   For these input files will be generated the ComputingDevice, Interface, Network, Service nodes and the "connectedTo", "connectedBy" and "provides" relationships.

2. Communication Matrix, COMOS and Software files

   For these input files will be generated all nodes and relationships mentioned above, plus the Software nodes and "installedOn" relationships.

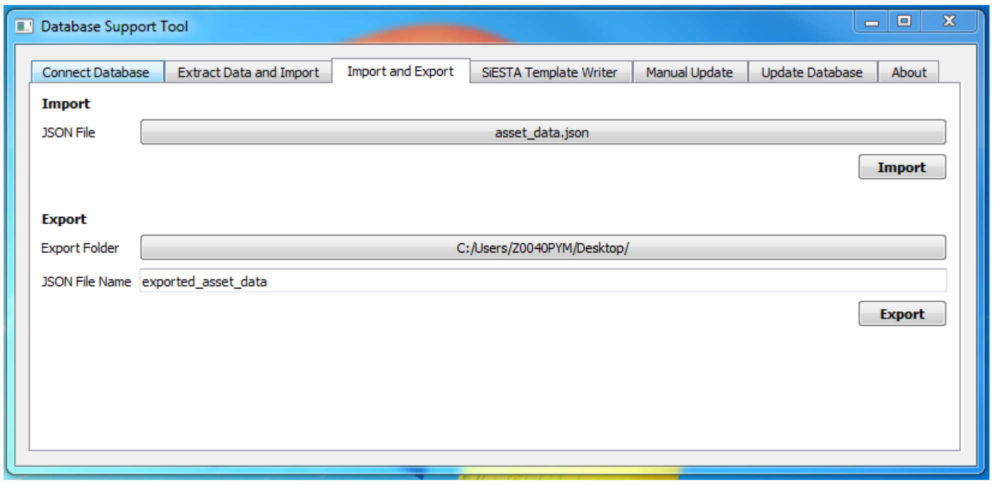
3. Communication Matrix, COMOS, Software and Test Case files

   For these input files will be generated all nodes and relationships mentioned above, plus the Test Cases nodes.

All other combinations are not working on the database support tool current version.

4.2.4.3 Import and Export

This tab allows to import data into the database from a previously generated JSON file with a valid format using the "Import" operation. This could be a file generated in the past by the previous functionality or a JSON file with the Vulnerability Scan Rules data. Figure 4.9 shows a screenshot of the user interface for this functionality, where it is possible to observe two areas, Import and Export.



JSON File: asset_data.json

Export Folder: C:/Users/Z0040PYM/Desktop/

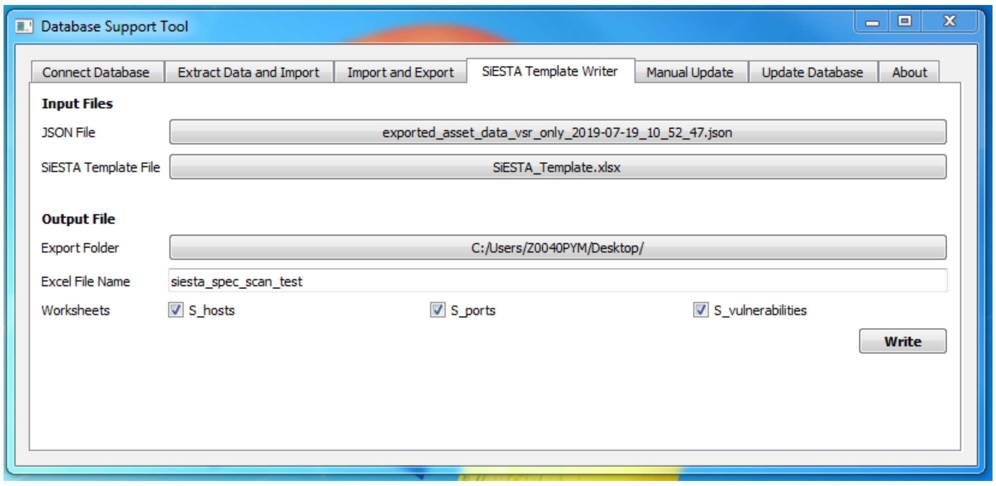
JSON File Name: exported_asset_data

FIGURE 4.9: User interface for "Import and Export" functionality.

It also allows data related to nodes and relationships, currently present in the database, to be exported through the "Export" process, creating a database backup in JSON format. The user can define a name for the exported file and store it in a chosen folder. This feature allows a given database status, at any point in time, to be restored and analyzed again.

4.2.4.4 SiESTA Template Writer

This functionality allows to write data about certain nodes and relationships to SiESTA Template Excel file, so that the generated Excel file can later be used as an input file for SiESTA's scanning tests. Figure 4.10 shows a screenshot of the user interface for this functionality, where it is possible to observe two areas, areas, Input Files and Output File.



Input Files

JSON File: exported_asset_data_vsr_only_2019-07-19_10_52_47.json

SiESTA Template File: SiESTA_Template.xlsx

Output File

Export Folder: C:/Users/Z0040PYM/Desktop/

Excel File Name: siesta_spec_scan_test

Worksheets: [✓ S_hosts] [✓ S_ports] [✓ S_vulnerabilities]

FIGURE 4.10: User interface for "SiESTA Template Writer" functionality.

On the input files section, the user can select the valid file JSON format (e.g. a database backup exported on the previous tab) and a clean SiESTA Template Excel file with a valid format.

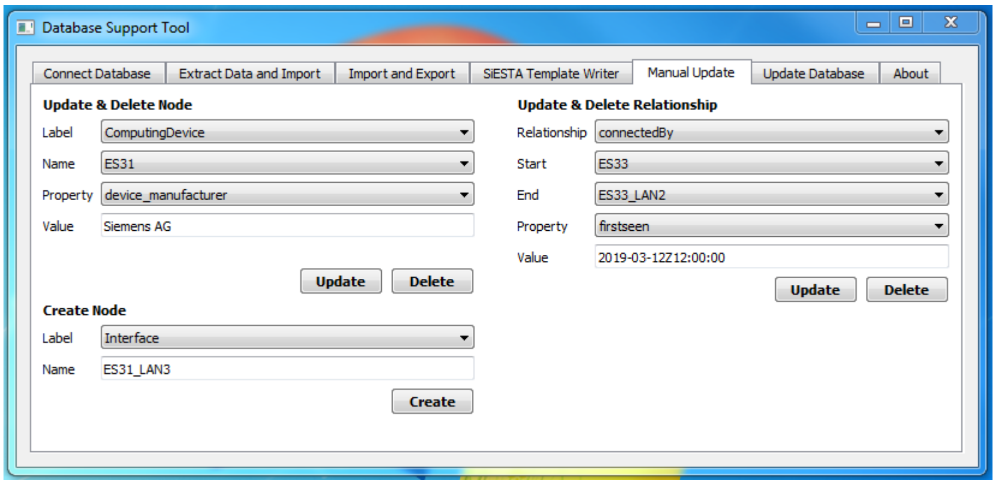
In the output file section, the generated file can be exported to a chosen folder and have a name defined by the user. If there is already a file with the same name in the chosen folder, the tool will ask the user if he wants to replace it. All fields must be filled in order to write the Excel file.

The "S_hosts", "S_ports" and "S_vulnerabilities" checkbox allow the user to choose which SiESTA Template spreadsheets will be written regarding the nodes and relationships information. The user must choose at least one checkbox for the output file to be generated.

4.2.4.5 Manual Update

This functionality allows the user to create, update and delete nodes, relationships and its properties present on the connected database, where only one transaction can be made at a time. It was decided to implement this functionality to overcome the barrier of users having to learn Cypher query language. Figure

4.11 shows a screenshot of the user interface for this functionality, where it is possible to observe three areas, Update & Delete Node, Create Node and Update & Delete Relationship.



Update & Delete Node

Label	ComputingDevice
Name	ES31
Property	device_manufacturer
Value	Siemens AG

Create Node

Label	Interface
Name	ES31_LAN3

Update & Delete Relationship

Relationship	connectedBy
Start	ES33
End	ES33_LAN2
Property	firstseen
Value	2019-03-12Z12:00:00

FIGURE 4.11: User interface for "Manual Update" functionality.

Update and Delete Node - This functionality's section allows to update node property values and to remove them. The Label combo box contains only the node types currently present in the database. The Name combo box will be filled with the existing nodes after selecting a Label value. The Property combo box will be populated with the properties defined for the selected node type within the Label combo box. Currently, it is not possible to create new properties for the nodes.

If the property to be updated is boolean type, it can only be updated with True or False values, lowercase or uppercase. If the property to be updated is a string enumerate, it can only be updated to a value defined within the node type property enumerate. To delete a node, at least the Label and Name combo box must be filled in.

Create Node - This functionality's section allows to create new nodes in the database. The Label combo box contains all node types defined in the data model. The users can define a new node's name to be created in the Name text box. If the name already exists for the selected node type, it will not be possible to create it.

After the new node has been successfully created, the user can edit its properties in the section "Update and Delete Node".

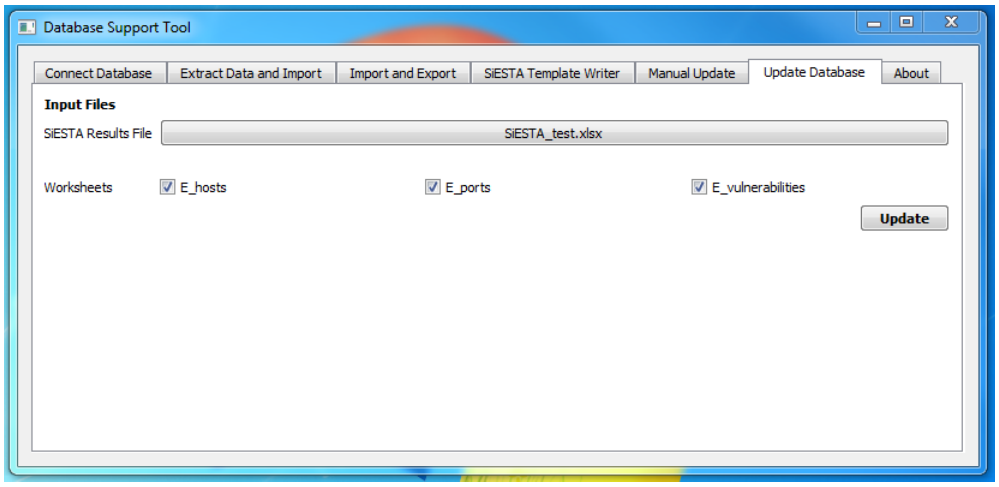
Update and Delete Relationship - This functionality's section allows to update relationship property values and remove them. The Relationship combo box contains all relationship types defined in the data model. The Start and End combo boxes will be filled with the existing nodes after selecting a Relationship value. The combo boxes' content is defined based on the nodes type that are at the edge of each relationship type.

The Property combo box will be populated according to the relationship type chosen. Currently, it is not possible to create new properties for the relationships. If the property to be updated is a date and time property, it must be updated with values that follow the format YYYY-MM-DDZ hh:mm:ss.

To delete a relationship it is necessary to set at least one value for the combo box Relationship, Start and End. A relationship must exist in the database to be deleted, otherwise the tool will inform the user that the operation is not possible.

4.2.4.6 Update Database

This functionality allows to update the database with the scan tests results performed by the SiESTA software. Figure 4.12 shows a screenshot of the user interface for this functionality, where it is possible to observe an area of Input Files.



Input Files

SiESTA Results File: SiESTA_test.xlsx

Worksheets: ☑ E_hosts ☑ E_ports ☑ E_vulnerabilities

FIGURE 4.12: User interface for "Update database" functionality.

The user uploads an Excel file exported from the SiESTA software, defines the evaluation spreadsheets to be used to update the database and clicks the "Update" button. Currently the tool is prepared to deal with "E_Hosts", "E_Ports" and "E_Vulnerabilities" spreadsheets. The user must choose at least one checkbox in order to update the database.

Chapter 5

Evaluation

This section is intended to describe the approach to the evaluation of the data model and the chosen database type, within the scope of the study conducted.

Firstly, some considerations the design-science approach to evaluation of research in Information Systems (IS) performed by Hevner et al [80], showing how the project followed the guidelines proposed by the evaluation method, described by the approach.

During the evaluation, a questionnaire was provided to a group of professionals that took part of the evaluation as participant sampling, as method for collecting feedback and evaluating the data model and the database. Thus is intended to analyse and extract opinions from their written answers, considering professionals' point of view as a part of the descriptive and qualitative evaluation method.

5.1 Evaluation Method

Hevner et al [80] proposed a research evaluation approach that describes a conceptual framework and a method with clear guidelines to evaluate design-science research in IS. According to this research, the design-science paradigm seeks to extend the boundaries of human and organisational capabilities by creating new and innovative artefacts.

Design-science research in IS also seeks to solve problems like unstable requirements and constrains based upon ill-defined environmental contexts or with complex interactions among subcomponents of the problem and its solution. The fundamental principle of design-science research is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artefact [80] .

The approach [80] proposes a set of guidelines for conducting and evaluating good design-science research.

Guideline number one - define that design-science research must produce a viable artefact in the form of a construct, a model, a method, or an instantiation. Artefacts usually are innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished. Artefact instantiation demonstrates feasibility both of the design process and of the designed product.

Furthermore, artefact constructed in design-science research are rarely full-grown information systems that are used in practice. The software prototype developed to support the asset management database is thus considered an artefact, as a result of a problem where a solution was sought through the design science research method.

Guideline number two - this guideline outline the problem relevance: the objective of design-science research is to develop technology-based solutions to important and relevant business problems [80]. Design science approaches this goal through the construction of innovative artefacts aimed at changing the phenomena that occur. The solution was developed to describe and organize the industrial networks asset data set throughout its life-cycle, solving the problem of not having a clear system overview as well as no well-defined data structure to organize all data about IT assets from different industrial sources. Therefore, it is possible to deliver a tool, that helps industrial network security teams, to manage their assets and what their configurations are.

Guideline number three - The utility, efficacy and quality of a design artefact is addressed on this guideline, saying these qualities must be rigorously demonstrated via well-executed evaluation methods [80]. The business environment establishes the requirements upon which the evaluation of the artefact is based. Thus, evaluation includes the integration of the artefact within the technical infrastructure of the business environment. In the solution developed, the requirements were defined through meetings between part of the project's stakeholders, which based on the use cases, defined what entities should be present in the data model, as well as all the business rules involved.

As a design evaluation method, the solution developed could be described as a descriptive evaluation given that is applicable to especially innovative artefacts for which other forms of evaluation may not be feasible [80].

Given the business environment, the evaluation of the artefact intends to be performed through the analysis of responses to the questionnaire on the data model and the solution presented. The questionnaire's goal is to extract insights from the feedback collected, helping the study evaluation process and introducing improvements in the data model. A questionnaire allows the same set of questions to be asked quickly to different people, while maintaining the integrity of the assessment. In order to participate in the questionnaire, a group of 8 professionals was gathered, all of them Siemens employees, with specialised training in IT security. The evaluation was carried out only using this group of Siemens employees for reasons of confidentiality. Although the sample of evaluators can be considered small to extract meaningful quantitative results, the experience of the professionals in the topic of the study conducted may bring qualitative opinions of great value. In addition, it allows opinions to be given anonymously, motivating the evaluators to give honest answers without anyone judging them.

It is also intended to do a final consideration and evaluation of the performance of the chosen database, considering the requirements initially chosen.

Guideline number four - enunciates that effective design-science research must provide clear and verifiable contributions in the areas of the design artefact, design foundations or design methodologies. Most often, the contribution of design-science research is the artefact itself. The artefact must enable the solution of heretofore unsolved problems [80]. The artefact delivered is solution for the problem proposed and it is a contribution for the industrial networks management area as an artefact in the business environment that produces significant value to the its research community.

Guideline number five - discusses research rigor saying that design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact. The environments in which IT artefacts must perform and the artefacts themselves may defy excessive formalism as important parts of the problem may be abstracted [80].

The developed prototype uses as input data provided by a remote testbed managed by a Siemens team, created to simulate a real industrial environment. These testbed environments allow to perform several iterations throughout the development process, allowing engineers to fine-tune the configurations of the devices involved until they reach levels of security and scalability to be migrated to a production environment.

Table 5.1 presents the main characteristics of the system used to build the prototype, displaying the software and which versions were used. Table 5.2 shows the hardware specification of the computers used during the development and while performing evaluation tests to the prototype.

Software	Version
Python	**3.7**
Py2Neo	**v4**
PyQT5	**5.12**
Neo4j	**3.2.14**
Windows	**10**

Table 5.1: Software versions used during the evaluation of the developed solution.

CPU	RAM
Intel i5 Dual Core	**8 GB**
Intel i7 Quad Core	**16 GB**

Table 5.2: Hardware specification of the machines used during the evaluation.

The detail degree of the data model in the developed solution displays the level of abstraction applied during the development process. The asset types attributes defined within the data model reflect the needs of use cases, excluding other attributes that would not add value to the scope defined.

Guideline number six - the proposed approach states design science as a search process inherently iterative to discover an effective solution to a problem, knowing that the search for the best design is often intractable for realistic information systems problems [80]. It also portrays the search for an effective artefact that should utilise the available means to reach desired ends while satisfing the laws in the problem environment. Design-science research often simplifies a problem by explicitly representing only a subset of the relevant subjects by decomposing a problem into simpler subproblems.

In collaboration with consultants involved in the Siemens project, was defined a roadmap of steps at the beginning of the designing process for the proposed approach. The figure 4.1 previously presented on Section Proposed Architecture, reflects the division of the initial problem into sub-processes, allowing the development of programmed components with different responsibilities.

At each stage of the roadmap, a development cycle was applied, performing several iterations to the prototype, managing the changes made to the code using a version control system. The use of this system allowed remote access to the project by other employees for testing using a blackbox policy, regularly sending feedback on possible existing problems. By sharing the input used in the tests it was possible to detect and correct the obstacles encountered at all stages, achieving the objectives and requirements initially proposed.

Guideline number seven - refers to the way to communicate the research conducted based on design-science research [80]. The results must be presented effectively both to technology-oriented, as well as, management-oriented audiences. It also states how important is for such audiences to understand the processes by which the artefact was constructed and evaluated. The emphasis must be on the importance of the problem and on the effectiveness of the solution approach realised in the artefact.

For the technology-oriented audiences it was built a PDF file as project documentation, where it is described how to use the solution developed and the roles of each component, establishing repeatability of the research project and building the knowledge base for further research.

For the management-oriented audience, the results were displayed in a presentation made to a number of managers and employees of the Siemens IT Security department. During this presentation, several topics were discussed topics such as the workflow project, the data model and the way some business rules were implemented. It was also presented a demo of the database support tool, where it was noticeable that the audience understood the objectives and general operation of the tool.

In summary, it is possible to state that the study developed successfully followed all the guidelines described by the method proposed by Hevner et al [80].

5.2 Database Evaluation

In order to evaluate the performance of the used database, it is intended to go through the requirements initially defined in chapter Database Requirements. This re-evaluation aims to verify if the initial choice of using a graph database was indeed a good choice after the solution's development, while describing which obstacles were encountered and how they were overcome.

Scalability - the definition for the scalability requirement aimed for a database that should be able to process a few hundred items, maintaining the same performance over time. Back then on the initially evaluation all the models covered were classified with rating of 3, given that the records volume wouldn't trigger lack of performance.

The solution proposed introduce a tool to support the import and export from a graph database. As an input data, it is given a JSON file with data structures representing the assets and their relationships. This asset data is then loaded into the database using predefined Cypher queries, built based on what was defined within the data model regarding object types, attributes and relationship types.

During development, different input file sizes were used, up to about 500 items, and there was no significant increase in runtime, with no loads exceeding 20 seconds of execution. The export functionality also showed great performance: using a method to export from the Neo4j Awesome Procedures On Cypher (APOC) library, largely optimised by Neo4j and its community, it was possible to export all data to a predefined data structure file, for most of the cases in less than 3 seconds. These performances verify what was initially said regarding the scalability of the chosen database, considering that it was up to what was required.

Data Schema - this requirement was meant to evaluate how the chosen database structure defines the objects and their constraints within the database. The used database allowed to create and update objects of an already existing type with new attributes without having to change the object schema. It was also possible to define that certain properties should be mandatory for the creation of certain objects types through the use of constraints. Throughout the development, the data model specification was updated and redefined. These changes were later manually reflected in the database. In this the base schema of each asset already created within the database way was kept up to date.

The difference between this database and the traditional relational ones is that it allows to update the data model as new objects are added to the database. On the other hand, in relational databases, the schema must be defined at the beginning, so that new records can be created, and no more fields can be added, unless the table definition is updated to handle the new columns.

The flexibility demonstrated by the graph database makes it possible to state that it is a good solution for projects where changes in the data model are constant, until the use cases requirements are met.

Data Visualisation - the graphical interface Neo4j Browser, allowed an easy and simple data visualisation for the database, representing the asset through nodes and relationships using arrows. This factor was determinant for the choice of the graph database model on the first evaluation moment. The end user could then easily observe the relationships, and make queries that filter the information displayed by the graph. The graphical interface of the database also allows the filtering of the node types through point and click, in addition to save predefined queries to more easily select a set of filtered data.

During development, the scale limit of the Neo4j graphics engine was tested, increasing the number of nodes and relationships displayed. A performance crash was detected when displaying over two thousand objects simultaneously, quite evident when dragging any of the nodes through the graphical interface. In order to verify if it was related to the hardware specification of the computer used, the same test was done later on a computer with a better processor, more memory and dedicated graphics card. The difference was notorious, being able to navigate through the interface freely, for the same two thousand objects.

With this, we conclude that the experience of the end user will always depend on the computer specification used, and the number of nodes displayed.

Despite the performance limitations for a large number of nodes, this seems to be a database type that brings something different to the problem under study, confirmed by the opinions extracted from the questionnaire. Two of the consultants highlighted the advantages of graphic representation over simple text. It was expected a great enthusiasm for being able to analyse an industrial network through a graph, instead of database tables or Excel spreadsheets. Another consultant stated that the hierarchies between Host, Interface and Service could be more evident, raising some concerns about the choice of using arrows to represent the relationships between the nodes.

When the participants of the questionnaire were asked if they would advise another database type (relational, document, key-value or any other) for this purpose, one of them pointed out that it would always depend on the requirements of the system. Although the evaluator agrees with the chosen database to serve the purpose, he considered that it may not be a scalable solution, in case it is necessary to view a larger number of nodes simultaneously. Another evaluator highlighted the fact that visualisation alone would not bring any real benefit to the business, such as efficiency or better customer documentation that other tools already do, and do well.

Participants were asked to rank from 1 to 5, if the graphical database is a good approach to view the database records and their relationships, describing networks of industrial assets. The average score obtained was 4, showing general acceptance for this database type to visualize this kind of data, despite a small sample of participants.

Query Language - as result of the initial assessment described earlier in this study, the graph database was not the immediate choice when it came to query language, as it required learning Cypher language, rather than the traditional and widely used SQL query language.

However, Cypher proved to be a surprise, allowing the use of MATCH (the equivalent of SELECT in SQL), MERGE (which allows defining scenarios for the case the object is being created or updated), and commands to identify chains of relationships between two or more different types of assets. It should also be noted the importance of the existence of the Py2Neo toolkit [77], which enabled the integration of Python scripts to automate the insertion and updating of records in the database.

For the operations necessary to perform in the context of the proposed solution, the query language of Neo4j proved to meet the expectations, being its syntax easy to understand, and with a large community available online.

Import and Export Data - initial research revealed that, at least, one implementation of each database model met the requirement of import and export data in CSV and JSON formats, and so all models were given the max score of 3. However, later it was discussed and defined that only the JSON file format would be used, in order to simplify the automated processes created.

The data structure within JSON files, to store asset data, has been defined based on the data structure exported by the *apoc.export.json* command from the Neo4j APOC plugin. Importing data with the same structure as the exported, makes easier for the user to update the database using a JSON file of a previous export, in order to rollback the database to a previous state. JSON has also proved to be a lightweight format for data exchange, generating small files and very simple to read.

About the developed database support tool, one of the participating evaluators suggested to minimize the necessary input files for the generation of the asset data JSON file. He added that a text with instructions for use should be included. This can be a factor that would decrease the risk of input mistakes while using the database support tool.

Backup and Snapshot - this requirement defines that the chosen database should allow the creation of backups, as well as comparison between two snapshots. The design has been changed to meet the requirement, since Neo4j Browser does not support this feature by default, and it is too complex to develop a custom solution to the study problem. Initially, it was ideally intended to create a graphical interface where two files with versions of a dataset could be loaded, and later visually enhance the differences between the two, to be noticeable what had been added, removed or changed between versions. This feature would allow to highlight the differences between current and intended state of the assets, their attributes and relationships.

Later, it was decided to ease the requirement, based on what was defined in the "Update Database with scan results" phase of the Proposed Architecture.

New objects would only be created, updated or removed based on certain criteria and business rules detected during results analysis of the scan performed by SiESTA. For the inserted or changed objects, the update date was defined for the test date and a status was defined, which made it possible to distinguish the objects that should or should not be on the network and that ended up being detected in the scan.

In short, it is considered that the database chosen had a performance that meets the defined expectations, despite some performance problems with Neo4j and the difficulty in implementing the interactive snapshots functionality. The database proved to be a plausible solution for the developed system, however it is considered that the choice will always depend on the context of the problem, requirements or dependencies with other systems.

5.3 Data Model Evaluation

This section is intended to describe the approach to the evaluation of the data model used during the proposed solution's development. In order to carry out the evaluation of the data model it is necessary to take into account some important constraints.

The study is being performed on a business environment, where the evaluation methods usually have to be adapted to the existing intelectual property and confidentiality of the organisation. The completeness the evaluation of a data model is something complex to perform: usually it is much more difficult to measure the quality of a specification than a finished product. The model will eventually be in constant evolution and optimisation, according to the need to integrate new use cases or update the existing ones.

Moody [81] conducted a study about an empirical evaluation of the use of quality metrics in practice. The author refer as an unexpected finding, that the subjective ratings and qualitative information were perceived to be more useful than the metrics for evaluating and improving the quality of data models. The opinions obtained from the questionnaire are not fully directed to the data model. However, some of the considerations extracted from the questionnaire have been included, to support the descriptive evaluation of the data model.

When the participants of the questionnaire were asked what aspects they liked on the data model presented, the model's good structure and simplicity were highlighted, focusing on the types of objects needed for the defined use cases requirements. This opinion was then acknowledgeed by another evaluator, who adds that the model makes a good coverage of relevant tools, usually used in the context of industrial networks and improves the visibility of network configurations.

On the other hand, when evaluators were asked what would be the next use cases to extend the solution functionality, it was pointed out that the data model could be extended to ensure greater compliance with security by managing accounts and accesses to host machines. It was also suggested that the tool could integrate with more software and tools than COMOS and SiESTA, thus allowing to concentrate a larger number of data sources. Two evaluators considered that the tool could be extended to cover use of cases such as patch management, and threat and risk analysis. These are cases that are considered to be of great importance in the context of IT industrial security and, when implemented, would make it possible to improve even more the present solution's value.

Chapter 6

Conclusions

This study is related to the life cycle management of assets in industrial networks and infrastructures. The motivation for this research lies on finding a solution to the current business problem, of not having a clear system overview and a well-defined data structure to organize all data about IT assets from different industrial sources, to ensure life cycle assets management.

The literature review carried out focuses on introducing the importance of asset management within industrial control systems, especially SCADA systems, as a key to ensuring the security and availability of essential services in today's societies. It was also conducted a research on industrial software tools currently used to support asset management and security scans, as well as identified industry standards and specifications for managing IT assets on organizations.

The proposed solution presents a database support tool to help managing the security of industrial assets, enabling the integration between COMOS engineering tool and SiESTA security scanner software. The tool aims to enable scan tests performed by SiESTA on industrial networks, using as input a network blueprint and asset data exported from COMOS software.

The necessary requirements for the idealised database were defined. Subsequently, a comparison was made between five types of database models with the objective of assessing which database model would be more appropriate for the problem under study and according to the defined database requirements. As a result of the comparison, it was considered that the graph model and Neo4j Software would be the best solution for the project needs.

A data model was elaborated, based on NIST Asset Identification Specification, and its objects and relationships defined, taking into consideration the requirements of the use cases: Host Discovery, Port Scanning and Vulnerability Management.

The evaluation method of the proposed solution was carried out through a questionnaire delivered to Siemens security consultants, regarding the database and the data model.

It was considered that the database chosen had a performance that met the stakeholders' expectations. It enabled storing asset data from multiple sources and network analysis through a use of the graph, despite some graphical performance issues with Neo4j and the lack of interactive snapshot comparison feature. Regarding the data model, its simplicity on detail was praised, focusing only the objects types and attributes needed for the defined use cases requirements. The evaluation also made it possible to raise improvements to be made to the proposed database support tool, as well as possible future use cases within the industrial security field.

6.1 Contributions

The proposed solution, developed to help manage assets on industrial networks, has shown great importance to the research area where it is inserted, as well as for the industry and Siemens in particular. It integrates different internal tools, like COMOS and SiESTA, with a different perspective from what has been done so far, and manages asset data, to remain consistent during the lifecycle of a certain product.

The definition of the core asset data model is helping security engineers to consider the system in a systematical way. Previously engineering tool included very little security attributes, or did not include at all. Another contribution was the integration with security scans, where sometimes the system information is quite limited, and without any specification or defined model. It should also be noted that the visualisation features of Neo4j graph database has been a very efficient way to let engineers to grasp the most critical asset information in restricted time.

6.2 Future Work

Since the proposed solution is a prototype, it is true that there is always room to improve the existing functionalities, and to address the development of new functionalities to attend relevant use cases for the business. The Siemens IT Security Department has done some Proof of Concepts with Grafana [82] to illustrate some key information based on its industrial infrastructure. It was suggested by members of the project team that the integration of the Neo4j database with Grafana, as a source of information, could add value to the solution developed.

One of the biggest bottle necks in current implementations has been the use of Excel files with thousands of lines. Although Excel is very human readable, it does not offer the efficiency of other formats. Since SiESTA also accepts XML as an input file, it is considered that using it, could introduce performance improvements, when compared to the Excel format.

The data model is another artefact that can be improved. Currently, it is designed to respond to the use of Host Discovery, Port Scan and Vulnerability Management cases. To cover new use cases, for instance Patch Management, new objects and attributes should be added to the model.

References

[1] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2007.

[2] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, “Guide to Industrial Control Systems (ICS) Security Supervisory Control and Data Acquisition (SCADA) Systems, Distributed Control Systems (DCS), and Other Control System Configurations,” *NIST Special Publication 800-82*, vol. Revision 2, no. Initial Public Draft, p. 255, 2014.

[3] T. J. Williams, “The Purdue Enterprise Reference Architecture,” *Computers in Industry*, vol. 24, pp. 141–158, 1994.

[4] T. Macaulay and B. L. Singer, *Cybersecurity for Industrial Control Systems*. CRC Press, 1 ed., 2011.

[5] Investopedia, “"What is an asset ?" - Corporate Finance & Accounting - Financial Statements,” 2019. Available at: `https://www.investopedia.com/terms/a/asset.asp` [Accessed on: 2019-05-07].

[6] S. Hazra, N. K. Campus, and P. Roy, “Development of an IT Asset Management Tool for Enterprise Information System,” vol. IV, no. I, 2008.

[7] International Association of Information Technology Asset Managers, “A Presentation about Asset Management Guidelines,” pp. 1–9, 2008.

[8] M. Stone, C. Irrechukwu, H. Perper, D. Wynne, and L. Kauffman, "IT Asset Management - NIST Cybersecurity IT Asset Management Practice Guide." 2018.

[9] E. Gelle, T. Koch, and P. Sager, "IT Asset Management of Industrial Automation Systems," in *12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)*, pp. 123–128, IEEE, 2005.

[10] I. Management, "Industrial Control System (ICS) Network Asset Identification and Risk Management - Master Thesis," 2011.

[11] H. Entscheidungssicherheit and G. Wettbewerb, "COMOS by Siemens – "Software solutions for optimized plant engineering"," tech. rep., 2018.

[12] Siemens, "Automation Systems - New standards in process control," 2019. Available at: `http://w3.siemens.com/mcms/process-control-systems/en/distributed-control-system-simatic-pcs-7/simatic-pcs-7-system-components/automation-systems/Pages/automation-systems.aspx?stc=wwiia306043` [Accessed on: 2019-10-17].

[13] "A Swiss Army Knife for Digital Security," 2019. Available at: `https://new.siemens.com/global/en/company/stories/research-technologies/cybersecurity/a-swiss-army-knife-for-digital-security.html` [Accessed on: 2019-10-17].

[14] Jiang Wei-hua, Li Wei-hua, and Du Jun, "The application of ICMP protocol in network scanning," pp. 904–906, 2004.

[15] M. Wolfgang, "A document on Host Discovery with Nmap." 2002.

[16] A. D. Montigny-leboeuf and F. Massicotte, "Passive Network Discovery for Real Time Situation Awareness," Tech. Rep. April 2004, 2004.

[17] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Surveying port scans and their detection methodologies,” *Computer Journal*, vol. 54, no. 10, pp. 1565–1581, 2011.

[18] M. D. Vivo, E. Carrasco, G. Isern, and G. de Vivo, “A review of port scanning techniques,” *ACM SIGCOMM - Computer Communication Review*, pp. 41–48, 1999.

[19] S. Staniford, J. A. Hoagland, and J. M. McAlerney, “Practical automated detection of stealthy portscans,” *Journal of Computer Security*, vol. 10, no. 1-2, pp. 105–136, 2002.

[20] T. Ahamad Ahanger, “Port Scan-A Security Concern,” *Certified International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 9001, no. 10, pp. 2277–3754, 2008.

[21] “Nmap: the Network Mapper - Free Security Scanner.” Available at: `https://nmap.org/` [Accessed on: 2019-08-14].

[22] “Nessus | Comprehensive Vulnerability Assessment Solution.” Available at: `https://www.tenable.com/products/nessus` [Accessed on: 2019-08-14].

[23] G. Giacobbi, “The GNU Netcat – Official homepage,” 2006. Available at: `http://netcat.sourceforge.net/` [Accessed on: 2019-08-14].

[24] R. A. Martin, “Managing Vulnerabilities in a Networked System,” vol. 10, no. 2, pp. 145–150, 2013.

[25] C.-w. Ten, S. Member, C.-c. Liu, and G. Manimaran, “Vulnerability Assessment of Cybersecurity for SCADA Systems,” *1836 IEEE Transactions on Power Systems*, vol. 23, no. 4, pp. 1836–1846, 2008.

[26] “MITRE Corporation,” 2018. Available at: `https://www.mitre.org/` [Accessed on: 2019-08-12].

[27] NIST, “NVD - Home,” 2016. Available at: `https://nvd.nist.gov/https://nvd.nist.gov/home.cfm` [Accessed on: 2019-12-26].

[28] Carnegie Mellon University, “CERT Vulnerability Notes Database,” 2012. Available at: `https://www.kb.cert.org/vuls/` [Accessed on: 2019-12-26].

[29] DIN, “Why do we need standards?,” 2019. Available at: `https://www.din.de/en/about-standards/use-standards` [Accessed on: 2019-07-23].

[30] International Organization for Standardization, “ISO/IEC 19770-1:2017 - Information technology – IT asset management – Part 1: IT asset management systems – Requirements,” 2017. Available at: `https://www.iso.org/standard/68531.htmlhttps://www.iso.org/obp/ui/{#}iso:std:iso-iec:19770:-1:ed-3:v1:en` [Accessed on: 2019-01-14].

[31] I. O. for Standardization, “ISO/IEC 19770-2:2015 - Information technology – IT asset management – Part 2: Software identification tag.” Available at: `https://www.iso.org/standard/65666.html` [Accessed on: 2019-05-07].

[32] Anton Pääkkönen, “Asset Management in an ICT Company using ISO/IEC 19770 - Master Thesis.” 2017.

[33] International Society of Automation, “ISA-62443-1-1-2007 Security for Industrial Automation and Control Systems Part 1-1: Terminology, Concepts, and Models,” 2018. Available at: `https://www.isa.org/store/products/product-detail/?productId=116720` [Accessed on: 2019-12-24].

[34] J. Wunder, A. Halbardier, and D. Waltermire, “Specification for Asset Identification 1.1,” *NIST Interagency Report 7693*, pp. 7–25, 2011.

[35] R. Cattell, “Scalable SQL and NoSQL Data Stores,” *ACM SIGMOD Record*, vol. 39, pp. 12–27, dec 2011.

[36] R. Angles and C. Gutierrez, “Survey of graph database models,” *ACM Computing Surveys*, vol. 40, no. 1, pp. 1–39, 2008.

[37] E. F. Codd, “A relational model of data for large shared data banks,” *Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.

[38] "ISO 9075:1987 - Information processing systems – Database language – SQL." Available at: `https://www.iso.org/standard/16661.html` [Accessed on: 2019-06-24].

[39] Google, "Cloud Bigtable | Google Cloud," 2019. Available at: `https://cloud.google.com/bigtable/?hl=pt-br` [Accessed on: 2019-12-21].

[40] Amazon, "Amazon DynamoDB - Overview," 2019. Available at: `https://aws.amazon.com/dynamodb/` [Accessed on: 2019-12-21].

[41] Y. Li and S. Manoharan, "A performance comparison of SQL and NoSQL databases A performance comparison of SQL and NoSQL databases," *978-1-4799-1501-9/13- 2013 Ieee*, no. November, pp. 15–19, 2015.

[42] DB-Engines, "DB-Engines Ranking - popularity ranking of database management systems," 2018. Available at: `https://db-engines.com/en/ranking` [Accessed on: 2019-06-21].

[43] J. D. Jing Han, Haihong E, Guan Le, "Survey on NoSQL Database," pp. 363–366, 2012.

[44] Neo4J, "5 Sure Signs It's Time to Give Up Your Relational Database - Neo4j Graph Database Platform," 2018. Available at: `https://neo4j.com/blog/five-signs-to-give-up-relational-database/` [Accessed on: 2019-05-07].

[45] J. Borchers, "Oracle SQL Developer," 2006. Available at: `https://www.oracle.com/database/technologies/appdev/sql-developer.html` [Accessed on: 2019-05-24].

[46] O. Corporation, "MySQL Database Software," 2019. Available at: `https://www.mysql.com/` [Accessed on: 2019-05-24].

[47] "An Overview of Graph Database Query Languages." Available at: `https://developer.ibm.com/dwblog/2017/overview-graph-database-query-languages/` [Accessed on: 2019-06-24].

[48] Neo4J, "Graph Database Use Cases and Solutions," 2019. Available at: https://neo4j.com/use-cases/ [Accessed on: 2019-05-24].

[49] Neo4J, "Neo4j Graph Platform – The Leader in Graph Databases," 2019. Available at: https://neo4j.com/ [Accessed on: 2019-05-24].

[50] OrientDB, "Graph Database | Multi-Model Database | OrientDB," 2019. Available at: https://orientdb.com/ [Accessed on: 2019-05-24].

[51] N. Ameya, P. Anil, and P. Dikshay, "Type of NOSQL databases and its comparison with relational databases.," *International Journal of Applied Information Systems*, vol. 5, no. 4, pp. 16–19, 2013.

[52] S. Zhang, "Application of document-oriented nosql database technology in web-based software project documents management system," *2013 IEEE 3rd International Conference on Information Science and Technology, ICIST 2013*, pp. 504–507, 2013.

[53] "The most popular database for modern apps | MongoDB." Available at: https://www.mongodb.com/ [Accessed on: 2019-05-24].

[54] "Couchbase NoSQL Database | Couchbase." Available at: https://www.couchbase.com/ [Accessed on: 2019-05-24].

[55] D. J. Abadi, P. A. Boncz, and S. Harizopoulos, "Column-oriented database systems," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1664–1665, 2014.

[56] AWS, "What is a Columnar Database? – AWS." Available at: https://aws.amazon.com/nosql/columnar/ [Accessed on: 2019-06-26].

[57] A. S. Foundation, "Apache Cassandra," 2019. Available at: http://cassandra.apache.org/ [Accessed on: 2019-05-24].

[58] A. S. Foundation, "Apache HBase – Apache HBase^{TM} Home," 2019. Available at: https://hbase.apache.org/ [Accessed on: 2019-05-24].

[59] "Redis - Database," 2019. Available at: `https://redis.io/` [Accessed on: 2019-05-24].

[60] "AWS | Amazon DynamoDB – NoSQL Online Datenbank Service." Available at: `https://aws.amazon.com/de/dynamodb/` [Accessed on: 2019-05-24].

[61] A. B. Bondi, "Characteristics of scalability and their impact on performance." 2004.

[62] D. Sullivan and J. Sullivan, "NoSQL Key-Value Database Simplicity vs. Document Database Flexibility," 2015. Available at: `http://www.informit.com/articles/article.aspx?p=2429466` [Accessed on: 2019-08-20].

[63] S. Jouili and V. Vansteenberghe, "An empirical comparison of graph databases," in *Proceedings - SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013*, pp. 708–715, 2013.

[64] NEO4j, "Neo4j Graph Platform – The Leader in Graph Databases," 2018. Available at: `https://neo4j.com/` [Accessed on: 2019-12-26].

[65] Titan, "Titan: Distributed Graph Database," 2014. Available at: `https://titan.thinkaurelius.com/http://thinkaurelius.github.io/titan/` [Accessed on: 2019-12-28].

[66] Sparsity Technologies, "Sparsity-technologies: Sparksee high-performance graph database," 2019. Available at: `http://sparsity-technologies.com/` [Accessed on: 2019-12-28].

[67] D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey, "Survey of graph database performance on the HPC scalable graph analysis benchmark," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6185 LNCS, pp. 37–48, 2010.

[68] A. S. Foundation, "Apache Jena," 2019. Available at: `http://jena.apache.org/` [Accessed on: 2019-12-28].

[69] Kobrix Software, “HypergraphDB - A Graph Database,” 2010. Available at: http://www.hypergraphdb.org/ [Accessed on: 2019-12-28].

[70] Claroty, “Claroty for OT Networks,” 2018. Available at: https://www.claroty.com/ [Accessed on: 2019-07-02].

[71] OCS, “OCS Inventory NG | Accueil,” 2019. Available at: https://ocsinventory-ng.org/?lang=en [Accessed on: 2019-07-02].

[72] ABB, “ABB Ability TM - Cyber Security Asset Inventory,” *ABB Whitepaper - Cyber Security for power generation and water*, 2018.

[73] Python Software Foundation, “Welcome to Python.org,” 2017. Available at: https://www.python.org/ [Accessed on: 2019-10-26].

[74] “openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files — openpyxl 2.6.2 documentation.” Available at: https://openpyxl.readthedocs.io/en/stable/ [Accessed on: 2019-06-06].

[75] “xlwings - Make Excel Fly! — xlwings dev documentation.” Available at: https://docs.xlwings.org/en/stable/index.html [Accessed on: 2019-06-06].

[76] “json — JSON encoder and decoder — Python 3.7.3 documentation.” Available at: https://docs.python.org/3/library/json.html{#}https://docs.python.org/3/library/json.html{%}23 [Accessed on: 2019-06-06].

[77] N. Small, “The Py2neo v4 Handbook,” 2018. Available at: https://py2neo.org/v4/ [Accessed on: 2019-06-06].

[78] “PyQt5 - Riverbank.” Available at: https://www.riverbankcomputing.com/software/pyqt/download5/ [Accessed on: 2019-06-06].

[79] “GitHub - pypa/pip: The Python Package Installer (recommended by PyPA).” Available at: https://github.com/pypa/pip [Accessed on: 2019-06-06].

[80] Hevner, March, Park, and Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, p. 75, dec 2004.

[81] D. Moody, "Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice," in *Proceedings of the Eleventh European Conference on Information Systems*, vol. 29, pp. 1337–1352, 2003.

[82] T. Ödegaard, "Grafana: The Open Observability Platform," 2019. Available at: `https://grafana.com/` [Accessed on: 2019-10-27].

Appendices

Appendix A - Questionnaire: Data Model Evaluation

This appendix is related to the questionnaire developed for the evaluation phase planned for this study. This questionnaire contains demographic questions (Siemens role), closed-ended questions (rated from 0 to 5), and open-ended questions.

About the questionnaire "Data Model Evaluation"

Dear study participant:

The purpose of this questionnaire is to evaluate the data model and database defined for this study. By participating, you are helping to find and solve possible existing problems that we are not aware of, and potentially suggesting solutions to them.

The questionnaire is voluntary and anonymous, only your role at Siemens is collected as personal information, for statistical and results analysis purposes.

The questionnaire contains questions and statements about the suitability of data model and database chosen regarding the study conducted for this project on asset management for ICS security maintenance.

In some of the questions you will be asked to write your opinion on an open-ended question. In other cases you will be asked whether you agree or disagree with the statement, making use of the scale used for the purpose. You may also have no actual opinion on the subject, thus marking the cell in the corresponding column. An example is shown below:

		Totally Disagree	Disagree	So-So	Agree	Totally Agree	No opinion
ID	Statement	1	2	3	4	5	
0	This data model is appropriate for the use cases presented.				X		

Questionnaire
Data Model Evaluation

.0.1 Personal Info

Role at Siemens: ____

.0.2 Closed-ended Questions

		Totally Disagree	Disagree	So-So	Agree	Totally Agree	No opinion
ID	Statement	1	2	3	4	5	
1	The solution proposed is capable of supporting asset management for industrial networks.						
2	Most important use cases (host, port and vulnerabilities scan) are well covered by the data model presented.						
3	A graph database it's a good approach to visualize database records and its relationships for industrial networks.						
4	It is possible to identify the project workflow phases with the features of the database support tool presented.						

.0.3 Open-ended Questions

What are the aspects you liked in data model presented ?

How would you improve the data model presented?

Are there any further uses cases that you think should be considered?

Would you advise another database type (relational, document, key-value or any other) for this purpose? If yes, which would be and why?