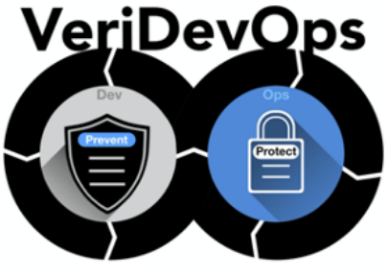

VeriDevOps

Automated Protection and Prevention to Meet Security Requirements in DevOps Environments

D3.1 Threat oracle engine specification, design and implementation - initial version

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957212. This document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Contract number:	957212
Project acronym:	VeriDevOps
Project title:	Automated Protection and Prevention to Meet Security Requirements in DevOps Environments
Delivery Date:	31/12/2021
Coordinator:	IKER
Partners contributed:	IKER, MI
Release Date:	31.12.2021
Version:	01
Revision:	MDH, ABO, MI
Abstract:	This deliverable provides an overview of the initial version of the Threat Oracle Engine (THOE) for known vulnerability analysis
Status:	PU (Public)

Table of Contents

List of Acronyms

CPE	Common Platform Enumeration
CVE	Common Vulnerability and Exposure
CVSS	Common Vulnerability Scoring System
DevOps	Software Development and IT Operations
ICS	Industrial Control Systems
ICT	Information and Communication Technology
IoT	Internet-of-Things
IPC	Information Processing Component
IT	Information Technology
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OT	Operational Technology
OWASP	Open Web Application Security Project
PLC	Programmable Logic Controller
POS	Part of Speech
SCAP	Security Content Automation Protocol
THOE	The Threat Oracle Engine Tool
VM	Vulnerability Management

Executive Abstract

This deliverable provides the specification and design of a vulnerability scanner, named Thread Oracle Engine (THOE) that is being developed by IKERLAN in VeriDevOps. This THOE aims at developing an active publicly-known vulnerability search service using international databases as the primary source of vulnerabilities. In this way, the purpose is to have up-to-date information on vulnerabilities that may affect industrial products or IT products. It is essential that the detection of vulnerabilities is carried out throughout the product life cycle. This deliverable corresponds to an initial version of THOE, the final version of THOE will be described in D3.2.

1. Introduction

The VeriDevOps project assumes a DevOps development process, starting from the initial security requirements of the system under test, ensuring preventive security at development and, last but not least, providing reactive protection at operations. WP3: Reactive Protection at Operations will incorporate security practices into operational environments. THOE is part of this operational phase and it will be in charge of early vulnerability identification. This deliverable describes the vulnerability detection system implemented in THOE, while the system characterization and threat identification tasks will be covered in deliverable *D3.3: Security Monitoring - security flaws detection mechanisms and tools initial version*.

First, the different technologies and architectures that can be used were analyzed and later THOE was designed. At the time of writing this deliverable, THOE is being implemented.

The current deliverable D3.1 is divided in the corresponding sections:

- motivation, in which we review a general state-of-the-art on vulnerabilities in industrial control systems,
- background, where we discuss the basic concepts on which the rest of the document relies on,
- vulnerability databases, where we list a set of vulnerability databases that will be used as source for the tool,
- THOE requirements, where we state the requirements to be fulfilled by the tool,
- THOE's design, in which we describe the global architecture, as well as, the components, workflows and interaction with the tool,
- implementation status, where we provide details about the current implantation state of the tool,
- usage in case studies, when we described the current and potential application of the tools in the case studies.

2. Motivation

Several studies highlight that the number of cyberattacks on Operational Technology (OT) is increasing [1]. OT refers to the set of technologies, software and hardware that organizations use for managing physical industrial equipment, assets, processes and events. Industrial Control Systems (ICS) are widely used in these OT environments to monitor and control industrial processes, including manufacturing, transport and pharmaceutical sectors, and critical infrastructures, such as electricity, water treatment plants, and oil and gas refineries [2].

Historically, ICSs ran on proprietary hardware and/or software that were physically isolated from external connections; today the situation is totally different. ICSs have adopted Information Technology (IT) solutions, such as commercial off-the-shelf (COTS) components, standard operating systems, and remote connectivity as well as cloud solutions. This evolution, together with the use of unsecure industrial protocols, such as, DNP3, OPC, MODBUS, increases the likelihood of security vulnerabilities and incidents [3] [4]. As a result, ICSs are subjected to the same type of vulnerabilities as any other system: including, buffer overflows, hardcoded credentials, authentication bypass, cross-site scripting, missing authentication, and vulnerabilities in hardware chips [5], among others.

According to Kaspersky [1], more than 150 industrial control systems-related vulnerabilities were discovered every year since 2012. Moreover, they showed that the most vulnerable ICS components were Human Machine Interface (HMI), electric devices, and SCADA systems. Vendors produced patches and new firmware for 85% of the published vulnerabilities. Most of the unpatched vulnerabilities, that is 74%, were considered being of high-level risk, but they were not addressed properly by vendors exposing a significant risk to the owners of the systems. Security challenges are emerging also in cloud environments, including a variety of issues such as misconfiguration issues and vulnerabilities in hardware chips.

Fast and effective detection and patching of known vulnerabilities are essential to effectively preventing cyberattacks. This issue was widely recognized in 2017 when the WannaCry ransomware exploited vulnerabilities for which Microsoft had published a patch two months before. Unfortunately, some organizations have not installed it, while the patch was also not available for legacy Windows versions (i.e. Windows XP or Windows Server 2003) when the attack was produced. This highlighted the importance of monitoring vulnerabilities even though no attack cases are around.

3. Background

In this section, the foundational security concepts of this research work are defined, along with MITRE's, NIST's and FIRST's Common Security Standards, as formal lists or dictionaries supporting security content automation.

3.1. Basic security definitions

3.1.1. Weakness

Weaknesses are flaws, faults, bugs, and other errors in software and hardware design, architecture, code, or implementation that, if left unaddressed, could result in systems, networks, and hardware being vulnerable to attacks [6] (e.g., buffer overflow).

3.1.2. Vulnerability

A vulnerability is a flaw in a software, firmware, hardware, or service component resulting from a weakness that can be exploited, causing a negative impact to the confidentiality, integrity, or availability of an impacted component or components. It can be used to gain remote or physical access to a system [7].

3.2. Common Platform Enumeration (CPE) scheme

CPE is an abstract structured naming scheme for describing and identifying applications, operating systems, software, and hardware, including industrial control systems, such as Supervisory Control And Data Acquisition (SCADA). The logical construct of a CPE is called "Well-Formed CPE Name (WFN)" [8]. The CPE scheme, however, cannot describe and identify specific instances of products (e.g., using serial numbers, particular licenses, or physically discernible products). CPE is operated by the NIST [9]. The latest version at the time this deliverable was written is version 2.3.

3.3. Common Weakness and Exposures (CWE)

CWE is a community-developed list of common software and hardware weakness types, each one associated with some CVEs (explained in the next subsection). CWE is operated by the MITRE Corporation[1]. The latest version at the time this deliverable was written is version 4.3.

[1] "MITRE Corporation." https://www.mitre.org/

3.4. Common Vulnerabilities and Exposures (CVE)

CVE is a list of common identifiers for publicly known cybersecurity vulnerabilities [10] operated by the MITRE Corporation. Each CVE includes a unique identification number, a description, one public reference, a reference to the software or hardware that is affected (using CPE), a reference to a weakness, and its severity. As Dimitriadis et al. state in [11], "it can be assumed that CVEs are "known knowns" (things we are aware of and understand) or specific vulnerabilities, while CWEs are "unknown knowns" (things we understand but are not aware of) or generic vulnerability types." The latest CVE version is always available on its official site.

CVE-2015-6154 is an example of a vulnerability that exploits the CWE-119 weakness present in Internet Explorer 8, allowing remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted website.

3.5. Common Vulnerability Scoring System (CVSS)

CVSS is a public framework that provides a standardized method for assigning quantitative values (scores) to security vulnerabilities (CVE) [12] according to their severity [13]. A CVSS score is a decimal number in the range [0.0, 10.0]. The latest version at the time this deliverable was written is version 3.1. As an example, the CVSS value for vulnerability CVE-2015-6154 is 9:3 out of 10. This value is classified as critical.

3.6. Common Vulnerability Reporting Framework (CVRF)

Common Vulnerability Reporting Framework (CVRF) [14] is an XML-based language that enables different stakeholders across different organizations to share critical security-related information in a single format, speeding up information exchange and digestion. It was created to fill a gap created by the lack of a standard framework for the creation of vulnerability report documentation. Many vulnerability databases allow downloading CVE lists using the CVRF format.

4. Vulnerability databases

For early vulnerability detection, THOE relies on public international vulnerability databases, being those the main information source to feed the tool. This section describes a list of publicly available vulnerability databases that will be used as THOE's data source. At the moment of writing this deliverable, only the NVD database is used by THOE. However the tool is designed in a modular way, in order to be flexible enough to easily add new data sources to the tool. Also note, that publicly available databases may also vary along the time: some of them may not be maintained anymore, while new ones may be created. So, in consequence, the list of vulnerability databases used by THOE once and after it is fully implemented may vary, as it is not a static list.

4.1. National Vulnerability Database

The National Vulnerability Database (NVD) [15] is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics to allow automation of vulnerability management, security measurement, and compliance.

All vulnerabilities are assigned a Common Vulnerabilities and Exposures (CVE) identifier and are scored using the Common Vulnerability Scoring System (CVSS), which is based on a set of equations using metrics such as access complexity and availability of a remedy.

Originally created in 2000, the NVD is a product of the NIST Computer Security Division, Information Technology Laboratory and is sponsored by the Cybersecurity & Infrastructure Security Agency of the U.S. government.

4.2. WPScan Wordpress Vulnerability Database

Started as a simple script, WPScan [16] has become a large software project to help identify vulnerabilities in self-hosted WordPress websites. It makes use of the WPScan WordPress Vulnerability Database, which is a database where WordPress vulnerabilities, plugin vulnerabilities and theme vulnerabilities compiled by the WPScan team and various other contributors are stored. CVE is used to identify each vulnerability and it provides search functionality, as well as an API.

4.3. Exploit Database

Exploit Database (ExploitDB) [17] is a CVE compliant repository of public exploits and the corresponding vulnerable software, developed for use by penetration testers and vulnerability researchers for the purpose of public security. Its aim is to serve as the most comprehensive collection of exploits, shellcode and papers gathered through direct submissions, mailing lists, and other public sources, and present them in a freely-available and easy-to-navigate database. ExploitDB is a non-profit project maintained by Offensive Security and the repository is updated daily with the most recently added submissions.

4.4. CVEdetails

CVEdetails provides an esy to use web interface to CVE vulnerability data [18]. CVEdetails was created by security consultant Serkan Özkan, as an effort to have an easy to use list of security vulnerabilities. It gathers vulnerability data from different sources and provides browse and search tools to list them. It also provides the ability to generate custom RSS feeds and API calls for specific searches.

4.5. CVE Program

The mission of the CVE Program [10] is to identify, define, and catalog publicly disclosed cybersecurity vulnerabilities. The vulnerabilities are discovered then assigned and published by organizations from around the world that have partnered with CVE Program.

4.6. VulDB

VulDB is a vulnerability database documenting and explaining security vulnerabilities and exploits [19]. Started as a personal project called Bugbase in 1997, after going through several steps, became vuldb.com and has become an important source for vulnerability handling, vulnerability management, exploit analysis etc. It uses MITRE standards such as CVE, CPE and CVSS. It provides a simple API for interaction.

4.7. Microsoft Security Response Center

The Microsoft Security Response Center (MSRC) [20] investigates all reports of security vulnerabilities affecting Microsoft products and services, and provides the information there as part of the ongoing effort to help you manage security risks and help keep your systems protected. It provides a RESTful API to get security updates programmatically, formatted according to the Common Vulnerability Reporting Framework (CVRF) XML-based language.

5. Requirements of THOE

The Threat Oracle Engine (THOE) is the tool for scanning the system for known vulnerabilities based on the information obtained from online vulnerability databases, such as, for example, NVD database. THOE is a solution for industrial control systems, including embedded systems with limited resources, and it will enable:

- Automatic detection of publicly known vulnerabilities affecting software and hardware used by a product during its development process and whole lifecycle.
- Searching for vulnerabilities published in different data sources, such as NVD, ExploitDB, etc. It will be flexible enough to allow adding new sources.
- Continuous monitoring for vulnerabilities and configuration
- Local or remote search of vulnerabilities

Therefore, THOE should fulfill the following list of requirements:

ID	Requirement	Criticality
THOE-010	THOE shall use NVD (National Vulnerability Database) as the main source for searching publicly-known vulnerabilities. https://nvd.nist.gov/	High
THOE-020	THOE shall search for vulnerabilities in NVD by means of hardware and software Common Platform Enumeration identifiers, that is, it shall provide query functionalities given a certain CPE identifier	High
THOE-030	THOE shall enable the visualization of vulnerabilities	High
THOE-040	THOE shall generate a report with the list of vulnerabilities and corresponding CVSS values that measures the vulnerability severity	High
THOE-050	THOE shall support the import of CPE-based asset inventory	High
THOE-060	THOE shall support the visualization and download of the logs information	Medium
THOE-070	THOE shall enable the configuration of a scheduler for searching vulnerabilities for a given product	Medium

Table 1. THOE Requirements

6. Design of THOE

THOE is composed of several building-blocks which have been designed as independent subsystems and each of them is intended to fulfill a specific task. The *Scheduling subsystem* manages configurations and schedules data acquisition from online databases. The *Threatfinder subsystem* gathers data from online vulnerability sources, and *the Vulnerability Database subsystem* formats and stores it. The *Interoperability subsystem* is in charge of providing interfaces to interact with THOE. Finally, the *Logging module* logs every event registered in the system.

The General architecture of THOE is depicted in Figure 1, where modules and their components and connections are shown. Each of the modules can be developed and tested independently, which simplifies and makes the development process more agile.

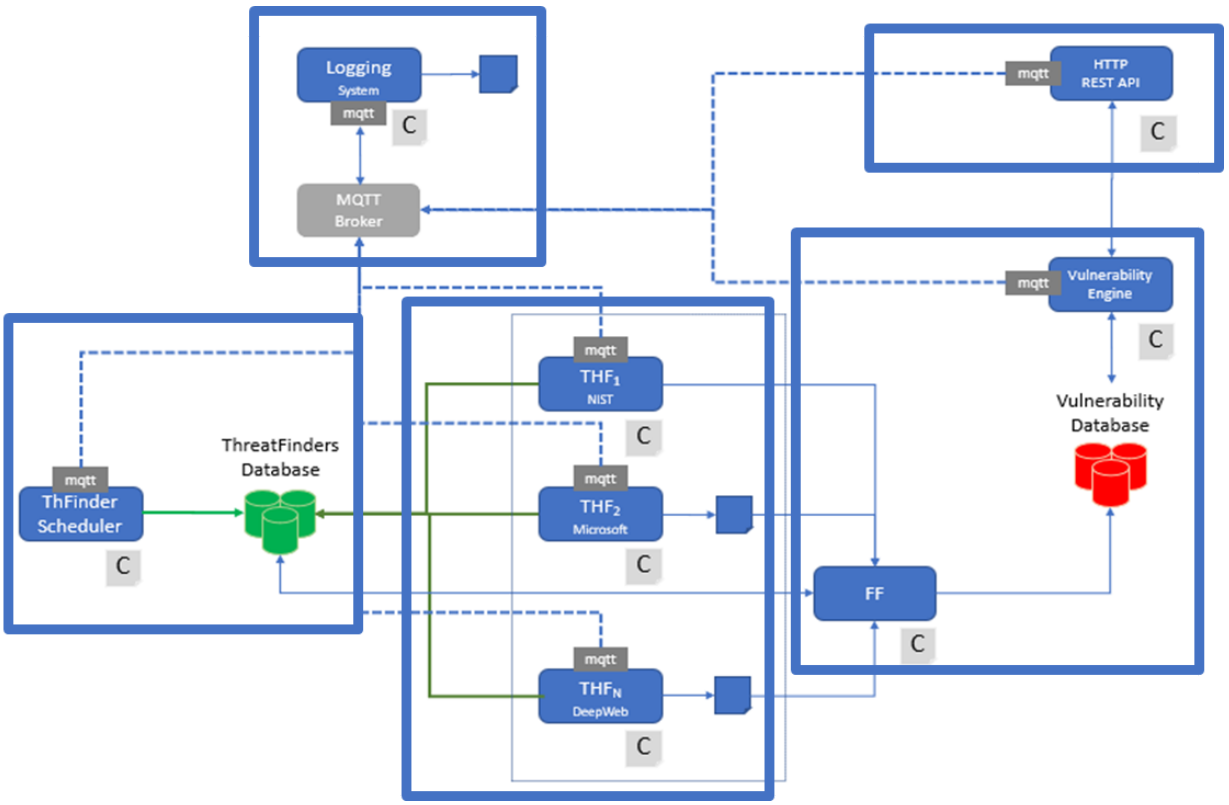

Figure 1. General architecture of THOE

6.1. Architecture

Following subsections describe more in detail each of the modules composing THOE

6.1.1. Scheduling subsystem

Scheduling subsystem is the one in charge of configuring and scheduling the vulnerability-related data acquisition from different sources, and is composed of a scheduler service and a database. The list of different public vulnerability sources and how often they should be consulted is configured here, as well as MQTT brokers and topics used for communications between different subsystems.

This module makes use of a database to store configurations. The scheduling configurations in the database will be later on used by the agents in the Threatfinder subsystem to gather the corresponding configuration. Communications in THOE are made through the MQTT protocol, which is a security enabled lightweight and efficient communication protocol with publish/subscribe architecture. Clients subscribe to topics to send and receive messages and a broker routes the messages to the appropriate destination clients. Broker and topic configuration data for each client in the subsystem is also stored in the database. Figure 2 shows the Scheduling subsystem components:

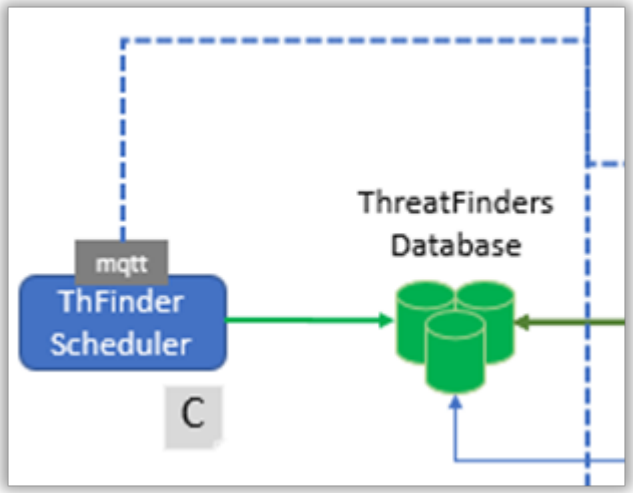

Figure 2. Scheduling subsystem

6.1.2. Threatfinder subsystem

Threatfinder subsystem is in charge of gathering vulnerabilities from online vulnerability databases. Since each online source may use different formats for data and may provide a different interface to interoperate with, an agent or connector called Threatfinder (THF) is implemented for each public data source. Different agents may share some methods, but each of them needs to implement specific functions to interoperate with the online source. THFs communicate with the scheduling subsystem in order to get the parameters they need to use to execute, such as execution schedule, source url, etc. When a THF downloads data from a

source, it communicates to the Vulnerability Database subsystem, where the vulnerability data will be stored. Figure 3 depicts the components of this module.

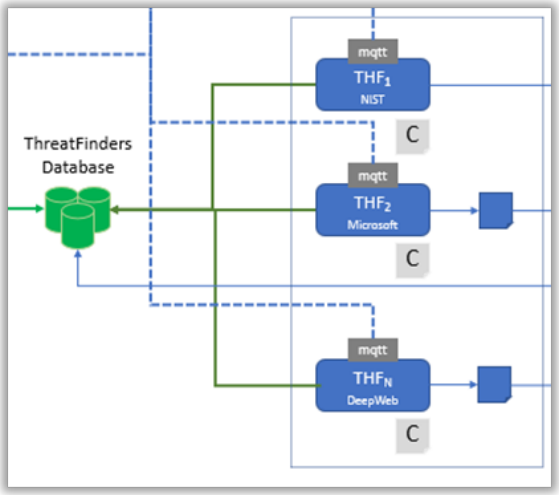

Figure 3. Threatfinder subsystem

6.1.3. Vulnerability Database subsystem

Is in the Vulnerability Database subsystem where all vulnerability data is stored in a normalized and structured format, using a relational database. Data gathered by the Threatfinder subsystem may be in different formats, both structured and unstructured. This subsystem provides a component called Format Factory, which is in charge of formatting the vulnerabilities gathered by the Acquisition module to the structure used by the database and inserting them in the database. The database stores the structured data, which will be used by the Vulnerability Engine component. The Vulnerability Engine module, is a search engine in charge of carrying out searches for vulnerabilities in the database. Figure 4 depicts the components in this module.

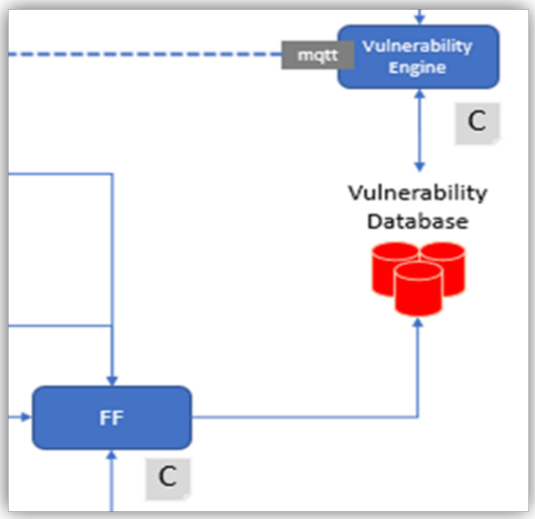

Figure 4. Vulnerability database subsystem

6.1.4. Interoperability interface subsystem

Interoperability interface subsystem (Figure 5) stands between the Vulnerability Engine module and the end user, and is in charge of providing an interoperability interface between them. This subsystem provides a RESTful API that allows making queries to the Vulnerability Database. The API will be used by a website where vulnerability searches could be performed, as well as for integration means for automation purposes. Given a CPE-based asset inventory, the API will be able to return the list of vulnerabilities affecting the platforms.

Figure 5. Interoperability interface subsystem

6.1.5. Logging subsystem

As shown in Figure 1, every subsystem in THOE is linked to the logging subsystem. Each of the subsystems report actions to the Logging subsystem, where logging service logs every action. Communication between each of the subsystems and the logging subsystems is carried out using the MQTT protocol, for which this subsystem implements a MQTT broker. Figure 6 depicts the components that take part in this subsystem.

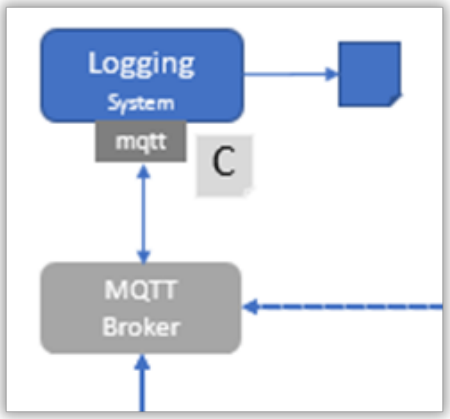

Figure 6. Logging subsystem

6.2. Databases

As mentioned in the previous section, THOE uses two databases to provide the functionality needed by its components. On the one hand, ThreatFinders database stores the configuration information to be used by the Threatfinder subsystem. On the other hand, the Vulnerability database stores the data of all known vulnerabilities in a structured way.

6.2.1. Threatfinders Database

Threatfinders database can be described as a management database. This database stores all the configurations and parameters that are necessary for the agents or connectors in the Threatfinder subsystem, such as descriptions, endpoints and other useful parameters. The configuration data of the agent execution scheduling is also stored in this database. Finally, the MQTT parameters to be used by the clients are also stored in the ThreatFinders database. The Figure 7 shows the structure of this database at the time of writing this deliverable.

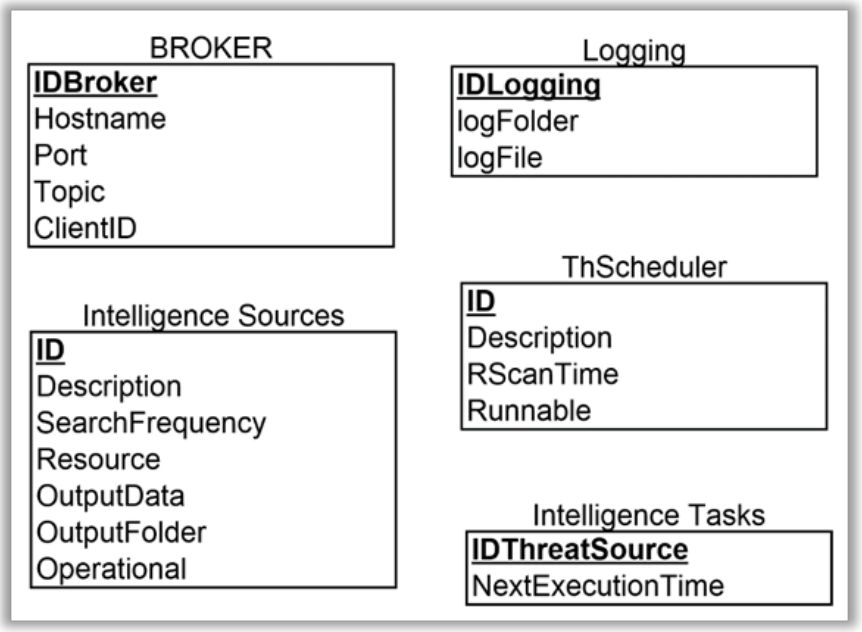

Figure 7. Threatfinders database schema

6.2.2. Vulnerability Database

The Vulnerability database stores structured data about all known vulnerabilities. The schema of this database has been created based on the National Vulnerability Database (NVD) schema from the NIST. Based on the structure of the JSON file format used by the NVD, the schema shown in Figure 8 has been generated, which can be used to store all the vulnerability-related data used by THOE.

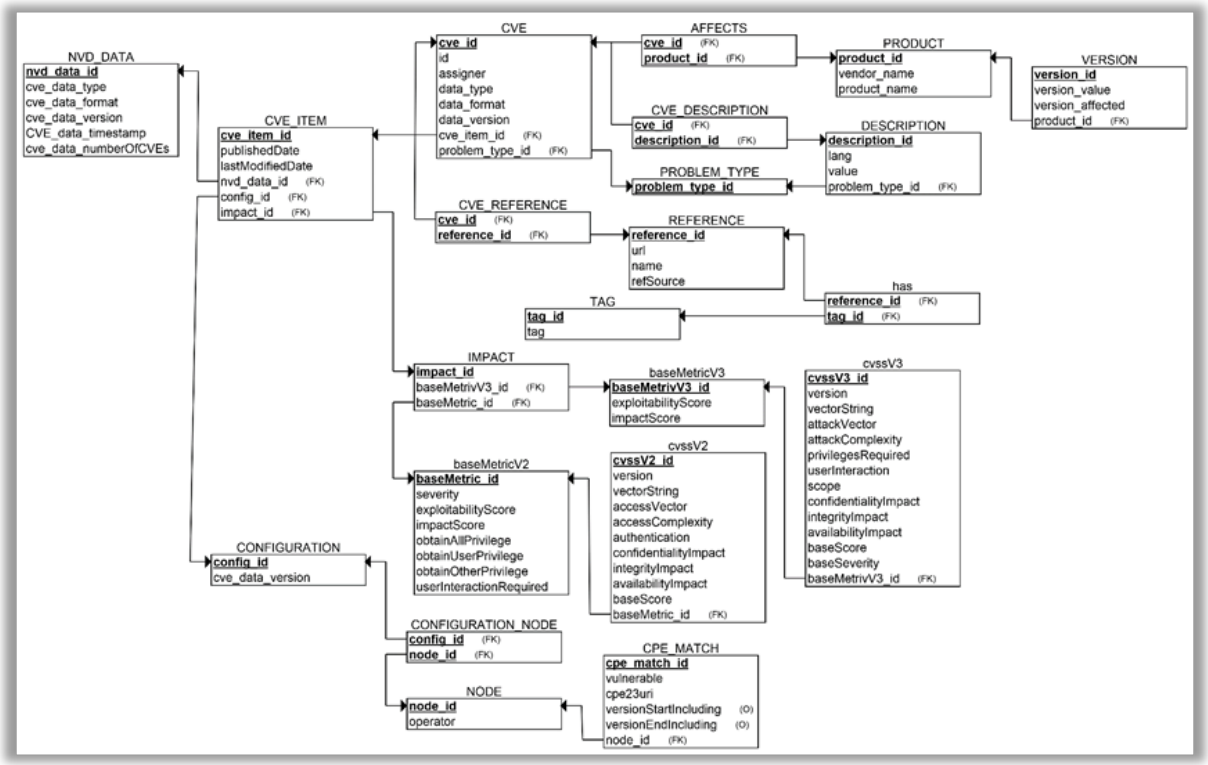

Figure 8. Vulnerability database schema

6.3. Functional Interfaces

As presented in Deliverable 1.4 VeriDevOps Framework Architecture and Roadmap, THOE makes use of several interfaces, both for feeding the tool and providing the desired outputs. As shown in Figure 9 the main inputs for the tool are the public vulnerability sources, in order to keep the internal vulnerability database up to date, and the CPE-based asset inventory, with the list of the platforms that vulnerabilities should be searched for.

On the other side, THOE provides a complete list of vulnerabilities, as well as a REST API to perform searches in the vulnerability database. The tool also provides a MQTT broker for MQTT communications and a log registry to allow the monitoring of all events occurring in the system.

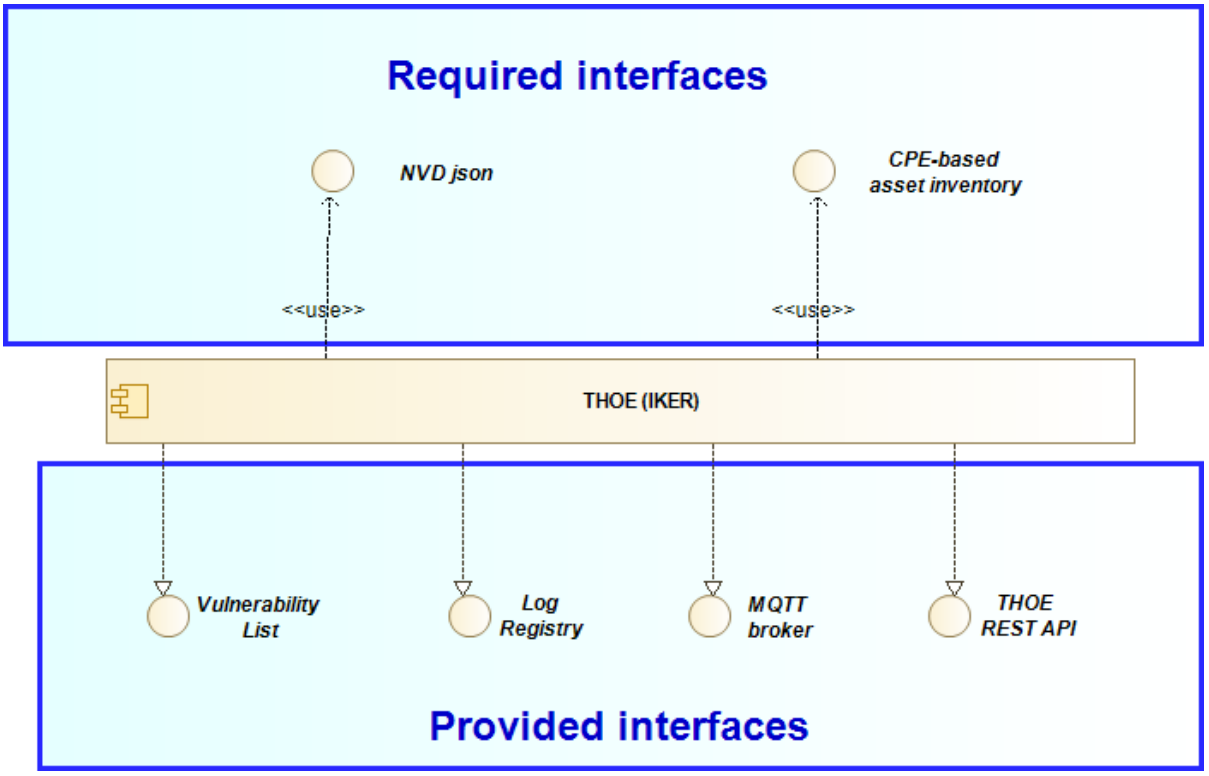

Figure 9. Functional Interfaces (Integration Means)

Some of the mentioned interfaces are specific to THOE tool. Table 2 gives details on them:

Name	Description
CPE-based asset inventory	THOE will be based on defined Common Platform Enumeration (CPE) identifiers for each hardware and software. THOE will use a CPE-based asset inventory for hardware and software to ensure that not only the operating system and installed applications are considered for vulnerability detection, but also libraries, packages, and so on. This asset inventory shall be updated whenever a software update/upgrade is performed.
Log Registry	The THOE provides a log registry. These logs are all traces generated in THOE for a specific system
MQTT broker	A MQTT broker runs in THOE so that any equipment can be connected to for getting vulnerability list and log information. Authentication/authorization will be required
THOE REST API	A Representational State Transfer (REST) API is an interface that supports sets of HTTP operations (methods), which provide a way to interact with THOE. By means of this interface, VeriDevOps tools or endpoints/devices can access THOE for checking current vulnerabilities, adding a new software asset to be scanned, modify vulnerability scheduler, and so on.

Table 2. Interfaces specific to THOE

6.4. Subordinates

Figure 10 and Table 3, which have been taken from the VeriDevOps D1.4 deliverable, describe the subordinate components used by THOE.

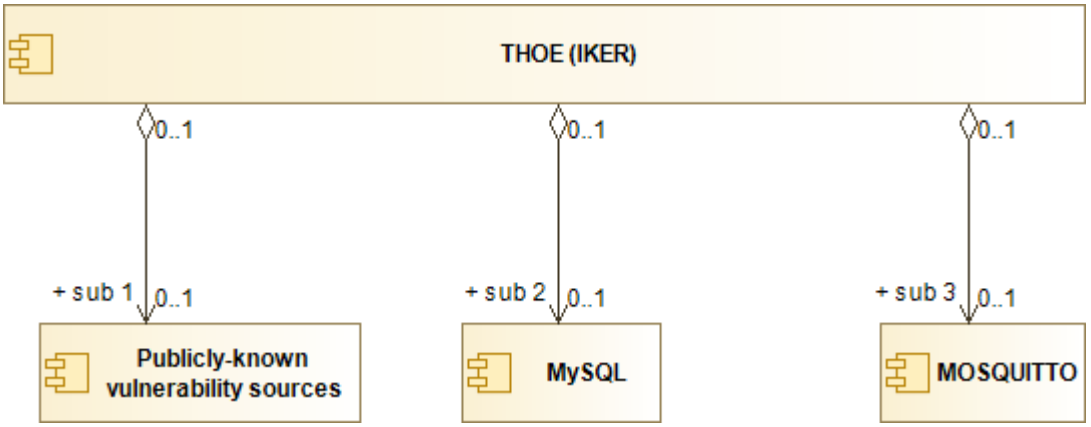

Figure 10. Subordinates

Name	Description
Publicly-known vulnerability sources	This sub-component searches inside of publicly known databases or sources for new vulnerabilities. NVD will be used as the main vulnerability source but other sources can be added.
MySQL	MySQL databases will be used for: - database for threat finder configuration - vulnerability database
MOSQUITTO	Mosquitto message broker that implements the MQTT protocol will be used. Mosquitto is lightweight and is suitable for use on all devices with limited and non-limited resources.

Table 3. Subordinates

7. Implementation status

THOE has been designed by IKERLAN, and it is currently in the development process. While some of the subsystems are fully implemented, others may be partially implemented, or not implemented at all yet.

In the Table 4 the implementation status of each subsystem is revisited. The implementation of the *Interoperability interface subsystem*, planned for M24, will enable a first integration of the tool in VeriDevOps.

Name	Release	Status	Notes
Scheduling subsystem	M32	**Working prototype**: a prototype which works with NVD database has been already implemented by IKERLAN	As THOE tool evolves, changes may be introduced.
Threatfinder subsystem	M15	**In progress**: a prototype of an agent has been implemented to gather vulnerability data from NVD. Agents for other sources pending to be implemented	As specific agents need to be implemented for each source, it will be in constant evolution.
Vulnerability Database subsystem	M15	**In progress**: a prototype has been implemented that stores data from NVD source.	Needs to adapt to different sources
Interoperability interface subsystem	M24	**Planned**: interoperability interfaces pending to be implemented	Interfaces will need to adapt to VeriDevOps use cases.
Logging subsystem	M32	**Working prototype**: a prototype has been already implemented	Future integrations may be added.

Table 4. Implementation status of subsystems

8. Case study applications

Case studies from both, ABB and FAGOR were presented in deliverable D1.1 Case Studies: Description, requirements analysis, and implementation. The THOE tool will be used in both cases for vulnerability identification.

8.1. FAGOR

In order to test the tools developed in the VeriDevOps context, FAGOR has deployed a test environment based on Azure infrastructure and containing two virtual machines with CentOS7, running FAGOR application (Figure 11). Regarding security monitoring, THOE will be used to identify vulnerabilities in FAGOR's test environment.

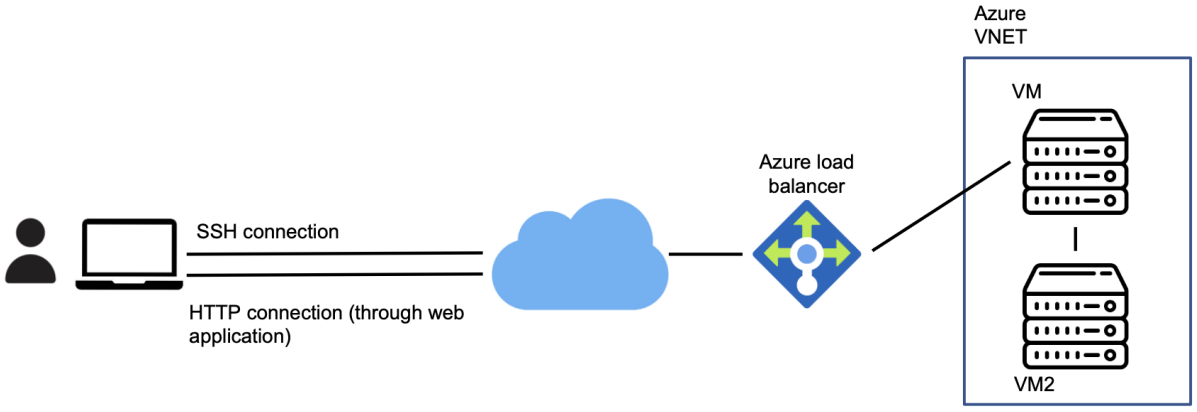

Figure 11. FAGOR test environment.

Additionally, FAGOR also aims to test THOE tool on it's real life Teleservice environment. The Teleservice environment is composed of several LInux and Windows servers, network devices such as firewalls and switches, and software used for remote assistance and VPN connections.

Since needs from customers may vary, and also their requirements for remote assistance may do, there is no unique Teleservice scenario. Along with customer's requirements, each one of the three different businesses from FAGOR may have their own ones. Depending on how a technician should connect to the customer's machine, one of four different scenarios can be applied. Some of the scenarios may share common resources, both network devices or servers and software, while others may be very specific for a customer or business. Currently, four different scenarios are contemplated in FAGOR's Teleservice

environment, which will be used in the context of WP3 Security Monitoring. Figure 12 depicts these scenarios. THOE will be tested in the whole Teleservice environment.

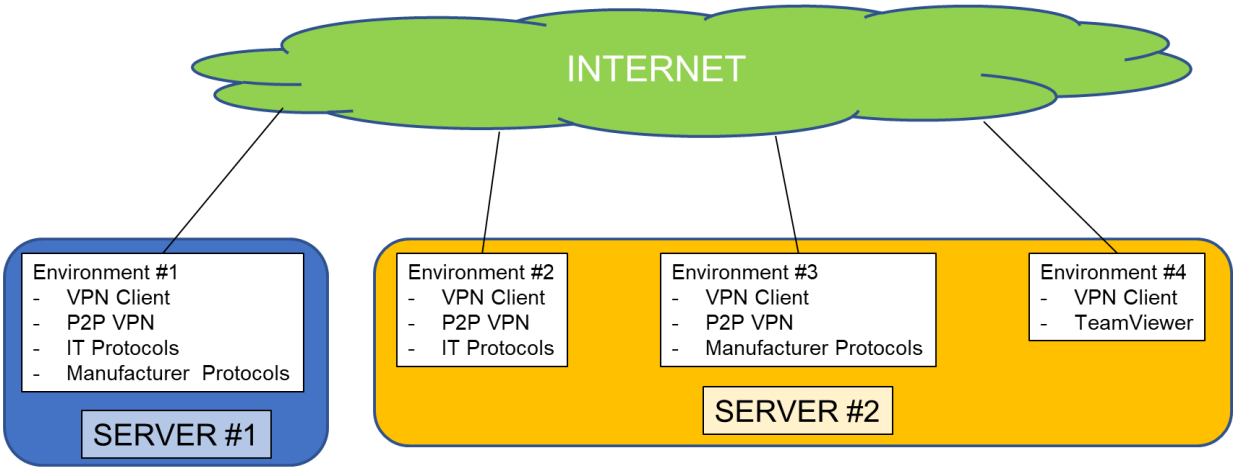

Figure 12. FAGOR Teleservice environment.

As stated in Section 6, the input for the THOE tool will be a CPE-based asset inventory. In this case, FAGOR will provision THOE with full asset inventory, while the output of the tool will be the full vulnerability list, including all vulnerability related data, that is CVE code, criticality, affected platforms and CVSS score for each of them.

Figure 13. THOE data flow.

The vulnerability database from THOE will be continuously updated with data from international vulnerability databases. Therefore, as soon as a new vulnerability is detected for a platform in the asset inventory, THOE will be able to notify about the newly detected vulnerability, allowing an early identification for taking further actions to secure the system.

8.2. ABB

One of the main expectations for ABB from the VeriDevOps project is to improve the way vulnerabilities are identified, security aspects are monitored and traces are analyzed, as stated in deliverable D1.1, previously mentioned. The requirement is to be able to handle bugs and vulnerabilities faster, instead of discovering these once the product is released.

ABB has defined multiple case study scenarios, one for each requirement or user story. THOE will be used in the case of study scenario ABB_S6: Known Anomaly and Vulnerability Identification presented in deliverable D1.1.

According to the use case scenario, when the crane is in the maintenance stage (in which security issues and software updates management processes are executed), the identification of known vulnerabilities in components and systems is needed, based on publicly known vulnerabilities in databases.

Vulnerabilities identification should cover not only control software, but also software tools for development and maintenance, as well as hardware and device configuration. The expectation in VeriDevOps is to be able to monitor these known vulnerabilities with the aim of ensuring security in operation.

In this sense, having the complete list of assets for which vulnerabilities need to be identified, thus, control software, tools for development and maintenance, as well as hardware and configurations, THOE will be able to provide an early and fast vulnerability identification, allowing ABB to ensure security in an effective way during operation.

9. Summary

The present document described the Threat Oracle Engine (THOE) tool that is being developed by IKERLAN in the context of the VeriDevOps project, and which is intended for vulnerability identification during the operation phase.

This deliverable presented an initial version of the tool, which is currently in the development process. The motivation for such a tool has been described, followed by some background context and requirement that it should fulfill. The document showed the design of the tool, as well as the implementation status of each building block. Finally, how the tool will be used by different actors in their use cases has been described.

Since this is still work in progress, it may suffer modifications, as well as improvements from what has been described in the present document. Modifications and improvements, as well as the description of the final version of the tool will be covered in the final version of this deliverable, that is D3.2 Threat oracle engine specification, design and implementation final version.

10. References

[1] O. Andreeva *et al.*, "Industrial control systems vulnerabilities statistics," *Kaspersky Lab, Report*, 2016 [Online]. Available: https://www.researchgate.net/profile/Sergey_Gordeychik/publication/337732465_INDUSTRIAL_CONTROL_SYSTEMS_VULNERABILITIES_STATISTICS/links/5de7842e92851c8364600e7e/INDUSTRIAL-CONTROL-SYSTEMS-VULNERABILITIES-STATISTICS.pdf

[2] M. Kravchik and A. Shabtai, "Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks," *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. 2018 [Online]. Available: http://dx.doi.org/10.1145/3264888.3264896

[3] K. Stouffer, J. Falco, and K. Scarfone, "Guide to Industrial Control Systems (ICS) Security - Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC)." 2011 [Online]. Available: http://dx.doi.org/10.6028/nist.sp.800.82

[4] "Communication network dependencies for ICS/SCADA Systems," 19-Dec-2016. [Online]. Available: https://www.enisa.europa.eu/publications/ics-scada-dependencies.

[5] S. Qayyum, S. Ashraf, M. Shafique, and S. Waheed, "Hardware devices security, their vulnerabilities and solutions," in *2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2018, pp. 439–445, doi: 10.1109/IBCAST.2018.8312261 [Online]. Available: http://dx.doi.org/10.1109/IBCAST.2018.8312261

[6] MITRE Corporation, "CWE - Common Weakness Enumeration." https://cwe.mitre.org/about/faq.html

[7] MITRE Corporation, "CVE - Common Vulnerabilities and Exposures. - Glossary" https://www.cve.org/ResourcesSupport/Glossary

[8] Brant A. Cheikes and David Waltermire and Karen Scarfone, "NIST Interagency Report 7695 - Common Platform Enumeration: Naming Specification Version 2.3," NIST interagency report, National Institute for Standards and Technology (NIST), Gaithersburg, Maryland, 2011.

[9] National Institute for Standards and Technology (NIST), "CPE - Common Platform Enumeration." https://nvd.nist.gov/products/cpe

[10] MITRE Corporation, "CVE - Common Vulnerability and Exposures." https://cve.mitre.org/index.html".

[11] A. Dimitriadis, J. L. Flores, B. Kulvatunyou, N. Ivezic, and I. Mavridis, "Ares: Automated risk estimation in smart sensor environments," Sensors, vol. 20, no. 16, 2020.

[12] National Institute for Standards and Technology (NIST), "National Vulnerability Database NVD — Vulnerability Metrics." https://nvd.nist.gov/vuln-metrics/cvss

[13] FIRST - Global Forum of Incident Response and Security Teams, “Common Vulnerability Scoring System (CVSS).” https://www.first.org/cvss/

[14] ICASI, “The Common Vulnerability Reporting Framework2”, https://www.icasi.org/cvrf/

[15] National Institute for Standards and Technology (NIST), “National Vulnerability Database NVD — Vulnerabilities.” https://nvd.nist.gov/vuln/full-listing

[16] “WPScan WordPress Security Scanner”, https://wpscan.com/wordpress-security-scanner

[17] Offensive Security, “Exploit Database”, https://www.exploit-db.com/

[18] “CVE Details - The ultimate security vulnerability datasource”, https://www.cvedetails.com/

[19] “VulDB Vulnerability Database”, https://vuldb.com

[20] Microsoft, “MSRC Security Update Guide”, https://msrc.microsoft.com/update-guide