Automated security analysis in a SCADA system

Akzharkyn Duisembiyeva

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, Finland August 13, 2020

Supervisor

Prof. Jan-Erik Ekberg and prof. Mathias Ekstedt

Advisor

Engla Ling

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Akzharkyn Duisembiyeva

Title Automated security analysis in a SCADA system

Degree programme School of Sciences

Major Security and Cloud Computing **Code of major** SCI3084

Supervisor Prof. Jan-Erik Ekberg and prof. Mathias Ekstedt

Advisor Engla Ling

Date August 13, 2020 **Number of pages** 112 **Language** English

Abstract

Supervisory control and data acquisition (SCADA) is a computer system for analysing, and monitoring data, as well as, controlling a plant in industries such as power grids, oil, gas refining, and water control. SCADA belongs to the category of critical systems that are needed to maintain the infrastructure of cities and households. Therefore, the security aspect of such a system has a significant role. The early SCADA systems were designed with the operation as the primary concern rather than security since they were a monolithic networked system without external access. However, the systems evolved, and SCADA systems were embedded with web technologies for users to monitor the data externally. These changes improved the efficiency of monitoring and productivity; however, this caused a problem of potential cyber-attacks to a SCADA system. One such example was Ukraine's power grid blackout in 2015. Therefore, it is beneficial for the security of a SCADA system to create a threat modeling technique that can understand the critical components of SCADA, discover potential threats, and propose possible mitigation strategies.

One issue when creating a threat model is the significant difference of SCADA from traditional Operational Technology (OT) systems. Another significant issue is that SCADA is a highly customisable system, and each SCADA instance can have different components. Therefore, for this work, we implemented a threat modeling language *scadaLang*, which is specific to the domain of a SCADA system. We started by defining the major assets of a SCADA system, attackers, entry surfaces, and built attacks and defense strategies. Then we developed a threat modeling domain-specific language *scadaLang* that can create a threat model for a particular instance of SCADA taking the differences in components and connections into account. As a result, we achieved a threat modeling language for SCADA, ensured the reliability of the results by peer-reviewing of an engineer familiar with the domain of the problem, and proposed a Turing test to ensure the validity of the result of *scadaLang* as the future development of the project.

Keywords threat modeling, DSL, MAL, cyber security, SCADA

Aalto University
School of Science

Aalto-universitetet, PB 11000, 00076 AALTO
www.aalto.fi
Sammandrag av diplomarbetet

Författare Akzharkyn Duisembiyeva

Titel Automatiserad säkerhetsanalys av SCADA system

Utbildningsprogram Säkerhet och molnberäkning

Huvudämne Säkerhet och molnberäkning **Huvudämnets kod** SCI3084

Övervakare Engla Ling

Handledare Prof. Jan-Erik Ekberg, Prof. Mathias Ekstedt

Datum August 13, 2020 **Sidantal** 112 **Språk** Engelska

Sammandrag

Supervisory control and data acquisition (SCADA) är ett datorsystem för att analysera och monitorera data samt kontrollera anläggningar för industrier såsom energisystem, olja, raffinering av gas och vatten. SCADA tillhör den kategori av kritiska system som krävs för att bibehålla städer och hushålls infrastruktur. Därför är säkerhetsaspekten av ett sådant system av stor roll. De tidiga SCADA systemen var utformade med funktionen som huvudsaklig oro istället för säkerheten då de var monolitiska nätverkssystem utan extern åtkomst. Systemen utvecklades emellertid och SCADA systemen blev inbyggda med webbteknologier så att användare kan monitorera data externt. De här förändringarna förbättrade effektiviteten av monitorering och produktivitet men skapade problemet med potentiella cyber-attacker mot SCADA systemen. Ett sådant exempel är Ukrainas energy systems elavbrott som skedde 2015. Därför är det fördelaktigt för säkerheten av SCADA systemen att skapa en hotmodelleringsteknik för att bättre förstå de kritiska komponenterna av SCADA, hitta potentiella hot och föreslå potentiella förmildrande strategier.

Ett problem för utvecklingen av en hotmodell är den stora skillnaden mellan SCADA från traditionella nätverkssystem inom industri. Ett annat stort problem är att SCADA är ett justerbart system och varje SCADA instans kan ha olika komponenter. Därför utvecklar vi i detta arbete ett språk för hotmodellering *scadaLang* som är specifikt för domänen SCADA system. Vi började med att definiera de huvudsakliga komponenterna av SCADA system, angriparna, attack ytorna och även bygga attacker samt försvarsstrategier. Sen utvecklade vi ett språk för hotmodelleringen som är domänspecifikt, *scadaLang* som kan skapa en hotmodell för en specifik instans av SCADA där skillnaderna på komponenter och sammankopplingar tas till hänsyn. Som resultat har vi skapat ett språk för hotmodellering för SCADA,verifierat resultat med hjälp av en ingenjör med domänkännedom och föreslagit ett Turing test för att förbättra verifieringen av resultatet som ett framtida arbete.

Nyckelord hotmodellering, DSL, MAL, cybersäkerhet, SCADA

Contents

List of Figures

List of Tables

Acronyms

AD Active Directory
API Application Program Interface
CC Control centre
CIA Confidentially, Integrity, Availability
CII Critical information infrastructure
Comm Communication
CVE Common Vulnerabilities Enumeration
CWE Common Weaknesses Enumeration
CVSS Common Vulnerability Scoring System
CNI Critical National Infrastructure
CDIO Conceive, Design, Implement, Operate
CDITO Conceive, Design, Implement, Test, Operate
DB Database
DBMS Database Management System
DE Data Engineering
DMZ Demilitarised Zone
DNP3 Distributed Network Protocol 3
DNS Domain Name Space
DoS Denial-of-Service
DDoS Distributed denial-of-service
DSL Domain-specific language
EPO ePolicy
ES Expert System
FAIR Factor Analysis of Information Risk
GUI Graphical User Interface
ICCP Inter-control communication protocol
ICS Industrial Control Systems
ICMP Inter Control Message Protocol
IDS Intrusion Detection Systems
IEC International Electrotechnical Commission
IED Intelligent Electronic Device
INSPIRE INcreasing Security and Protection through Infrastructure REsilience
IT Information Technology

IPS Intrusion Prevention System
HMI Human Machine Interface
nHMI new Human Machine Interface
KTH Royal Institute of Technology
KTMM KTH Threat Modeling Method
MAL Meta Attack Language
MiM Man-in-Middle attack
MAC Mandatory Access Control
NIS Network Information System
NIST National Institute of Standards and Technology
NGFW Next Generation Firewall
NTP Network Time Protocol
NVD National Vulnerability Database
ODBC Open Data Base Connectivity
OS Operating System
OT Operation Technology
PASTA Process for Attack Simulation and Threat Analysis
PBX Private Branch Exchange
PCN Process Control Network
PCS Process Control System
PLC Programmable Logic Controllers
ProdZone Production Zone
ROC Remote Operator Console
RDB Real-time database
RTU Remote Terminal Unit
SCADA Supervisory control and data acquisition
SIS Safety Instrumented Systems
SHARP Security-Hardened Attack Resistant Platform
SELinux Security-enhanced Linux
SQL Structured Query Language
SSO Site security officer
TCP Transmission control protocol
TTC Time-To-Compromise
UDP User datagram protocol

UML Unified Modeling Language

VIKING Vital Infrastructure, Networks, Information and Control Systems Management

WAN Wide Area Network

Glossary

Admin accounts type of account with high privileges. 28

Adversary The malicious actor. 64

ANSI/ISA a compliance institute that offers security standards for organisations. 14

CIA properties of the system that relate to the confidentiality, integrity and availability of communication between entities of this system. 37

CISA: Industrial Control Systems US governmental standard for homeland security. 8, 58, 87

CNI (the UK), a term used by governments to group the fields of industries, sectors and assets that are necessary for the economy, daily life of society and security. 6

Domain-specific language The language designed to solve the specific set of problems. 18

ExploitDB a vulnerability database for collecting and maintaining information about discovered computer security vulnerabilities. 46

IEC 60870-5-104 IEC standard used for control in electrical engineering and power system automation applications.. 64

IEC 622443-1 IEC standard used for the security of Industrial Control System (ICS) networks. 14

IEC 62254-1 IEC standard used for the manufacturing operations management domain (reference model for computer integrated manufacturing). 14

KTH threat modeling method Adaptation of the PASTA and FAIR methodologies with the following phases: 0. Scope and Delimitations; 1. Business Analysis; 2. System Definition and Decomposition; 3. Threat Analysis; 4. Attack and Resilience Analysis; 5. Risk Assessment and Recommendations.. 28

Mimikatz Windows tool for post-exploitation and dumping passwords memory, PINs and Kerberos tickets. 48, 52

MITRE non-governmental organisation that conducts security research. 7, 8, 25, 26, 33, 37, 39, 41, 43, 45, 46, 50, 51, 54, 58, 60, 71, 72, 75, 84, 85, 87

Modbus a communication protocol originally designed for programmable logic controllers, and nowadays commonly available for connection of industrial electronic devices. 13

OCTAVE Operationally Critical Threat, Asset, and Vulnerability Evaluation, a threat modeling technique. 15

Sandia National Laboratories Part of the PCS security project of the Institute for Information Infrastructure Protection (http://thei3p.org). 14

SCADA is a computer system for analysing, monitoring data and controlling a plant in industries as electricity, oil, gas refining and water control.. 6

Service accounts type of non-human account dedicated for automation of services. These accounts are more privileged than a user, but do not have admin privileges. 28

STRIDE threat modeling framework proposed by Microsoft. 15, 16

Trust boundary an abstract logical zones that is different from another trust boundary due to access levels. 55

User accounts type of account that offers least privilege level and allows to have a limited read permissions only. 28

1 Introduction

Supervisory control and data acquisition (SCADA) is a computer system for data monitoring and analysis, as well as controlling a plant in industries such as power grids, oil, gas refining, and water control. SCADA is an essential core of Process Control System (PCS) [1], a key component of Critical National Infrastructure (CNI) [2], and have a signification role in daily life of Swedish population [3]. In early SCADA systems, there was no importance assigned to cyber-security, and, therefore, the topic of security analysis of SCADA systems is relatively new [4]. Section 1.1 provides a short introduction about SCADA and its security aspect. This thesis assesses the security of SCADA and proposes a tool to investigate potential attacks and vulnerabilities. Section 1.2 provides a problem statement for the work, section 1.3 describes the goals, while section 1.4 showcases the research questions in this work. Section 1.5 discusses the ethical concerns and issues related to sustainability. In section 1.6, we discuss the scope of the project and its delimitation. Section 1.7 navigates through the further chapters of the report.

1.1 Motivation

The Ministry of Justice of Sweden [3] states that the electric power industry, which utilizes systems like SCADA, has a major impact on national well-being. The security topic of SCADA has national importance, which was stressed by the Protective Security Act (SOU 2015:25). Early SCADA was a standalone, isolated system where cyber-security attacks were less priority than operation security. The significant change in the security of the SCADA system happened from the 2000s when these systems became embedded with web technologies for efficient monitoring [5]. These drastic shifts lead to a host of desirable advantages like the increase of data inter-exchange and easy management of production capabilities [5]. However, these drastic shifts also lead to disadvantages, the most important of which was the rise of cyber-attacks.

According to Ahmed et al. [6], the architecture of SCADA systems drastically evolved compared to their earlier predecessors. The early-stage SCADA systems consisted of input/output devices, and the signals were transmitted between the master and remote units. Later, the communication between units became wireless and relied on IP. The key change occurred when the internal network and external components started to integrate, allowing the SCADA to connect with the corporate intranet. These internal components had direct access to the Internet, making the previously closed system available for connecting to the outside world, which led to potential threats.

One popular approach towards mitigating potential security vulnerabilities is a threat modeling. Threat modeling is a risk assessment technique that identifies potential threats to the system. Threat modeling consists of several phases. First of all, we assess the system and identify assets that are components of the system and actors who interact with the system. The next phase is to define potential attackers and their capabilities. This phase is followed by building attack graphs to identify

potential threats for each attacker. Finally, a threat model proposes a protection strategy and generates a risk assessment for the system. Creating threat models can be time-consuming, as the attack graphs grow larger and become more complex. To mitigate this drawback, we can use Domain-specific language (DSL) for attack simulations. Domain-specific language (DSL) for attack simulations provides a way to automate the generation of attack graphs. One such language is Meta Attack Language (MAL) [7].

In this work, we extend MAL to build a threat modeling language for SCADA, and create a tool that can generate a threat model for an instance of SCADA using our language.

1.2 Problem Statement

As described in the section 1.1, the attacks on SCADA increased drastically in recent years. Moreover, SCADA is a highly customisable system and can have different setups. It makes it difficult to create a generalised framework for all instances of the same SCADA system. How can we assess the security of SCADA? We can find important information in configuration files. However, to our best knowledge, no threat modeling tool can make use of this information. Can we utilize important information in configuration files to build a threat modeling tool for a generalised SCADA system?

1.3 Goals of the Thesis Work

This project aims to assess the security of a SCADA system, and develop a Domain-specific language (DSL), which we can use to generate a threat model for a particular instance of SCADA. We can achieve our goal with two objectives. The first objective is to develop a domain-specific threat modeling language specification based on MAL that would fit the SCADA architecture. For this, we make use of state-of-the-art methodologies for threat modeling (Factor Analysis of Information Risk (FAIR), KTMM, MITRE), which we thoroughly explore in chapters 3 and 4. The second objective is creating a threat model for a SCADA instance using the developed language. For that, we get the configuration files of a particular SCADA system. Based on information in the files, we generate a threat model for this instance.

1.4 Research Question

The project focuses on building the threat modeling language for SCADA. In this work, we ask the following research questions:

- How can we build a DSL for threat modeling specifically fit for SCADA? Can we utilise state-of-the-art methodologies for threat modeling (FAIR, MITRE) of SCADA?
- How can we apply this language for generating a threat modeling tool for a particular SCADA instance using configuration files? Do these configuration

files provide sufficient data to successfully model and assess the security of a SCADA system?

1.5 Ethics and Sustainability

The general purpose of this work is to provide a security assessment for a SCADA system using available data from configuration files and open-source security knowledge bases. We take all the data for attack steps from the open-source MITRE knowledge base. The general assets for the work correlate with assets described in MITRE knowledge base. There was no fabrication while taking the images of attack steps or reporting of results. The report and the degree work do not have plagiarised content. All the materials in this work are properly cited. The work does not leak any confidential information about a SCADA system.

The work output cannot be exploited with bad intent by attackers. Threat modeling provides the analysis of the system and the theoretical description of potential attack steps. At first glance, these attack steps could leak information on how the system could be exploited. However, the work uses attacks from open-source MITRE, which is publicly accessible. If the attacker prefers so, he or she can use the attack techniques from the MITRE knowledge base directly as a guideline. In addition to that, the names of assets and associations that were described in this work are not unique terms for a particular company, but rather an obfuscation. The protection mechanisms are also taken from MITRE knowledge base, or CISA: Industrial Control Systems, in case the protection mechanism are not available in MITRE. All these measures solve the problem of using this work as a directive of breaking into the SCADA system.

On the economic and sustainability front, we made the best effort to reduce time and computation while creating a threat model. In other words, time and complexity required for proposing a threat model. For example, our proposed solution's main idea is to generate a threat model for a particular SCADA instance considering the proper assets. The instance might have a different setup from another instance, and, therefore, we propose attack steps that match only the needed asset. This avoids building unnecessary and redundant attack steps that do not possess a value for a threat model.

Our work contributes to making the SCADA system more secure and sustainable. The insecure SCADA systems could be categorised as unsustainable systems in terms of operation and environmental impact. SCADA suffers from inefficient patch delivery, time-consuming maintenance, and expensive hardware and software repair [8]. In addition to that, the system is geographically distributed with weak protection, and a successful exploit on one of the sites could spread on others [9]. Other issues connected with the impact of security breaches is the environmental pollution caused by SCADA incidents [10]. Examples of such breaches took place in Siberian Pipeline (1982), Bellingham, WA Gas Pipeline (1999), Maroochy Water System(2000), and others [10]. Contributing to the security of SCADA would benefit the sustainability of this system in terms of operation and ecology.

1.6 Delimitation

The work has the following delimitation due to time constraints. The following is not in the focus of this work:

- This work does not consider attacks against safety systems and protection against manual misconfiguration.
- This work does not consider any attacks toward field units. We consider field units to be an entry surface in this work.
- This work proposes a Turing test for enhancing the validation of results but does not implement it. Turing test for enhancing of the validation of results is a future extension of work.

1.7 Structure of the Report

Chapter 2 provides the background information to make a reader familiar with the notions used in this work. Chapter 3 gives information about the analysis of collected data and methodology. Chapter 4 presents *"scadaLang"*, DSL for threat modeling based on MAL for a general threat modeling for SCADA. Chapter 5 presents a threat model for a particular instance of SCADA using proposed *scadaLang*. Chapter 6 presents the results and findings. In chapter 7 we discuss our work and state limitations and future work. Chapter 8 concludes the work and summarizes our findings.

2 Background

This chapter gives the necessary background information for the reader to get familiar with the main notion and terms used in this work. Section 2.1 gives information about SCADA systems, security issues in SCADA, communication media in SCADA and the notion of zones. Section 2.2 explains the important definitions for threat modeling that are used for this work, as well as gives information about Common Vulnerability Scoring System (CVSS), a security metric used in threat modeling for describing criticality of an attack step. Section 2.3 dives into the MAL language framework. Section 2.4 provides information about previous work in this topic.

2.1 SCADA Overview

In this section we explain what SCADA means. We discuss the architecture of SCADA and security issues in this system.

2.1.1 SCADA architecture overview

Modern SCADA systems have a complex heterogeneous architecture that consists of geographically distributed components [2]. Figure 2.1 demonstrates the simplified version of how the SCADA works. In general, a Control centre (CC) and field sites comprise a SCADA system.

Control centre (CC) is the hub of the system consisting of the following units: HMI, database management system (Historian), and SCADA server. Based on the configuration, the system can also have the load sharing server, Inter-control communication protocol (ICCP) server, backup servers, replicas, antivirus servers, directory servers, workstations, and other. CC consists of a primary site and backup sites. The main functionality of CC is planning, monitoring, and control.

CC is connected with the Corporate Network (Company Support Zone), and the communication can be established through a network connection. Older versions of SCADA used Private Branch Exchange (PBX) for this communication, while modern SCADA uses internet connections between Corporate Network and CC. This connection is needed in SCADA to ensure the constant company support for CC.

On the other hand, field sites are connected with CC via satellite, radio/microwave/cellular networks or Wide Area Network (WAN), and are geographically distributed in various locations. Field sites are equipped with Programmable Logic Controllers (PLC) or Remote Terminal Unit (RTU) that are needed to control the state of the system on-site. Field sites also send regular health checks to CC. Communication Frontend servers accept the data from all field devices. Human operators can access this data through Human Machine Interface (HMI), and Historian archives it.

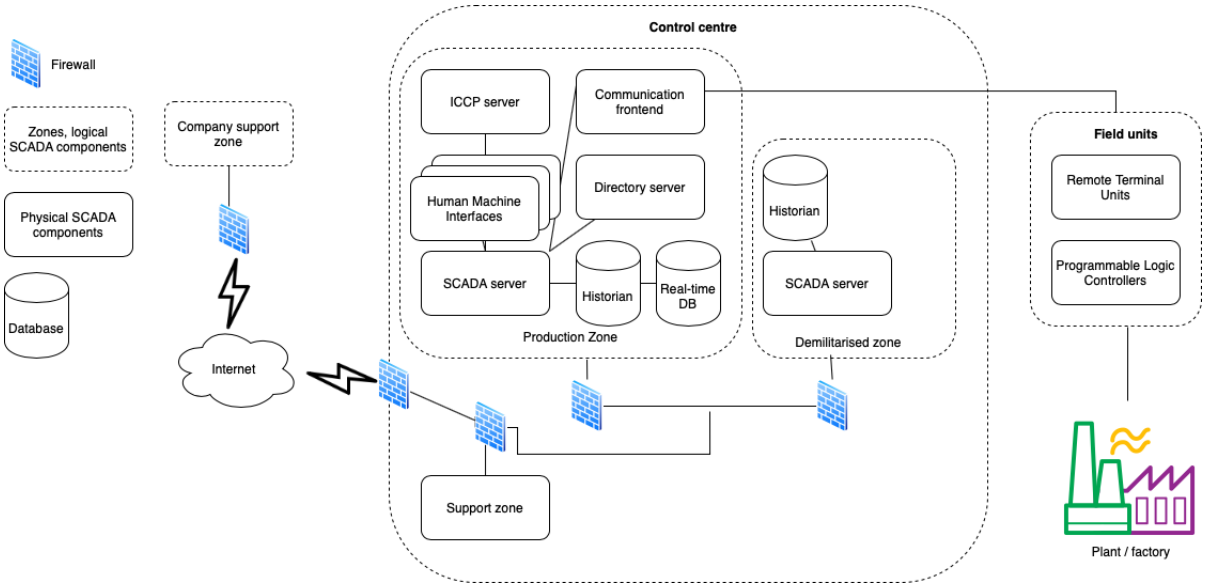

Figure 2.1: A typical SCADA Architecture in a simplified logical view [6]

2.1.2 SCADA security assessment

The early SCADA systems did not consider cyber-security attacks as a major concern due to its standalone nature [11]. The priority was given to operational security. Fernandez et al. [1] mentions that PCS requirements were the main requirements that SCADA needed to fulfil during the early stage. The assessment of the security of the system included only the physical security of components and networks. However, the situation changed in the last decade, since the early security standard became outdated [2]. Therefore, the SCADA security standards needed to be reworked, and National Institute of Standards and Technology (NIST) published the risk assessment and security guide in 2014 in a document *"System Protection Profile Industrial Control Systems"*. SCADA systems control critical industrial systems, and, therefore, serve as a fascinating target for attackers [12]. One of the most well-known examples is the *Stuxnet* virus that was designed to harm automated systems, such as SCADA, in 2010 [10]-2012 [13]. As a result, the Stuxnet virus affected 50,000 - 100,000 computers worldwide. The example of Stuxnet inspired attackers to create more powerful and sophisticated attacks, such as *"Flame"*, which is considered an espionage tool, and *"Duqu"* which used the same techniques as Stuxnet [10].

According to Schneider, Obermeier, and Schlegel [12], the security in SCADA systems differ from traditional Operation Technology (OT) system in some ways. First of all, some systems have unprotected legacy parts that are of particular interest. Secondly, the main requirement for these systems is constant availability. This constant availability requirement is related to the fact that this system is connected to physical systems supported by a large number of sensors. These sensors also serve as an attack surface. Thirdly, unlike the OT systems, the SCADA system cannot be turned off, which, in the case of forensics, complicates the data acquisition and analysis [6]. Ahmed et al. [6] mentioned the following potential attacks in

SCADA:

- *Security Component Malfunctioning*
- *Insider Attacks*
- *Unauthorized Access*
- *Technology Improvement as a threat*
- *Shifting of Trust Boundaries*

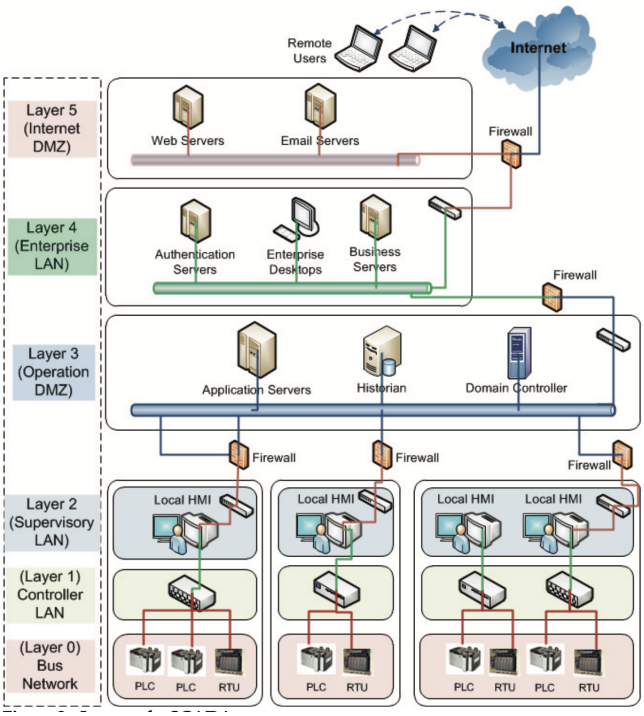

Figure 2.2: Layers of a SCADA system. Figure taken from [6]

Figure 2.2 introduces the hierarchical concept of *layers*, which is a convenient way to group the components of SCADA based on their purpose, location and functionality. The *regulatory levels* handle process control [14], while *supervisory control layers* are responsible for the diagnosis of the system [15].

According to Men et al. [16], HMI is the most straightforward way to get into the system. The first way to achieve it is social engineering or phishing of a human operator. The second way is to obtain the administrative privileges on the Communication frontend that communicates with remote units on the field. Obtaining administrative access to Communication frontend can lead to sending incorrect data to a CC and all connected RTUs. Thirdly, there is a threat connected with the external SCADA system. Taking over the Inter-control communication protocol (ICCP) would lead to gaining access to a CC. The potential threats involve internal SCADA's trust boundary as well. The key component of the system is a

Directory Service. Directory Service is responsible for authentication, and in case of obtaining the administrative right on Directory Service, there is a great threat to the whole system.

2.1.3 SCADA communication protocols

When it comes to describing the communication means in SCADA, it is necessary to pay attention to the fact that the choice of communication means or protocol depends on the noise and speed of data transmission [17].

Currently, the most prevalent protocols include Modbus [1] and Distributed Network Protocol 3 (DNP3) [17] that is used to communicate with PLC. Table 2.1 presents some of the protocols in a SCADA system.

Protocol	Description
Modbus	A widely used, industrial control protocol which relies on a simple request/reply procedure between a CC and field devices. There are two main variants Serial and TCP [18].
DNP3	The primary SCADA protocol used in the electrical power grid [19]. The protocol supports three main communication models between a CC and field devices, unicast, broadcast, and a mode for unsolicited responses from field devices [20].
ICCP	The standard CC-to-CC communication protocol. The protocol can run on top a range of transport layer protocols, but it is often used on top of TCP/IP to establish a point to point connection between to CCs [21].
IEC 60870-5-101 / 104	Communication between components of a SCADA system relies on IEC 60870-5-101 and IEC 60870-5-104 international standards [22]. These standards were designed by International Electrotechnical Commission (IEC), and enable message transmission between CC and remote field units. These standards also include remote control protocols IEC 60870-5-101 and IEC 60870-5-104 which use dedicated optical fibers, digital radio links or mobile networks [23]. These protocols define a communication between CC to field units.

Table 2.1: SCADA protocols overview.

2.1.4 Security zones in SCADA

Mahan et al. [24] defines *security zone* as "a collection of information systems connected by one, or more, internal networks under the control of a single authority and one security policy". The important features of security zones or *zones* are as follows [24]:

- Zones should separate the critical assets from common ones;
- Assets under the same zone have the same zone policy;
- The boundary of the zones should be accurate, and the communication between the assets in this zone to other external assets should be filtered;
- Any services, application and protocols that are carrying information from the zone to outside environment should be blocked;

The number of zones vary based on necessity of the system, however, security standards (ANSI/ISA, IEC 622443-1, IEC 62254-1) and Sandia National Laboratories [25] propose the generalised zones. Table 2.2 groups these zones together.

IEC 622443-1, IEC 62254-1	**ANSI/ISA**	**Sandia National Laboratories**
Automation Zone (process, safety and protection, basic control/local control)	Enterprise (corporate) Zone	Corporate Zone
Operation Control (supervisory control) Zone	CC Zone (primary and backup CC)	Process Control Zone
Operation support (operation management) Zone	DMZ (Historian and ROC)	DMZ
Business support Zone (Business planning and logistics)	Site Control Zone (local operator and engineering workstations, servers, and SCADA devices)	
Corporate (enterprise and common services) IT zone		
External Integration Zone (third-party services)		

Table 2.2: Generalised zones in SCADA

Figure 2.2 demonstrate the example of zoning of SCADA system. DMZ is also referred as *"screened subnet"*. This zone is located logically between two logical networks. The purpose of DMZ is being a shield for an internal protected network with releasing restricted access to data in the internal zone for external sources. The

recommendation by [24] states that the system should consist of external and internal DMZ.

External DMZ should provide a public access to external-facing servers without any traffic between servers inside DMZ as depicted in Figure 2.2.

Enterprise Zone is also termed as “Corporate Support Zone" to support SCADA remotely for customers.

Secure Production Zone (ProdZone) has the highest priority, and it is responsible for processes handling. This zone is also termed as Process Control Network (PCN). Byres, Karsch, and Carter [26] mentions that this zone should be isolated from both Corporate (Enterprise) Zone and Internet systems through the firewalls.

2.2 Threat Modeling

In this section, we discuss the concept of threat modeling and provide the essential definitions that we use in this report.

2.2.1 Taxonomy of threat modeling techniques

UcedaVelez and Morana [27] shows that the majority of attacks took place when the system was unintentionally designed with flaws and security bugs or when security was not prioritised. This correlates with the issues at SCADA as described in section 2.1.2. Therefore, the notion of *“threat modeling"* becomes important.

Definition 2.1. Threat modeling - “a strategic process aimed at considering strategic attack scenarios and vulnerabilities within a proposed or existing application environment to identify risk and impact levels". The application environment refers to the object of the threat modeling process [27].

Threat modeling is a risk assessment technique that identifies potentials threats to the system. These threats could be resolved while developing the system. Examples of threat modeling approach are STRIDE developed by Microsoft [28], OCTAVE created by CERT Division of the SEI [29], Factor Analysis of Information Risk (FAIR) [30] or Process for Attack Simulation and Threat Analysis (PASTA) introduced by UcedaVelez and Morana [27]. Royal Institute of Technology (KTH) also developed a threat modeling technique KTMM [31]. KTMM is a combination of FAIR and PASTA. In summary, the core of threat modeling techniques as STRIDE and PASTA is identifying the main assets (described in definition 2.2), the attacker (described in definition 2.3), actors of the system and their roles (administrative with higher privilege or user with lower privilege), the data flow and architecture of the system. After analysing this information in this work, we can develop an attack tree (described in definition 2.4). Finally, knowing the potential vulnerabilities and issues, we can propose the risk assessment (described in definition 2.5), mitigation strategy, and how it can reduce the damage or possibility of an attack.

Definition 2.2. Asset - in general, can relate to “component of the system, function or process, data" which value can be “associated with business critically" [27].

The assets can be divided into groups or categories based on different factors. For the categorisation of assets in this paper, we selected the KTH Threat Modeling Method (KTMM) categorisation of assets. KTMM has convenient and simple asset types. Below are categories of the assets that we use for this paper:

- *Function* - this category includes components “that feature some kind of behaviour".
  - *Services* - this subcategory includes applications and features that face users (actors).
  - *Platforms* - this subcategory includes non-end user-facing applications and software, unlike services subcategory. These assets build an infrastructure for services, e.g., Operating System (OS), middleware, Database Management System (DBMS), and servers.
  - *Hardware* - this subcategory includes all physical assets that serve as deployment infrastructure for platforms and services, e.g., computers, work stations.
- *Data* - this category includes assets that constitute some information in the system, e.g., log files, credentials, configuration data.
- *Networks* - these assets include communication media that connect functions and transmit data, e.g., Ethernet, firewalls, routers.

Definition 2.3. Attacker (adversary) - a malicious actor with the intent to disrupt the Confidentially, Integrity, Availability (CIA) properties of the system.

Attackers use *vulnerabilities* to exploit the system. Attackers can differ based on their motivation, skills, and resources. Script kiddies have beginner skills with a motivation to learn and limited resources. They do not impose a critical risk to the SCADA system. On the contrary, cyber-criminals and government-sponsored groups of attackers have high motivation, skills, financial resources, time, and equipment. Therefore, these types of attackers are a serious threat to the SCADA system. As for SCADA, the attacks from the second category have a higher possibility of occurrence based on previous attacks described in 2.1.2. As an example, *Sandworm Team* is a cyber-espionage group of attackers that is related to the Ukrainian energy sector attack (2015) [32].

Definition 2.4. Attack tree - a threat modeling technique popularised by Bruce Schneider in 1999 [33] that places the attacks in a tree form where the root is a goal of an attack and leaves are the starting point to achieve it [33].

The nodes in a tree are divided into *OR* and *AND* nodes. When the node is marked with *OR* label, any attack branch of this node could be used (e.g., cheapest attack branch). In the case of *AND* label of the node, all the attack branches must be fulfilled [33]. Other risk assessment techniques adopted the method of building attack trees as STRIDE and PASTA.

Definition 2.5. Risk assessment - "is a process of identifying the risks to system security and determining the probability of occurrence, the resulting impact, and additional safeguards that would mitigate this impact" [34]. PASTA model is integrated with a generic risk assessment process as NIST Risk Assessment Methodology in Special Publication 800-30.

After building a risk assessment based on potential attacks, we can suggest mitigation for the particular issue.

2.2.2 CVSS score

CVSS is a system that allows us to assign numeric values of severity and impact of a vulnerability based on calculations by researchers [35]. The severity level of attacks is divided into low, mid, high, and critical impact levels by Stouffer, Falco, and Scarfone [36]. This work focuses on version 3 of the CVSS scoring model.

Rating	**CVSS Score**	**Example**
None	0.0	No impact
Low	0.1-3.9	Insignificant impact on business or operation
Medium	4.0-6.9	Limited access to key components of the system
High	7.0-8.9	Significant data loss or downtime of the system
Critical	9.0-10.0	Root-level compromise of key components of the system or infrastructure devices

Table 2.3: Severity levels in CVSS v3 system [37]

The CVSS v3 model uses three categories of metrics: *base*, *temporal* and *environmental*. Base metrics are the characteristics of a vulnerability that are constant, regardless of time or user environments. On the other hand, temporal metrics take into account properties of a vulnerability that might change over time, like patches. Environmental metrics gives the option to customise the score based on desired user environment factors. For this work, we only take into account the Base metrics. Table 2.3 represents the severity scores in Base metrics in CVSS v3.

ID	Metric	Description	Values
AV	Attack Vector	Reflects how remote the attacker can be to deliver the attack against the vulnerable component. The more remote, the higher the score.	Not Defined(X), Physical(P), Local(L), Adjacent(A), Network(N).
AC	Attack Complexity	Reflects the existence of conditions that are beyond the attacker's control for the attack to be successful.	Not Defined(X), High, Low(L).
PR	Privileges Required	Reflects the privileges the attacker need have on the vulnerable system to exploit the vulnerable component.	Not Defined(X), High(H), Low(L), None(N).
UI	User Interaction	Reflects the need for user interaction to deliver a successful attack.	Not Defined(X), Required(R), None(N).
S	Scope	Reflects when the vulnerability can affect resources beyond the authorization privileges intended by the vulnerable component.	Not Defined(X), Unchanged(U), Changed(C).
C	Confidentiality	Measures the impact to the confidentiality of information stored on the impacted system.	Not Defined(X), None(N), Low(L), High(H).
I	Integrity	Measures the impact to the integrity of information stored on the impacted system.	Not Defined(X), None(N), Low(L), High(H).
A	Availability	Measures the impact to the availability of the impacted component.	Not Defined(X), None(N), Low(L), High(H).

Figure 2.3: Summary of CVSS v3 Base Metrics. Figure taken from [38]

2.3 Meta Attack Language (MAL)

In this section, we discuss the Meta Attack Language (MAL) developed by KTH, give the explanation of the language and probability distribution at MAL.

2.3.1 MAL Syntax and Overview

MAL [7] is a type of Domain-specific language that provides the syntax and compiler for building further language specification for a problem designed by KTH. In this work, we discover whether this technique can suit threat modeling of SCADA. MAL language utilises the terms *"Asset"*, *"Attacker"*, *"Attack"*, *"Association"*, *"Category"*, and *"Defense"* described in previous section 2.2.1.

Below we can see the sample code[1] for creating a MAL specification:

```
category System {
  asset Computer {
     let allFolders = folder.subFolder*

     | connect
        -> access
     E firewallExists
        <- firewall
        -> firewall.bypass
     E! noFirewall
        <- firewall
        -> firewallBypassed
     | firewallBypassed @hidden
        -> access
     | vulnerability
        -> compromise
     & access
        -> compromise
  & compromise
        ->  allFolders().accessFolder
           // Let substitution
```

[1] https://github.com/mal-lang/mal-documentation/wiki/MAL-Code-Examples. Accessed on 29.06.2020

```
    }
    asset Folder {
        | accessFolder
            -> stealSecrets
        | stealSecrets
    }
}
category Security {
    asset Firewall {
        & bypass [Bernoulli(0.2)]
            -> computer.firewallBypassed
        # hardened
            -> bypass
    }
}
associations {
    Computer [computers] * <-- Protect --> 0..1 [firewall] Firewall
    Computer [computer] * <-- Contains --> * [folder] Folder
    Folder [folder] 1 <-- Contains --> * [subFolder] Folder
}
```

Asset are mainly zones, communication media, database records, accounts, and servers in the SCADA system. MAL utilises the *Abstract Assets* type, which, in contrary to standard *Asset* cannot be instantiated. In addition to that, MAL enables the inheritance between parent and child assets, which helps avoid multiple definitions of the same attack steps in case a child class extends the base functionality of a parent asset. The code below exemplifies the inheritance between assets in MAL[2].

```
asset Parent
    {
    [parent logic]
    }

asset Child extends Parent
    {
        [parent + child logic]
    }

asset OperatingSystem
    {
        [OperatingSystem logic]
    }

asset Linux extends OperatingSystem
    {
        [OperatingSystem + Linux logic]
    }
```

Attacks are the ways to access the assets. For the work, one of the ways to get these attacks are using the adversarial attack model at MITRE Industrial Control Systems (ICS) knowledge base. MAL files contain the attack steps. Each step is either logical conjunction (OR in attack trees in 2.4, | in MAL) or disjunction of other steps (AND in attack trees in 2.4, & in MAL). One attack step can lead to another attack step, which is denoted as -> in MAL. In other words, to reach the attack step *"stealSecrets"* in asset *"Folder"*, we need to perform *"accessFolder"* attack step first.

```
asset Folder {
        | accessFolder
            -> stealSecrets
        | stealSecrets
    }
```

Association is MAL defines the connection between two assets that can lead to an attack from one asset to another. In other words, the association provides information about how one sensitive asset could be reached from another.

[2]The code was slightly modified because of a typo in the original resource. The authors were notified about the error.

```
associations {
 Computer [computers] * <-- Protect --> 1 [firewall] Firewall
 Computer [computer] * <-- Contains --> * [folder] Folder
 Folder [folder] 1 <-- Contains --> * [subFolder] Folder
}
```

In the example above, we can observe associations between two assets *Computer* and *Firewall*, *Computer* and *Folder*, *Folder* and *Folder*. The associations are called *Protect* and *Contains*. MAL language supports following *relations* between assets (denoted as *multiplicities in associations* in MAL):

- Many-to-Many (e.g., * <– Contains –> *),
- One-to-Many (e.g., 1 <– Contains –> *),
- Many-to-One (e.g., * <– Contains –> 1),
- One-to-One (e.g., 1 <– Contains –> 1),

A *Firewall* can be connected to zero or more assets *Computer* through the role *computers* (`[computers]`) and a *Computer* can be connected to one *Firewall* through the role *firewall*. The roles are denoted as [role], or in this case (`[computers]`) or (`[firewall]`). In MAL this is known as a *role*.

Category in MAL is a group of assets that are connected based on the security properties and rules. In the aforementioned example we can see two categories `"System"` and `"Security"`. MAL language allows us to choose names for categories and grouping logic based on the developer's judgment.

Defense in MAL is the mitigation strategy that helps to fight against the potential attack. Defense against a particular attack step is denoted as # in MAL.

```
category Security {
   asset Firewall {
      & bypass [Bernoulli(0.2)]
         -> computer.firewallBypassed
      # hardened
         -> bypass
   }
}
```

2.3.2 Testing at MAL

MAL language supports unit testing that can be used to test attack steps and specific functions of the language. In order to run tests, a developer needs to use the *JUnit* library of Java.

Let's consider the following assets with attack steps:

```
asset Network {
   | access
     -> hosts.connect
 }

 asset Host {
   | connect
     -> access
   | authenticate
     -> access
   | guessPassword
     -> guessedPassword
   | guessedPassword [Exponential(0.02)]
     -> authenticate
   & access
 }
```

In order to write a unit test we need to instantiate a model of *scadaLang* with two test objects of *Host* and *Network*. If two assets have an association, MAL generates an `add` function between two objects. This function is written as `<asset1>.add<Role2>`, where *asset1* is a name of the first asset, which is a *Network* in our case, and *Role2* is name of the role in a association with the capitalised first letter [39].

After creating assets and associations, we can initiate an attacker and provide an entry point as an initial attack step. After that, we can test whether we can expect to reach another attack step and how difficult the attack step was. MAL provides several assertion methods for the test. Table 2.4 describes these methods[3]. In our sample scenario, an attacker needs to spend some effort to access a server object of type *Host*, starting from accessing a *Network*.

```
public class ExampleLangTest {
  @AfterEach
  public void deleteModel() {
    Asset.allAssets.clear();
    AttackStep.allAttackSteps.clear();
    Defense.allDefenses.clear();
  }
}

public class TestGuessPassword extends ExampleLangTest {
  private static class GuessPasswordModel {
    public final Network internet = new Network("internet");
    public final Host server = new Host("server");

    public GuessPasswordModel() {
      internet.addHosts(server);
    }
  }

  @Test
  public void testGuessPassword() {
    var model = new GuessPasswordModel();

    var attacker = new Attacker();
    attacker.addAttackPoint(model.internet.access);
    attacker.addAttackPoint(model.server.guessPassword);
    attacker.attack();

    model.server.access.assertCompromisedWithEffort();
  }
}
```

[3]Table from original source has some typos. The table in this work has some modifications compared to original source. Authors were notified about these errors.

Assertion method	Description
assertUncompromised()	Check for unsuccessful compromise of the attack step.
assertUncompromisedFrom(<parent>)	Check for unsuccessful compromise of the attack step from the specified parent attack step.
assertCompromisedInstantaneously()	Check for successful and immediate compromise of the attack step.
assertCompromisedWithEffort()	Check for successful compromise of the attack step but only after some effort/-time is spent.
assertCompromisedInstantaneouslyFrom(<parent>)	Same as assertCompromisedInstantaneously from specified parent attack step.
assertCompromisedWithEffortFrom(<parent>)	Same as assertCompromisedWithEffort from specified parent attack step.

Table 2.4: Assertion methods for writing tests in MAL [39]

2.3.3 Probabilistic model of MAL

MAL uses a probabilistic model to estimate Time-To-Compromise (TTC) for each attack step. TTC is used to measure the effort for an attacker to perform an attack. In MAL, TTC is evaluated in the number of days that attacker needs to perform an attack step [7] successfully.

MAL language uses *global* and *local* TTC measurements. Local TTC refers to a single attack step, while global TTC refers to several attacks steps to achieve a target asset. The probability of global TTC is computed as

$$\Phi(A) = P(T_{glob}(A) = t),$$

where A is Attack step, T_{glob} is TTC the asset starting from initial step [7].

MAL uses the probability distribution model, where given $n, m, k \in \mathbb{N}, n < m < k$:

- 5 % of attacks success can happen within n time (s),
- 50 % of attacks success can happen within m time (s),
- 95 % of attacks success can happen within k time (s).

Distribution	**Parameters**	**Limits**	**Expected Values**
Bernoulli	p, probability	$0 \leq p \leq 1$	E(Bernoulli(p)) = p
Binomial	n, trials and p, probability	$0 \leq n, 0 \leq p \leq 1$	E(Binomial(n, p)) = n * p
Exponential	λ, rate	$0 < \lambda$	E(Exponential(λ)) = $\frac{1}{\lambda}$
Gamma	k, shape and θ, scale	$0 < k, 0 < \theta$	E(Gamma(k, θ)) = k * θ
LogNormal	μ, mean and σ, standard deviation	$0 < \sigma$	E(LogNormal(μ, σ)) = $e^{(\mu+\sigma^2/2)}$
Pareto	m, minimum and α, shape	$0 < m, 0 < \alpha$	E(Pareto(m, α)) = ∞ if $m \leq 1$, otherwise $\frac{(m * \alpha)}{(\alpha - 1)}$
TruncatedNormal	μ, mean and σ, standard deviation	$0 < \sigma$	E(TruncatedNormal(μ, σ)) = μ
Uniform	min and max	min $\leq$ max	E(Uniform(min, max)) = $\frac{(min + max)}{2}$

Table 2.5: Distribution functions available in MAL [40]

In this work, we focus on two probability distributions: *Bernoulli* and *Exponential*. In Table 2.5, the *Bernouli(0.5)* function represents the 50 % certainty of an attack success. For all uncertain attack steps, we multiply the probability of distribution by *Bernouli(0.5)*, while, in case of certain attack steps, we do not need this multiplication. *Exponential* function here is responsible for attack difficulty. This function shows how many days it is needed for an attacker to compromise an asset. As an example, *Exponential(0.1)* is mapped to 10 days to compromise, while *Exponential(0.01)* is mapped to 100 days to compromise. The more difficult attack step for an asset is, the more days it takes to compromise an asset.

In the following example, [4] we can see how probability distributions are attached to an attack step in MAL.

```
category Systems
{
  asset Computer
  {
    | compromise [Exponential(0.1)]
  }
}
```

[4] https://github.com/mal-lang/malcompiler/wiki/Supported-distribution-functions. Accessed on 29.06.2020

Here, to compromise a *Computer*, an attacker needs ten days with the certainty that this attack will succeed.

2.4 Related Works

Our research on previous works covers two topics: model-driven security engineering and DSL, and threat modeling projects for the domain of SCADA. In subsection 2.4.1 we cover the notion of model-driven security engineering and DSL. In subsection 2.4.2 we discuss previous works on threat modeling for SCADA.

2.4.1 Model-driven security engineering and Domain-specific languages

Model-driven security engineering focuses on reusing the design models to perform a security analysis [41]. In other words, model-driven security engineering outputs security implementations automatically based on the security specifications model. A model-driven security engineering facilitated the application of DSL to express the security requirements of a system or a software [42]. Using the DSL, we can build systems or tools that assess the security of a system or software during phases of building, deployment, and production. Previous languages for security as *CORAS* [43], *UMLSec* [44] and *SecDSVL* [45] are good examples of such DSLs. CORAS is a model-driven risk-analysis language for threat modeling [43]. UMLSec extends the capabilities of Unified Modeling Language (UML) by assessing the security of the software as well [44]. However, one issue with UMLSec is that the security assessment happens at the beginning of the building of software, which overlooks the potential issues that can in the middle or end of the building. SecDSVL is also a DSL for enterprise security modeling [45] that extends the capabilities of UMLSec. In addition to that, Almorsy and Grundy [45] introduced a way to ensure the changes to the deployed system after finding security issues. The disadvantage of these languages, however, is a lack of automated analysis. Instead, these languages focus on producing security properties with manual analysis [42].

One way of addressing such a lack of automated analysis is an attack graphs-based DSL. *Topological Vulnerability Analysis (TVA) system* is one example of such a tool which models the security conditions in the network for multi-step network penetration, and applies the attacks from the database of exploits [46]. TVA system also proposes the computation of hardening measures and assess the alerts. Another example of attack graphs-based DSL is Meta Attack Language (MAL). MAL produces the probabilistic attack graph from a given system specification. This DSL combines the attack graphs-based DSL with a model-driven security engineering approach.

There are good examples of how MAL could be extended for creating threat modeling languages for various domains: *vehicleLang* and *corelang*. *vehicleLang* is dedicated for modeling of cyber-attacks towards the domain of vehicles, and *corelang* is a core threat modeling language designed for modeling standard attacks towards an abstract Information Technology (IT) system [42].

2.4.2 Threat modeling projects for SCADA

KTH research groups worked on a Vital Infrastructure, Networks, Information and Control Systems Management (VIKING) project from 2008 to 2011 to build a threat modeling framework for determining security issues in a SCADA system [47]. According to to [21], the main security challenges in SCADA are as follows:

- Diversity of technology stack in SCADA. The field RTUs can have a code written in the '70s, while the CC is the most modern system with the newest technology stack.
- ICCP was not designed with the CIA properties in mind.
- The Denial-of-Service (DoS) attacks are a prevalent threat to the continuous availability of SCADA.

The main limitations of a VIKING project was a focus on data exchange between a CC and other components of SCADA, and application layer security [47].

INcreasing Security and Protection through Infrastructure REsilience (INSPIRE) is another project which focuses on the security of SCADA.INSPIRE provides directives for properly configuring, managing, and securing Critical information infrastructure (CII), such as SCADA [48]. INSPIRE presents a simulation tool *PSS-Cincal CRISP*, which could be used in various industries. Nevertheless, the main disadvantage of this project is the high-level approach in finding potential threats, which does not provide any concrete action points or protection mechanisms [48].

Recently, MITRE created a knowledge base for attacks against Industrial Control Systems (ICS), *ATT&CK for Industrial Control Systems (ICS)* [49]. According to MITRE knowledge base for attack adversaries towards ICS, the attacks towards these systems are different from attacks from other MITRE knowledge bases [49]. First of all, attackers focus on disturbing the system and causing harm to human operators. Secondly, ICS operators are working continuously 24/7 and are responsible for collecting information about the state of the system. This makes them a target for attackers to incorrectly accept the incoming information about the state of the system. The third difference is the heterogeneous nature of ICS. Unlike other systems, SCADA has various environments, hardware, software, and communication protocols, which complicates the process of creating the attack adversaries and collecting the unified techniques applicable for all SCADA systems. The main advantages of this knowledge base are giving a full vision of the system, information about previous attacks and attackers, and covering all the possible security threats.

3 Methodology

This chapter describes the process of collecting and analysing data that is needed for creating a threat modeling language for a SCADA system based on MAL, as well as building a specific SCADA system threat model based on the proposed language. Section 3.1 describes the selected methodology technique and the reasoning for its selection. Section 3.2 describes how we collect the data from available sources and process it. In section 3.3, we present our suggested risk assessment strategy to incorporate Common Vulnerability Scoring System (CVSS) to MAL. Section 3.4 outlines the results assessment and validation process.

3.1 Method

This work follows *Engineering Conceive, Design, Implement, Test, Operate (CDITO)* method. CDITO enhances Conceive, Design, Implement, Operate (CDIO) [50] framework with additional testing process. This method concentrates on creating and operating new products or systems focusing on the strategic importance of research and understanding the industry needs [51]. This method suits our needs, since we first of all design our language (choose the scope of assets and TTC), implement (gather all needed data and create a language based on MAL), perform available tests (MAL unit tests) and then operate. Method of surveying can be used for enhancing the validity of output using Turing tests. This is elaborated further in section 3.4.

3.2 Data Collection

We collect relevant data that fall under these categories: *Assets* and *Actors*. According to MITRE [52], assets are software, hardware, OS, communication protocols and embedded devices. *Actors* are the key engineering roles that work directly with the SCADA system.

3.2.1 Assets and Layers selection

Due to the heterogeneous nature of ICS systems, it is necessary to generalise and categorise assets. Section 2.1.1 explains the architecture of a SCADA system. The generalisation of ICS assets are as follows [52]:

- Control server (referred further as *"App server"*)
- Data historian (referred further as *"Database"*)
- Field Controller/RTU/PLC/Intelligent Electronic Device (IED) (referred further as *"RTU"*)
- HMI
- Input/Output Server (referred further as *"Communication front end"*)

- Safety Instrumented System/Protection Relay
- Engineering workstation (laptops, mobile devices)

Table 3.1 represents assets available at SCADA for this work, and categorises into three group based on the functionalities: *network*, *function* (platform and service) and *data* as described in 2.2.1. For the convenience, *network* is related to communication means, *function/service* is the category of assets that face the external clients or actors, and *function/platform* are internal assets that include infrastructure and are not accessible for external components.

Asset	KTH threat modeling method Type
Production zone	network
Enterprise zone	network
PCN	network
DMZ	network
Development zone	network
Global zone	network
Management zone	network
ICCP zone	network
Training zone	network
Firewalls	network
Data diodes	network
Router	network
IDS	network
ICCP frontend server	function/platform
ICCP backend server	function/platform
Directory Server (e.g., AD, Redhat Directory Server)	function/platform
Control centre (CC)	function/platform
HMI Servers, Thin Client, new HMI	function/platform
Field units, RTU	function/platform
Communication front end	function/platform
Backup servers, replicas	function/platform
DNS, NTP, NIS servers	function/platform
Data Engineering servers	function/platform
Alarm	function/platform
PostgreSQL database	data
Real-time database	data
Oracle database	data
Service accounts	function/service
User accounts	function/service
Admin accounts	function/service
ICCP	network

Table 3.1: List of assets in a SCADA and their categorisation for this work based on KTMM

3.2.2 Data Sources and Processing

We collect the data used in this work from various documents and configuration files:

- IP packet filter rules (configured with *iptables* utility program),

- configuration files,
- output of configuration tools.

The project uses different configuration files that represent different SCADA instances. Having necessary input data to work with is important to build a threat modeling language for SCADA. Later, using this language, we can build a threat modeling tool for various SCADA setups.

When the SCADA setup is built, configuration files store information about the hosts, networks, databases, and products. However, configuration files lack information about the firewall rules. We can find the rules for packet transmission and traffic that can via the use of *iptables* command inside Linux hosts.

The steps for data processing in this work are as follows:

- Reviewing the provided script files and documents to assess the assets at SCADA.
- Collecting and defining assets that are relevant to the scope of this work
- Identifying threats and threat actors
- Summarising the most harmful threats
- Iteration over the found results
- Providing the mitigation strategies
- Generating a risk assessment

3.3 Risk Assessment Strategy Design

One of the challenges of creating a threat model for SCADA is performing an accurate risk assessment for a particular system. Not all of the attacks that utilise the same strategy have the same impact on different SCADA [4] systems, and therefore, we need to evaluate which attacks have a higher impact on a specific system than others.

In this work, we map CVSS scoring system, which measures the severity of a vulnerabilities, into probabilistic model of MAL. As described in section 2.3.3, probabilistic model of MAL uses TTC metric. TTC is computed as

$$\text{TTC} = \text{attack difficulty} * \text{certainty of attack success}$$

.

There is an explicit relation between attack difficulty and CVSS score. The *Base score* and *Environmental score* of CVSS scoring system considers the *Attack Complexity* as a factor that determines the score. Keramati, Akbari, and Keramati [53] performed a similar mapping with regards to CVSS temporal scoring and attack difficulty. Keramati, Akbari, and Keramati [53] stated that the higher the temporal score is, the easier it is for an attacker to make an exploit. We consider this relation in this work.

Since the attack difficulty offered by MAL describe the Time-To-Compromise (TTC) for an attack step, we map the *Attack Complexity* of CVSS into TTC supported by MAL discussed in more details in section 2.3.3. First of all, we define which factors are considered for *Attack Complexity* of CVSS. Attack complexity describes the situation when there exist any conditions that can prevent an attacker from exploiting a vulnerability. Whenever such conditions exist, an attacker needs to dedicate more time to collect additional information about the target asset or prepare an exploit [54].

Holm, Ekstedt, and Andersson [55] describe the relation between security metrics using CVSS information and TTC as a Pearson correlation[5]. Researchers took into consideration several security metrics that use CVSS data and concluded that the more CVSS data is taken into consideration, the higher the Pearson correlation coefficient is between security metrics using CVSS information and TTC.

For this work, we implicitly derive the certainty of the success of an attack from CVSS score. Sawilla and Ou [56] explains that the "likelihood the attack path can lead to a successful exploit" is a necessary factor in determining the criticality of an attack.

CVSS rating	**Ordinal distribution**	**Mapping to attack difficulty**	**Mapping to certainty of success**
Critical	EasyAndCertain	attack is straightforward with lower than medium difficulty.	if the attack is successful, then priority of resolving is high
High	HardAndCertain	difficult to exploit.	if the attack is successful, then priority of resolving is high
Medium	HardAndUncertain	difficult to exploit.	if the attack is less successful, then priority of resolving is lower
Low	EasyAndUncertain	the least difficult to exploit.	if the attack is less successful, then priority of resolving is lower

Table 3.2: Mapping CVSS score to MAL probability distribution

In this work, we designed a way to map a CVSS score of vulnerabilities into MAL distribution probability. First of all, we collect the CVSS scores for Common Vulnerabilities Enumeration (CVE) vulnerabilities that could be used to exploit an attack step. Then we compute an average of the score of all CVE vulnerabilities.

[5]Pearson correlation coefficient is a statistic that measures linear correlation between two variables

We use this value as an *Attack severity score*. Attack severity score follows the same categorisation (low, mid, high, critical) as CVSS score for vulnerabilities.

The mapping of Attack severity into MAL probabilistic model for this work follows the rules:

- *Severity Level: Critical* -> straightforward, no any special authentication credentials are needed, root-level compromise of servers or infrastructure devices, lower than medium difficulty. For the scope of this work, an attacker would require ten days to complete an attack step since an attacker needs to perform a straightforward exploit without any special authentication credentials or obtaining valid accounts through time-consuming phishing or social engineering campaign.

- *Severity Level: High* -> difficult to exploit. For the scope of this work, it takes 100 days for an attacker to complete an attack step, as an attacker needs to perform time-consuming attack preparation as a phishing campaign or obtain root access to the system.

- *Severity Level: Medium* -> less difficult than hard but still difficult, require user authentication. For the scope of this work, it takes 100 days for an attacker to complete an attack step, as an attacker needs to perform time-consuming attack preparation as a phishing campaign to get the user account. This attack takes the same time as a high severity attack. However, the probability of success is only 50 percent.

- *Severity Level: Low* -> requires local or physical system access, little impact on the organization, the least difficult. For the scope of this work, it takes ten days for an attacker to complete an attack step as an attacker already has local or physical system access.

Table 3.2 summarises the mapping of CVSS score to attack difficulty and certainty and describes which probability function was selected to map the CVSS score.

3.4 Assessing the Reliability and Validity of DSL for Threat Modeling based on MAL

The process of evaluation of DSL for threat modeling based on MAL consists of three parts: assessing the reliability, assessing the validity of the language by SCADA experts, and enhancing a formal validity of *scadalang* in a real-life scenario using the Turing test. The first part of the evaluation includes testing the implementation and logic in the first step. These tests provide an estimation of whether the DSL based on MAL is capable of giving a reliable threat model for a specific problem. In our case, the specific problem is generating a threat modeling language for SCADA. The explanation of writing tests for DSL for threat modeling using MAL is given in appendix B. The second part includes cross-referencing with another expert that had an experience with MAL, and engineer experts in SCADA. Experts can guarantee

that the language could be applied to generate a threat model for a domain of SCADA using *scadaLang*.

The enhancement of formal validation of *scadaLang* can strengthen the results and ensure that *scadaLang* is applicable for threat modeling of SCADA in real-life scenario. One example of such enhancement of validation is using the Turing test by inviting experts in the industry (in our work, experts of SCADA). DSL for threat modeling based on MAL fall under the definition and categorisation of Expert System (ES) because it aims to offer a solution for a complex problem in a specific domain (which needs a human expert in solving issues for this domain) "at the level of human expert performance" [57].

Turing test is a method to estimate whether a program can perform a "thinking like a human expert" [58]. Typically, in the case of validating ES, we can run numerous tests and compare the output with the output from an expert in this domain based on his/her judgment and opinion. The success rate of ES can be calculated subjectively and objectively by an expert [57]. In our case, an expert needs to compare the attack path prepared by a human and the attack path generated from the language towards the same goal, starting from the same entry point. These experiments can be repeated with several experts. Then, these two outputs can be given to another independent expert that needs to distinguish whether a human expert or an ES proposed the specific solution. This independent expert compares two attack paths and gives an independent judgment used as a success rate.

The output of such a Turing test can enhance the validity of the threat model generated by DSL based on MAL for a particular SCADA instance.

4 Implementation of *scadaLang*

This chapter present a DSL for language of a SCADA system called *scadaLang* based on MAL. Section 4.1 describes the scope and assumption for a *scadaLang*. Section 4.2 presents the assets that we selected for *scadaLang*. Section 4.3 and section 4.4 describe the actors in the system, accounts and potential attackers. Section 4.5 lists all potential attacks per each general asset in SCADA in *scadaLang*. Section 4.6 demonstrate the associations between assets and attacks encoded in the asset. Section 4.7 proposes mitigation strategy for aforementioned attacks. Section 4.8 generates a risk assessment for proposed attacks in a SCADA system. The code for *scadaLang* can be found in appendix D

4.1 Scope and Assumptions

The assumptions for *scadaLang* are as follows:

- We assume that attacks originate from outside SCADA. The scope of this language does not consider the attacks inside the Enterprise/Corporate Zone, which we discussed in section 2.1.4.
- We assume that MITRE gives valid information about attacker groups and attack techniques.
- Assets that belong to *Safety Instrumented Systems (SIS)/Protection Relay* are not considered into the scope of this language.
- Assets that belong to *Engineering workstation (laptops, mobile devices)* are not considered into the scope of this language.
- This work focuses on the attacker with relatively high skills and motivation that targets SCADA system with malicious intent. One such example is a rogue employee (both current or former employees).

4.2 Assets and Categories

In this section we describe the assets and categories for *scadaLang*. In 4.2.1 we group assets into categories, while in 4.2.2 we present assets in *scadaLang*.

4.2.1 Categories

As described in section 2.3.1, categories are used for grouping of assets. For *scadaLang*, we divide the assets into three category: *System*, *Communication* and *Security*. As mentioned in section 3.2, all assets are categorised into KTMM types: *function/service*, *function/platform*, *data* and *network*. In *scadaLang*, the mapping from MAL types into categories is as follows:

- Function/platform -> System category

- Function/service -> Security category
- Network -> Communication category
- Data -> Data category

The naming for the categories is subjective, and based on our preferences. To the current extend of MAL, these categories do not affect neither assets, nor attacks, nor defense, nor risk assessment. The main purpose of categories is to organise assets together.

4.2.2 Domain Feature Matrix

The *domain* of our work is a SCADA system. Table 4.1 contains these domain-specific assets divided into aforementioned categories.

As explained in section 4.2, some assets as RTU are considered out of scope for this work. Three servers: DNS, NIS, and NTP are present in the language specification, but they have lower importance in this language, and, thus, marked as out of the scope in the Table 4.1.

Category	Asset	scadaLang	Out of scope
System	ICCP server	x	
	HMI	x	
	RTU		x
	Communication front end	x	
	Alarm	x	
	App Server	x	
	Database	x	
	Antivirus server	x	
	Directory service	x	
	Backup server	x	
	Product	x	
	DNS server		x
	NIS server		x
	NTP server		x
	DE server	x	
	nHMI server	x	
	Software		
Security	User Account	x	
	Admin Account	x	
	Service Account	x	
	Vulnerability		
Comm	Firewall	x	
	Zones	x	
	Router		x
	Data diodes	x	

Table 4.1: Asset feature matrix

Mainly, assets are obtained from configuration files mentioned in 3.2. If some important information is missing, we can use other sources of information about the traffic and firewall rules.

4.3 Actors

In this section, we describe the actors of the system. For the convenience, we selected only actors that are key to a SCADA, since they have access to multiple components of the system. In this work, we do not focus on all actors in a SCADA system. Table 4.2 focuses on actors that primarily work with such assets that can serve as an entry point for potential attacks.

Actor	Description
HMI Operator	monitoring and controlling the power transmission network
Admin User	monitoring system health and updating control system configuration
Admin Directory Service User	monitoring service account and access through Directory Service
Trainer and students	running simulations
Data engineer	updating the power system model
Production planner	viewing historical data and creating plans
Field engineer	sending the data from RTU to Communication front end

Table 4.2: List of main actors in SCADA

4.4 Attacker Profiles

As discussed in section 4.1, in our work we mainly focus on *Rogue Employees*. Rogue actors that we consider for this language in SCADA for this work are as follows:

- *HMI Operator* can is considered rouge, when the accounts to HMI were leaked. This way the attack path can start from HMI.
- *Field engineer* can also be a rogue actor to generate the attacks starting from RTU to the CC.
- *Admin User* of other SCADA that is connected to the current SCADA can be rogue, when the first system got compromised.

External SCADA is another system which communicates with the current SCADA through ICCP. Having a connection to another SCADA system is necessary to ensure the guaranteed availability of the service in case of any critical issues. If the secondary SCADA fails to attacks, the primary SCADA becomes vulnerable to attacks through ICCP.

Field units include all components of SCADA that collect information about the state of the system in the field. Therefore, we can assume that malicious actors could misconfigure the data that field units send to CC.

4.5 Attacks

In this section we demonstrate attack techniques and vulnerability exploits for SCADA. This work proposes the attack techniques from *ATT&CK matrix for ICS* as described in section 4.1. For mapping of *ATT&CK matrix for ICS* knowledge base into our attacks we considered the following rules:

- MITRE provides a matrix of attack steps (referred as “techniques") grouped into “tactics". Some techniques might occur in several tactics. In our work, we renamed some attack steps. In appendix C, we give an explanation of which attack step in *scadaLang* refer to which technique in MITRE.
- The tactic *“Impact"* refers to goals of attack path in *scadaLang*. We do not consider any goal of attack path outside of the tactic *“Impact"*. We do not modify the impact of an attack based on our judgement. Instead, we only follow *ATT&CK matrix for ICS* to determine an impact for an attack step.
- We do not have attacks that do not have a correspondence to *ATT&CK matrix for ICS*.

The attack techniques vary based on target key assets. Most importantly, *availability* of the SCADA is the main priority from CIA properties. This is due to necessity to continuously preserve the industrial processes putting data loss and confidentiality aside as less priority. Overall, we summarised the goals of attackers into the following categories:

- *Manipulating sensitive data*,
  - Loss of view,
  - Manipulation of view,
  - Theft of Operational Information[6]
- *Disrupting the safety of operation in SCADA*
  - Damage to property [7]
  - Loss of safety[8]
- *Disrupting the availability of the system.*
  - Loss of Productivity and Revenue[9]
  - Denial of control

These aforementioned goals match *“Impact"* tactics at MITRE as described in section 4.1. Subsection 4.5.1 provides the information about the entry surface for the attacks. Further subsections discuss the attacks defined for each asset in section 4.2. For convenience in this report, we used the alphanumeric codes, e.g. “ICCPS1" (for an attack in ICCP server, where number was selected for notation purpose only)

[6] https://collaborate.mitre.org/attackics/index.php/Technique/T882. Accessed on 29.06

[7] https://collaborate.mitre.org/attackics/index.php/Technique/T879. Accessed on 29.06

[8] https://collaborate.mitre.org/attackics/index.php/Technique/T880. Accessed on 29.06

[9] https://collaborate.mitre.org/attackics/index.php/Technique/T828. Accessed on 29.06

4.5.1 Entry surface

Entry surface include the assets and techniques that are the closest to malicious actor due to the system configuration (e.g. front-end web servers or applications for external users) or due to vulnerabilities (e.g. the kernel version, malicious library version found, possibility of buffer overflow due to weak code, lack of data sanitising and validation). As for SCADA, this work focuses on three entry surfaces as discussed in 2.1.2:

- External SCADA
- Field units
- HMI

4.5.2 ICCP server (frontend and backend servers)

- ICCPS0.Server compromise
- ICCPS1.Accessing logs to collect information
- ICCPS2.Obtain valid account
- ICCPS3.Remote System discovery
- ICCPS4.Remote File copy
- ICCPS5.Stealing Passwords from Memory

Figure 4.1 presents the relations between attack steps and which attack steps lead to another.

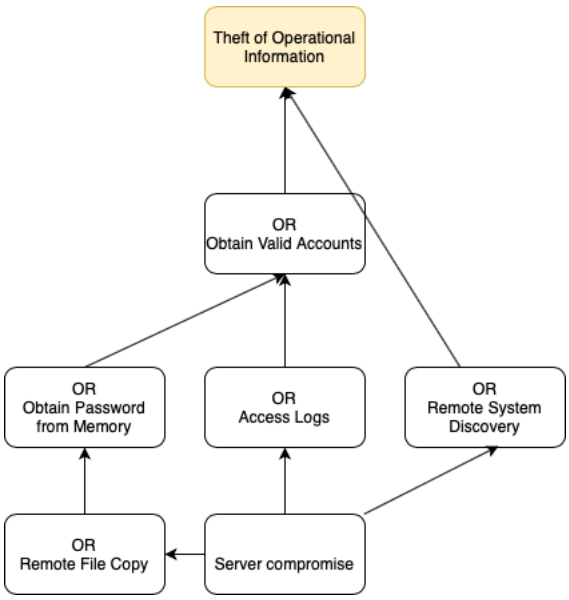

Figure 4.1: Attacks relations for ICCP server

The starting point for an attacker in this and further attacks is compromising the server, i.e., getting physical access to the server. The access may not be an administrative one. In this work, we use the terms “host" and “server" interchangeably.

After an attacker could get into the server, there are attack paths: enumerating the host, uploading the file, or looking for other hosts from this host. We enumerate[10] the host to find any potentially useful information for gaining administrative access [59]. To enumerate the host, an attacker could check the logs (MITRE attack technique nr. T811) for any valid accounts (MITRE attack technique nr. T859). As for uploading the file, an attacker can copy the malicious files to this host (e.g., using Metasploit, an attacker can place a Mimikatz tool in a remote host to crack the system credentials or perform a *pass-the-hash* attack). This attack step corresponds to a remote file copy (MITRE attack technique nr. T867). The third attack path is trying to look for other hosts in the network. This attack direction corresponds to remote system discovery (MITRE attack technique nr. T846. An attacker can use tools like *nmap/Zenmap* to scan the open ports in other hosts. Such a scenario can happen when an attacker wants to get into the App server through the ICCP server.

The impact of *Obtaining valid account*, *Remote system discovery* and *Obtain password from memory* attacks is *Theft of Operation Information*. Theft of Operation Information includes obtaining valid accounts, information about other hosts in the network, and others.

4.5.3 RTU

- RTU1.Remote System Discovery

We do not consider attacks at the field against RTU in the scope of this work. For this work, we consider a compromised RTU as an access point. Therefore, we obfuscate these attacks.

4.5.4 Communication front end

- CF0.Server compromise
- CF1.Obtain valid accounts
- CF2.Using default credentials
- CF3.Accessing logs to collect information
- CF4.Service discovery
- CF5.Administrative access to Communication front end
- CF6.Service stop

[10] Enumeration is a procedure where an attacker establishes an active connection to the target hosts to discover potential attack vectors in the system. This procedure is used for further exploitation of the system [59].

- CF7.Shut down Communication front end
- CF8.Remote System discovery
- CF9.Remote File copy
- CF10.Stealing Passwords from Memory (Mimikatz)

This asset is accessed by a *Field Engineer* discussed in section 4.3. The main way to access the Communication front end discussed at this work is from RTU. In case we consider that this field engineer can be rogue or an attacker obtained access to an account, we have a threat that he/she can send the wrong commands and disrupt the system. Another way to access the Communication front end is through Directory Service as *Admin User*. In case Admin Directory Service User is rogue, the access to the Communication front end is compromised. To our best knowledge, the Communication front end does not have replicas, so shutting down a Communication front end could disrupt the system.

Figure 4.2 presents the relations between attack steps and which attack steps lead to another.

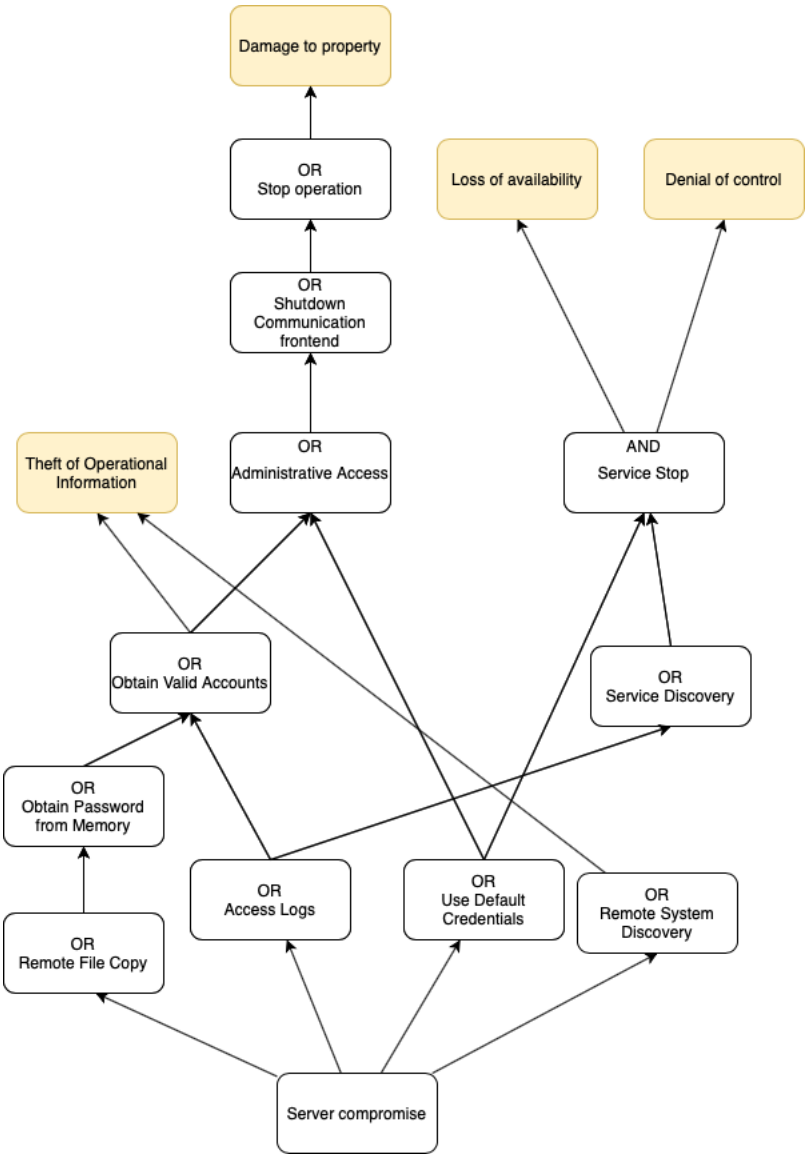

Figure 4.2: Attacks relations for Communication front end server

The starting point for an attacker is compromising the server, i.e., getting physical access to the server. The attacker can then proceed to enumerate the host, i.e., collect relevant information about the server. An attacker can find relevant information by manually inspecting logs and other interesting system files or utilising automated enumeration tools(MITRE attack technique nr. T811). Such an approach can lead the attacker to obtain valid accounts for some systems (MITRE attack technique nr. T859). Alternatively, an attacker can utilise system discovery tools, e.g., using *nmap*, and scanning techniques to discover relevant connected systems and components to pivot the attacks (MITRE attack technique nr. T846). Moreover, an attacker can utilise remote file copy techniques to bring malicious tools to the server (MITRE attack technique nr. T867). Such tools could allow the attacker to obtain an admin account from memory (MITRE attack technique nr. T843/T845). After getting

administrative access to the Communication frontend, an attacker can try to shut down Communication frontend.

The impacts of attack paths are *Theft of Operation Information*, *Damage to property*, *Loss of availability* and *Denial of Control*. Theft of Operation Information includes obtaining valid accounts, information about other hosts in the network, and others. Damage to property includes shutting down the service to harm to operation. Loss of availability, in return, is similar to damage to property. However, it focuses on disrupting the availability for a longer period rather than damaging. Denial of control is a similar impact as a loss of availability.

4.5.5 HMI + Thin client

- HMI0.Server compromise
- HMI1.Accessing logs to collect information
- HMI2.Obtain valid account
- HMI3.Man-in-the-Middle
- HMI4.Modify alarm settings
- HMI5.Administrative access to HMI
- HMI6.Remote system discovery
- HMI7.Place a ransomware

This asset is assessed by an *Operator* discussed in Table 4.2. In case we consider that this Operator can be rogue or an attacker obtained access to an account, we have a threat that he/she can send the wrong commands and disrupt the system. HMI can be a Graphical User Interface (GUI) or a server with GUI. Since there is no information about this in configuration files, we consider the second case. The assumption is then that HMI is a Windows server with GUI.

Figure 4.3 presents the relations between attack steps and which attack steps lead to another.

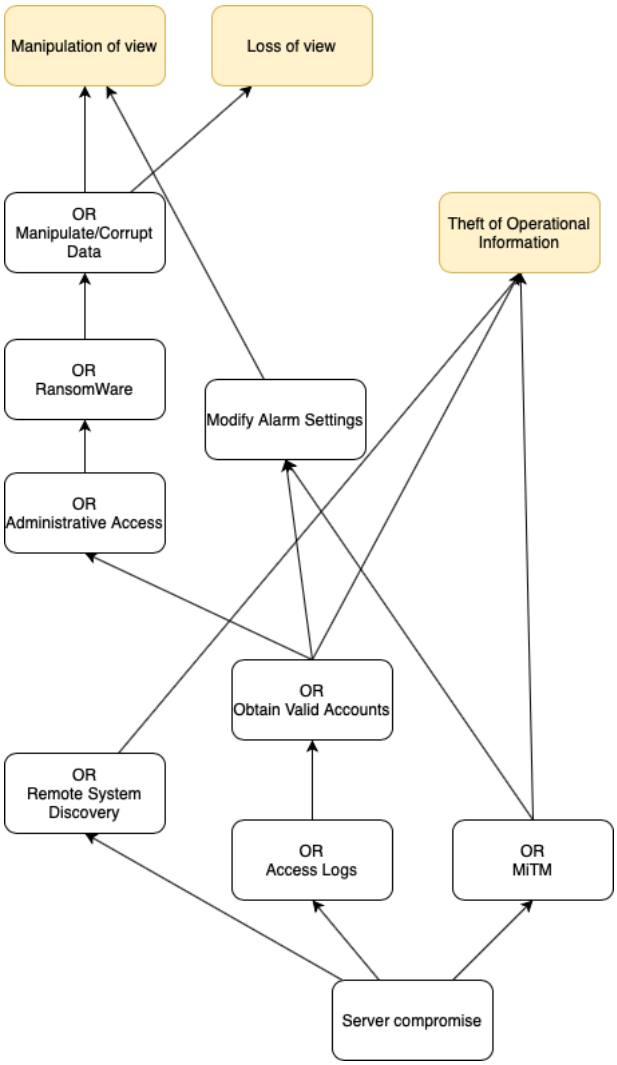

Figure 4.3: Attacks relations for HMI

The starting point for an attacker is compromising the server, i.e., getting physical access to the server. After an attacker acquires access to a server, the next steps could be either enumerating the host (referred as checking the logs in the figure) (MITRE attack technique nr. T811) for any valid accounts (MITRE attack technique nr. T859), or remote system discovery (MITRE attack technique nr. T846), e.g., using *nmap* command to find other hosts in the network, or Man-in-Middle attack (MiM) attack (MITRE attack technique nr. T859) to sniff the communication between HMI and other SCADA components, e.g., Alarm. After an attacker obtains an admin account to HMI, an attacker can place ransomware (e.g., LockerGoga, nr. S0008 in MITRE) that can erase or corrupt a data visible for Operator in GUI.

There are ways to manipulate data other than implementing ransomware. . Nevertheless, we follow MITRE suggested attacks, which highlighted ransomware as an impact for *"Manipulation of view"*.

The impacts of attack paths are *Theft of Operation Information*, *Manipulation of view*, and *Loss of view*. Theft of Operation Information includes obtaining valid

accounts, information about other hosts in the network, and others. Manipulation of view includes editing the information visible in GUI at HMI for an Operator, while the loss of view involves shutting down the GUI.

4.5.6 Alarm

- AL1.Boot / Unboot alarm
- AL2.Start alarm

"Boot alarm" and *"start alarm"* are commands in configuration file given as the input for generating a *scadaLang* as described in 3.2.2. Malicious use of either of them can harm the system. The impact for both attacks AL1 and AL2 are loss of safety.

4.5.7 App server

- APS0.Server compromise
- APS1.Obtain valid accounts
- APS2.Using default credentials
- APS3.Accessing logs to collect information
- APS4.Service discovery
- APS5.Administrative access to App server
- APS6.Service stop
- APS7.Remote System discovery
- APS8.Install rogue master device
- APS9.Data exfiltration
- APS10.Man-in-the-middle-attack
- APS11.Masquerading

App server is composed of several Linux servers and replicas. All of them constitute one asset for this work. Shutting down App server is not beneficial, because there are several replicas. APS11.Masquerading is disguising a malicious application as a standard tool.

Figure 4.4 presents the relations between attack steps and which attack steps lead to another.

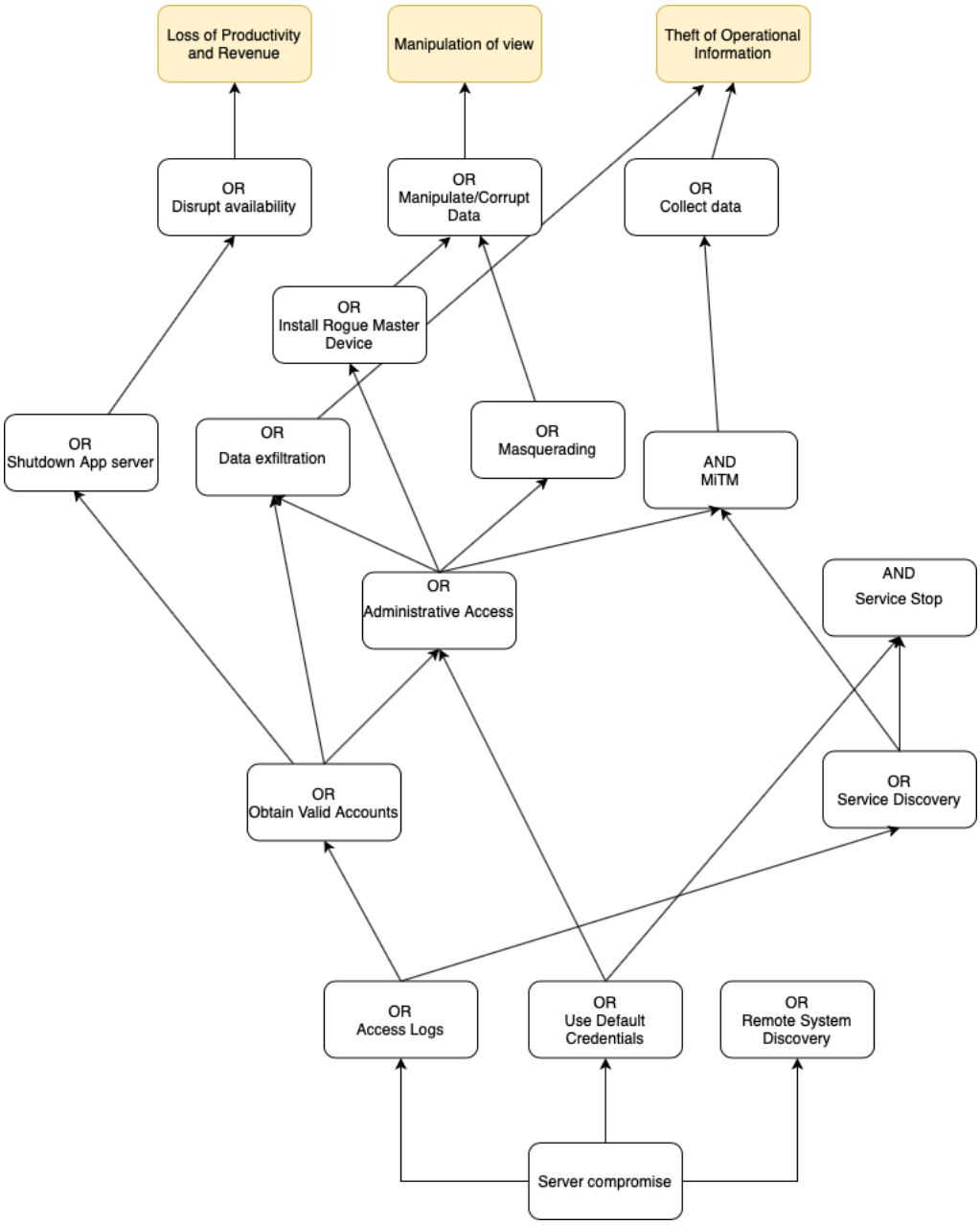

Figure 4.4: Attacks relations for App server

The starting point for an attacker is compromising the server, i.e., getting physical access to the server. The attacker can then proceed to enumerate the host, i.e., collect relevant information about the server. An attacker can find relevant information by manually inspecting logs and other interesting system files or utilising automated enumeration tools (MITRE attack technique nr. T811). Such an approach can lead the attacker to obtain valid accounts for some systems (MITRE attack technique nr. T859). Alternatively, an attacker can utilise system discovery tools, e.g., using *nmap*, and scanning techniques to discover relevant connected systems and components to pivot the attacks (MITRE attack technique nr. T846). The attacker can acquire

administrative access by utilising valid accounts from the enumeration or in case the App server has the default username and password (MITRE attack technique nr. T812). After accessing the administrative access to App server, an attacker can either proceed to data exfiltration (MITRE attack technique nr. 867), or masquerading (MITRE attack technique nr. 849), or MiM (MITRE attack technique nr. 830) or installing rogue master device (MITRE attack technique nr. 848). Masquerading involves techniques to disguise malicious application as standard (genuine) software. An attacker can use this technique to install spyware in the system that is difficult to find since it pretends to look like standard software. MiM includes sniffing the communication between App Server and other components of a SCADA system to learn about other hosts or study the underlying network.

The impacts of attack paths are *Theft of Operation Information*, *Manipulation of view*, and *Loss of productivity and revenue*. Theft of Operation Information includes obtaining valid accounts, information about other hosts in the network, and others. Manipulation of view includes editing the information visible in GUI at the App server for an App server Admin. Loss of productivity and revenue is similar to the loss of availability. However, since the App server focuses on monitoring and processing the data, disrupting the availability of hosts could lead to ineffective operations in the SCADA system.

4.5.8 Postgre and Oracle Database

- DB1.Bypass input validation
- DB2.SQL injection
- DB3.Command-line execution through SQL injection
- DB4.Data exfiltration
- DB5.Accessing logs to collect information
- DB6.Obtain valid account
- DB7.Administrative access to a Database (DB)
- DB8.Data corruption

SCADA has variations of DB to use. In this work, we focus on Oracle DB and Postgre DB as the most commonly employed relational database solutions for large scale systems. Since Postgre DB are open-source and do not require licenses to purchase and renew, there is a trend to shift to open-source and free alternatives, e.g., shifting from Oracle DB to Postgre DB. Therefore, making specific attacks from ExploitDB instead of focusing on general attacks from MITRE is important. The attacks for Oracle DB are from 2010, which makes them outdated. We found fewer vulnerabilities in ExploitDB and CVE/Common Weaknesses Enumeration (CWE) for Oracle DB. Compared to Oracle DB, Postgre DB has more vulnerabilities recorded in ExploitDB and CVE/CWE.

Figure 4.5 presents the relations between attack steps and which attack steps lead to another.

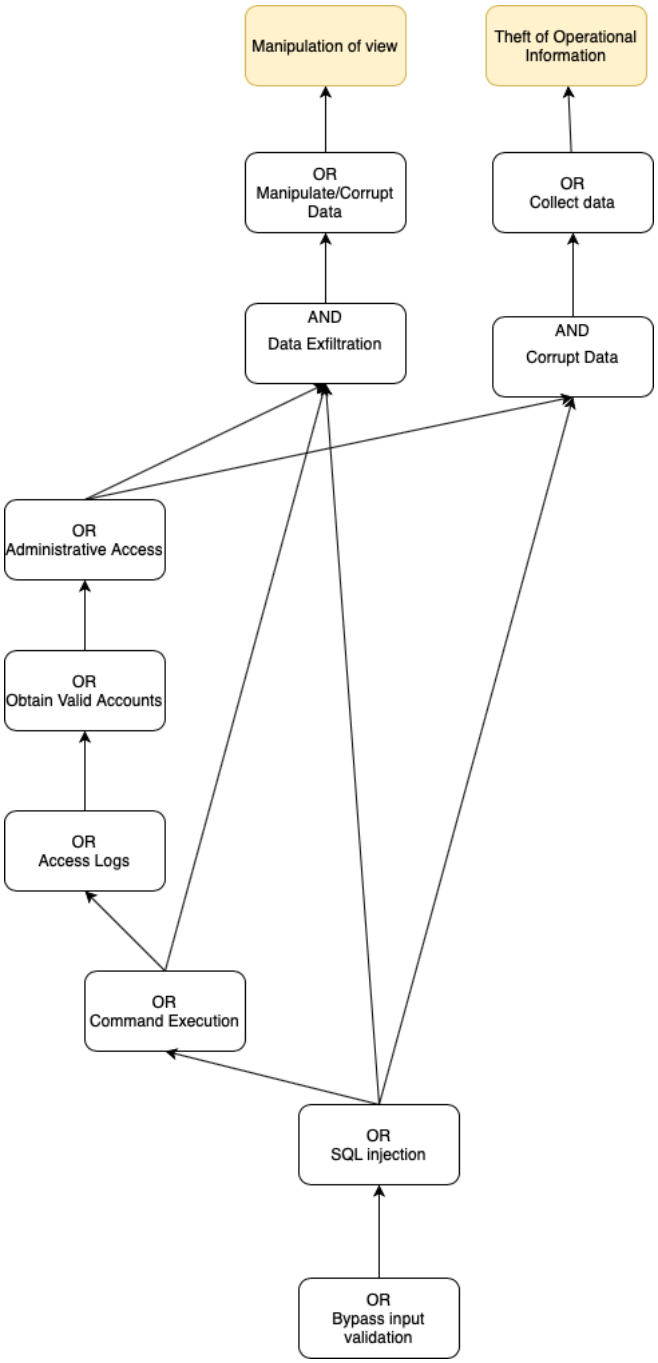

Figure 4.5: Attacks relations for the Database

The entry point for an attacker is bypassing user input validation that can occur when the user input is not sanitised and validated on the client-side. Such vulnerability can lead to Structured Query Language (SQL) injection, e.g., when there are no prepared statements, and an attacker could insert a malicious SQL

statement. With the help of such an attack, the attacker can delete the tables in DB or query user and admin DB accounts. After successful SQL injection, an attacker can execute further malicious commands. One such example is accessing logs in DBMS to obtain valid accounts to get admin access in DBMS.

The impacts of attack paths are *Theft of Operation Information*, and *Manipulation of view*. Theft of Operation Information includes obtaining valid accounts, information about other hosts in the network, and others. Manipulation of view includes editing the information in DB that is shown in other components of SCADA.

4.5.9 Real-time Database

- DB1.Bypass input validation
- DB2.SQL injection
- DB3.Command line execution through SQL injection
- DB4.Data exfiltration
- DB5.Accessing logs to collect information
- DB6.Obtain valid account
- DB7.Administrative access to Oracle Database
- DB8.Data corruption

Real-time DB uses modelled SQL and utilises Oracle DB technologies. We can assume that attacks towards Oracle and PostgreSQL apply here.

4.5.10 Antivirus Server

- AV0.Server compromise
- AV1.Obtain valid accounts
- AV2.Using default credentials
- AV3.Accessing logs to collect information
- AV4.Service discovery
- AV5.Service stop
- AV6.Man-in-the-middle-attack
- AV7.Modify the configuration of antivirus agents and repositories
- AV8.Remote File copy
- AV9.Stealing Passwords from Memory (Mimikatz)

The antivirus server in a SCADA system used for this work is implemented with ePolicy (EPO) server. However, other providers are also available. EPO server [60] is a central repository for storing all antivirus software (McAfee) installations and configurations. To the best of our knowledge, EPO server uses Windows OS. However, for this work, the Antivirus server is not a target asset. Attackers would access the antivirus central repository to misconfigure the antivirus software in other hosts. After misconfiguring the antivirus on other hosts, an attacker can proceed further to other services or data.

Figure 4.6 presents the relations between attack steps and which attack steps lead to another.

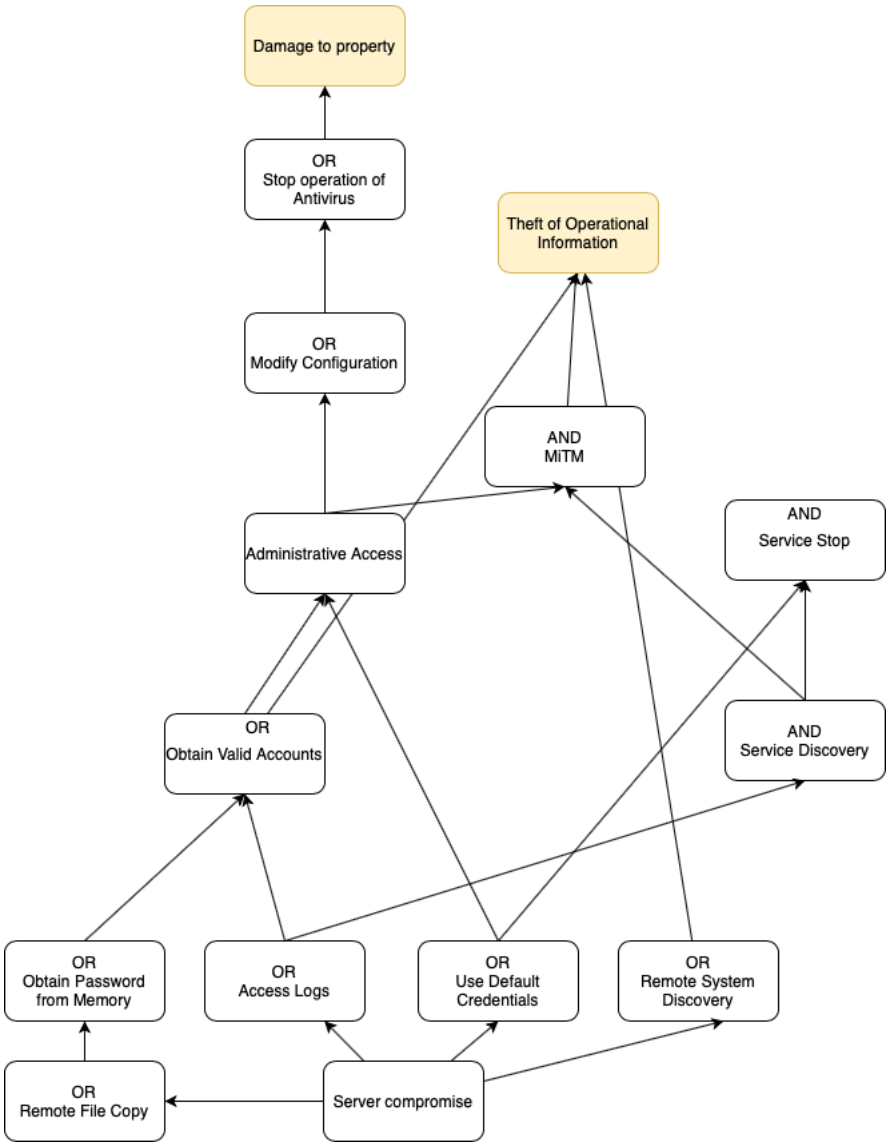

Figure 4.6: Attacks relations for Antivirus server

The starting point for an attacker is compromising the server, i.e., getting physical access to the server. The attacker can then proceed to enumerate the host, i.e., collect

relevant information about the server. An attacker can find relevant information by manually inspecting logs and other interesting system files or utilising automated enumeration tools(MITRE attack technique nr. T811). Such an approach can lead the attacker to obtain valid accounts for some systems (MITRE attack technique nr. T859). Alternatively, an attacker can utilise system discovery tools, e.g., using *nmap*, and scanning techniques to discover relevant connected systems and components to pivot the attacks (MITRE attack technique nr. T846). Moreover, an attacker can utilise remote file copy techniques to bring malicious tools to the server (MITRE attack technique nr. T867). Such tools could allow the attacker to obtain an admin account from memory (MITRE attack technique nr. T843/T845). After getting administrative access to the antivirus server, an attacker can try to modify the antivirus configuration files. This attack step can lead to the stop of antivirus software in other hosts. The attacker can achieve this attack if the Antivirus server is left with the default username and password (MITRE attack technique nr. T812). The impacts of attack paths are *Theft of Operation Information*, and *Damage to property*. Theft of Operation Information includes obtaining valid accounts, information about other hosts in the network, and others. Damage to property includes shutting down a central repository to harm other hosts.

4.5.11 Backup server

- BS0.Server compromise
- BS1.Obtain valid accounts
- BS2.Using default credentials
- BS3.Accessing backup files to collect information
- BS4.Administrative access to Backup server
- BS5.Data exfiltration

Backup server is a valuable target, since attacker can exfiltrate data (files, configuration data, access logs) from backup servers. Backup servers are responsible for automated backup and recovery supports disk storage or tape storage targets. Figure 4.7 presents the relations between attack steps and which attack steps lead to another.

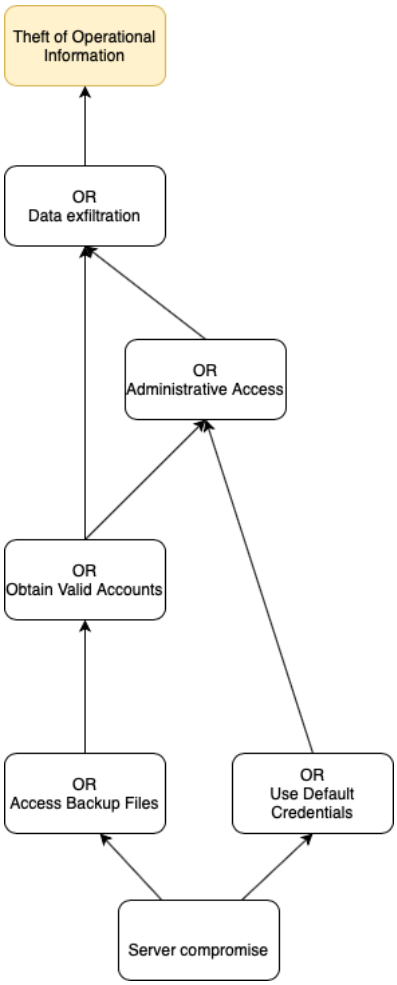

Figure 4.7: Attacks relations for Backup server

The starting point for an attacker is compromising the server, i.e., getting physical access to the server. After an attacker had access to a server, the next steps could be accessing backup files (MITRE attack technique nr. 811) or looking for default credentials (MITRE attack technique nr. 812) to get admin access of backup server. The factory/manufacturer of a backup server could set default credentials. The backup files can contain valid accounts (MITRE attack technique nr. 859), which can grant admin access for an attacker.

The impact of attack paths is *Theft of Operation Information*. Theft of Operation Information includes obtaining valid accounts, information about other hosts in the network, and others.

4.5.12 Directory Service

Directory Service in this scope of work are mainly focused on Windows AD, but the general attacks can be generalised for other OS [61].

- AD0.Server compromise

- AD1.Obtain AD Admin Account
- AD2.Remote File copy
- AD3.Stealing Passwords from Memory using (Mimikatz)
- AD4.Remote system discovery

A Directory service, e.g., Active Directory (AD), provides the methods for storing data in the network and providing access to this data for network users and administrators. Directory Service is not a target itself; the administrator's account in a Directory Service is a target to reach assets as App server or Communication frontend.

Figure 4.8 presents the relations between attack steps and which attack steps lead to another.

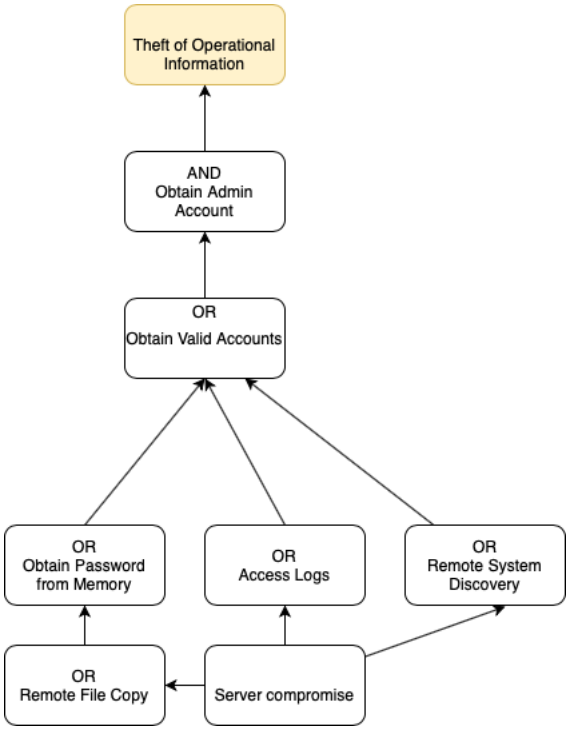

Figure 4.8: Attacks relations for Directory Service

4.5.13 Product

A product is a vendor software or program, residing on App Servers. Depending on the system, products might also reside on HMI [14]. In this work, we focus on products that reside on App Servers. The functionality of products in SCADA systems are as follows: [14]:

- Providing Open Data Base Connectivity (ODBC) for database;
- Providing libraries of Application Program Interface (API) support for programming languages, e.g., C, C++, Visual Basic 6;

- Facilities for exporting configuration data;
- Support archiving and logging of databases;

The possible attacks against the product are as follows:

- PR1.Product infection
- PR2.Product deletion/corruption

The product itself is not a target but rather a stepping stone for further attack. The *Product* asset facilitates and supports the processes in SCADA. The impact of attacks PR1 and PR2 are loss of revenue and productivity.

4.5.14 Other servers: DNS, NIS, NTP

- DNS0.Server compromise
- DNS1.DNS poisoning/spoofing
- DNS2.Shut down DNS service on a port
- NTP0.Server compromise
- NTP1.Stop service
- NTP2.Alter clock configurations

These asset are present in the system, however, these servers are not considered as important in this work. We assume that the attacker obtained service account to DNS server. NIS was removed from consideration.

4.5.15 Data Engineering / new HMI server

- DE/nHMI0.Server compromise
- DE/nHMI1.Remote file copy
- DE/nHMI2.Stealing Passwords from Memory
- DE/nHMI3.Obtain valid accounts
- DE/nHMI4.Using default credentials
- DE/nHMI5.Accessing logs to collect information
- DE/nHMI6.Service discovery
- DE/nHMI7.Administrative access to server
- DE/nHMI8.Data exfiltration

For this work, DE/nHMI servers are primarily used to store data for analytic purposes. Further capabilities of DE/nHMI servers as running Communication front end and HMI applications are considered out of scope.

Figure 4.9 presents the relations between attack steps and which attack steps lead to another.

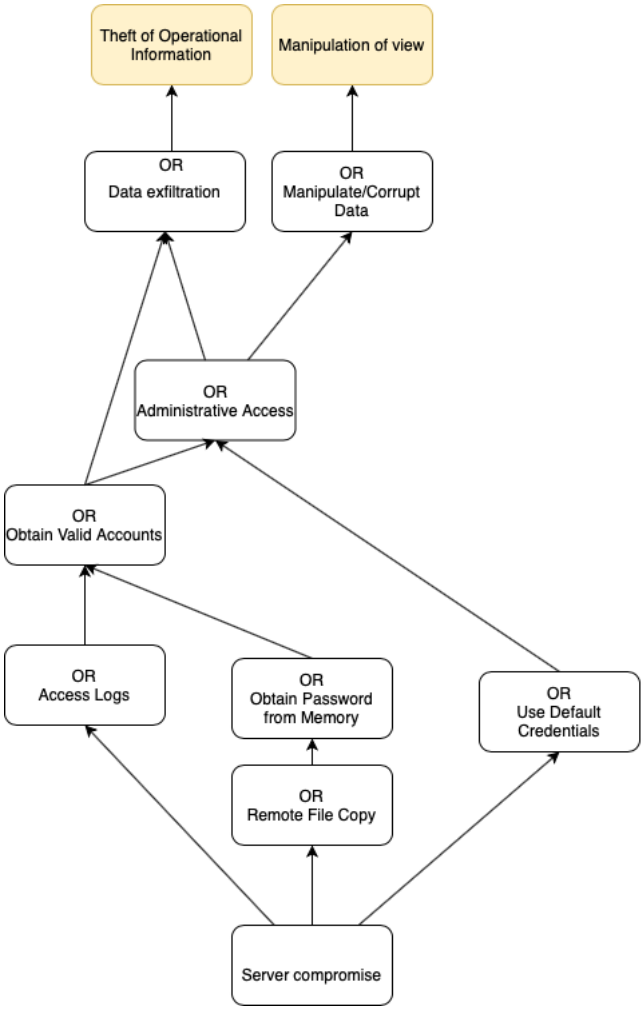

Figure 4.9: Attacks relations for DE server

The starting point for an attacker is compromising the server, i.e., getting physical access to the server. After an attacker had access to a server, the next steps could be enumerating the host (referred as accessing the logs (MITRE attack technique nr. 811)), remote file copy (MITRE attack technique nr. 867) or looking for default credentials (left as a plain text in files or system logs) (MITRE attack technique nr. 812) to get admin access of DE/nHMI. The ultimate goal for an attacker is to get an admin account toDE/nHMI, which could give an opportunity to either exfiltrate the data or corrupt the data (MITRE attack technique nr. T831/T832).

The impacts of attack paths are *Theft of Operation Information*, and *Manipulation of view*. Theft of Operation Information includes obtaining valid accounts, information

about other hosts in the network, and others. Manipulation of view includes editing the information in DE/nHMI servers.

4.5.16 Accounts

- SAC1.Brute force
- SAC2.Bribing SCADA engineer described in Table 4.2
- AC1.Brute force
- AC2.Phishing SCADA engineer
- AC3.Bribing SCADA engineer described in Table 4.2

Service account is type of account given for automation purposes for automated user. The admin on the system can give such type of credential to some automated systems, e.g. Continuous Integration (CI)/Continuous Delivery (CD) pipeline.

4.5.17 Firewall

- FW1.Bypass Firewall

A misconfigured firewall may enable a reverse connection. A reverse connection allows the attacker to establish communication from the compromised device to the attacker's device, rather than the typical forward connection from the attacker's device to the compromised device.

4.5.18 Zones

We can consider a zone as a Trust boundary. Once an attacker is inside one zone, he/she can access servers within this zone. In this work scope, we use Zones as assets, but do not focus on attacks towards the zone. Instead, we focus on attacks towards the hosts located in this zone.

4.5.19 Data diode

Data diode is a hardware solution, and, thus, requires physical attacks. We do not focus on physical attacks against data diode.

4.5.20 Router

Router information is not given in the configuration files, however, we consider that the router exists in SCADA system.

4.6 Associations

Associations between assets demonstrate how we can reach one asset from another. Associations between assets also carry information about the attack from one asset to another. The names of associations follow the techniques under tactics “Initial Access", “Execution" and “Discovery" in *ATT&CK matrix for ICS*. Figure 4.10 represents all of the assets and associations between assets that we consider for *scadaLang*.

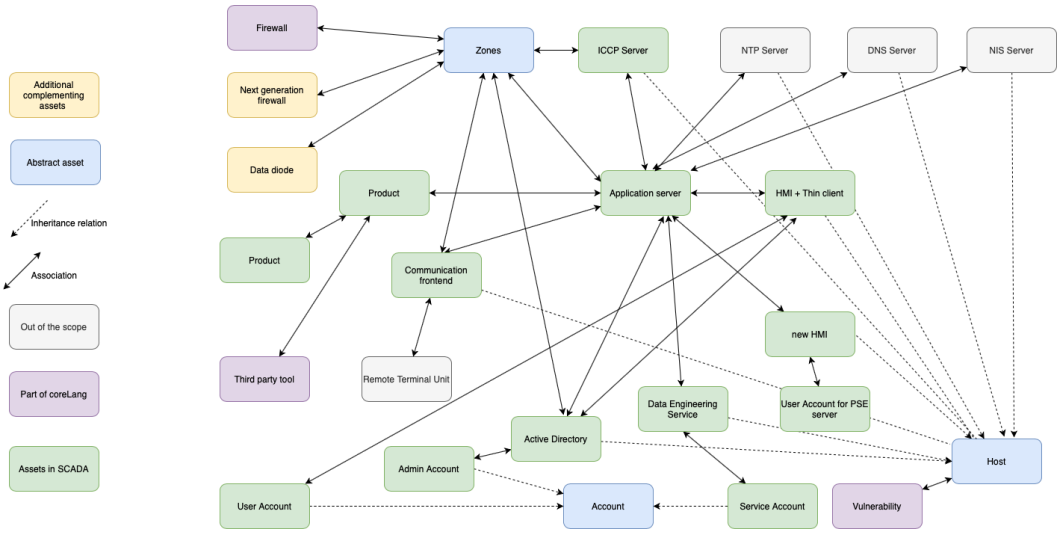

Figure 4.10: General assets and associations selected for this implementation

Table 4.3 demonstrates the associations between assets and the attacks from configuration files. We match naming for these associations with MITRE techniques naming convention for consistency.

Association	**Asset 1**	**Asset 2**	**Starting attack**	**Resulting attack**
Discover	App Server	ICCP server	remote system discovery AND obtain valid account	server compromise
Discover	RTU	Comm front end	remote system discovery	server compromise
Discover	Comm front end	App server	remote system discovery AND obtain valid account	server compromise
Discover	HMI	App server	remote system discovery AND obtain valid accounts	server compromise

Connect	App server	Alarm	MiM	start alarm, unboot alarm
Connect	HMI	Alarm	MiM	start alarm, unboot alarm
Access	App server	Oracle DB, Postgre DB, Real-time DB	remote system discovery AND obtain valid accounts	bypass input validation
Discover	App server	Antivirus server	remote system discovery AND obtain valid accounts	server compromise
Discover	App server	Backup server	remote system discovery AND obtain valid accounts	server compromise
Access	App server	Product	MiM	product infection, product corruption
Access	Host	Zone	gain initial access	server compromise
Compromise	User Account	Host	remote system discovery	brute force, bribing SCADA Engineer, phish SCADA Engineer
Compromise	Admin Account	Host	remote system discovery	brute force
Compromise	Service Account	Host	remote system discovery	brute force, bribing SCADA engineer
Physical access	Diode	Zone	physical shutdown	gain initial access
Bypass	Firewall	Zone	bypass firewall	gain initial access
Reconfigure	Firewall	Router	access router	reconfigure router
Access	Host	Directory Service	remote system discovery AND obtain valid accounts	server compromise

Table 4.3: Associations between assets and encoded attacks

4.7 Potential Mitigation for SCADA

In this section we will propose potential mitigation strategy for the attacks presented in section 4.5. The mitigation strategy is based on MITRE and, in case we cannot get information from MITRE, we also add mitigation strategy from CISA: Industrial Control Systems [62]. Tables in this section provide the protection mechanisms for attacks per each asset in the scope of this threat model for SCADA.

4.7.1 ICCP server (frontend and backend servers)

Attack	Protection
ICCPS2	Multifactor Authentication
ICCPS5	Hardening Profile
ICCPS3	IDS, Hardening Profile

Table 4.4: List of protection mechanisms for Inter-control communication protocol (ICCP) asset in SCADA

IDS is a tool to protect the system by raising the alarm whenever the security of the system was compromised, such that a Site security officer (SSO) can perform countermeasure actions [63]. The difference between IDS and traditional firewalls is their capabilities. The traditional firewall has a set of rules applied to deny or allow traffic in a network. A firewall prevents the intrusion if we define a proper set of rules for a network. Disadvantages of such a system are the “all-or-nothing" approach [64] and lacking an alarm, and, hence, ignoring the system’s potential vulnerability. On the contrary, IDS monitors the network system, uses the common pattern from previous attacks (signatures), and starts an alarm in case of suspicious activity. IDS reports these activities for SSO for further investigations. Additional component and extension to IDS,Intrusion Prevention System (IPS), prevents detected intrusions and performs access control [65].

Next Generation Firewall (NGFW) combines the functionality of both. Additional functionality for NGFW has a deeper inspection mechanism with the help of antivirus, anti-spam, and IDS allows it to detect more unintended activity than a traditional firewall can do [64]. There are several products in this market, e.g. *Fortigate*[11]

Nevertheless, SCADA is different from the classic OT system, and, therefore, the detection and prevention tools might not meet the requirements of SCADA [66]. Zhu and Sastry [66] mentions that academic and industry community made an effort to adapt IDS specifically for SCADA. According to Zhu and Sastry [66], the main issues are as follows:

[11] https://www.fortinet.com/products/next-generation-firewall.html. Accessed on 29.06

- The simulated data can be different from real-life measurements for gas, oil, electricity, and others. Such separation can cause difficulty in differentiating between abnormal and desired data to be sent to CC.
- SCADA heavily depends on the ability to have a 24/7 availability with CC getting data from ICCP server and taking a decision process. Therefore, the authentication mechanism between field units and CC needs to detect the malicious commands in a real-time while having a continuous response. However, the authentication mechanism can complicate the work of IDS.
- SCADA has a static topology with simple data transmission protocols as described in 2.1.3

The solutions proposed by Zhu and Sastry [66] are *Model-Based IDS using Modbus/TCP*,*Anomaly-Based Intrusion Detection*, *Configurable Embedded Middleware-Level Detection*, *Security-Hardened Attack Resistant Platform (SHARP)* and other probabilistic approaches. *Hardening* is a set of rules in OS that prevents some unintended actions.

4.7.2 RTU

In the scope of this threat model, we cannot provide a protection mechanism for RTU, since we do not focus on attacks towards RTU. Instead we consider only that RTU is an entry surface for attacks as mentioned in 4.5.1.

4.7.3 Communication front end

Attack	Protection
CF1	Multifactor Authentication
CF2	Hardening Profile
CF5	On-demand Account
CF8	IDS
CF10	Hardening Profile

Table 4.5: List of protection mechanisms for Communication front end asset in SCADA

Similar to 4.7.1, the IDS could be used to detect abnormal activity in the communication between App Server and Communication front end. On-demand accounts can be done by allocating admin account for a limited period of time and only upon a request.

4.7.4 HMI + Thin client

Attack	Protection
HMI2	Multifactor Authentication
HMI3	IDS
HMI4	Multifactor Authentication
HMI5	On-demand Account
HMI7	Hardening Profile, Scheduled Backup, Malware Detection and Antivirus

Table 4.6: List of protection mechanisms for Human Machine Interface (HMI) asset in SCADA

To protect access to important components of HMI we should use multi-factor authentication. As for *HMI7*, various protection mechanisms against a ransomware, e.g. LockerGoga, include regular backup on the server, additional layer of control of application, server and traffic, strengthen the access control policy [67].

4.7.5 Alarm

The possible way to mitigate unauthorised access to Alarm is to use IDS and monitor the network.

4.7.6 App server

Attack	Protection
APS3	Encryption
APS5	On-demand Account
APS10	Network Segregation
APS10	IDS

Table 4.7: List of protection mechanisms for App server asset in SCADA

As mentioned in 4.5.7, for the scope of this work, App servers are composed of Linux servers. Security-enhanced Linux (SELinux) is an extension for Linux Kernel that enhances the standard Linux restrictions for access permissions by providing Mandatory Access Control (MAC) list. As for Network Segregation to encounter *APS10.Man-in-the-middle attack* from App server, there are protection mechanisms in MITRE T830[12]. One such mitigation is that we should ensure that ICS and IT

[12] https://collaborate.mitre.org/attackics/index.php/Technique/T830. Accessed on 29.06

networks cables are kept separate. IDS should report any abnormal activity in the communication between App servers and other SCADA components.

4.7.7 Databases

Attack	Protection
DB2	Input Sensitisation
DB2	Prepared Statement
DB3	Input Sensitisation
DB3	Prepared Statement
DB5	Encryption
DB7	On-demand Account

Table 4.8: List of protection mechanisms for Database assets in SCADA

Database assets, like Real-time database (RDB) and Postgres/Oracle DB, require constant protection as they enable the operation of SCADA. The logs in DBMS could be encrypted and accessed by DBMS Admin user. DBMS Admin user should be given an account and access only on demand and for a limited period of time.

4.7.8 Antivirus Server

Attack	Protection
AV2	Hardening Profiles
AV6	IDS
AV7	On-demand Account

Table 4.9: List of protection mechanisms for Antivirus Server asset in SCADA

An example of hardening rules is not allowing the default credentials on Antivirus server.

4.7.9 Backup server

Attack	Protection
BS4	Data Encryption for a Storage
BS4	On-demand Account
BS5	IDS

Table 4.10: List of protection mechanisms for Backup Server asset in SCADA

In SCADA all the previous information about the state of the system and metrics are preserved in a backup server. This asset has a high value for an attacker, as it contains the sensitive industrial information. The access should be granted only upon a request for a limited period of time. Moreover, it is necessary to constantly check the data in the backup servers and monitor server failures. IDS should monitor the traffic to back up server especially on off-hour times, when the traffic is not expected.

4.7.10 Directory Service

SCADA system has providers of directory service. For this work, we focus on AD from Windows.

Attack	Protection
AD1	On-demand Admin Account

Table 4.11: List of protection mechanisms for Directory Service asset in SCADA

As for the defense mechanisms for Directory Service, the most important is to protect the access to admin account on Directory Service.This can be achieved through on-demand accounts.

4.7.11 Product

Attack	Protection
PR1, PR2	Application Security Guidelines

Table 4.12: List of protection mechanisms for Product asset in SCADA

4.7.12 Other servers: DNS, NIS, NTP

As stated in 4.2, these services have a less importance in this work. Therefore, we do not focus on defense mechanisms for DNS, NIS, and NTP.

4.7.13 Data Engineering / new HMI server

Attack	Protection
DE/nHMI6	Whitelists

Table 4.13: List of protection mechanisms for Data Engineering (DE)/new Human Machine Interface (nHMI) asset in SCADA

Whitelisting includes limiting the hosts to discover from DE/nHMI server. This also includes the protocols that DE/nHMI server does not allow, e.g. Inter Control Message Protocol (ICMP) ping.

4.7.14 Accounts

Attack	Protection
AC1, AC2	Multi-factor Authentication for User Account

Table 4.14: List of protection mechanisms for Accounts asset in SCADA

As mentioned in section 2.1.2 SCADA system focuses on availability rather than confidentiality and integrity of data. Thus, controlling authorisation and access levels is important.

4.7.15 Firewall

The majority of SCADA implementations have the *security gateways* located at the border of the zone to manage the traffic flow. Security gateways (for instance, firewalls) resolve the features as the network traffic management and security screening of the zone [68]. Firewalls are general and standard solutions for security between zones, but have security flaws, e.g., missed security patches, configuration mistakes, A lack of deep packet inspection or Distributed denial-of-service (DDoS) attacks. More advanced and stronger security solutions between zones include data diodes.

Stouffer, Falco, and Scarfone [36] defines *data diode* as "is a network appliance or device allowing data to travel only in one direction". Since data diodes are hardware solutions without connection to the Internet, we can eliminate any possible software-based attacks [69]. Nevertheless, this solution is expensive and mainly used for the military, where the network zones are categories into low and high-security zones [70]. In addition to that, the solution of using data diodes has some drawbacks: using a one-directional User datagram protocol (UDP) protocol over two-directional Transmission control protocol (TCP), which does not ensure the safety and reliability of data transmission [71]. Therefore, we cannot guarantee that this solution can work as a panacea for all issues related to communication systems.

4.7.16 Zones

Giani et al. [17] mentioned that the prevalent protocols used for communication between components in SCADA lack authentication and confidentiality mechanisms. In order to resolve this issue, *IEC 60870-5-104* suggests protection mechanism as constant monitoring for abnormal activity [22]. Nevertheless, Maynard, McLaughlin, and Haberler [72] raise a concern that in a real-life case, the protection mechanism is delayed due to legacy equipment, operation, and cost management.

The importance of protecting the physical layer is connected with the robust capabilities of Adversary when it has physical access to the system and communication networks [5]. Such attacks towards a physical layer have an impact on the whole SCADA. The primary strategy in this category is IDS and appropriate firewall policies.

4.8 Risk Assessment

As discussed in section 3.3, we map a Severity score per an attack into MAL probability distribution. In chapter 6, we discuss how these probability values affected the selection of attack paths by an adversary. Tables in this section present these scores and how attacks affect CIA properties.

For evaluating the Attack severity score, we computed an average score per each factor in CVSS framework for several CVE vulnerabilities that we can exploit to make an attack. Table 4.15 demonstrates an example of computing Attack severity score for an attack *HMI4. Modify alarm settings*.

CVSSv3 factor	**CVE-2016-5787**	**CVE-2020-6992**	**CVE-2018-17904**	**HMI4**
Attack Vector (AV)	Local	Local	Network	**Local**
Attack Complexity (AC)	Low	Low	Low	**Low**
Privileges Required (PR)	Low	High	None	**Low**
User Interaction (UI)	Required	None	Required	**Required**
Scope (S)	Changed	Unchanged	Changed	**Changed**
Confidentiality (C)	Low	High	Low	**Low**
Integrity (I)	Low	High	Low	**Low**
Availability (A)	Low	High	None	**Low**
Result of attack severity	Medium	Medium	Medium	**Medium**

Table 4.15: CVSS for vulnerabilities and mapping to HMI4 attack step

Further Attack severity computation follows a similar logic. However, this approach has limitations:

- The Attack severity is an average of scores. This approach has limitations, e.g. for *Attack complexity* factor, average of *Low, High* is *Medium*, but *Medium* does not exist for *Attack complexity* factor in CVE scoring system. The better approach would have been using a Mode[13]. Nevertheless, using a Mode of scores also has limitations, e,g., a Mode for *None, None, High* is *None*, which does not properly represent the score.

- The CVE scores for the same product vary based on the vendor and version of the model. We compute an average score for SCADA vulnerabilities that we

[13]Mode is the value that appears most often in the data set

found from National Vulnerability Database (NVD). However, each SCADA system has different vendors. There could be vendors or versions of a model that are not in this particular SCADA system. Therefore, the score might get changed significantly.

4.8.1 ICCP server (frontend and backend servers)

Attack	CIA Risks	Severity
ICCPS1	CI	1.5
ICCPS1	CI	2.0
ICCPS3	A	2.5
ICCPS4	IA	3.0
ICCPS5	CI	2.0

Table 4.16: Severity scores and CIA impact for Inter-control communication protocol (ICCP) attacks

4.8.2 RTU

Attack	CIA Risks	Severity
RTU1	A	2.5

Table 4.17: Severity scores and CIA impact for Remote Terminal Unit (RTU) attacks

4.8.3 Communication frontend

Attack	CIA Risks	Severity
CF1	CI	2.0
CF3	CI	1.5
CF4	CA	1.5
CF5	CIA	4.0
CF6	A	2.0
CF7	A	4.0
CF8	A	2.5
CF9	IA	3.0
CF10	CI	2.0

Table 4.18: Severity scores and CIA impact for Communication Frontend attacks

4.8.4 HMI + Thin client

Attack	CIA Risks	Severity
HMI1	CI	1.5
HMI2	CI	2.0
HMI3	CIA	1.5
HMI4	I	4.0
HMI5	CI	4.0
HMI7	A	3.0

Table 4.19: Severity scores and CIA impact for Human Machine Interface (HMI) attacks

4.8.5 App server

Attack	CIA Risks	Severity
APS1	CI	2.0
APS2	CI	2.5
APS3	CI	1.5
APS4	CA	1.5
APS5	CIA	4.0
APS6	A	2.0
APS8	A	2.0
APS9	CIA	4.5
APS11	A	3.0

Table 4.20: Severity scores and CIA impact for App server attacks

4.8.6 Databases

Attack	CIA Risks	Severity
DB1	CIA	2.0
DB2	C	4.5
DB3	CI	2.0
DB5	CI	1.5
DB6	CI	2.0
DB7	CI	4.0
DB8	CIA	4.5

Table 4.21: Severity scores and CIA impact for Database attacks

4.8.7 Antivirus Server

Attack	CIA Risks	Severity
AV1	CI	2.0
AV3	CI	1.5
AV4	CA	1.5
AV5	A	2.0
AV6	CIA	1.5
AV7	IA	4.0
AV8	IA	3.0
AV9	CI	2.0

Table 4.22: Severity scores and CIA impact for Antivirus Server attacks

4.8.8 Backup server

Attack	CIA Risks	Severity
BS1	C	3.8
BS3	CI	3.5
BS4	CIA	4.0
BS5	CIA	4.5

Table 4.23: Severity scores and CIA impact for Backup Server attacks

4.8.9 Directory Service

Attack	CIA Risks	Severity
AD1	CIA	4.0
AD2	IA	3.0
AD3	CI	2.0
AD4	A	2.5

Table 4.24: Severity scores and CIA impact for Directory Service attacks

4.8.10 Product

Attack	CIA Risks	Severity
PR1	IA	4.0
PR2	A	3.5

Table 4.25: Severity scores and CIA impact for Product attacks

4.8.11 Data Engineering / new HMI server

Attack	CIA Risks	Severity
DE/nHMI1	IA	3.0
DE/nHMI2	CI	2.0
DE/nHMI3	CI	2.0
DE/nHMI4	CI	2.5
DE/nHMI5	CI	1.5
DE/nHMI6	CA	1.5
DE/nHMI7	CIA	4.0
DE/nHMI8	CA	4.5

Table 4.26: Severity scores and CIA impact for DE/nHMI Service attacks

4.8.12 Accounts

Attack	CIA Risks	Severity
AC1	CI	3.5
AC2	CIA	3.5
AC3	CIA	4.0

Table 4.27: Severity scores and CIA impact for Account attacks

4.8.13 Firewall

Attack	CIA Risks	Severity
FW1	A	3.5

Table 4.28: Severity scores and CIA impact for Firewall attacks

5 Creating a Threat Model for a SCADA Instance Using *scadaLang*

In this chapter we provide an explanation for the tool that generates a threat model for an instance of SCADA using *scadaLang*.

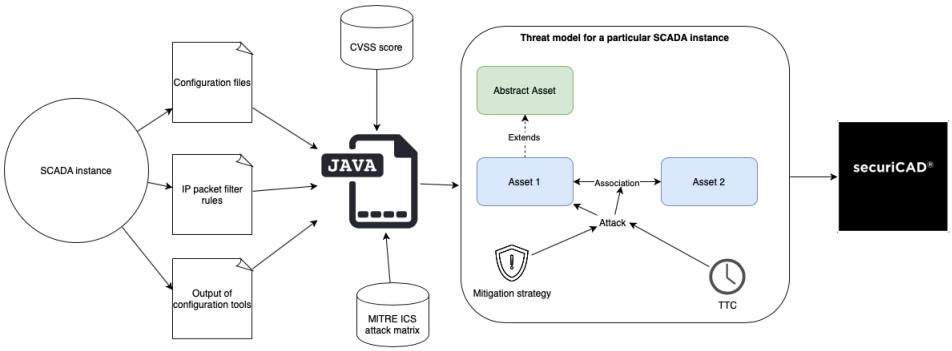

Figure 5.1: Creating a threat model for a particular SCADA instance based on *scadaLang*

Figure 5.1 presents how we can get generate a threat model for a particular SCADA instance from configuration files, and IP packets routes described in section 3.2.2. Each SCADA instance has configuration files that describe how the components of SCADA are connected with each other. We use the information in configuration files to map into assets and associations defined by *scadaLang* in section 4.2. The MITRE attacks defined in section 4.5 is collected into a JSON file. The Severity scores described per each attack in section 4.8 and protection mechanisms described in section 4.7 are also collected into JSON files. The output of this work are *scadaLang* files (*.mal) for a SCADA instance. The generated (*.mal) file can then be packaged into (*.scad) to be opened with be *SecuriCAD* software developed by *Foreseeti*[14]. The software allows us to make attack simulations that will be explained further in the upcoming chapter 6.

The configuration file that we use for parsing needs to match the following format (.JSON):

```
"Host":{
  ""<host name>":{
    "IPAddr":<ip address>
    "LAN":<zone name>
  },
  <all hosts>
}
"Network":{
  <Zones>
}
"UserAccounts":{
  <Accounts>
}
"appl_server":{
  ""<App server name>":{
    "account":<account name>
    "additional_nodes": <host names>
    "applications":<product names>
    "dbi":<database name>
    "supervised_by":<app server name>
```

[14] https://www.foreseeti.com. Accessed on 29.06

```
  }
  <all app servers>
}
"backup_server":{
  <backup servers>
}
"de_server":{
  "DATAENG":{
    "PRODUCTS":{
      <products>
    }
    "account":<account name>
    "appl_servers":<app server name>
    "databaseId":<database name>
    "normal_nodes":<host names>
  }
}

"mmi_server":{
  <HMIs>
}
"networkServices":{
  "Inhouse":{
    "DNSServers":<host name>
    "KerberosServers":<host name>
    "NISServers":<host name>
    "NTPServers":<host name>
  }
}
"routing":{
  "Inhouse":{
    "default":{
      ""<Zone name>":<host name>
    }
  }
}
```

The configuration file that matches such format is parsed into assets. In upcoming sections 5.1, 5.2 and 5.3, we discuss the parsed assets. The associations between assets are also defined in the configuration file. For the convenience, we grouped the assets into categories as explained in section 4.2.1.

One example of how the tool parses the configuration data into assets and associations is as follows:

- We have the aliases (*<App server name>*) for application servers under *appl_server*. We create an Asset *App server* for each app server that we parse.
- We parse the values from keys *account, additional_nodes, applications, dbi, supervised_by* to establish the association between an app server and other SCADA components.

Other assets and associations are also created based on parsed information from the configuration file.

This tool needs to get an input configuration file. If the vendor does not have a configuration file, this tool might not be fit. Hence, the vendor needs to fill the information according to the aforementioned structure.

The information file from MITRE knowledge base is collected into the following JSON file:

```
{
"assetsAttackTags":{
   <asset name1>: [<asset tag1>, <asset tag2>]
   <asset name2>: [<asset tag1>, <asset tag2>]
   ...
}
"attacks": {
   <asset tag>: [
      <attacks>
   },
```

```
    ...
},
"potentialAttacksInAssociation": {
    <association name>: [<attack asset 1>, <attack asset 2>],
    ...
},
"attacksChain": {
    <asset tag>: [
        <attack name>: [<attacks>]
    ],
    ...
}
```

The Attack severity scores are written in the JSON file:

```
{
<attack name1>: <score1>,
<attack name2>: <score2>,
...
}
```

5.1 System Assets

Assets 4.5.2 - 4.5.15 are grouped together into the category of *System assets*.

Regarding Communication front end, OS for Communication front end can vary based on instance of SCADA. Therefore, for this work in 4.5.4 *scadaLang* presents OS-independent attack techniques.

Based on provided SCADA configuration, we can have different DBMS. Therefore, we can have either *OracleDatabase* or *PostgreDatabase*. Based on information in configurations, our tool selects the attacks that are relevant to the particular DBMS.

Logically, HMI has a GUI for Operators described in section 4.3. Thin Client can be located in a separate zone, e.g., Production zone. In particular configurations, HMI and Thin Client are located in the same machine.

As for the antivirus servers, backup servers, DE servers, and nHMI servers, not all instances have these servers. When our tool generates a threat model for an instance of SCADA, the servers mentioned above are added if they are present in the configuration.

Directory Servers (e.g., AD, Redhat Directory Server) are assets responsible for managing authorisation and access. Based on the configuration settings, Directory Servers can be Windows- or Linux-based.

5.2 Security Assets

Assets in section 4.5.16 are grouped together into the category of *Security assets*. Overall, *scadaLang* covers “User Account", “Service Account" and “Admin Account" assets discussed in 4.5.16. At least one administrative account should be present for our tool to generate a threat model in a SCADA instance.

5.3 Communication Assets

Assets in 4.5.17- 4.5.20 are grouped together into the category of *Communication assets*. Security zones in SCADA system is a highly customisable part of the system. The names of zones can differ, as well as how other assets are located in zones.

Overall, as mentioned in section 2.1.4, industry standards define the number of zones for a system like SCADA. Following these, our tool parses the zones into assets and connects to the hosts located in the zone.

Not all information about the communication can be fully inferred from configuration files. Our tool needs to get other information as DNS rules. For the future improvements, *Zone* asset should be parsed as *Trust boundary* instead.

6 Results of Attack Simulations

In this chapter, we discuss the results of the attack simulations. Each attack has an entry point as described in section 4.5.1, and leads to an impact defined by MITRE as explained in section 4.5. Section 6.1 describe an attack scenario with an impact on availability of SCADA. In section 6.2, we can see an example of attack simulation that leads to theft of operation information, while in section 6.3 we can examine an attack simulating with the purpose of loss of control. Section 6.4 demonstrates one example of the loss of safety, however, in this work, we do not cover manual system misconfigurations that could lead to harm to a human operator due to the loss of safety.

6.1 Loss of Availability

Loss of availability is the process when the essential components or the system were disputed, which led to the failure of delivering the services or products.

Figure 6.1 demonstrates how an attacker can stop the service at Communication Frontend starting from RTU.

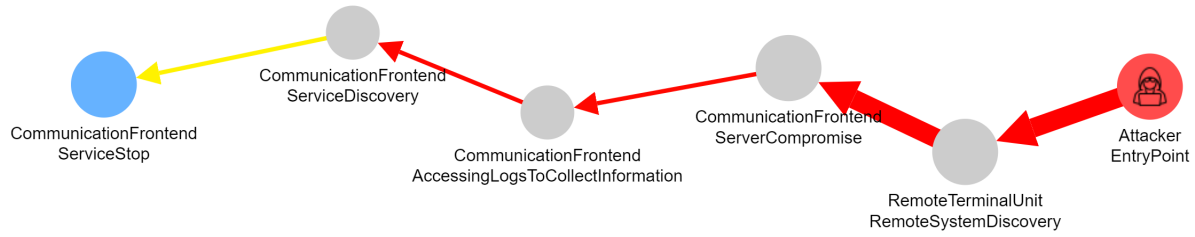

Figure 6.1: Attack steps to shut down Communication frontend

In Figure 6.1, we can observe the attack simulation result produced by SecuriCAD. According to the documentation of SecuriCAD, the yellow paths are more difficult to perform than red ones. Moreover, the thick lines are more important than thin ones. As for the exclamation mark, the attack steps that have it can be reached with additional paths [73].

Figure 6.2 demonstrates additional paths that were not selected by an attacker.

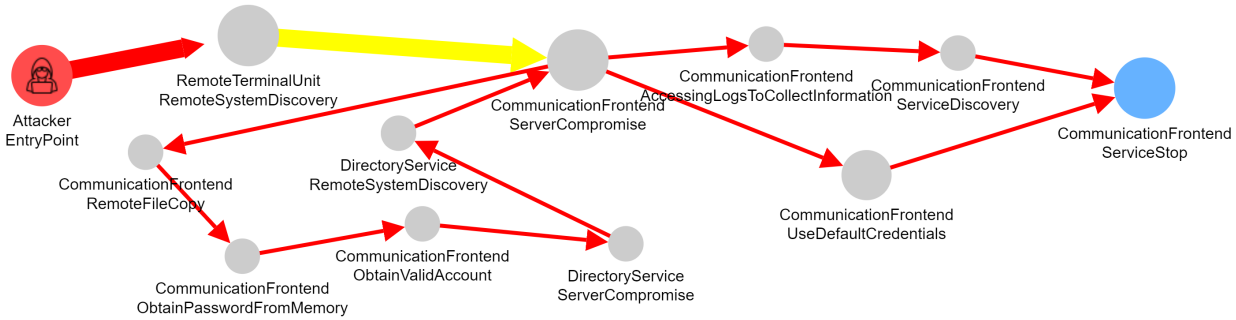

Figure 6.2: Alternative path to shut down Communication frontend

As mentioned in section 4.8, each step has a probability distribution. Based on this value, an attacker would prefer one attack path instead of other. Figure 6.3 summarises the alternative paths depicted in Figure 6.2. The red path in the figure demonstrates the least difficult and most certain attack path. The grey and yellow paths denote the alternative path that was not selected for this simulation.

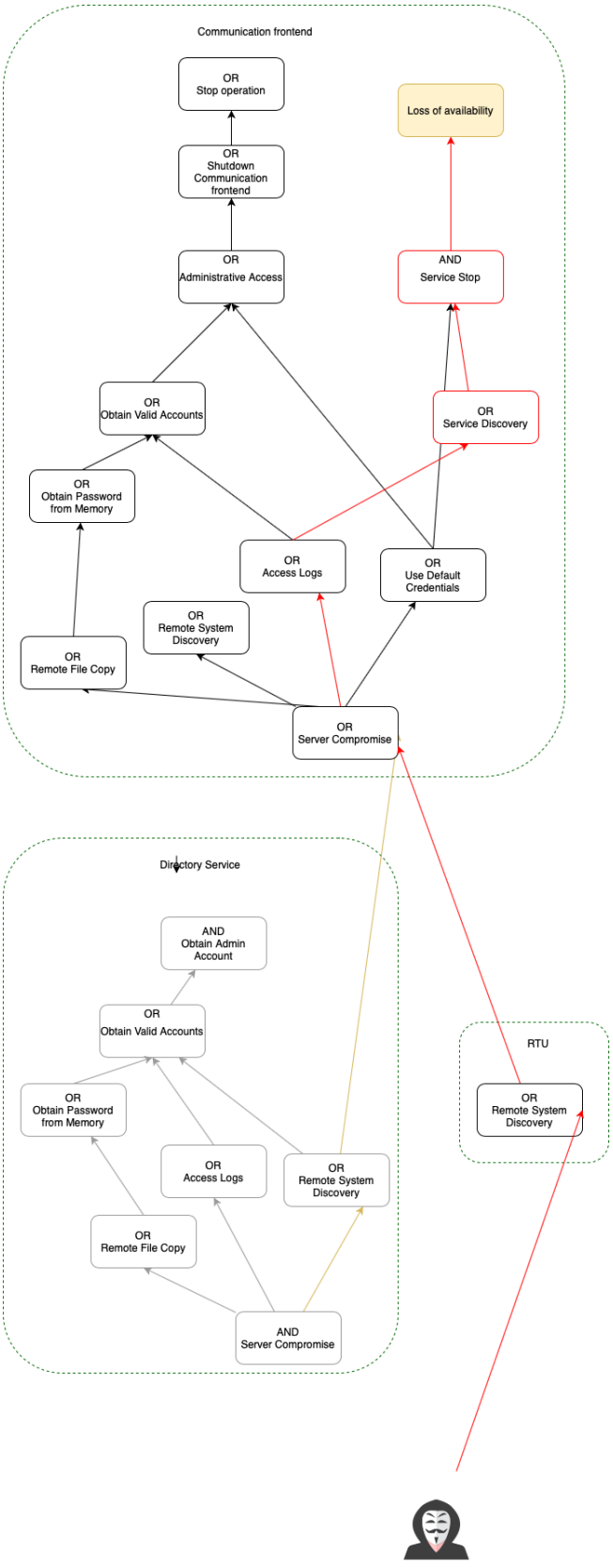

Figure 6.3: Full attack steps to shut down Communication frontend

6.2 Theft of Operational Information

Theft of operational information is the process when an attacker obtains access to operational information in a production environment to gain industrial knowledge or learn about future operations in the system.

Figure 6.4 demonstrates how an attacker can get into *Backup server* starting from RTU.

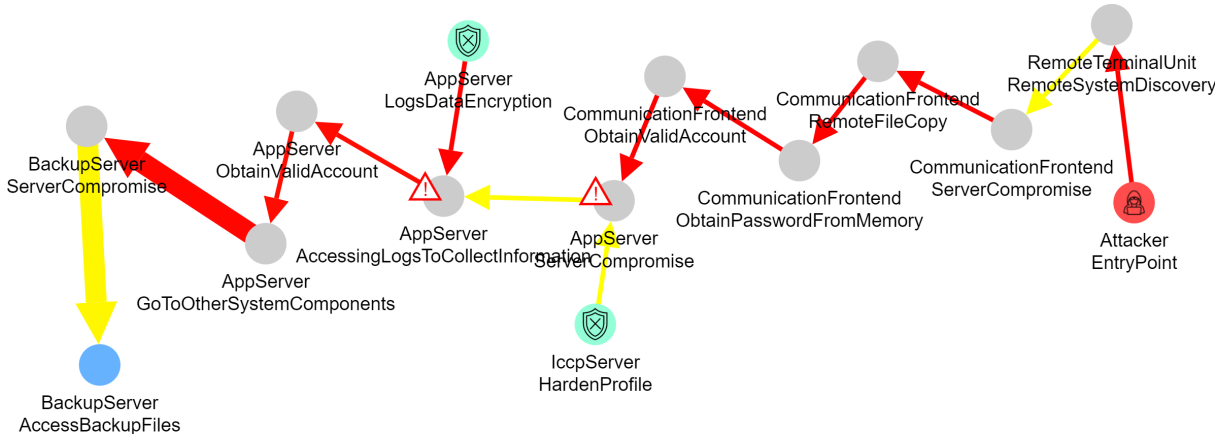

Figure 6.4: Attack steps to steal operational information

In this scenario, an attacker could access the files stored in a backup server, which leads to disrupting the confidentiality properties of the system.

Same as Figure 6.1, in Figure 6.4, we can observe the attack simulation result produced by SecuriCAD. According to the documentation of SecuriCAD, the yellow paths are more difficult to perform than red ones. Moreover, the thick lines are more important than thin ones. As for the exclamation mark, the attack steps that have it can be reached with additional paths [73].

Figure 6.5 summarises the full path depicted in Figure 6.4, where an attacker could choose to go through ICCP server. SecuriCAD did not generated additional paths for this attack. The red path in the figure demonstrates the least difficult and most certain attack path. The grey and yellow paths denote the alternative path that was not selected for this simulation.

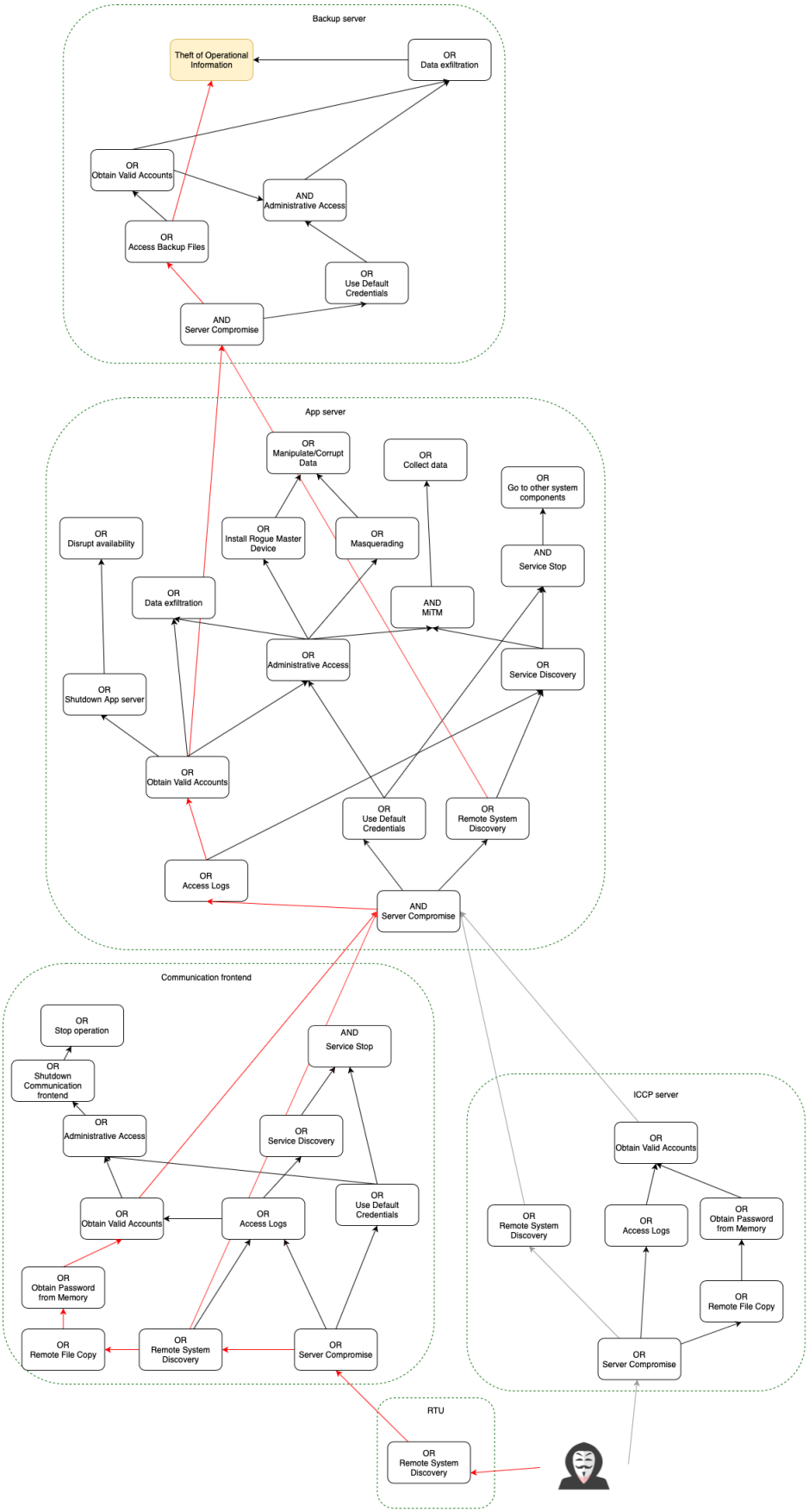

Figure 6.5: Full attack steps to access files in Backup server

6.3 Loss of Control

A loss of control is a process when an operator is not capable of running any commands. Figure 6.6 demonstrates how an attacker can get into *App server* starting from RTU.

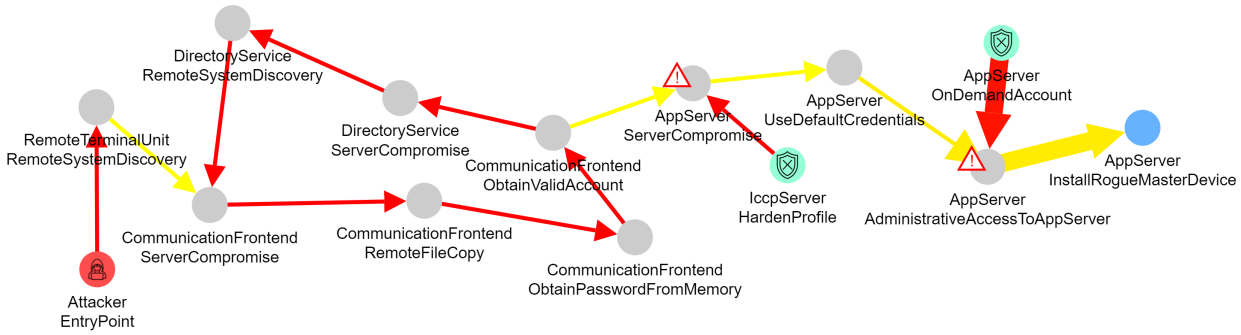

Figure 6.6: Attack steps to control slave devices

In this scenario, an attacker would need to install a rogue master device to leverage control server functions and send malicious commands to slave devices. This action affects the integrity of communication between master and slave devices, since an attacker may impersonate the genuine master device.

Figure 6.7 summarises the full path depicted in Figure 6.6. SecuriCAD did not generated additional paths for this attack. The red path in the figure demonstrates the least difficult and most certain attack path. The grey and yellow paths denote the alternative path that was not selected for this simulation.

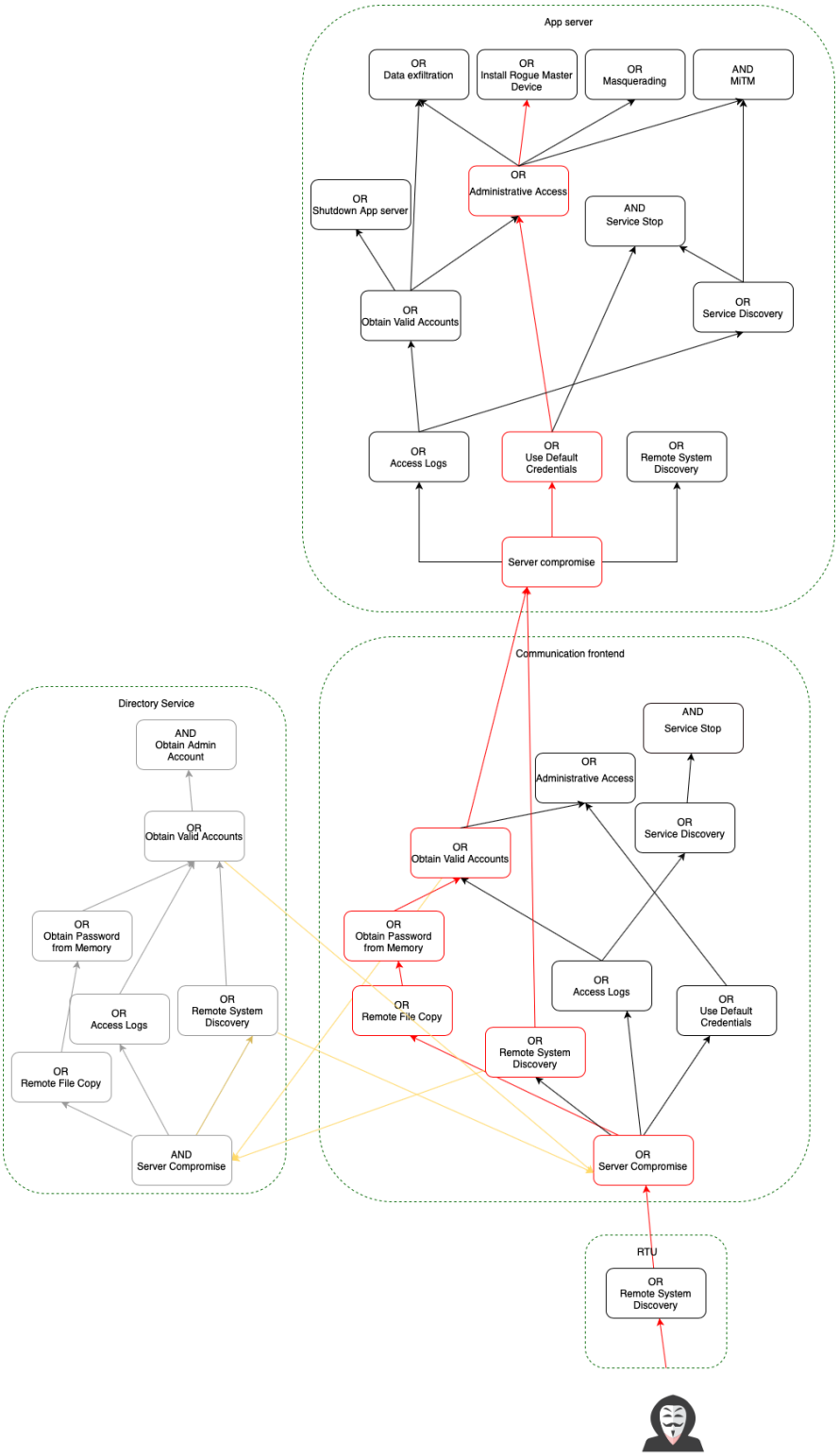

Figure 6.7: Full attack steps to install a rogue master device

6.4 Loss of Safety

A loss of safety is a process when a particular set of operations is not accomplished, leading to unsafe conditions or threats to the system's operation. One such example is starting the wrong alarm or dismissing the actual alarm. In SCADA systems, in the worst case, this can lead to missing a failure of a critical process. Figure 6.8 demonstrates how an attacker can get into *Alarm* starting from HMI.

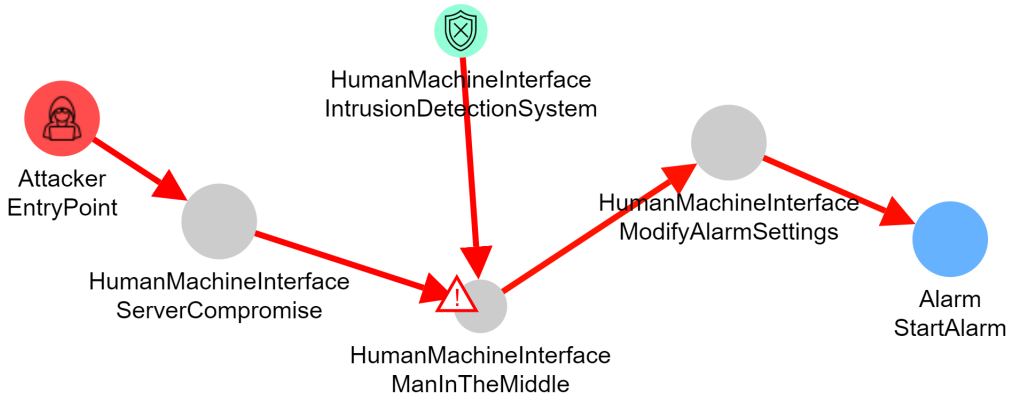

Figure 6.8: Attack steps to start the wrong alarm

In this scenario, an attacker could connect to *Alarm* through *HMI* to start a wrong alarm. In the worst case, an attacker would dismiss an alarm, which would allow the attacker to hide his presence or facilitate reaching further goals. Figure 6.9 provides the alternative paths that attacker did not choose.

Same as Figure 6.1, 6.4 and 6.6, in Figure 6.8, we can observe the attack simulation result produced by SecuriCAD. According to the documentation of SecuriCAD, the yellow paths are more difficult to perform than red ones. Moreover, the thick lines are more important than thin ones. As for the exclamation mark, the attack steps that have it can be reached with additional paths [73].

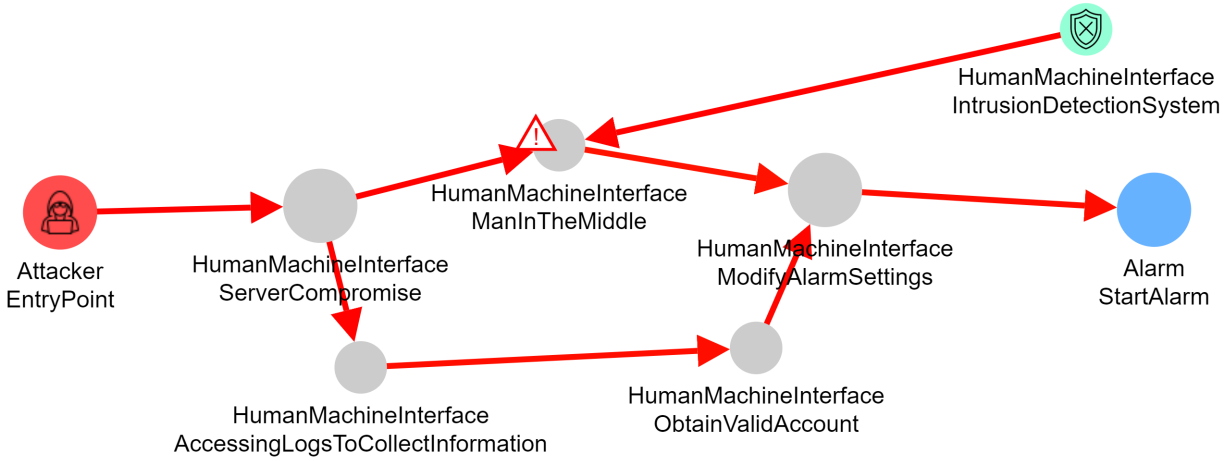

Figure 6.9: Alternative attack paths to start the wrong alarm

Figure 6.10 describes the full attack paths to *Alarm*. The red attack steps denote the path that is more likely to be chosen by an attacker, as it is less difficult and more certain. The grey and yellow attack steps constitute an alternative attack path described in Figure 6.9

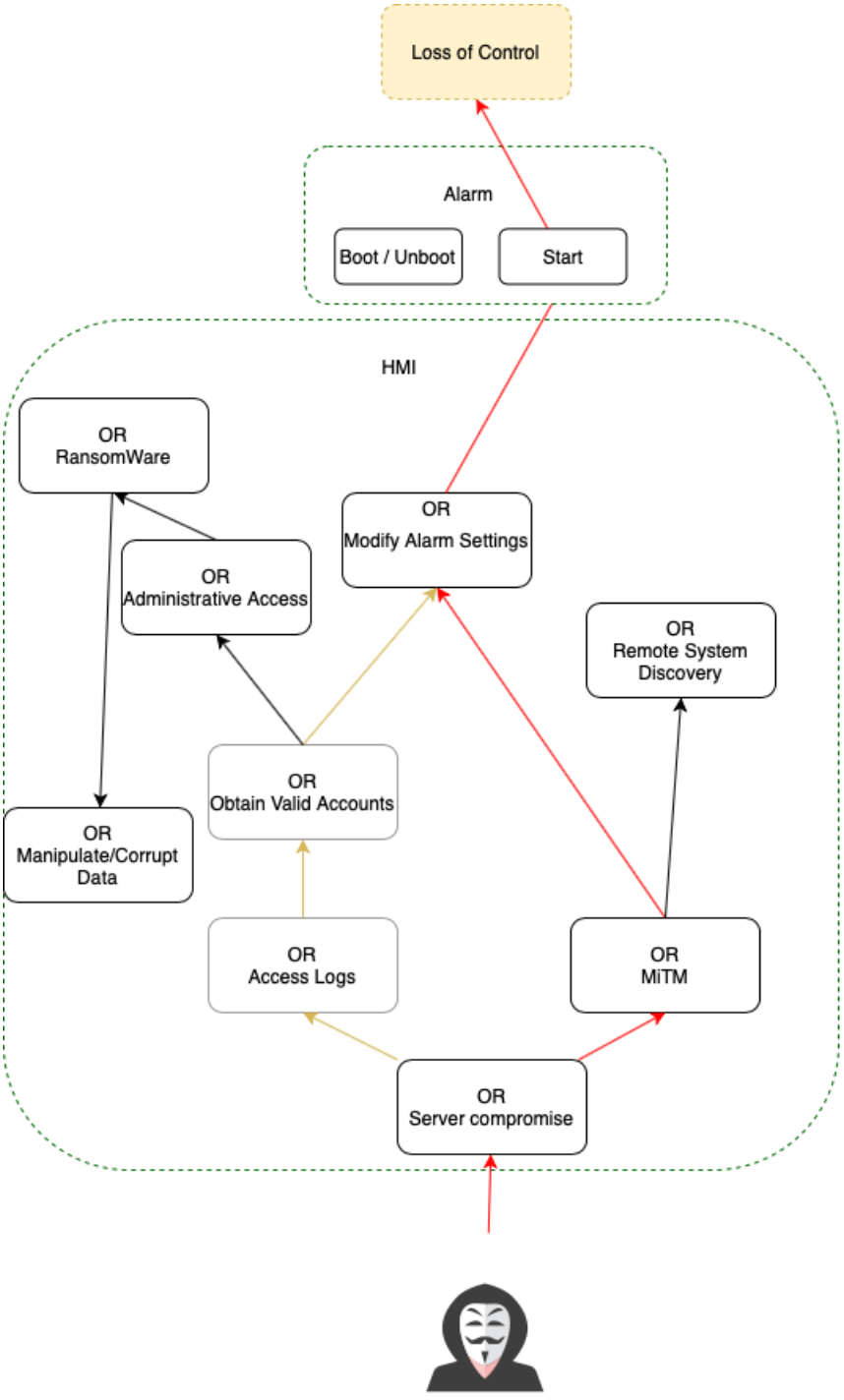

Figure 6.10: Full attack paths to start the wrong alarm

7 Discussion, Limitations, and Future Work

In this chapter, we discuss our work, state the limitations, and propose some improvements. In section 7.1, we discuss if our solution is valid or not, if our approach and methodology were right, and address related work. In section 7.2, we state the limitations of our work and explain we decided not to include them in this work. In section 7.3, we propose some improvements and future work.

7.1 Discussion

This work presents a *scadaLang* and a tool for generating a threat model for a specific instance of SCADA using *scadaLang*. SCADA experts validated *scadaLang*, and an example of a threat model that a tool can generate. *scadaLang* combines the research and technical knowledge, since we use MITRE knowledge base with attacks specific for a SCADA system, and use configuration files from a real SCADA system to understand the components and their connections. Therefore, *scadaLang* resolves the problem of previous works in this field, where the tools could not either match SCADA requirements or proposed attacks that are not suitable for a SCADA system. Regarding the approach and methodology for this work, we used the CDITO method for this work and found it convenient and applicable for creating *scadaLang* and the tool. Moreover, we propose other methods as surveying for enhancing the validity of *scadaLang*.

As for mapping of the Attack severity score into TTC for an attack in *scadaLang*, we generated our own model. The mapping is based on many assumptions and simplifications. We compute an average of CVSS scores of CVE vulnerabilities from NVD that could lead to this particular attack. Computing average has limitations, which could give us a change in the Attack severity score. In addition to that, CVE vulnerabilities of products depend on the version, model, and vendor, which also affects the Attack severity score.

7.2 Limitations

This work had some limitations in the implementation that is the topic of further improvement. First of all, the work did not include any manual configuring of safety instructions. As mentioned in MITRE, one of the attacker goals is causing an outrage that could potentially lead to harm for a SCADA engineer [74]. Therefore, SCADA systems have safety instructions on the components of the system, e.g., tags attached to hardware, or Safety Instrumented Systems (SIS) to automate actions to keep safety in a plant. An attacker might try to misconfigure these safety systems manually. However, in this work, we do not consider such types of attacks. Secondly, the scope of the work is limited to assets and attacks in the CC. Any attacks towards field units, RTUs, are not considered in the language specification. In this work, we focus on a single attack from RTU to the Communication front end and consider RTU as an entry point for attackers.

Additional limitations of this work include suggested mapping of CVSS score into

attack difficulty and certainty. A possible further improvement to this work might be enhancing the proposed model of computing the Time-To-Compromise (TTC) with considering additional factors that are relevant to SCADA, e.g., zones and required availability of critical components.

7.3 Future work

First of all, one further improvement to the project includes obtaining information from other files, e.g., DNS records. Currently, we use the configuration files for collecting assets and associations in *scadaLang*. We use MITRE knowledge base for the information for attacks and mitigation strategies. Using *scadaLang* we can generate a threat model for a given SCADA instance. By considering the supplementary information provided in other files, we can cover further metrics for attacks and mitigation strategies, which would increase its accuracy.

The other future work is an enhancing of validation of *scadaLang* through the Turing test. To ensure that the attacks presented in this works are not distinguished from attacks made by a human, this work can utilize a Turing test. A Turing test relies on the judgment and analysis of experts on how this work performs. For achieving better results, it is vital to access the proposed model over time continuously.

The third future work is multiple attacker profiles. In this work, we only consider a single attacker profile, choosing depth over breadth. A future work proposal would be to explore the other attacker profiles interested in cyber-attacks towards SCADA. Table 7.1 proposes a number of those attackers profiles to considered for SCADA threat modeling. Script kiddies are less motivated to attack SCADA due to the low return value and high complexity of the system compared to their low skills and resources. On the other hand, ethical hackers are more motivated by the complexity of the system and aim to document and disclose their findings. Hacktivists, in particular political hacktivists, have a higher motivation to perform malicious actions on SCADA due to its critical importance. *State-sponsored groups*, *Cyber-criminals* and *Rogue Employees* pose the highest risk to SCADA due to their high skill level, vast resources, and strong motivation towards malicious intent. State-sponsored groups target critical infrastructure systems, like SCADA, in order to politically attack a country.

Attacker type	**Motivation**	**Skills**	**Resources**	**Examples**
Script kiddies	Low	Low	Low	-
Ethical hackers	Low-Mid	Mid	Mid	-
Hacktivists	Mid	Mid-High	Low-Mid	-

Cyber-criminals	Mid-High	High	High	Vitek Boden, 2000 Maroochy Shire cyber event [75]
Rogue employees	Mid	Mid	Mid-High	Actors
State-sponsored	High	High	High	Sandworm, ALLANITE, Lazarus group[15]

Table 7.1: List of attacker profiles for SCADA threat model

[15] https://collaborate.mitre.org/attackics/index.php/Groups Accessed on 25.06.2020

8 Conclusions

The security of SCADA has great importance since this category of systems belong to Critical National Infrastructure (CNI). The early SCADA was an isolated monolithic system relying on hardware security. The availability of SCADA was more priority over confidentiality and integrity. Nevertheless, the development of technologies and the necessity of more efficient managing of SCADA led to making the system more comprehensive. In return, this made the system vulnerable to cyber-attacks.

This work presents the *scadaLang*, threat modeling DSL based on Meta Attack Language (MAL). In particular, the work presents the MAL specification of the language prepared for SCADA addressing the potential attacks and present vulnerabilities that are specific to a SCADA system. We use MITRE knowledge base for the attacks and proposed mitigation strategies. In case a mitigation strategy for an attack is not given in MITRE, we use other sources, e.g., CISA: Industrial Control Systems.

Another focus of this work was automating the security analysis of a SCADA system. SCADA is a highly customisable system and can have different setups. This work utilized the information given in configuration files and documents per each system to generate a threat model for each setup. As a result, we developed a software that can create a threat model using *scadaLang* per specific given instance of SCADA.

We collected assets from different configuration files. We discovered that the assets vary per each configuration. Some assets are present in all configurations (App server), while other assets as Backup Server are only available in some of them. To examine how to reach one asset from another, we used the information from configuration files and output of *iptables* command. We discovered three assets that can serve as an entry point for an attacker.

We simulated attack steps for four different SCADA instances. For each attack step, we also enabled a defense mechanism and attached a TTC value. SecuriCAD makes a simulation, and an attacker chooses the least difficult step with a higher probability of success for attack simulation. We made several simulations with different impacts (e.g., loss of control, loss of availability, theft of operational information), and for each attack simulation, we observed additional paths. We discovered that additional paths that an attacker did not proceed with during simulating in SecuriCAD require either more steps to perform or needs to compromise more assets, or have a lower TTC value.

References

[1] Eduardo B Fernandez et al. "On building secure SCADA systems using security patterns". In: *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*. 2009, pp. 1–4.

[2] Yulia Cherdantseva et al. "A review of cyber security risk assessment methods for SCADA systems". In: *Computers & security* 56 (2016), pp. 1–27.

[3] Ministry of Justice of Sweden. "A national cyber security strategy". In: (2016). Accessed: 18.04.2020, https://www.government.se/4ada5d/contentassets/d87287e088834d9e8c08f28d0b9dda5b/a-national-cyber-security-strategy-skr.-201617213.

[4] Eric J Byres, Matthew Franz, and Darrin Miller. "The use of attack trees in assessing vulnerabilities in SCADA systems". In: *Proceedings of the international infrastructure survivability workshop*. Citeseer. 2004, pp. 3–10.

[5] Simon Duque Anton et al. "Two decades of SCADA exploitation: A brief history". In: *2017 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE. 2017, pp. 98–104.

[6] Irfan Ahmed et al. "SCADA Systems: Challenges for Forensic Investigators". In: *ABB Corporate Research* (2012).

[7] Pontus Johnson, Robert Lagerstrom, and Mathias Ekstedt. "A Meta Language for Threat Modeling and Attack Simulations". In: *ARES 2018: Proceedings of the 13th International Conference on Availability, Reliability and Security*. 2018, pp. 1–8.

[8] Robert E Johnson. "Survey of SCADA security challenges and potential attack vectors". In: *2010 International Conference for Internet Technology and Secured Transactions*. IEEE. 2010, pp. 1–5.

[9] Jason Stamp et al. "Sustainable security for infrastructure SCADA". In: *Sandia National Laboratories, Albuquerque, New Mexico (www. sandia. gov/scada/-documents/SustainableSec urity. pdf)* (2003).

[10] Bill Miller and Dale Rowe. "A survey SCADA of and critical infrastructure incidents". In: *Proceedings of the 1st Annual conference on Research in information technology*. 2012, pp. 51–56.

[11] Jian Guan, James H Graham, and Jeffrey L Hieb. "A digraph model for risk identification and mangement in SCADA systems". In: *Proceedings of 2011 IEEE International Conference on Intelligence and Security Informatics*. IEEE. 2011, pp. 150–155.

[12] Johannes Schneider, Sebastian Obermeier, and Roman Schlegel. "Cyber Security Maintenance for SCADA Systems". In: *ABB Corporate Research* (2017).

[13] Wang Yong et al. "Stuxnet Vulnerabilities Analysis of SCADA Systems". In: *ABB Corporate Research* (2012).

[14] Axel Daneels and Wayne Salter. "What is SCADA?" In: (1999).

[15] Saurabh Amin et al. "Cyber security of water SCADA systemsPart I: Analysis and experimentation of stealthy deception attacks". In: *IEEE Transactions on Control Systems Technology* 21.5 (2012), pp. 1963–1970.

[16] Jiaping Men et al. "Finding sands in the eyes: vulnerabilities discovery in IoT with EUFuzzer on human machine interface". In: *IEEE Access* 7 (2019), pp. 103751–103759.

[17] Annarita Giani et al. "A testbed for secure and robust SCADA systems". In: *ACM SIGBED Review* 5.2 (2008), pp. 1–4.

[18] Peter Huitsing et al. "Attack taxonomies for the Modbus protocols". In: *International Journal of Critical Infrastructure Protection* 1 (2008), pp. 37–44.

[19] Samuel East et al. "A Taxonomy of Attacks on the DNP3 Protocol". In: *International Conference on Critical Infrastructure Protection*. Springer. 2009, pp. 67–81.

[20] Gordon Clarke, Deon Reynders, and Edwin Wright. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004.

[21] György Dán et al. "Challenges in power system information security". In: *IEEE Security & Privacy Magazine* 10.4 (2012), pp. 62–70.

[22] Yi Yang et al. "Intrusion detection system for IEC 60870-5-104 based SCADA networks". In: *2013 IEEE power & energy society general meeting*. IEEE. 2013, pp. 1–5.

[23] Ensotest. "Introduction to the IEC 60870-5-104 standard". In: (2020). Accessed: 08.08.2020, https://www.ensotest.com/iec-60870-5-104/introduction-to-the-iec-60870-5-104-standard/.

[24] Robert E Mahan et al. *Secure data transfer guidance for industrial control and SCADA systems*. Tech. rep. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2011.

[25] Steven Cheung et al. "Using model-based intrusion detection for SCADA networks". In: *Proceedings of the SCADA security scientific symposium*. Vol. 46. Citeseer. 2007, pp. 1–12.

[26] Eric Byres, John Karsch, and Joel Carter. "Firewall deployment for scada and process control networks". In: *Centre for Protection of National Infrastructure, Government Digital Service* (2005).

[27] Tony UcedaVelez and Marco M Morana. *Risk centric threat modeling*. Wiley Online Library, 2015.

[28] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. "A descriptive study of Microsofts threat modeling technique". In: *Requirements Engineering* 20.2 (2015), pp. 163–180.

[29] Christopher J Alberts and Audrey Dorofee. *Managing information security risks: the OCTAVE approach*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[30] Jack Freund and Jack Jones. *Measuring and managing information risk: a FAIR approach*. Butterworth-Heinemann, 2014.

[31] Robert Lagerstrom. "Attack simulations of viable cities (smart facilities)". In: (2020). Accessed: 22.06.2020, https://www.hbv.se/contentassets/5c2b43290a84481dbe13cc197ebece3d/presentation-robert-lagerstrom-kth.pdf.

[32] MITRE. "Groups". In: (2020). Accessed: 10.05.2020, https://attack.mitre.org/groups/.

[33] Nataliya Shevchenko et al. *Threat modeling: a summary of available methods*. Tech. rep. Carnegie Mellon University Software Engineering Institute Pittsburgh United , 2018.

[34] Ronald S Ross. *Guide for conducting risk assessments*. Tech. rep. 2012.

[35] Pengsu Cheng et al. "Aggregating CVSS base scores for semantics-rich network security metrics". In: *2012 IEEE 31st Symposium on Reliable Distributed Systems*. IEEE. 2012, pp. 31–40.

[36] Keith Stouffer, Joe Falco, and Karen Scarfone. "Guide to industrial control systems (ICS) security". In: *NIST special publication* 800.82 (2011), pp. 16–16.

[37] Umesh Kumar Singh and Chanchala Joshi. "Quantitative security risk evaluation using CVSS metrics by estimation of frequency and maturity of exploit". In: *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 1. 2016, pp. 19–21.

[38] Luca Allodi et al. "Estimating the assessment difficulty of CVSS environmental metrics: an experiment". In: *International Conference on Future Data and Security Engineering*. Springer. 2017, pp. 23–39.

[39] MAL-lang. "Instantiating Language Models". In: (2020). Accessed: 16.07.2020, https://github.com/mal-lang/mal-documentation/wiki/Instantiating-Language-Models.

[40] MAL Compiler. "Supported distribution functions". In: (2020). Accessed: 18.04.2020, https://github.com/mal-lang/malcompiler/wiki/Supported-distribution-functions.

[41] Linzhang Wang, Eric Wong, and Dianxiang Xu. "A threat model driven approach for security testing". In: *Third International Workshop on Software Engineering for Secure Systems (SESS'07: ICSE Workshops 2007)*. IEEE. 2007, pp. 10–10.

[42] Simon Hacks et al. "Creating Meta Attack Language Instances using ArchiMate: Applied to Electric Power and Energy System Cases". In: *2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE. 2019, pp. 88–97.

[43] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. *Model-driven risk analysis: the CORAS approach*. Springer Science & Business Media, 2010.

[44] Jan Jürjens. “Towards development of secure systems using UMLsec”. In: *International Conference on Fundamental Approaches to Software Engineering*. Springer. 2001, pp. 187–200.

[45] Mohamed Almorsy and John Grundy. “Secdsvl: A domain-specific visual language to support enterprise security modelling”. In: *2014 23rd Australian Software Engineering Conference*. IEEE. 2014, pp. 152–161.

[46] Sushil Jajodia and Steven Noel. “Topological vulnerability analysis: A powerful new approach for network attack prevention, detection, and response”. In: *Algorithms, architectures and information systems security*. World Scientific, 2009, pp. 285–305.

[47] Network and Systems Engineering. “Summary of VIKING results”. In: (2020). Accessed: 18.07.2020, https://www.kth.se/polopoly_fs/1.407987.1550158302!/Menu/general/column-content/attachment/FinalProjectReport_WEBversion.pdf.

[48] Salvatore DAntonio et al. “INcreasing Security and Protection through Infrastructure REsilience: The INSPIRE Project”. In: *Critical Information Infrastructure Security: Third International Workshop, CRITIS 2008, Rome, Italy, October 13-15, 2008*. Vol. 5508. Springer. 2009, p. 109.

[49] MITRE foundation. “Overview of MITRE ICS”. In: *MITRE ICS* (2020). Accessed: 18.04.2020, https://collaborate.mitre.org/attackics/index.php/Overview.

[50] CDIO. “Worldwide CDIO initiative”. In: (2020). Accessed: 30.07.2020, http://www.cdio.org.

[51] Helene Leong. “Designing a CDIO Programme: The CDIO Syllabus and Standards”. In: (2020). Accessed: 30.07.2020, http://www.kanazawa-it.ac.jp/cdio/english/file/slide10_leong.pdf.

[52] MITRE foundation. “Assets in MITRE ICS”. In: *MITRE ICS* (2020). Accessed: 18.04.2020, https://collaborate.mitre.org/attackics/index.php/All_Assets.

[53] Marjan Keramati, Ahmad Akbari, and Mahsa Keramati. “CVSS-based security metrics for quantitative analysis of attack graphs”. In: *ICCKE 2013*. IEEE. 2013, pp. 178–183.

[54] First. “Common Vulnerability Scoring System version 3.1: User Guide”. In: (2020). Accessed: 18.04.2020, https://www.first.org/cvss/user-guide.

[55] Hannes Holm, Mathias Ekstedt, and Dennis Andersson. “Empirical analysis of system-level vulnerability metrics through actual attacks”. In: *IEEE Transactions on dependable and secure computing* 9.6 (2012), pp. 825–837.

[56] Reginald E Sawilla and Xinming Ou. “Identifying critical attack assets in dependency attack graphs”. In: *European Symposium on Research in Computer Security*. Springer. 2008, pp. 18–34.

[57] Osman Balci and Eric P Smith. *Validation of expert system performance*. Tech. rep. Department of Computer Science, Virginia Polytechnic Institute & State , 1986.

[58] Ayse Pinar Saygin, Ilyas Cicekli, and Varol Akman. "Turing test: 50 years later". In: *Minds and machines* 10.4 (2000), pp. 463–518.

[59] Infosec. "What is Enumeration?" In: (2020). Accessed: 10.08.2020, https://resources.infosecinstitute.com/what-is-enumeration.

[60] McAfee. "Introduction to McAfee ePolicy Orchestrator". In: (2020). Accessed: 16.05.2020, https://www.mcafee.com/enterprise/en-us/downloads/trials/epolicy-orchestrator.html.

[61] McAfee. "Managing McAfee ePO users with Active Directory". In: (2020). Accessed: 16.05.2020, https://docs.mcafee.com/bundle/epolicy-orchestrator-5.9.x-product-guide/page/GUID-17CC4F49-DDAA-4282-A778-5B4D71BE236B.html.

[62] Department of US homeland security. "CISA: Industry Control Systems". In: (2016). Accessed: 18.04.2020, https://www.us-cert.gov/ics.

[63] Stefan Axelsson. *Intrusion detection systems: A survey and taxonomy*. Tech. rep. Technical report, 2000.

[64] Eric Geier. "Intro to Next Generation Firewalls". In: (2011). Accessed: 12.05.2020, https://www.esecurityplanet.com/security-buying-guides/intro-to-next-generation-firewalls.html.

[65] John R Vacca. *Network and system security*. Elsevier, 2013.

[66] Bonnie Zhu and Shankar Sastry. "SCADA-specific intrusion detection/prevention systems: a survey and taxonomy". In: *Proceedings of the 1st workshop on secure control systems (SCS)*. Vol. 11. 2010, p. 7.

[67] Trend Micro. "What You Need to Know About the LockerGoga Ransomware". In: (2019). Accessed: 28.05.2020, https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/what-you-need-to-know-about-the-lockergoga-ransomware.

[68] Yi Sun et al. *Distributed computer network zone based security architecture*. US Patent 9,419,941. 2016.

[69] Paul Francis Mevec and Ibrahim A Marhoon. *Systems, methods, and computer medium to securely transfer business transactional data between networks having different levels of network protection using barcode technology with data diode network security appliance*. US Patent 9,210,179. 2015.

[70] Industrial Control Systems Cyber Emergency Response Team. "Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies". In: (2016). Accessed: 18.04.2020, https://www.us-cert.gov/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf.

[71] Boo-Sun Jeon and Jung-Chan Na. "A study of cyber security policy in industrial control system using data diodes". In: *2016 18th International Conference on Advanced Communication Technology (ICACT)*. IEEE. 2016, pp. 314–317.

[72] Peter Maynard, Kieran McLaughlin, and Berthold Haberler. "Towards understanding man-in-the-middle attacks on iec 60870-5-104 scada networks". In: *2nd International Symposium for ICS & SCADA Cyber Security Research 2014 (ICS-CSR 2014) 2*. 2014, pp. 30–42.

[73] SecuriCAD. "Attack Paths". In: (2020). Accessed: 18.07.2020, https://community.securicad.com/attack-paths/.

[74] MITRE. "MITRE ATT&CKő for Industrial Control Systems: Design and Philosophy". In: (2020). Accessed: 30.07.2020, https://collaborate.mitre.org/attackics/img_auth.php/3/37/ATT%26CK_for_ICS_-_Philosophy_Paper.pdf.

[75] Marshall Abrams and Joe Weiss. "Malicious control system cyber security attack case study–Maroochy Water Services, Australia". In: *McLean, VA: The MITRE Corporation* (2008).

A Overall Attack Graph

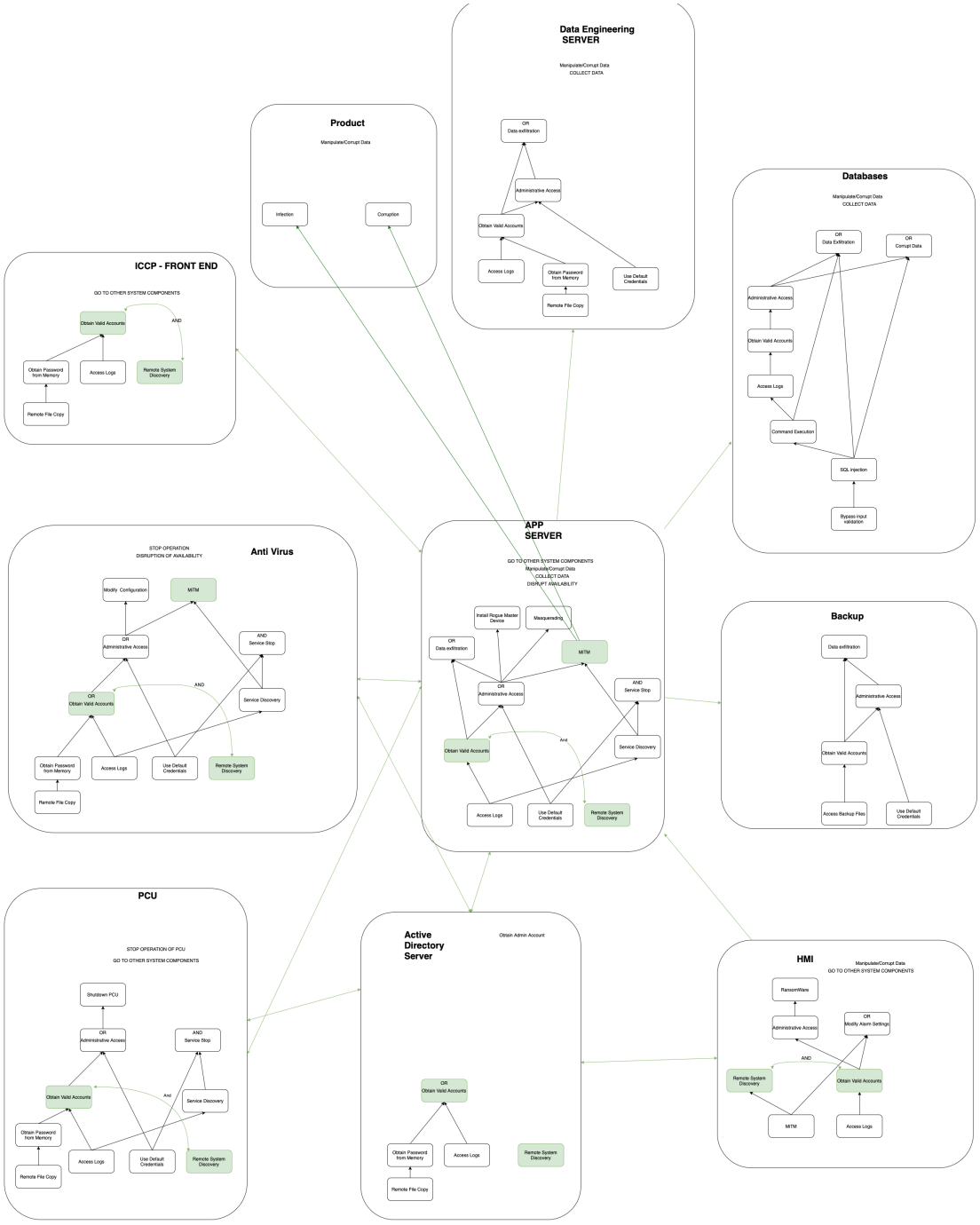

Figure A.1: Overall attack graph

B Example of Unit Testing a DSL Based on MAL

In this example, we create a test model *DatabasesModel* with several test cases. For this work, we use the *Java* language. Let us review how the user can bypass input validation and access data in a database. We assume that the attacker is already in a server and is trying to corrupt data in a database. For this scenario, bypassing put validation is an entry point. As we can see after this attack succeeded, we managed to perform further attack steps as accessing logs to collect information, administrative access to the DB, and data corruption. The situation can be different when the defense mechanisms as sanitising client-side input and validating client-side input are enabled. In this case, the attacker needs some effort to perform further attack steps.

```
public class TestDatabases extends SCADALangTest {
    private static class DatabasesModel {
        public final OracleDatabase oracleDatabase = new
            ↪ OracleDatabase("testOracleDB");
        public final PostgreDatabase postgreDatabase = new
            ↪ PostgreDatabase("testPostgreDB");
        public final RealTimeDatabase realTimeDatabase = new
            ↪ RealTimeDatabase("testRDB");
        public final AppServer appServer = new AppServer("testAppServer");

        public DatabasesModel() {
            oracleDatabase.addAppservers(appServer);
            postgreDatabase.addAppservers(appServer);
            realTimeDatabase.addAppservers(appServer);
            appServer.addOracledatabases(oracleDatabase);
            appServer.addPostgredatabases(postgreDatabase);
            appServer.addRealtimedatabases(realTimeDatabase);
        }
    }

    @Test
    public void testOracleDatabasesBypassInputValidation() {
        var model = new DatabasesModel();

        var attacker = new Attacker();
        attacker.addAttackPoint(model.oracleDatabase.bypassInputValidation);

        attacker.attack();
        model.oracleDatabase.sqlInjection.assertUncompromised();
        model.oracleDatabase
                .commandLineExecutionThroughSQLInjection.assertUncompromised();
        model.oracleDatabase.dataExfiltration.assertUncompromised();
        model.oracleDatabase.accessingLogsToCollectInformation.assertUncompromised();
        model.oracleDatabase.obtainValidAccount.assertUncompromised();
        model.oracleDatabase.administrativeAccessToDatabase.assertUncompromised();
```

```
        model.oracleDatabase.dataCorruption.assertUncompromised();

    }

    @Test
    public void testOracleDatabasesBypassInputValidationWithDefense() {
        var model = new DatabasesModel();

        var attacker = new Attacker();
        attacker.addAttackPoint(model.oracleDatabase.bypassInputValidation);

        attacker.attack();

        var defense1 = new Defense("sanitizeClientSideInput");
        var defense2 = new Defense("validateClientSideInput");
        defense1.isEnabled();
        defense2.isEnabled();
        model.oracleDatabase.bypassInputValidation.assertUncompromised();
        model.oracleDatabase.sqlInjection.assertUncompromised();
        model.oracleDatabase
                .commandLineExecutionThroughSQLInjection.assertUncompromised();
        model.oracleDatabase.dataExfiltration.assertUncompromised();
        model.oracleDatabase.accessingLogsToCollectInformation.assertUncompromised();
        model.oracleDatabase.obtainValidAccount.assertUncompromised();
        model.oracleDatabase.administrativeAccessToDatabase.assertUncompromised();
        model.oracleDatabase.dataCorruption.assertUncompromised();
    }

    @Test
    public void testPostgreDatabasesBypassInputValidation() {
        var model = new DatabasesModel();

        var attacker = new Attacker();
        attacker.addAttackPoint(model.postgreDatabase.bypassInputValidation);

        attacker.attack();
        model.postgreDatabase.sqlInjection.assertUncompromised();
        model.postgreDatabase
                .commandLineExecutionThroughSQLInjection.assertUncompromised();
        model.postgreDatabase.dataExfiltration.assertUncompromised();
        model.postgreDatabase.accessingLogsToCollectInformation.assertUncompromised();
        model.postgreDatabase.obtainValidAccount.assertUncompromised();
        model.postgreDatabase.administrativeAccessToDatabase.assertUncompromised();
        model.postgreDatabase.dataCorruption.assertUncompromised();
    }

    @Test
    public void testPostgreDatabasesBypassInputValidationWithDefense() {
```

```
    var model = new DatabasesModel();

    var attacker = new Attacker();
    attacker.addAttackPoint(model.postgreDatabase.bypassInputValidation);

    attacker.attack();

    var defense1 = new Defense("sanitizeClientSideInput");
    var defense2 = new Defense("validateClientSideInput");
    defense1.isEnabled();
    defense2.isEnabled();

    model.postgreDatabase.sqlInjection.assertUncompromised();
    model.postgreDatabase
            .commandLineExecutionThroughSQLInjection.assertUncompromised();
    model.postgreDatabase.dataExfiltration.assertUncompromised();
    model.postgreDatabase.accessingLogsToCollectInformation.assertUncompromised();
    model.postgreDatabase.obtainValidAccount.assertUncompromised();
    model.postgreDatabase.administrativeAccessToDatabase.assertUncompromised();
    model.postgreDatabase.dataCorruption.assertUncompromised();
}

@Test
public void testRealTimeDatabasesBypassInputValidation() {
    var model = new DatabasesModel();

    var attacker = new Attacker();
    attacker.addAttackPoint(model.realTimeDatabase.bypassInputValidation);

    attacker.attack();
    model.realTimeDatabase.sqlInjection.assertUncompromised();
    model.realTimeDatabase
            .commandLineExecutionThroughSQLInjection.assertUncompromised();
    model.realTimeDatabase.dataExfiltration.assertUncompromised();
    model.realTimeDatabase.accessingLogsToCollectInformation.assertUncompromised();
    model.realTimeDatabase.obtainValidAccount.assertUncompromised();
    model.realTimeDatabase.administrativeAccessToDatabase.assertUncompromised();
    model.realTimeDatabase.dataCorruption.assertUncompromised();
}

@Test
public void testRealTimeDatabasesBypassInputValidationWithDefense() {
    var model = new DatabasesModel();

    var attacker = new Attacker();
    attacker.addAttackPoint(model.realTimeDatabase.bypassInputValidation);

    attacker.attack();
```

```
        var defense1 = new Defense("sanitizeClientSideInput");
        var defense2 = new Defense("validateClientSideInput");
        defense1.isEnabled();
        defense2.isEnabled();

        model.realTimeDatabase.sqlInjection.assertUncompromised();
        model.realTimeDatabase
                .commandLineExecutionThroughSQLInjection.assertUncompromised();
        model.realTimeDatabase.dataExfiltration.assertUncompromised();
        model.realTimeDatabase.accessingLogsToCollectInformation.assertUncompromised();
        model.realTimeDatabase.obtainValidAccount.assertUncompromised();
        model.realTimeDatabase.administrativeAccessToDatabase.assertUncompromised();
        model.realTimeDatabase.dataCorruption.assertUncompromised();
    }
}
```

C All attacks and Mapping MITRE Into Threat Modeling of SCADA

Asset	Attack Code	MITRE code
ICCP Server	ICCPS1	T811
	ICCPS2	T859
	ICCPS3	T846
	ICCPS4	T867
RTU	RTU1	T846
CF	CF1	T859
	CF2	T812
	CF3	T811
	CF4	T846
	CF5	T859
	CF6	T881
	CF7	T816
	CF8	T846
	CF9	T867
HMI	HMI1	T811
	HMI2	T859
	HMI3	T830
	HMI4	T838
	HMI5	T859
	HMI6	T846
	HMI7	S0008
App server	APS1	T859

Asset	**Attack Code**	**MITRE code**
	APS2	T812
	APS3	T811
	APS4	T846
	APS5	T859
	APS6	T881
	APS7	T846
	APS8	T848
	APS9	T867
	APS10	T830
	APS11	T849
Database	DB4	T867
	DB5	T811
	DB6	T859
	DB7	T859
	DB8	T867
Anti-virus server	AV1	T859
	AV2	T812
	AV3	T811
	AV4	T846
	AV5	T881
	AV6	T830
	AV8	T867
Backup Server	BS1	T859
	BS2	T812
	BS3	T811
	BS4	T859
	BS5	T867
Active Directory	AD1	T859
	AD2	T867
	AD4	T846
Product	PR1	T873
	PR2	T873
	DNS2	T816
NTP server	NTP1	T816
DE Server	DE1	T867
	DE3	T859
	DE4	T812
	DE5	T811

Asset	Attack Code	MITRE code

Table C.1: All Attacks

D Code of *scadaLang*

```
#id: "org.mal-lang.scadalang"
#version: "1.0.0"

category Networking{
abstract asset Zone{
      | gainInitialAccess [Exponential(1)]
        -> bypassFirewall
    | bypassFirewall
   }

asset ProductionZone extends Zone
{   | gainInitialAccess [Exponential(1)]
        -> ntpserver.serverCompromise,
           physicalShutdownDiod,
           directoryservice.remoteSystemDiscovery,
           dnsserver.remoteSystemDiscovery,
           ntpserver.remoteSystemDiscovery,
           directoryservice.serverCompromise,
           dnsserver.serverCompromise,
           nisserver.remoteSystemDiscovery,
           nisserver.serverCompromise
    | bypassFirewall [Bernoulli(0.5)]
          -> firewall.bypassFirewall
     | physicalShutdownDiod [Exponential(0.1)]
          -> diod.physicalShutdownDiod
}
asset DemilitarizedZone extends Zone
{   | gainInitialAccess [Exponential(1)]
          -> accessToDMZ
      | accessToDMZ
      | bypassFirewall [Bernoulli(0.5)]
          -> firewall.bypassFirewall
```

```
      | physicalShutdownDiod [Exponential(0.1)]
          -> diod.physicalShutdownDiod
}

asset SupportZone extends Zone {
      | bypassFirewall [Bernoulli(0.5)]
          -> firewall.bypassFirewall
      | physicalShutdownDiod [Exponential(0.1)]
          -> diod.physicalShutdownDiod
}

asset Firewall extends Host
{     | bypassFirewall [Bernoulli(0.5)]
        ->
          accessRouter,
          productionzone.bypassFirewall,
          supportzone.bypassFirewall,
          demilitarizedzone.bypassFirewall
      | accessRouter
        -> router.reconfigureRouter
}
asset Diod
{     | physicalShutdownDiod [Exponential(0.1)]
        -> productionzone.physicalShutdownDiod,
          demilitarizedzone.physicalShutdownDiod,
          supportzone.physicalShutdownDiod
}
asset Router
{     | reconfigureRouter
        -> firewall.accessRouter
}
asset NextGenerationFirewall
{     | exploitVulnerableFunction
}
asset DNSServer extends Host
{     | serverCompromise [Bernoulli(0.5)]
        -> shutdownDNSService,
          poisonDNS
      | shutdownDNSService
      | poisonDNS
      | remoteSystemDiscovery [Bernoulli(0.5)]
        -> serverCompromise
}
asset NISServer extends Host
{     | serverCompromise [Bernoulli(0.5)]
```

```
    |  remoteSystemDiscovery [Bernoulli(0.5)]
       -> serverCompromise
}
asset NTPServer extends Host
{   |  serverCompromise [Bernoulli(0.5)]
       -> alterClockConfigurations,
          stopService
    |  alterClockConfigurations [Bernoulli(0.5)]
    |  stopService
    |  remoteSystemDiscovery [Bernoulli(0.5)]
       -> serverCompromise
}
}

category Security{

abstract asset Account{
     |  bruteForce [Bernoulli(0.5)]
    |  bribeSCADAEngineer [Exponential(0.1)]
   }

asset AdminAccount extends Account{
    |  phishSCADAEngineer [Bernoulli(0.5)]
   }

asset ServiceAccount extends Account{ }

asset UserAccount extends Account{
    |  phishSCADAEngineer [Bernoulli(0.5)]
   }
}

category System{
abstract asset Host {
    |  serverCompromise [Bernoulli(0.5)]
}
asset DirectoryService extends Host
{   |  serverCompromise [Bernoulli(0.5)]
       -> obtainPasswordFromMemory,
          remoteSystemDiscovery,
          accessLogs
    |  obtainPasswordFromMemory [Bernoulli(0.5)]
       -> obtainValidAccount
    &  remoteSystemDiscovery [Bernoulli(0.5)]
       -> cfs.serverCompromise,
```

```
            hmis.serverCompromise,
            obtainValidAccount
      |   accessLogs
            -> obtainValidAccount
      & obtainValidAccount [Bernoulli(0.5)]
            -> obtainAdminAccount
      | obtainAdminAccount
}
asset  AppServer
{     |   serverCompromise [Bernoulli(0.5)]
            -> remoteSystemDiscovery,
              accessingLogsToCollectInformation,
              useDefaultCredentials
      &  remoteSystemDiscovery [Bernoulli(0.5)]
      |   useDefaultCredentials  [Bernoulli (0.5) ]
            -> administrativeAccessToAppServer,
            serviceStop
      &  obtainValidAccount [Bernoulli(0.5) ]
            -> administrativeAccessToAppServer,
              dataExfiltration ,
              goToOtherSystemComponents
      |   dataExfiltration  [Exponential(0.1)]
            -> dataCorruption
      &  serviceStop  [Bernoulli (0.5) ]
      |   accessingLogsToCollectInformation [Exponential(1)]
            -> serviceDiscovery,
                obtainValidAccount
      &  serviceDiscovery  [Exponential(1)]
            -> serviceStop,
              manInTheMiddle
      &  manInTheMiddle [Exponential(1)]
            -> goToOtherSystemComponents
      |   administrativeAccessToAppServer [Exponential(0.1)]
            -> dataExfiltration,
                installRogueMasterDevice,
                masquerade
      |   installRogueMasterDevice [Bernoulli(0.5)]
            -> collectData
      |   collectData
      |   dataCorruption [Exponential(0.1)]
      |   goToOtherSystemComponents
            -> audiblealarm.startAlarm,
                audiblealarm.unbootAlarm,
                                    products.productInfection,
                                    iccpserver .serverCompromise,
```

```
                                    postgredatabases.bypassInputValidation,
                                    oracledatabases.bypassInputValidation,
                                    backupserver.serverCompromise,
                                    realtimedatabases.bypassInputValidation,
                                    products.productCorruption,
                                    products.productInfection,
                                    cfs.serverCompromise,
                                    antivirus.serverCompromise,
                                    hmis.serverCompromise
    |  masquerade [Bernoulli(0.5)]
    # onDemandAccount
      -> administrativeAccessToAppServer
    # networkSegregation
      -> manInTheMiddle
    # logsDataEncryption
      -> accessingLogsToCollectInformation
}

asset OracleDatabase
{   |  bypassInputValidation [Bernoulli(0.5)]
      -> sqlInjection
    |  sqlInjection [Exponential(0.1)]
      -> commandLineExecutionThroughSQLInjection
    |  commandLineExecutionThroughSQLInjection [Bernoulli(0.5)]
      -> accessingLogsToCollectInformation
    |  accessingLogsToCollectInformation [Exponential(1)]
      -> obtainValidAccount
    &  obtainValidAccount [Bernoulli(0.5)]
      -> administrativeAccessToDatabase
    |  administrativeAccessToDatabase [Exponential(0.1)]
      -> dataCorruption,
         dataExfiltration
    &  dataCorruption [Exponential(0.1)]
    &  dataExfiltration [Exponential(0.1)]
    # sanitizeClientSideInput
      -> sqlInjection,
        commandLineExecutionThroughSQLInjection,
        bypassInputValidation
    # validateClientSideInput
      -> bypassInputValidation
    # usePreparedStatement
      -> sqlInjection,
        commandLineExecutionThroughSQLInjection
    # onDemandAccount
      -> administrativeAccessToDatabase
```

```
}
asset PostgreDatabase
{       |  bypassInputValidation [Bernoulli(0.5)]
           -> sqlInjection
        |  sqlInjection  [Exponential(0.1)]
           -> commandLineExecutionThroughSQLInjection
        |  commandLineExecutionThroughSQLInjection [Bernoulli(0.5)]
           -> accessingLogsToCollectInformation
        |  accessingLogsToCollectInformation [Exponential(1)]
           -> obtainValidAccount
        &  obtainValidAccount [Bernoulli(0.5)]
           -> administrativeAccessToDatabase
        |  administrativeAccessToDatabase [Exponential(0.1)]
            -> dataCorruption,
               dataExfiltration
        &  dataCorruption [Exponential(0.1)]
          &  dataExfiltration  [Exponential(0.1)]
        # sanitizeClientSideInput
           -> sqlInjection ,
              commandLineExecutionThroughSQLInjection,
              bypassInputValidation
        # validateClientSideInput
           -> bypassInputValidation
        # usePreparedStatement
           -> sqlInjection ,
              commandLineExecutionThroughSQLInjection
        # onDemandAccount
           -> administrativeAccessToDatabase
}
asset RealTimeDatabase
{    |  bypassInputValidation [Bernoulli(0.5)]
                -> sqlInjection
             |  sqlInjection  [Exponential(0.1)]
                -> commandLineExecutionThroughSQLInjection
             |  commandLineExecutionThroughSQLInjection [Bernoulli(0.5)]
                -> accessingLogsToCollectInformation
             |  accessingLogsToCollectInformation [Exponential(1)]
                -> obtainValidAccount
             &  obtainValidAccount [Bernoulli(0.5)]
                -> administrativeAccessToDatabase
             |  administrativeAccessToDatabase [Exponential(0.1)]
                 -> dataCorruption,
                    dataExfiltration
             &  dataCorruption [Exponential(0.1)]
             &  dataExfiltration  [Exponential(0.1)]
```

```
                # sanitizeClientSideInput
                    -> sqlInjection ,
                       commandLineExecutionThroughSQLInjection,
                       bypassInputValidation
                # validateClientSideInput
                    -> bypassInputValidation
                # usePreparedStatement
                    -> sqlInjection ,
                       commandLineExecutionThroughSQLInjection
                # onDemandAccount
                    -> administrativeAccessToDatabase
}
asset BackupServer
{   | serverCompromise [Bernoulli(0.5)]
        -> accessBackupFiles,
           useDefaultCredentials
    | accessBackupFiles [Bernoulli(0.5)]
        -> obtainValidAccount
    | useDefaultCredentials [Bernoulli(0.5)]
        -> administrativeAccess
    & obtainValidAccount [Bernoulli(0.5)]
        -> administrativeAccess,
           dataExfiltration
    | administrativeAccess [Exponential(0.1)]
        -> dataExfiltration
    | dataExfiltration [Exponential(0.1)]
    # encryptStorageData
        -> administrativeAccess
}
asset IccpServer extends Host
{   | serverCompromise [Bernoulli(0.5)]
        -> remoteFileCopy,
           remoteSystemDiscovery,
           accessingLogsToCollectInformation
    | remoteFileCopy [Bernoulli(0.5)]
     -> obtainPasswordFromMemory
    & remoteSystemDiscovery [Bernoulli(0.5)]
    | accessingLogsToCollectInformation [Exponential(1)]
        -> obtainValidAccount
    & obtainValidAccount [Bernoulli(0.5)]
        -> goToOtherSystemComponents
    | goToOtherSystemComponents
        -> appservers.serverCompromise
    | obtainPasswordFromMemory [Bernoulli(0.5)]
        -> obtainValidAccount
```

```
    # hardenProfile
        -> appservers.serverCompromise
    # intrusionDetectionSystem
        -> remoteSystemDiscovery
}
asset DataEngineeringServer extends Host
{   |  serverCompromise [Bernoulli(0.5)]
        -> remoteFileCopy,
           accessLogs,
           useDefaultCredentials
    |  remoteFileCopy [Bernoulli(0.5)]
    |  accessLogs
    |  useDefaultCredentials [Bernoulli(0.5)]
    |  obtainPasswordFromMemory [Bernoulli(0.5)]
    |  stealingPasswordsFromMemory [Bernoulli(0.5)]
    |  obtainValidAccount [Bernoulli(0.5)]
        -> administrativeAccess
    |  administrativeAccess [Exponential(0.1)]
    |  accessingLogsToCollectInformation [Exponential(1)]
    |  serviceDiscovery [Exponential(1)]
    |  administrativeAccessToDEServer [Exponential(0.1)]
    |  dataExfiltration [Exponential(0.1)]
    # whitelistHosts
        -> serviceDiscovery
    # onDemandAccount
        -> administrativeAccessToDEServer
}
asset Product
{   |  productInfection [Bernoulli(0.5)]
    |  productCorruption [Exponential(0.1)]
    # applyApplicationSecurityGuidelines
        -> productInfection,
           productCorruption
}
asset Alarm
{   |  unbootAlarm [Bernoulli(0.5)]
    |  startAlarm [Bernoulli(0.5)]
}
asset HumanMachineInterface
{   |  serverCompromise [Bernoulli(0.5)]
         -> accessingLogsToCollectInformation,
            manInTheMiddle
    |  accessingLogsToCollectInformation [Exponential(1)]
        -> obtainValidAccount
    &  obtainValidAccount [Bernoulli(0.5)]
```

```
        -> modifyAlarmSettings,
           administrativeAccessToHMI
    |   modifyAlarmSettings [Exponential(0.1)]
        -> audiblealarm.unbootAlarm,
           audiblealarm.startAlarm
    |   manInTheMiddle [Exponential(1)]
        -> modifyAlarmSettings,
           remoteSystemDiscovery
    &   remoteSystemDiscovery [Bernoulli(0.5)]
        -> appservers.serverCompromise,
           directoryservice .serverCompromise
    |   administrativeAccessToHMI [Exponential(0.1)]
        -> placeRansomware
    |   placeRansomware [Bernoulli(0.5)]
    #   hardenProfile
        -> placeRansomware
    #   intrusionDetectionSystem
        -> remoteSystemDiscovery,
           manInTheMiddle
    #   scheduleBackup
        -> placeRansomware
    #   malwareDetectionAndAntivirus
        -> placeRansomware
    #   onDemandAccount
        -> administrativeAccessToHMI
}
asset  AntivirusServer extends Host
{   |   serverCompromise [Bernoulli(0.5)]
        -> remoteFileCopy,
           remoteSystemDiscovery,
           accessingLogsToCollectInformation,
           useDefaultCredentials
    |   remoteFileCopy [Bernoulli(0.5)]
        -> obtainPasswordFromMemory
    &   remoteSystemDiscovery [Bernoulli(0.5)]
    |   useDefaultCredentials [Bernoulli(0.5)]
        -> serviceStop,
           administrativeAccess
    &   obtainValidAccount [Bernoulli(0.5)]
        -> appservers.serverCompromise
    &   serviceStop [Bernoulli(0.5)]
    |   administrativeAccess [Exponential(0.1)]
        -> manInTheMiddle,
           modifyConfigurationOfAntivirusAgentsAndRepositories
    |   accessingLogsToCollectInformation [Exponential(1)]
```

```
        -> serviceDiscovery,
             obtainValidAccount
    |   serviceDiscovery  [Exponential(1)]
        -> manInTheMiddle,
           serviceStop
    &   manInTheMiddle [Exponential(1)]
        -> appservers.serverCompromise
    |   modifyConfigurationOfAntivirusAgentsAndRepositories [Exponential(0.1)]
    |   obtainPasswordFromMemory [Bernoulli(0.5)]
        -> obtainValidAccount
    |   stealingPasswordsFromMemory [Bernoulli(0.5)]
}
asset  CommunicationFrontend extends Host
{   |   serverCompromise [Bernoulli(0.5)]
        -> remoteFileCopy,
           remoteSystemDiscovery,
           accessingLogsToCollectInformation,
           useDefaultCredentials,
           rtus.remoteSystemDiscovery
    |   remoteFileCopy [Bernoulli(0.5)]
    -> obtainPasswordFromMemory
    &   remoteSystemDiscovery [Bernoulli(0.5)]
    |   useDefaultCredentials [Bernoulli(0.5)]
      -> serviceStop,
           administrativeAccessToCommFrontend
    &   obtainValidAccount [Bernoulli(0.5)]
        -> appservers.serverCompromise,
           directoryservice.serverCompromise
    &   serviceStop  [Bernoulli(0.5)]
        -> shutDownCommFrontend
    |   accessingLogsToCollectInformation [Exponential(1)]
        -> serviceDiscovery,
           obtainValidAccount
    |   serviceDiscovery  [Exponential(1)]
        -> serviceStop,
           obtainValidAccount
    |   administrativeAccessToCommFrontend [Exponential(0.1)]
        -> shutDownCommFrontend
    |   shutDownCommFrontend [Exponential(0.1)]
    |   obtainPasswordFromMemory [Bernoulli(0.5)]
     -> obtainValidAccount
    # onDemandAccount
        -> administrativeAccessToCommFrontend
}
asset  RemoteTerminalUnit
```

```
{   |  remoteSystemDiscovery [Bernoulli(0.5)]
        -> cfs.serverCompromise
}
}

associations{
DirectoryService [directoryservice] 1
    <--LinuxDirectoryServiceAccess--> 1 [useraccount] UserAccount
DataEngineeringServer [dataeng] 1 <--DataEngineeringAccess--> 1
    [serviceaccount] ServiceAccount
DirectoryService [directoryservice] 1
    <--WindowsDirectoryServiceAccess--> 1 [administrator] AdminAccount
HumanMachineInterface [humanmachineinterfaces] * <--MMIAccess--> 1
    [appserveraccounts] AdminAccount
AppServer [appservers] 1 <--AntivirusAccess--> * [antivirus]
    AntivirusServer
AppServer [appservers] * <--AlarmAccessFromAppServer--> 1
    [audiblealarm] Alarm
CommunicationFrontend [cfs] 1 <--DirectoryServiceAccess--> 1
    [directoryservice] DirectoryService
BackupServer [backupserver] 1 <--BackupServerAccess--> 1 [appservers]
    AppServer
HumanMachineInterface [hmis] 1 <--DirectoryServiceAccess--> 1
    [directoryservice] DirectoryService
AppServer [appservers] * <--AppServerAccess--> * [hmis]
    HumanMachineInterface
HumanMachineInterface [hmis] * <--AlarmAccessFromMMI--> 1
    [audiblealarm] Alarm
Product [products] 1 <--ProductAccess--> 1 [appservers] AppServer
CommunicationFrontend [cfs] * <--RtuAccess--> * [rtus]
    RemoteTerminalUnit
AppServer [appservers] * <--DatabaseAccess--> * [realtimedatabases]
    RealTimeDatabase
AppServer [appservers] * <--DatabaseAccess--> * [postgredatabases]
    PostgreDatabase
IccpServer [iccpserver] * <--AppServerAccess--> * [appservers]
    AppServer
AppServer [appservers] * <--DatabaseAccess--> * [oracledatabases]
    OracleDatabase
CommunicationFrontend [cfs] * <--RtuAccess--> 1 [appservers] AppServer
Firewall [firewall] * <--SupportZoneAccess--> * [supportzone]
    SupportZone
AppServer [appservers] 1 <--AppServerAccess--> 1 [supportzone]
    SupportZone
Diod [diod] 1 <--DiodAccess--> 1 [demilitarizedzone] DemilitarizedZone
```

ProductionZone [lans] * <−−NISServerAccess−−> * [nisserver] NISServer
ProductionZone [lans] * <−−NTPServerAccess−−> * [ntpserver] NTPServer
Firewall [firewall] 1 <−−RouterAccess−−> 1 [router] Router
ProductionZone [lans] * <−−DNSServerAccess−−> * [dnsserver] DNSServer
Diod [diod] 1 <−−DiodAccess−−> 1 [productionzone] ProductionZone
Firewall [firewall] * <−−ProductionZoneAccess−−> * [productionzone]
ProductionZone
Diod [diod] 1 <−−DiodAccess−−> 1 [supportzone] SupportZone
ProductionZone [lans] * <−−DirectoryServiceAccess−−> *
[directoryservice] DirectoryService
AppServer [appservers] 1 <−−AppServerAccess−−> 1 [productionzone]
ProductionZone
Firewall [firewall] * <−−DemilitarizedZoneAccess−−> * [demilitarizedzone]
DemilitarizedZone
AppServer [appservers] 1 <−−AppServerAccess−−> 1 [demilitarizedzone]
DemilitarizedZone
}