Analysis of Software Defined Networks as a Mechanism for Enforcing Corporate Security Policies in OT Networks

A Dissertation
Presented in Partial Fulfillment of the Requirements for the
Degree of Doctor in Philosophy
with a
Major in Computer Science
in the
College of Graduate Studies
University of Idaho
by
Sandeep Gogineni Ravindrababu

Approved by

Major Professor: Jim Alves-Foss, Ph.D.

Committee Members: Jia Song, Ph.D.; Daniel Conte de Leon, Ph.D.;
Yacine Chakhchoukh, Ph.D.

Department Administrator: Terry Soule, Ph.D.

May 2022

Abstract

Cyber Security has been given a high priority for operational technology systems in recent years after specific cyber-incidents targeting them. Previously, these systems were primarily concerned with reliability; however, cyber security is now viewed as a critical aspect in avoiding production damage and financial losses. According to certain studies, replacing traditional networks in OT systems with software-defined networks (SDN) minimizes cyber-attacks due to the features provided by these networks. SDN networks have various advantages over traditional networks, due to the separation of the data plane and control plane. The concern is whether SDN networks are more dependable than existing traditional networks, and whether we can take advantage of all of SDN's characteristics when connecting with OT systems. Furthermore, deploying cyber security on a network infrastructure necessitates the creation and implementation of security policies that define the authorized communication between network devices. There is, however, a distinction to be made between security policies and the technologies that implement them. There is also often a distinction between intended policy and deployed or configured policy. Therefore there is a need to confirm compliance between policy and reality in a network. This is especially true in operational technology systems where there is a lot of network infrastructure and special purpose devices which can not be scanned or analyzed using traditional cyber security tools.

To address the cyber security issues in operational technology systems, this dissertation reviews cyber-incidents reported on them and summarizes possible attacks on each of their sub-systems to gain broader insight into vulnerabilities present in them and uses the common vulnerability exposure database to enumerate trends. Then, a process is formally developed and evaluated through a proof of concept tool to detect the security policy implemented in the control rules of an SDN switch deployed in an industrial control system network. These rules were analyzed to determine if this security policy is compliant with the organization's high-level policies.

Acknowledgments

I would first like to thank wholeheartedly to my advisor, Dr. Jim Alves-Foss, for his support, encouragement, and guidance throughout my graduate studies. I wouldn't have traveled this far if he hadn't put his faith in me. He continues to inspire me in numerous ways and has helped me become the person I am today. I would also like to thank my other committee members, Dr. Jia Song, Dr. Daniel Conte de Leon, and Dr. Yacine Chakhchoukh for their valuable insights on my dissertation work.

I would like to thank all my instructors for their hard work and dedication in providing me with a comprehensive and valuable education. I would like to thank the department chair Dr. Terry Soule, Ms. Arvilla Daffin and other staff in department of Computer Science for their help during my study in the department. I also thank all the staff of the College of Graduate Studies for their help and support throughout my study at UI.

I would like to thank my dear friends Shashi A, Sharath C, Preethi T, Ranjitha R, and Harshitha K who supported me at difficult times and instilled confidence in me to achieve my dream. I will always be indebted to Shashi A, it was her advice for me to apply for PhD and without her support this would'nt have come true. She has always been a support system and backbone for me and i wish she continues to be that in future too.

I would also like to thank my UofI friends Aman G, Chaithanya B, Charmi G, Tushar A, Varsha V and others for giving me wonderful memories during my studies. I wish to acknowledge the Funding Agency Schweitzer Engineering Laboratories (SEL), for supporting me during the course of my graduate studies.

Last, but certainly not least, I would like to thank Shashikala V (Mom), Ravindrababu G (Dad), Kavya O.R (Wife), Santosh K (Uncle), Varalakshmi M (Aunt), Indrani (Aunt) and other family members for their understanding, support, encouragement, patience and unconditional love which have helped me make this dissertation a reality.

I firmly believe in GOD and thank GOD for giving me an opportunity to get educated, and also for his blessings on everyone.

Dedication

To my best friend Shashi AswathnarayanaReddy.....

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advancement of technology in operational technology systems (OT), which includes industrial control systems (ICS) and their sub-systems, has overcome the nature of standalone and isolated networks by integrating the systems with Information Technology (IT) systems, resulting in increased accuracy and performance in terms of results and reliability. This progress has ramifications in terms of cyber-attacks, which have been on the rise in recent years. Because of this, ICS systems are recommended to follow standard procedures, protocols, or guidelines developed by the government or organizations to protect against cyber-attacks. The National Institute of Standards and Technology (NIST) has proposed guidance for securing ICS networks consisting of supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and programmable logic controllers (PLC) [SFS11]. This guide clearly states that system vulnerabilities in ICS can occur in hardware or software due to misconfigurations and poor maintenance of devices deployed across complex ICS networks.

In ICS networks, security misconfigurations are security policies that are incorrectly configured across networking devices. A security policy is a collection of rules that specify what actions are permitted or prohibited in the system. In general, organizations develop policies to achieve their objectives and defend themselves from cyber threats. The network administrator enforces these policies through the organization's networking equipment, such as switches and routers. As a result, how well these policies are enforced within the networking devices determines the level of security of an ICS network.

Misconfigurations often contribute to the disclosure of confidential data or enable unauthorized naccess, which aids adversaries in compromising the network. Misconfiguration problems can be mitigated in several ways, one of which is to adopt SDN technology. Due to the versatility of SDN networking, which includes dynamic reconfiguration, unified policy management, flow control, fault tolerance, programmability, and more, it is easy to configure SDN devices in complex environments [NS17]. However, Venugopal et al., [VAFR19a] specify that poor configuration using SDN technology will still result in insecure networks.

In order to address security misconfiguration issues in ICS networks using SDN technology, there is a need to develop a formal process to check whether the implemented policy is in line

with the organization's policy, and this is our primary research objective. This dissertation examines the concept of a policy-driven method, which retrieves the implemented high-level policies in a SDN switch, analyzes them, and compares them to corporate security policies defined by the organization. This will aid in the development of a general solution for handling critical features in operational technology systems without requiring a complete revamp of current systems.

Section 1.1 of this chapter introduces core concepts related to this work. The problems that this study aims to solve are presented in Section 1.2. The motivation for this project is discussed in Section 1.3. The remainder of the dissertation is summarized in Section 1.4.

1.1 Background

This section briefly introduces the basic concepts of OT systems, SDNs, and High-level Policies.

1.1.1 Operational Technology Systems

OT systems are computing systems that manage industrial operations and monitor or control physical devices. This category includes systems such as oil and gas monitoring, power usage on energy networks, and hydropower plant management, among others. OT systems process operational data and employ a variety of hardware and communication protocol technologies. DNP3, MODBUS, and Profibus are standard protocols used by OT systems. Due to advancements in technology, the standalone nature of OT systems has dwindled over time, resulting in their integration with corporate networks, which provides dynamic feedback to respond in realtime, raising cyber threats.

ICS components, such as SCADA, DCS, and PLCs, are intended to be used for monitoring and controlling critical infrastructure systems . By combining commercial-off-the-shelf (COTS) components for better business operations, ICS systems have evolved from isolated to highly interconnected networks. As a result of this evolution, the number of computer-network-based attacks has increased.

Several researchers have demonstrated the vulnerabilities, attacks, and countermeasures for the sub-systems in ICS systems (e.g., [DSMF08], [NFCMT09], [HCPS08]) have analyzed that SCADA protocols and architectures used in the various critical infrastructure sectors have security vulnerabilities. Yan et al. identified vulnerabilities and develop cyberattack scenarios in the wind farm SCADA system. Mallouhi [MANC$^+$11a] describes a set of SCADA cyber attacks against a MODBUS TCP testbed. Yang et al. [YLS$^+$11] have described the impact of cyberattacks on the smart grid. Liu et al. [LNR11] described false data injection attacks against state estimation algorithms in electric power systems. These findings support the need to take action to identify classes of vulnerabilities and attacks affecting operational technology systems, as well as the necessity to provide a general solution for preventing cyberattacks on OT systems.

1.1.2 Software Defined Networks

Traditional networks are being phased out in favor of SDN, which offer significant capabilities such as policy administration, flow control, fault tolerance, programmability, and more. The benefits of integrating SDN with OT systems are numerous, and the handling of policies is regarded as one of the most significant aspects of sophisticated OT systems. Policies are kept in SDN in terms of logical connections or flows, rather than being stored directly.

SDNs have a number of advantages over traditional networks, and because the data plane and control plane are separated, there are reduced vulnerabilities and threats. ICS systems, SCADA systems, embedded devices, and other OT components are increasingly being connected with IT networks, which has both advantages and downsides in terms of production and cyber security.To date there hasn't been much written about the vulnerabilities of OT systems adopting SDN. An initial focus of this study was on reviewing potential or existing vulnerabilities in SCADA systems and SDN networks. The detailed review of SDN and its architecture is presented in Chapter 3, with a comparison with traditional SCADA systems in Chapter 4.

1.1.3 High-Level Policies

One strategy to mitigate cyber-attacks is to use a policy-driven approach, in which high-level policies are specified and implemented to provide a generic solution without changing the complete implementation of existing systems. Some works conducted by researchers with respect to policies are as follows: Guel [Gue10] provided the overview for developing security policies, Machalo et al. [MGSFW14a] worked on QoS and policy management in SDN, Lara and Ramamurthy [LR14] implemented the security policies using OpenFlow and Ben-Itzhak et al. [BIBC+15a] have created a framework called EnforSDN for specifying the network policies with SDN. Chapter 5 outlines the details about Policies, different policy languages, and comparisons between them.

1.2 Problems

Any complex OT system requires a thorough understanding of its design and implementation. A complex system is made up of multiple sub-systems, and how the sub-systems are configured is crucial to the system's overall security. The policy-driven approach is used to configure this. Traditional networks, SDNs, or a combination of both can make up a sub-system. These sub-systems components can be configured manually or automatically. One issue with the automatic approach is that there is no formal method for verifying network configurations and whether or not modifications to those configurations are in accordance with corporate security policies specified by the organizations.

The research presented in this dissertation is concerned with two issues: Is it possible to create a technique for mapping Implemented Security Policies (ISP) to Corporate Security

Policies (CSP) and determining whether ISP satisfies CSP? Is it possible to examine the effort of formal verification of policies in an ICS using this technique?

To summarize, a formal study of SDN as a technique for enforcing CSPs in OT networks is a difficult procedure that has not been explored previously, and this research offers a practical solution to this problem.

1.3 Motivation

The purpose of this dissertation is to develop a methodology for analyzing and validating security policies deployed throughout ICS networks via SDN technology, as well as comparing them to high-level corporate security policies.

In achieving this goal, this dissertation has achieved the specific objectives:

- Analysis of vulnerability trends and attacks in OT systems.
- Analysis of SCADA system security properties using traditional and SDN networks.
- Development of a proof-of concept tool to extract implemented policies from an SDN switch. SDN Controller generates a backup configuration file which consists information about system. The details of the systems are stored as logical flows. Retrieving the logical flows and mapping them to policies is the primary objective.
- Development of a theory to map the ISP and CSP into the formal model.
- Extension of the proof-of concept tool's capabilities to convert the high-level policies defined at the organization level to the low-level flow rules using a scenario-based approach.
- Development of a formal model to map the ISP with the policies specified by the organization using mathematical and relational operations.
- Demonstration of how to analyze if ISP in an SDN switch is consistent with respect to CSP.

In conclusion, the work provided in this dissertation leads to a practical technique that will assist product owners in determining whether policy requirements or objectives are correctly implemented in sub-systems. Using SDN technology, network administrators will be able to analyze and validate the policies applied in sub-systems without the need for manual intervention. Finally, network configurations and modifications to those configurations can be checked to see if they comply with corporate security policies.

1.4 Dissertation Overview

Chapter 2 presents a brief review of the relevant research on OT Systems, including ICS systems, SCADA systems, power and smart grid systems, embedded devices, PLC's and outlines the vulnerabilities and attacks reported in CVE database.

SDN networks, their workflow, and the vulnerabilities and attacks based on SDN architecture and control flow operations are all covered in Chapter 3

The investigation of SCADA system security properties utilizing traditional and SDN networks is presented in Chapter 4

Policy-Based Network Management (PBNM) is discussed in Chapter 5, along with its benefits, several types of policy languages, and a comparison of three different policy languages: Ponder, XACML, and HERMES.

Chapter 6 illustrates how to retrieve data from the backup configuration file generated by SDN controller automatically and store it as flow rules in the database. The final section of this chapter explains how the retrieved flow rules are transformed into ponder authorization policies for further analysis.

Chapter 7 validates the security policies implemented in SDN switch deployed across ICS network.

Finally, in Chapter 8, the work is summarized and a number of objectives for further research in the fields of ICS system security and SDN technology are proposed.

Chapter 2

Background

The material in this chapter was originally published as "Analysis of Vulnerability Trends and Attacks in OT Systems." [GRAF22], and has been slightly edited for content and presentation.

Cyber Security has been given an high priority for OT systems in recent years after specific cyber-incidents targeting them. Earlier, these systems were focused mainly on reliability and at present security is also considered as an important factor to avoid production damage and financial losses. To improve the cyber security in industrial systems, it is necessary to understand the flaws and provide countermeasures to mitigate the cyber-attacks.

2.1 Operational Technology Systems

OT systems are computing systems that can manage industrial operations which monitors or controls physical devices. Systems like oil and gas monitoring, power consumption on electricity grids, hydropower plant management, etc. falls in this category. OT systems process operational data and use various technologies for hardware design. Standard network protocols used by OT systems are DNP3, MODBUS, and Profibus. The standalone nature of OT systems has diminished over the years due to the integration with corporate networks. This integration provides dynamic feedback for network operators to respond in realtime, consequently reducing the security. ICS, SCADA, Control Systems, Embedded Systems, Smart Grid Sytems, and Power systems are all part of OT systems.

OT systems have transformed from isolated to highly interconnected networks and this development has led to increase in computer network-based attacks [Cze16]. Several significant cyber-incidents on OT systems were listed by Hemsley, et al. [HF+18]. Figure 2.1 depicts the list of cyber-attacks on OT systems from the year 2000 to present. These incidents on critical and safety infrastructure have caused huge financial losses to industries and government. Vulnerabilities that were reported in Common Vulnerabilities and Exposures (CVE) database on OT systems or shown in Figure 2.2. To secure OT systems it is very important to develop an understanding the causes of these vulnerabilities and ways to mitigate them. The main focus of

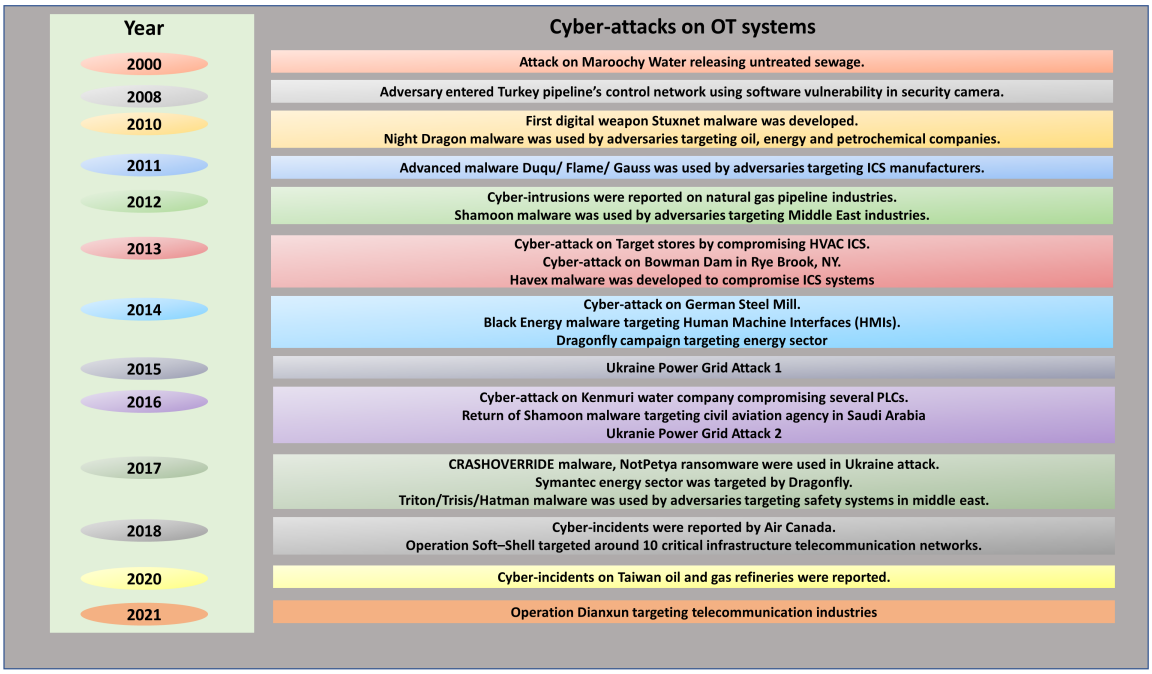

Figure 2.1: Cyber-Attacks On Operational Technology Systems

this chapter is to present the evolution of cyber-incidents reported on OT systems in the CVE database. Primarily we focus on Smart Grid and Power systems, SCADA systems, Embedded systems, PLCs, and Remote Terminal Units (RTUs). In subsequent sections we present the trends and attacks on these systems individually, which will help to give a broader insight on the causes of cyber-attacks in these systems.

2.1.1 Industrial Control Systems (ICS)

ICS components are designed for monitoring, controlling the industrial control systems or critical infrastructure systems, including SCADA, DCS, and PLC. ICS systems have transformed from isolated to highly interconnected networks by incorporating COTS components for improving the business processes. The consequence of this development has increased computer-network based attacks.

Several researchers demonstrated the vulnerabilities, attacks, and countermeasures for the subsystems in ICS systems (e.g., [DSMF08], [NFGMS11], [HCPS08]) have that SCADA protocols and architectures used in the various critical infrastructure sectors have security vulnerabilities. Yan et al. identify vulnerabilities and develop cyberattack scenarios in the wind farm SCADA system. Mallouhi [MANC$^+$11a] describes a set of SCADA cyber attacks against a MODBUS TCP testbed. Yang et al. [YLS$^+$11] have described the impact of cyberattacks on the smart grid. Liu et al. [LNR11] described false data injection attacks against state estimation algorithms in electric power systems. This research justifies the need for action to identify

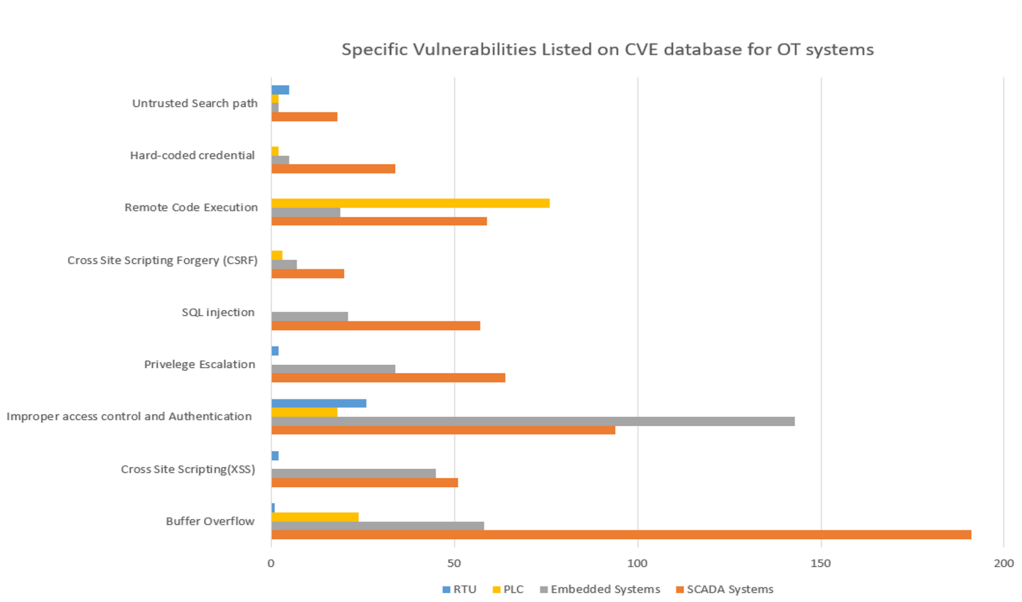

Figure 2.2: Vulnerabilities Reported On Operational Technology Systems

classes of vulnerabilities and attacks for operational technology systems.

2.1.2 Cyber Physical Systems

A CPS is a mechanism of monitoring, controlling or integrating the operations based on computer algorithms and communication core. In CPS, both physical and software components are deeply integrated to exhibit unique behavior while interacting with each other. Smart Grids, Autonomous Automobile Systems, Process Control Systems, Robotics Systems, Automatic Pilot are some of the examples that are considered to be CPS.

In CPS, process control is often referred as embedded systems, which mainly focuses on computation and CPS is considered to be similar to Internet of Things (IoT) with similar architecture. CPS brings together the discrete and powerful logic of computing to monitor and control the continuous dynamics of physical and engineered systems for tolerating the failures in both physical and cyber domains [RLSS10].

Some of the common failures of CPS are described by Mitchell and Chen [MC16] like,

- *Attrition Failure:* This kind of failure occurs when the CPS doesn't have enough control nodes or actuators to accomplish its intended functions.
- *Pervasion Failure:* This occurs when the density of compromised control nodes or actuators is too high.

- *Exfiltration Failure:* This occurs when the attacker secretly monitors the network and then leaks data for financial benefits.

2.1.3 Smart Grid Power Systems

The Smart Grid and Power Systems have enhanced the reliability and efficiency of supplying electricity through broadband and distribution, resulting in a cost-effective approach for power infrastructures, which impacts economic stability and development of power grids. The Smart Grid Power Systems are considered critical in terms of economy and security because of advancements in distributed intelligence technologies. However, appropriate security enforcement does not get supported by these techniques and create new vulnerabilities in power networks, leaving them open to a wide range of cyber-physical attacks. One such attack is described in detail by E-ISAC [EI16].

Security in smart grid power systems are composed of two main aspects: physical security and cyber security [LZL+17]. physical security tries to overcome the disturbances caused by natural disasters or any other physical attacks and helps to prevent the power systems from a complete blackout. Cyber security tries to overcome the weakness caused by the integration of physical systems with cyber systems. Earlier, people believed that the cyber-attacks were incapable of threatening the security of industrial operations. However, cyber-attacks have resulted in many security problems in recent years and have become a critical concern for both industrial control system users and clients [Boo]. Smart Grid Power Systems introduce enhancements and improved capabilities to the conventional power networking making it more complicated and vulnerable to different types of attacks. These vulnerabilities allow attackers to access the network.

According to the CVE database, four cyber-incidents were reported on smart grids in the year 2016. There are several other incidents on smart grids that are not made publicly available because of security constraints. Several researchers have exposed vulnerabilities of these systems that are briefly described by Aloul et al. [Alo12] and some of them are as follows:

- Customer Security.
- The number of devices (IEDs) has increased.
- Physical Security.
- There are no trust mechanisms implemented between power devices.
- There are compatibility issues between IT protocols and ICS protocols.

Based on these vulnerabilities, Hahn et al. [HG11] described attacks like CPU exhausting carried out on the application Layer. Data flooding and buffer overflow attacks took on network and transport layers, man-in-the-middle attacks on MAC layer, and jamming attacks on the physical layer targeting the availability of smart grids and power systems. Attackers can

also launch Denial-of-Service (DoS) attacks targeting the topology of the smart-grid [Alo12], preventing operators from making appropriate decisions. General attacks on smart grid power systems are:

- Malware Spreading
- Accessing Database
- Compromising the communication equipment
- Injecting False Information
- Network Availability
- Eavesdropping
- Utilizing the vulnerabilities present in industrial control protocols (MODBUS, DNP3, Profibus.).

Specific attacks on smart grid power systems described are as follows [APMR18], [MCHL14], [MRLG11], [YQST12], [YLR11]:

Load-Altering attacks attempt to control or change certain load types that are accessible through the internet to damage the grid through circuit overflow or disturbing the balance between the power supply and demand.

Denial-of-Service on smart grids is an attack where an adversary disrupts some or all the remote control system components.

Random attack are attacks used to overcome the detection mechanism implemented by the central system.

False Data Injection attack In this, the attacker knows the system model, including parameters that lead to the control of a subset of sensors and sends false inputs to the primary system.

Load Redistribution attack attempts to maximize the system operation cost subject to attacking resource limitation under the logical assumption that the control center implements practical corrective actions to minimize the operation cost based on the false state estimation outcome.

Economic attack is where an adversary intends to buy virtual power at the lower-priced node and sell at the higher-priced node.

Energy Deceiving is an attack where an adversary can inject either the forged energy information or forged link state information into the energy request and response message among nodes.

Open-loop Dynamic Load Altering is an attack where an adversary tries to manipulate vulnerable load without monitoring the grid conditions causing the grid to be impacted while implementing the attack.

Closed-loop Dynamic Load Altering is an attack where an adversary continuously monitors the grid conditions by hacking into a power system monitor infrastructure to control the trajectory on the target load buses based on the grid operating conditions.

Sphear Phishing is a cyber-attack method used to compromise systems and networks and gather information using social engineering techniques. A phishing email is designed to prompt a response from the recipient, such as clicking on a link or opening an attachment. Through the answer, the recipient may download malware or be redirected to a website prompting them to provide sensitive information, such as login credentials.

Credential Theft is an attack where an adversary tries to retrieve vital information like login details, etc.

Data Exfiltration is an attack where an adversary tries to exfiltrate the necessary information by discovering hosts and devices and plans an attacking concept to cause the power outage.

VPN Access is an attack where an adversary looks for existing point-to-point VPN implementations at trusted third-party networks or through remote support employee connections where split tunneling is enabled.

For the attacks mentioned above, researchers have proposed the following countermeasures:

- Metke et al. [ME10] proposed countermeasures for cyber attacks by introducing mechanisms like PKI standards, automated trust anchor security, certificate attributes, smart grid PKI tools, and Trusted Computing.
- Manandhar et al. [MCHL14] described a mathematical model and designed a robust framework that uses the Kalman filter with X2 detector and euclidean detector to detect the DoS attacks, random attacks, and false data-injection attacks.
- Electricity Information Sharing and Analysis Center (E-ISAC) mentioned how to overcome the power system's vulnerabilities based on the attack on the Ukrainian Grid [EI16].
- Liang et al. [LZL+17] proposed mechanisms like protecting a set of necessary measurements and PMU-based protection mechanisms for defending against economic, load distribution, and energy deceiving attacks.

2.1.4 Supervisory Control and Data Acquistion Systems

SCADA Systems are typically used to monitor, gather and process real-time data or operational data to perform specific industrial organizations operations. SCADA system consists of components like PLC, RTU, Human-Machine Interfaces (HMI), end-devices, control servers, and sensors. These components communicate internally to process data and network administrators analyzes the processed data to make meaningful decisions. Earlier, the traditional SCADA system goal was to perform reliable operations from isolated locations using data historians or other proprietary technologies to handle data. This would make the process operations

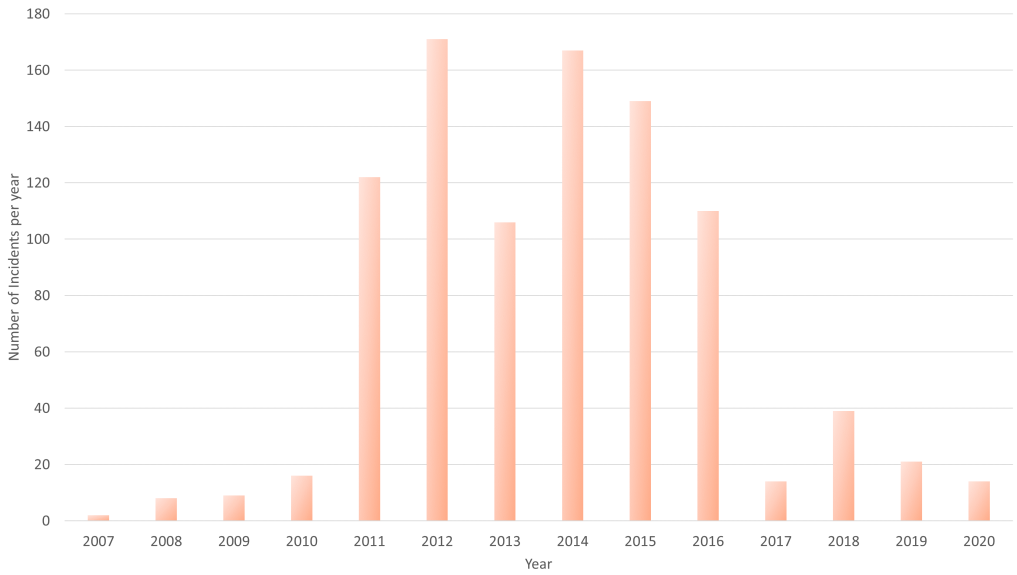

Figure 2.3: SCADA System Vulnerabilities Reported In CVE Database

complicated and inefficient. Modern SCADA systems solve this problem by leveraging with IT systems. This integration has reduced the gap between IT and SCADA systems. Modern SCADA systems allow real-time data to be accessed remotely through the internet. Due to this, the interconnectivity and remote accessibility of the systems have increased in SCADA networks, making them vulnerable to cyber-attacks.

According to vulnerabilities reported in the CVE database [CVE], since 2007, around 948 incidents have been reported. Figure 2.3 depicts the number of vulnerability incidents reported every year on SCADA systems. The incidents reported peaked during the years 2011 to 2016 and decreased in later years. The graph represents the analysis to the best of our knowledge, based on the CVE database. The vulnerabilities like buffer overflow, XSS, improper access control and authentication, privilege escalation, SQL injection, CSRF, remote code execution, hard-coded credential, and untrusted search path were found in these incidents.

RTUs are integral part of SCADA systems which monitors field data and sends to SCADA control servers. There are specific incidents which were reported against RTUs in CVE database. Figure 2.4 depicts the number of incidents reported on RTUs, where 43 incidents were reported in the CVE database on RTUs from 2010 to 2020. The maximum number of incidents were reported during the years 2018 to 2020. From the two figures Fig 2.3, Fig 2.4 we can see that incidents reported on SCADA systems from 2018 to 2020 were mostly based on RTUs. Interdependency between SCADA components will be one of the important aspects that should be considered in the future and it is necessary to develop countermeasures for SCADA internal components like RTUs and PLCs.

Based on these Vulnerabilities, the possible attacks on the SCADA systems are as fol-

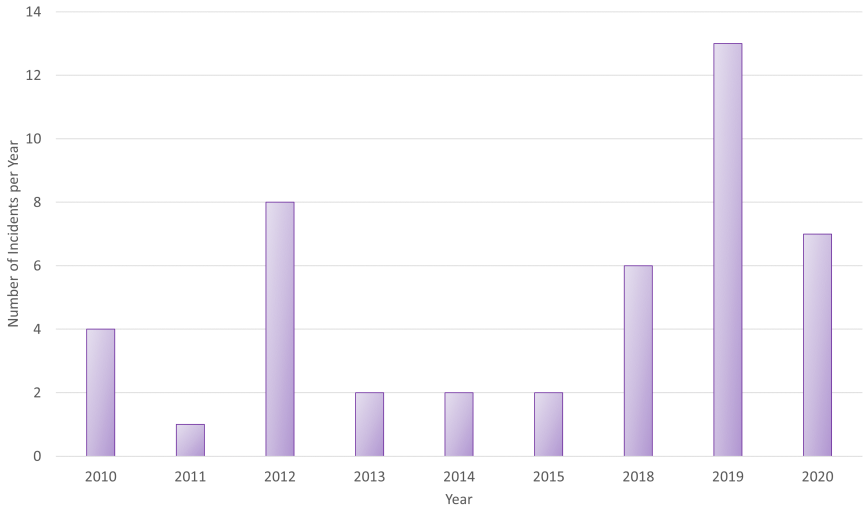

Figure 2.4: RTU Vulnerabilities Reported In CVE Database

lows [CRS09a], [CDF+06], [GKR+08], [KLKP09a], [MG13], [TLM08a]:

RADIUS Remote Authentication Dial-In User Service DoS attack causes damage by limiting or denying access to its resources by denying the service to authorized users.

ICT Worm Infection occurs because of the vulnerabilities found in the software installed on SCADA servers. Worms try to replicate and spread throughout the network. Then, by closing all the network connections, it isolates the SCADA systems.

Process Network Malware Infection attack is carried out by injecting the worm into the process network like RTUs, communicating using protocols like MODBUS or DNP3. Worms carry malware code in the payload by spreading themselves using resource hosts and executing malicious code on SCADA systems.

Phishing attack in SCADA systems are carried out by creating a fake website with a malicious code. This makes the user believe that they are connected to the legitimate website. When entered with their credentials, the adversary steals the credentials to get the SCADA system's direct access.

War Dialing Scanning attack executes the scripts on the surrounding numbers to detect potential connections once the main phone number is determined. Then the subsequent attacks are performed to penetrate into the SCADA control server.

Traffic Sniffing is used to capture the packets traversing within the network through a network analyzer.

Password Cracking is a software program that repeatedly tries to guess a password to gain unauthorized access to a network. An adversary can also use brute force or dictionary techniques to crack the password.

Warm Restart is an attack where a command is sent from the master controller to PLC, forcing it to execute an immediate reboot. Warm restart belongs to a family of DoS attacks and occurs on the DNP3 protocol. Multiple WarmRestart commands stop the PLC operations and result in DoS of the PLC.

Man-in-the-Middle attack is accomplished by first gaining unauthorized access to data. Next, communication between the Human Machine Interface (HMI) and the MODBUS server is spoofed to send attacks to either device and finally send busy exceptions to the HMI.

TCP SYN Flooding attack opens many connections on the target device. It leaves the connections open for the system to allocate resources for each connection. By sending multiple SYN packets, the target resources are exhausted, forcing the system's shutdown.

TCP ACK Flooding attack is carried out by sending multiple TCP packets with ACK enabled on it. The ACK packets indicate the target system has received the data. Continually performing this, an adversary can shut down the system.

Buffer Flooding attack is performed by sending multiple events to a device that temporarily buffers SCADA data before retrieving it in the control station. This limits the buffering of critical alerts from authorized devices, negatively impacting the control station's situational awareness.

Integrity attack is carried out by sending wrong inputs to other devices in the SCADA systems, which results in the corruption of data.

Reconnaissance attack are performed to gather control system network information, map the network architecture, and identify the device characteristics such as a manufacturer, model number, supported network protocols, system address, and system memory map. Example: Discovering vulnerable ports by using software like port scanning.

Response injection attack take three forms; first, response injection attack originate from the control of a PLC or RTU or network endpoints, which are servers that respond to queries from network clients. Second, response injection attack capture network packets and alter contents during transmission from server to client. Finally, response injections may be crafted and transmitted by a third-party device to the network. These attacks occur because of a lack of authentication features in ICS network protocols to validate packets origin, thus enabling attackers to capture, modify, and send response packets.

Command Injection attack injects false control and configuration commands into a control system.

Format String attack is performed against user data to compromise the root privilege account without inserting any external code.

Spoofing attack is carried out by jamming the devices across the network through interference with the device radiofrequency.

Replay attack is carried out by transmitting the malicious packets repeatedly.

Destroying a Node attack occurrs because of a lack of physical access security, which is beneficial for an attacker to destroy the nodes or devices, especially in sensor networks.

Environment Tampering is where an attacker can change the sensor readings by tampering in the deployment area.

Cryptanalysis attack refers to transforming encrypted data into plain text without having prior knowledge of encryption processes.

Exploit attack An attacker becomes accustomed to the system's vulnerabilities and launches an attack based on those vulnerabilities.

Sybil attack In this attack, a malicious device pretends to have multiple identities to impersonate other legitimated devices across the network. This attack is carried out in Sensor Networks.

Replication attack Here, an attacker attempts to add one or more devices with the same name as current device. Routing attacks decrease the availability of the system as the attacker tries to create false routes.

Time Synchronization Attacks resluts in mainly devices out of sync with each other. Sensor networks rely on synchronization to perform correctly using time-synchronized protocols. Attack on these protocols results in out of sync between devices.

Slander attack can cause each device to accuse the other if misbehavior occurs across the network. This is possible to detect if there is a implementation of distributed detection mechanism.

For the attacks mentioned above, researchers have proposed the following countermeasures:

- Fovino et al. [NFCMT09] proposed mitigation strategies for RADIUS, ICT Worm infection, Process Network Malware Infection, Phishing attacks, and DNS poisoning attacks, through the process of filtering and monitoring, and by specifying rules for TCP/IP and SCADA protocols.

- Three steps for intrusion detection were developed and proposed by Cheung et al. [CDF+06] for Model-based intrusion detection.

- Vulnerability assessment framework consisting of three levels, system vulnerability, scenario vulnerability, and access point vulnerability to detect attacks like directed attacks and intelligent attacks, is presented by Chee-Wooi et al. [TLM08a].

- Mallouhi et al. [MANC+11a] developed a module with a 100 percent detection rate to detect TCP protocol attacks (TCP SYN, TCP ACK, Man-in-the-middle) with low false-positive alerts.

- Jin et al. [JNY11a] proposed to use a rule-based policy approach to overcome the Buffer-overflow attacks.

- Giani et al. [GKR+08] have developed mathematical and computational models for the interaction between the physical and infrastructure processes to overcome Integrity attacks in SCADA systems.

These mitigation strategies are confined to a particular testbed of SCADA architecture and cannot be considered universal. With the advancement of modern SCADA systems, many more new vulnerabilities are expected in the future. It is necessary to understand the trends and attacks of the incidents reported to build a secure and robust SCADA system.

2.1.5 Embedded and Firmware Systems

Advancement in hardware devices and communication technology has led to an increase in many embedded devices in critical communication infrastructures. Embedded systems are computing systems built into an extensive network, designed for dedicated functions to interact with general-purpose systems. These systems include hardware, software, and other mechanical parts [YLR11], [APMR18]. Extensive use of Embedded devices in the Internet of things (IoT) has caused security breaches in OT systems.

One of many problems associated with embedded devices is firmware. Firmware is the software for the embedded device that is loaded and run when the device starts. It is usually stored in permananent storage in the device. Microprocessors are considered the core of electronic systems, and firmware software can effectively use various hardware interfaces associated with them. Firmware has become the primary component of any automated method instead of many custom made hardware. For each equipment we can use more general purpose microprocessor and specialized software. Due to this advancement, securing this has become an essential aspect in the real-world [PMB15].

Analysis of Embedded Systems vulnerabilities from CVE database are provided by Kocher et al. [KLMR04], and some of them are as follows:

- Programming Errors: Control flow attacks like input parsing vulnerabilities leads to buffer overflow attacks. Memory management problems such as pointer exception occur because of not following coding standards while developing software programs.
- Web-Based Vulnerability: Lack of updates or patches for web applications used by embedded devices are exposed to specific attacks because of the vulnerabilities existing in the web interfaces.
- Weak Access Control or Authentication: Use of default or weak passwords. Some systems have hard-coded passwords to keep them simple. This provides backdoor access to those who know, makes it easier for attackers to bypass.
- Improper use of Cryptography: Use of random generators for generating cryptographic keys or vulnerabilities in protocols have a severe impact on embedded devices.

According to vulnerabilities reported in the CVE database, since 2000, around 843 incidents have been reported. Figure 2.5 depicts the number of vulnerability incidents reported every year on embedded systems. Until the year 2012, there were not may incidents reported in

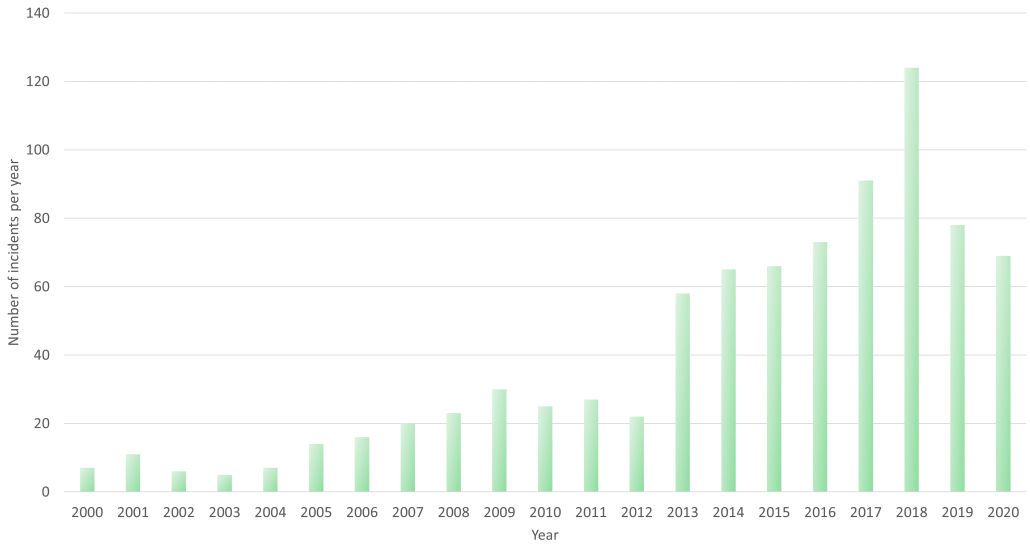

Figure 2.5: Embedded Device And Firmware Vulnerabilities Reported In CVE Database

CVE database. From the year 2013 to 2018, maximum number of incidents were reported and in the years 2019 and 2020 incidents reported were less than the year 2018. The graph in Figure 2.5 represents the analysis to the best of our knowledge, based on the CVE database. The vulnerabilities like buffer overflow, XSS, improper access control and authentication, privilege escalation, SQL injection, CSRF, remote code execution, hard-coded credential, and untrusted search path were found in these incidents.

The effect of these vulnerabilities causes a different type of attacks on Embedded and Firmware Systems as mentioned in several papers [PMB15], [BA01], [CCS13], [Mil11]:

Control Hijacking attack is where an adversary tries to divert the program's normal flow running on the embedded device, resulting in executing the code injected by the attacker. Reverse Engineering is a process of obtaining sensitive information by analyzing the software in the embedded device. Using this technique, the attacker can find the vulnerabilities of code like input parsing errors.

Malware attack is where an adversary tries to infect an embedded system with malicious software known as malware, which can modify the device's behavior, resulting in severe consequences. Brute-force search attacks like exhaustive key search can occur because of the weak cryptography and weak authentication methods in embedded devices. An adversary bypasses the system with brute force search attacks by using random credentials to login.

Eavesdropping or Sniffing is a passive attack, where an attacker observes the messages sent and received by an embedded device.

Injecting Crafted Packets or Input is an attack method against the protocols used by embedding devices. Both packet and input crafting attacks exploit parsing vulnerabilities in

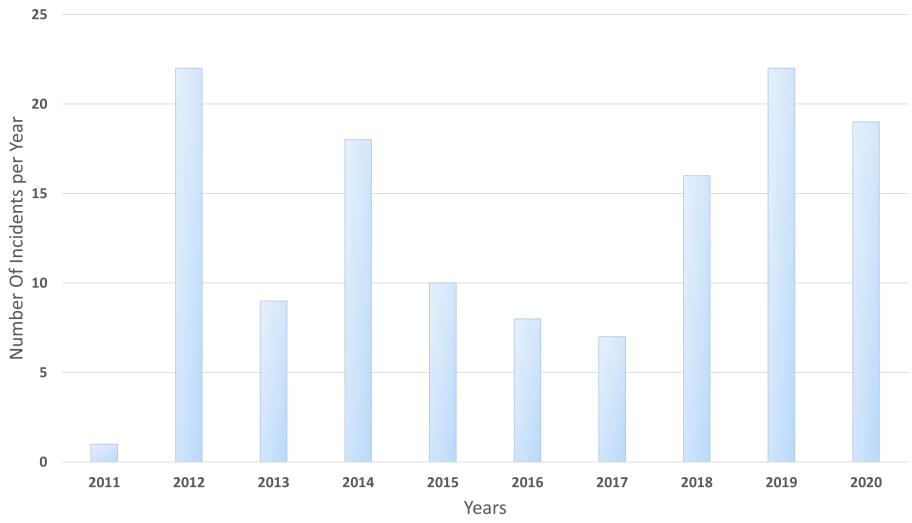

Figure 2.6: PLC Vulnerabilities Reported In CVE Database

protocol implementations or other programs. Also, replaying previously observed packets or packet fragments is considered a particular form of packet crafting, which can be an effective method to cause protocol failures.

Known-key attack occurs because of the vulnerability in the standard protocols. An adversary takes old key and supplies it twice in the second transaction, resulting in a known terminal key. They are considered the core of all zeroes, as the key is exclusive- or' ed with itself. Thus, it enables an attacker to encrypt the PIN key under a terminal master key so that an ATM can verify customer PINs while the network is down. So now the attacker can obtain the PIN key encrypted under the all-zero key. Decrypt it using his computer and is then able to compute any customer's PIN.

Two-Time Type attack is carried out by writing a program that maps the possible key and data transformations between different vital types, computing the transitive closure, and scanning the composite operations for undesirable properties.

Meet-in-the-Middle attack exploits try to abuse all the keys of a particular type. It is usually only necessary to get one of them and leads to a natural time-memory tradeoff in a key search. Firmware Modification attacks like reprogramming the battery affect devices consisting of design flaws, operating systems of different versions, and instruction set architectures.

CIH Virus rewrites the BIOS firmware online, causing BIOS firmware program to crash and fail while loading the systems.

Control-Flow attack occurs when information like return addresses are stored together with functions. These function variables come from untrustworthy sources, and sufficient checks are not in place, leading to memory corruption. This corruption allows an adversary to corrupt

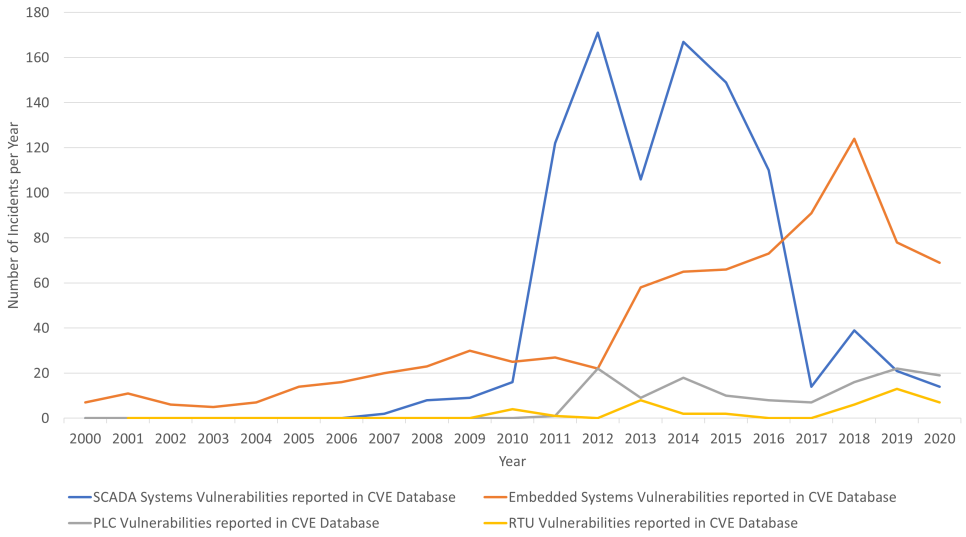

Figure 2.7: Comparison Of Incidents Reported In CVE Database On SCADA, Embedded, PLCs and RTUs

the control flow information stored on the stack.

Stack-overflow attack occurs while allocating too much data on the stack or when the stack's depth grows large. In both cases, the stack exhausts its available memory. It overlaps with other memory sections like the BSS section causing both a reliability problem and a security problem.

Configuration Manipulation attack allows an attacker to modify an embedded device's critical configuration parameters to force it to misbehave. These attacks are concerned with programmable devices such as PLCs, which can control the programming logic process.

For some of the attacks mentioned above, researchers have proposed the following counter-measures:

- Hou et al. [HLC17] proposed Automatic Binary Structure Randomization (ABSR), which disables unnecessary features and removes the unused binary from the firmware image to overcome the firmware modification attack.
- The CIH virus were peak during the years 1996 and 1998 [CJ03]. Numerous solutions to overcome this attack are given, like implementing fuzzy testing, behavioral analysis, homology analysis, etc.
- Split Stack and instruction-based memory control techniques were developed by Francillon et al. [FPC09] to overcome the control-flow and stack-overflow attacks in embedded and firmware devices.

- Memory verification and control-flow integrity techniques are used as countermeasures to overcome configuration manipulation attacks and control-flow attacks on embedded systems [Abb16].

2.1.6 Programmable Logic Controller (PLC)

Programmable Logic Controllers [McL11] are real-time systems that closely monitor and control plant devices to keep the process functioning correctly. In general, PLC receives information from input devices, processes the data, and triggers outputs based on the preprogrammed parameters. This program in PLC considers as logic and executes within the control flows.

Research work on PLCs like detection of vulnerabilities and attacks on PLC's concerning embedded systems are investigated by Abbasi et al. [Abb16]. In this work, the authors have described the different attack techniques used by an adversary to tamper the I/O operations in PLC and also showed how certain detection mechanisms implemented in PLCs can be evaded. Sandaruwan et al. [SRO13] have revealed the vulnerabilities of PLCs through attack vectors affecting critical infrastructure, described Stuxnet attacks and proposed solutions to overcome the weaknessess in PLCs. McLaughlin [McL11] constructed an dynamic payload, based on the process observation in control system and claimed that this process will significatly lowers the attacks agains PLCs. Gonzalez and Hinton [AGH14] have demonstrated the feasibility of power fingerprinting technology to monitor PLC and detect malicious software using Siemens S7 PLC.

According to vulnerabilities reported in the CVE database, since 2011, around 132 incidents have been reported. Figure 2.6 depicts the number of vulnerability incidents reported every year on PLCs. The peak of incidents were reported from 2012 to 2014 and again there was a rise from 2018 to 2020. The above graph represents the analysis to the best of our knowledge, based on the CVE database.

Some of the attacks on PLC are described as follows; **Pin Control attack** consists of misusing the pin control functionalities of the embedded systems at runtime, causing physical damage, communication block, and manipulation of reading or writing values from a device, to be a legal process. Because of the vulnerability embedded system, an attacker can modify the multiplexing registers, which leads to the disconnect of the legitimate device.

PLC Malware Stuxnet is considered to be the PLC malware, and Stuxnet intends to reach the nodes in the SCADA systems that connect to the PLC's operating the target plant MTUs. Once executing on an MTU, Stuxnet uploads its payload of static code blocking the PLC, resulting in the process to be under malicious control.

By-Pass Logic attack In general PLCs contains main memory and register memory. Register memory contains some variables assosiacted with main logic. This register memory are allowed to access by other PC's in industrial plants with read and write operations across PLC networks. When an adversary gains access to one of these machines, he can change the values arbitarily and cause the system to collapse.

Brute-Force Output attack In PLCs, an PLC operator can remotely change the output forcefully by connecting to the PLC through a network or internet with out any authentication.

When an adversary gain the access to PLC, he can force the output to shutdown some valves etc, which will lead to awful concequences.

Replay attack In this attack, an adversary gets some information about PLC and uses the same information to compromise the system later.

S7 Authentication Bypass attack In this attack, an adversary can easily bypass the authentication by intercepting the valid user packet and replay that packet at later to authenticate himslef. This is because of lack of security in the protocol.

The comparison of cyber-attacks on SCADA, embedded systems, PLCs, and RTUs are given in Fig 2.7. From the graph, the incidents reported on all four of the systems were at peak during the years 2010 to 2018. Even if there is a decrease in the number of incidents reported in past two years, still it is important to understand these trends and address the vulnerabilities in these systems.

Chapter 3

Software-defined Networks (SDN)

SDNs have been used for years by information technology (IT) systems to provide quick responses to meet the organization's changing needs. The same concept has been applied to Operational Technology (OT) systems to increase the performance and enable fault-tolerance capabilities for networked control systems. Features like secure communication and fault tolerance can be achieved through the use of SDN. This approach has significant advantages such as securing the control plane, restricting unauthorized access, and enabling the visibility to monitor devices across the network.

OT Systems using SDN are considered a modern approach to build secure and reliable systems. Researchers have shown that it is a dynamic technology for building secure and reliable systems. The SANS 2019 survey report on OT/ICS systems [FWD19] have mentioned that the adoption of SDN in an OT domain enables the flexibility and dynamic logical segmentation of contemporary ICS. By complementing Network Access Contol (NAC), SDN technology enhances network resiliency and provides segmentation when more devices are connected to ICS.

There are several publications regarding the architecture of SDN, features of SDN, advantages of SDN, performance measurement of SDN, challenges of SDN and difference between traditional and SDN networking. Machado et al. [MGSFW14b] worked on QoS and policy management in SDN, and Lara [LR14] have implemented the security policies using OpenFlow and Ben-Itzhak et al. [BIBC+15b] created a framework called EnforSDN for specifying the network policies. Scenario-based programming for SDN has been developed by Yifei et al. [YLA+18]. Most of the research publications primarily focused on behavior and implementation of SDN technology. Minimal research has been conducted on how to retrieve the logical flows implemented on the SDN switch. This gap has motivated us to describe a process and develop a tool for retrieving the implemented logical flows on SDN switch for formal analysis and verification purposes.

Other works have proven that SDN technology can increase operational performance and efficiency of OT systems. Cruz et al. [CQSM16] illustrated how interoperability and performance using SDN can be achieved in real-time networks. Hadley et al. [HNS17] have summarized the

advantages of SDN on control systems. Thubert et al. [TPE15] proposed SDN-based centralized methods for global optimizations on the Internet of Things (IoT). And a survey on SDN for IoT has been conducted by Bizanis et al. [BK16] emphasizing the impacts of SDN on 5G networks. A framework for network resilience using SDN has been designed by Modarresi et al. [MGS17], and Clements et al. [CSNN20] have proposed solutions by evaluating SDN to reduce the digital attack surface of nuclear security systems.

3.1 SDN Architecture

SDN is a network architecture, which enables the system to be programmable using software applications to improve the performance of the network. The goal of SDN is to enable cloud and network engineers and administrators to respond quickly to changing business requirements via a centralized control console [KAL+18]. The four key features of SDN architecture are:

- *Centralized View*: In SDN, controller is centralized by maintaining the configuration details of all the devices, which enables to view across the entire network.
- *Dynamic Configuration*: In SDN, network administrators can configure and optimize the traffic flows dynamically through built-in programs without depending on third-party software.
- *Flexibility*: Separation of the control plane from the data plane enable network administrators to configure the traffic flow to meet changing demands.
- *Programmability*: In SDN, network controls are programmable because it is decoupled from the data plane.

SDN Architecture consists of Data Plane, Control Plane, Application Layer, and protocols for communication between them. Communication between controller and data plane is carried out through a southbound application programming interface (API) like OpenFlow protocol, widely used in SDN networks. The communication between controller and application layer is carried out using northbound API like procera or frenetic.

The Data plane is responsible for forwarding Ethernet data packets. In contrast, the Control plane located in the flow controller will program and configure all the switches in the network to provide benefits like flexibility, speed, and security. The Application layer simplifies the network configuration process. It enables the user to configure their interfaces using abstraction, a mechanism to interact with the low-level programming rules and actions, to control the flow of the packet.

An SDN has two main components, the Flow controller and the network appliance (SDN Switch). Being placed centrally, the controller will have a global view of the network to improve its decision-making process. It facilitates network administrators to program the entire network easily. Problems that had previously taken a long time to detect and resolve, such as

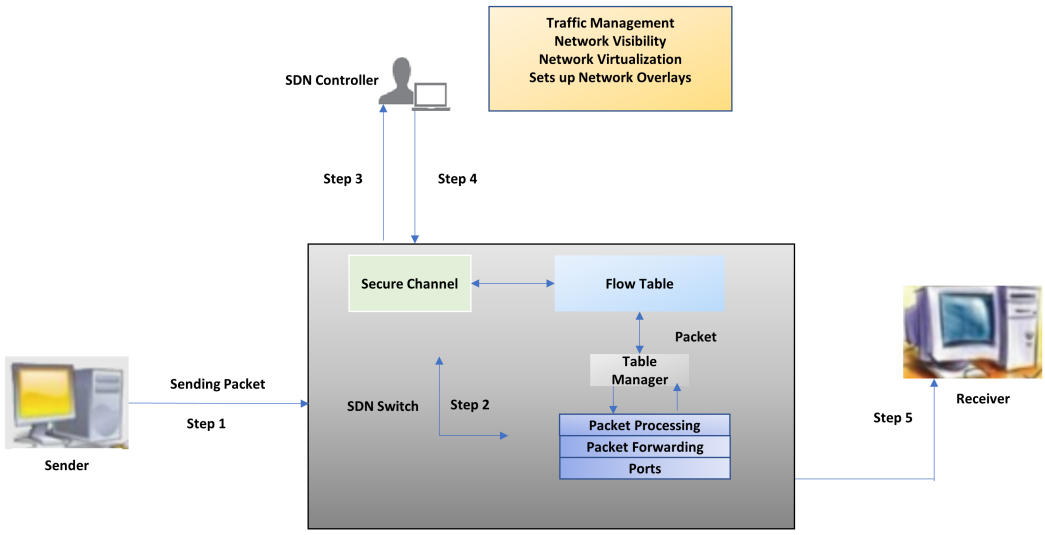

Figure 3.1: Depicting The Basic Workflow Of SDN Switch.

configuration issues, the detection and mitigation of attacks, and new protocol implementation for legacy network devices, can now be fixed in a matter of hours with SDN [RS16]. For this research, we focus on the rules programmed into the switch by the controller. It is important to note that in data centers and cloud infrastructures, the flow controller is an integral part of the network, dynamically installing new rules and re-configuring the network. In an ICS system, which requires consistency, reliability, and resilience, it is more common to use the flow controller to configure the network and then take it offline, reverting to a static network configuration. It is this static configuration that is the focus of this research.

3.1.1 Workflow of SDN

The basic workflow of a traditional SDN switch is depicted in Fig. 3.1. The default nature of the SDN is that upon receiving a data packet at the switch, the switch checks the entries in its flow table. If there is any flow associated with that particular packet, it processes the packet according to the flow rules in the table. In a dynamic configuration, such as that found in a traditional IT network, if there are no flows related to the received packet, the switch forwards the packet to the controller. This is the significant difference between an SDN switch and a traditional switch, where the latter broadcasts the received packet to all the nodes connected to the switch, hoping to find one that responds. The SDN Flow Controller then checks if there are any rules associated with the received packet. If any rules are present in the controller flow table, it sends the packet back to the switch with the appropriate rules by adding the flow rules to the switch flow table. If there are no rules in the controller flow table, then the packet is dropped. This nature of SDN is known as deny-by-default.

In a static configuration, such as an ICS network where there is no flow controller present,

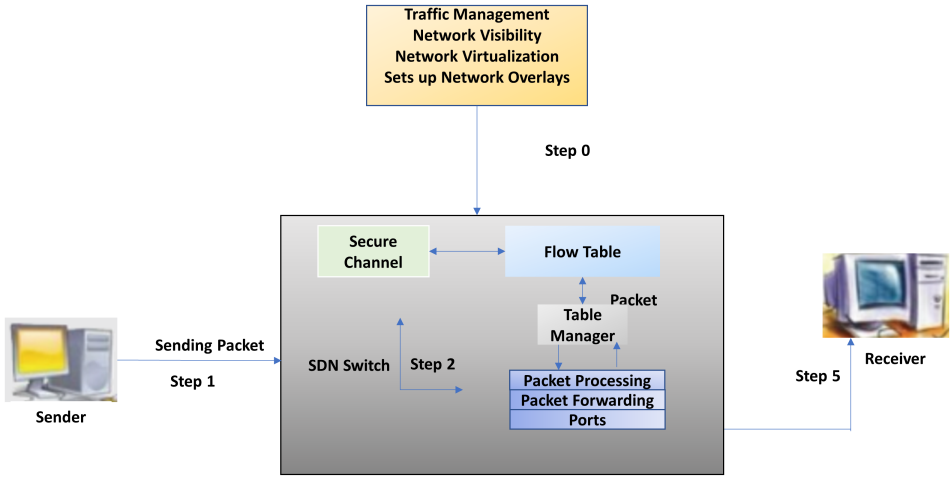

Figure 3.2: Workflow Of SDN Switch Across OT Networks

the switch will just drop the packet. It is recommended that ICS networks create a default rule that forwards the unrecognized packet to a logging station for analysis, since static networks should never receive unknown data packets. Since the rules are all defined statically, it makes sense to evaluate the configuration data that is used to configure the SDN swtich. For this research, our experiments were based on the Schweitzer Engineering Labs SEL-5056 SDN Flow controller and switch SEL-2740S [Sch19]. The workflow of SDN switch across OT networks is shown in Fig 3.2. The static configuration of OT networks allows network administrators to pre-define the flow rules. In this case, the SDN switch will automatically detect the nodes across the network. And when the packet arrives at the switch ingress port, it checks its internal flow table, and if entry is found, it forwards the packet to the appropriate destination through an egress port. For unexpected traffic or flow rules that are not pre-defined, SDN switch drops the packet at its ingress port.

3.1.2 SDN Flow Controller

A flow controller is software that constitutes a control plane and configures all the network appliances on how to forward data packets. It has three components [Sch19]: match, actions and counters.

The *Match* component determines which control plane rules to apply to each packet entering the switch port. The match rules are applied to data packet headers, such as source and destination IP addresses and TCP/UDP ports, or similar header fields in other protocols, such as GOOSE. The match rules are searched in-order, with the first successful match being used

to select the action for the packet. The *Action* component instructs the switch what to do with the packet, such as forward, modify or drop the packet. The *Counter* component is used to monitor the overall status and health of the network by updating log counters concerning the packet.

A typical SDN controller will maintain an internal control flow table specifying all of the rules for the network components it controls. This will typically involve several network switches. The controller then pushes out the appropriate subset of rules to each SDN switch. A secure network will utilize digital signatures and other encryption technology to authenticate this programming of the switches. Attacks against the controller and the authentication to the switches are beyond the scope of this dissertation. Rather the scope of this work is understanding the network as configured.

In a normal network design, high-level policies are defined at the organization level and implemented across all the network components by a network administrator. For example, the policy that communication between two end-devices should consist of authorized messages [VAFR19b] is considered as high-level policy. This is converted to low-level policies and implemented in the switches. The SEL-5056 Flow controller has an internal database that is used to store the control flow information of the network. The flow controller stores the high-level policies as logical connections in the database for security purposes and can automatically generates and exports a configuration backup file. For this dissertation, the backup file generated by the SEL 5056 Flow controller is called backup.db. This file consists of configuration and security certificates of all switches and hosts in the system. It can be stored on the same computer where the controller is running, or on backup media. The latter case is recommended so that if the controller running the system fails or is corrupted, the user can import the external backup file to restore the network configuration. Deleting the database removes all of the certificates and credentials from the controller, and the SDN switches will no longer be accessible across the network.

3.2 Classification of SDN Vulnerabilities

The vulnerabilities in SDN are outlined by several researchers are [ACP+17], [DMSK16], [FP18], [KAL+18], [RS16]:

- Communication bottleneck between the data and control planes can easily overload the control traffic in many situations.
- Hardware-Based challenge, like the controller, might be the primary choice for attackers because the controller is the logically centralized device, which functions as the brain of a network. Therefore, many traditional attack approaches and several new attack behaviors are instrumental in making a controller disabled.
- Protocol-Based Challenges. Different OpenFlow versions except the first edition make

mutual authentication optional between controller and switches, which can lead to a data information leak. Because of this, man-in-the-middle attacks have occurred, and attackers can steal network information and modify flow rules in the switches to take control of the whole network. Because of these vulnerabilities, Denial of Service (DoS) attacks are possible. A DoS attack is an attempt made by a malicious user to compromise the regular functioning of a network. If this attack comes in a group of hosts, instead of one host, then this is called a distributed denial of service (DDoS) attack.

3.2.1 Vulnerabilities Based on SDN Architecture

SDN architecture consists of three planes: application plane, control plane and data plane. The list of vulnerabilities concerning SDN architecture is described briefly in Table 3.1, and Fig 3.3 outlines the attacks that can be exploited based on the vulnerabilities listed in Table 3.1. The innermost area is segregated into three sections: application, control, and data plane. The middle area represents the possible vulnerabilities associated with each section, and the outermost area represents the potential attacks carried out by exploiting these vulnerabilities.

Application Plane consumes SDN communications and networking services for end-user business requirements and does not provide authentication, authorization, and accountability mechanisms. Exploration of this vulnerability by the adversary in the application plane will lead to an access control attack. The different attacks viable on the application plane are DDos attack, storage attack, malicious attack, and control message attack.

Control Plane provides control functionality and supervises the packet-forwarding behavior through an open interface. Misconfiguration of controllers leads to unauthorized control access vulnerability, and when exploited by an adversary, they can change the controller's internal implementation leading to hash collison attack. Other possible attacks on the control plane are, link spoofing attack, availability attack, software hack attack, saturated and single point of failure attack, and inference attack.

Data Plane consists of forwarding elements, including physical switches and virtual switches, is accessible via an open interface to forward packets. As mentioned in Fig 3.3, attacks like unbounded Flow state memory allocation and device attack are performed by exploiting the specific vulnerabilities concerning protocols and flow table mechanisms used.

The classification of vulnerabilities and attacks on SDN architecture helps to determine if it impacts SCADA systems. For example, most ICS using SCADA systems are static, and when SDN technology is implemented on these systems, the controller has limited scope. Therefore, the control plane's vulnerabilities do not significantly impact SCADA systems performance or security.

3.2.2 Vulnerabilities Based on Control Flow Operations

The control flow operations inside SDN switches determine the communication between end-devices. There are three types of control flow operations:

Table 3.1: Vulnerabilities On SDN Architecture

SDN Architecture	Vulnerabilities
Application Plane	Exhausting CPU, memory, bandwidth, and socket resources. Lack of authentication, authorization, and accountability mechanisms. Access to the shared storage is permitted. Fraudulent Rule Insertion. Lack of monitoring systems which helps to check for unauthorized changes or malicious attemp.t Can send control messages continuously at any time. Allowance of executing System EXIT command and can disconnect all the controller instances. Security rules and configuration conflicts.
Control Plane	No mechanism to ensure integrity and originality of LLDP packets. Able to flood the controller with packet-in messages. System variable such as time can be altered to turn the controller offline. Scalability issues. Flow table capacity is limited. Parsing errors results in buffer overflow vulnerability on controller. Unauthorized controller access leads to change the internal implementation of the controller causes hash collision attack. Attack Traffic can be mixed with normal traffic and is hard to differentiate. Limited computing and storage resources of the controller. Distribution of access-control in multi-controllers leads to DoS attacks. Inconsistent configuration of multiple Controllers causes DDoS attacks. Flow rule modification vulnerability to modify packets. Improper message configuration exchange vulnerability. Lack Of TLS mechanisms in controllers.
Data Plane	Can identify rules using input buffer to analyze the forwarding policy. Prone to hardware and software bugs. Inconsistency in flow rules. Requirement of a large memory space for each switch causes an adversary to exhaust the memory of the device. Network protocol vulnerabilities. Forward policy discovery vulnerability. Switch flow table flooding. If the listener Mode is activated on the switch, there is the possibility to establish a connection from the controller to the switch with no built-in authentication and access control. Lack of authentication mechanisms in the data plane. Enormous number of packets can be sent in short time

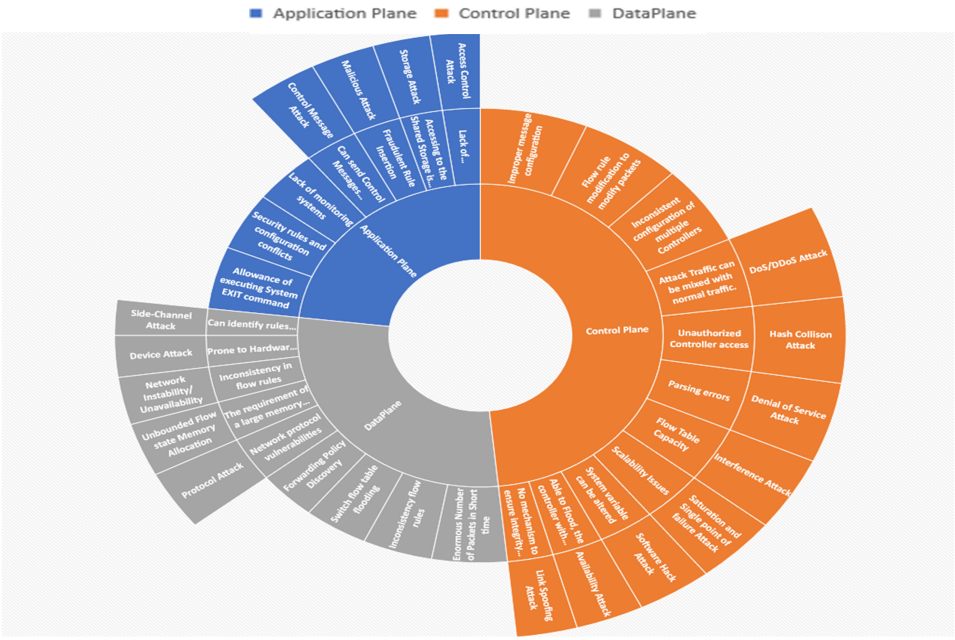

Figure 3.3: Vulnerabilities And Attacks In Application Plane, Control Plane And Data Plane

Table 3.2: Vulnerabilities On Control Flow Operations

SDN Control Flow Operations	Vulnerabilities
Symmetric Flows	Weak authentication during the handshake step The adversary can persistently replay meaningless handshake messages to exhaust the internal storage of the controller. An adversary can alter the message carrying OpenFlow version information. An adversary can alter the message carrying OpenFlow header information.
Asymmetric Flows	A listener mechanism allows applications to register to receive specific messages from the data plane. The attackers can interfere to cause the applications to drop the message. Malicious applications can alternatively implement an infinite loop to prevent other apps from acting on the message. Control messages between the controller and the switch are unencrypted, an attacker located between them can guess what topology is constructed by sniffing control messages in a passive manner. An adversary can intercept the control message and then change some field values of the control messages with malicious intent.
Intra-controller control flows	No access control mechanisms to limit API usage among applications. Exploiting the vulnerability of the mechanism that dynamically controls applications running on the controller. For example, the malicious application can dynamically unload a Firewall application. Notifications can be turned off dynamically.

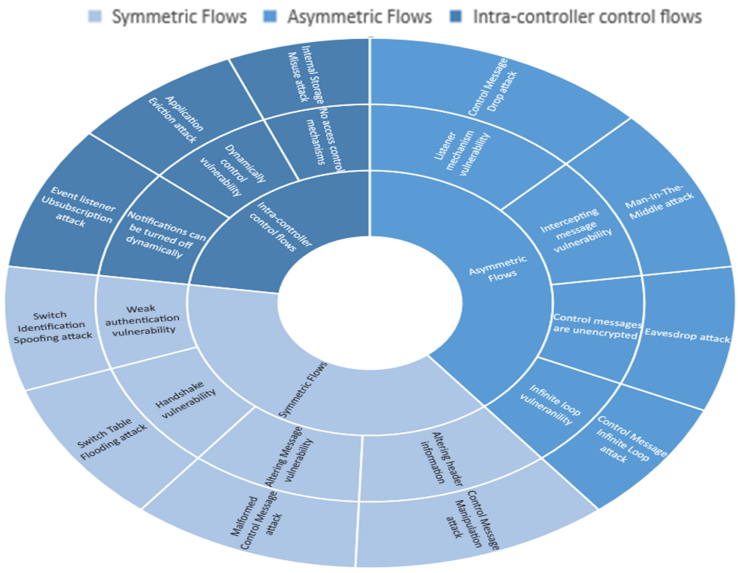

Figure 3.4: Vulnerabilities And Attacks Based On SDN Symmetric Flows, Assymetric Flows and Intra-Controller Flows

- Symmetric Flows: In this operation the SDN component sends a request to another component and receives a reply. These are often termed as bi-directional flows. Since it waits for the response, an adversary can reply with meaningless messages to exhaust the controller's internal storage, leading to switch table flooding attack. Other attacks concerning to Symmetric flows are switch identification spoofing attack, malformed control message attack, and control message manipulation attack.

- Asymmetric Flows: Operations involving unidirectional flows, i.e., components that do not require a reply, are considered asymmetric control flow operations. The possible attacks on asymmetric flows are control message drop attack, control message infinite loop attack, and man-in-the-middle attack.

- Intra-Controller Control Flows: In this, the flow operations are initiated by applications running on a controller. When applications interact with one another or uses the controller's internal services, the controller exposes API's. This limits the API usage among applications and leads to internal storage misuse attack. Other attacks are the application eviction attack and event listener unsubscription attack.

The Vulnerabilities for SDN control flow operations are summarized in Table 3.2, and the attacks concerning them are outlined in Fig 3.4. The innermost section is segregated into three parts: symmetric, asymmetric, and intra-controller control flows. The middle section represents the possible vulnerabilities concerning each part, and the outermost section refers

to the potential attacks carried out by exploiting specific vulnerabilities. For example, in symmetric flows, altering message vulnerability leads to malformed message vulnerability attack and listener mechanism vulnerability in asymmetric flows leads to control message drop attack.

3.3 Summary

In this Chapter, introduction to SDN and its workflow are presented. Then, the vulnerabilities based on the SDN architecture and its control flow operations are summarized. These threats on SDN warrants research to determine if any of the vulnerabilities influence the security and performance of OT sytems integrated with SDN technology. This is examined in detail in Chapter 4.

Chapter 4

Security Properties for SCADA Systems Using Traditional and Software-Defined Networks

Software-defined Networks (SDN) have several advantages over traditional networks, and since the data plane and control plane are separated, there are few vulnerabilities and attacks that exist. Operational Technology (OT) systems like Industrial Control Systems (ICS), SCADA systems, Embedded Systems, etc. integrated with Information Technology (IT) networks are increasing, which has both advantages in terms of production and disadvantages in terms of cyber-attacks. Some researchers have shown that replacing SDN networks with traditional networks in OT systems reduces the cyber-attacks based on the features offered by SDN networks. The question here is, how reliable are SDN networks when compared with existing traditional networks, and can we utilize all the features provided by SDN when integrating with OT systems. This chapter addresses these questions by analyzing the standard security features incorporated by the SDN and Traditional networks.

4.1 SCADA SYSTEMS

SCADA systems are used to control or monitor critical infrastructure systems, electric power generation transmission, water management systems, oil and gas systems, and other distributed processes. Originally SCADA systems were built for standalone systems or isolated networks without much connectivity to external networks. Things have changed over the years as new technologies like “Smart Grids” have been active in advanced computing and have made power grids operate effectively. Consequently, SCADA systems have become a primary target for cyber attacks for numerous reasons. Still, there are no practical solutions to protect against these attacks, which is reasonable since SCADA systems with in process networks or corporate networks are heterogenous and the operating systems used by these systems are old. To protect

these systems using anti-virus, one should have the latest updated version of the operating system. By their nature, SCADA systems are not suitable for up-gradation as, to install updates, all of the production units that SCADA system manages should be off, which may lead to loss of service. Even if somehow SCADA systems operate to migrate, there is no guarantee that the new software supports all the control system features.

SCADA systems collect data from remote facilities and send commands to the physical process creating a feedback control loop. There are three components in the SCADA System:

- Remote Terminal Units (RTU's): In RTU, the collection of data is achieved by connecting to a physical system.
- Human-Machine Interface (HMI): The data collected in the RTU's are compiled and formatted in HMI's; network operators can make supervisory control decisions.
- Communication Infrastructure: This is responsible for connecting the SCADA components through the internet.

Commonly used standard communication protocols in SCADA systems are MODBUS, DNP3, and Profibus. There are several vulnerabilities found in these protocols when combined with TCP/IP [FCMT09], [JNY11b]. The encapsulation of MODBUS over TCP/IP is known as MODBUS TCP/IP protocol.

TCP/IP application data unit consists of a header and MODBUS Protocol Data Unit. This migration of TCP/IP network for MODBUS client makes the systems vulnerable to specific attacks against the operating system, even if they are not against standard MODBUS Protocol [FCMT09]. DNP3 protocol is considered to be reliable in terms of performance, but it has many vulnerabilities that can cause the failure of the control system [JNY11b], [MANC$^+$11b]. New DNP3 controlled devices use TCP/IP-based connections, and ensure security mechanisms by encrypting the DNP3 communications. Enhanced versions like DNP3 secure authentication or DNPSec are still in the evaluation phase, and the majority of the DNP3 controlled devices are working with little security.

The communication protocols designed for Industrial Control SCADA systems considered the nature of the standalone system and isolated networks. Rebuilding the protocols, taking security into consideration is an infeasible approach as most of the systems are deployed with MODBUS and DNP3 protocols.

The types of attacks the SCADA system is vulnerable to and the countermeasures for these attacks have been examined by researchers. For example, the security properties, threat models, and the design space in SCADA systems is illustrated by Cardenas et al. [CRS09b]. Fovino et al. [FCMT09] investigated malware attacks on SCADA systems. Analysis of cyber threats on SCADA systems was done by Kang et al. [KLKP09b]. Cheung et al. [CDF$^+$07] developed model-based intrusion detection systems for SCADA networks. A detailed study of Vulnerability Assessment of Cybersecurity for SCADA Systems was conducted by Ten et al. [TLM08b].

4.1.1 Vulnerability Assessment of SCADA Systems

Identifying vulnerabilities of any network are essential, and vulnerabilities in SCADA System are as follows:

- Lack of authentication and authorization mechanisms in protocols (e.g., MODBUS, DNP3, and Profibus).
- SCADA system protocols are different from IT systems, and they have not been built considering IT scenarios, which have led to massive cyber-attacks over the years.
- SCADA system protocols use the master and slave concept, but they lack mechanisms to check the integrity of the packets sent from master to slave and vice-versa.
- An attacker can easily get access to the devices because of the lack of authentication among sensor networks and SCADA system components like actuators, SCADA servers, and RTU's.
- Network segregation between SCADA networks and IT networks are inefficient.
- Lack of physical access security.
- Susceptible to a single point of failure.

For example, when there is no authentication between SCADA components, with minimal effort, an adversary can quickly get access to the SCADA system and perform a read and write operation. Exploiting this vulnerability leads to a DNS Forgery attack [ZJS11]. Standard protocols like MODBUS and DNP3 are integrated with TCP/IP protocol, which enables the blind port scan feature by default, and this helps the adversary to take advantage and perform Idle Scan Attack. Once the adversary enters the system, he can start injecting malicious packets and try to saturate the system resources by sending TCP connection requests faster than the system can process. This process leads to a TCP SYN attack, and if this is successful, the adversary may gain control and shut down the system [CDF+07].

In this chapter, we classify the vulnerabilities and attacks of SCADA systems based on the SCADA hardware, SCADA software, Protocols, and some layers of Open System Interconnection (OSI) stack. Table 4.1 presents the vulnerabilities based on Network Layer, Transport Layer, Application Layer, SCADA software, SCADA Hardware, MODBUS TCP/IP protocol, and DNP3 Protocol. When the adversary exploits the vulnerabilities, it leads to attacks like Smurf attack, TCP SYN attack, DNS Forgery Attack, Integrity Attack, Buffer Overflow Attack, etc. Fig 4.1 presents the list of attacks on SCADA systems that an adversary can perform by exploiting the vulnerabilities. For example, an adversary can exploit the network layer's vulnerability by sending the fake ARP messages. This confuses the switches connected across the network, leading to Arp Spoofing/Poisoning attack. The vulnerability exists because of the lack of access-control mechanisms among SCADA components.

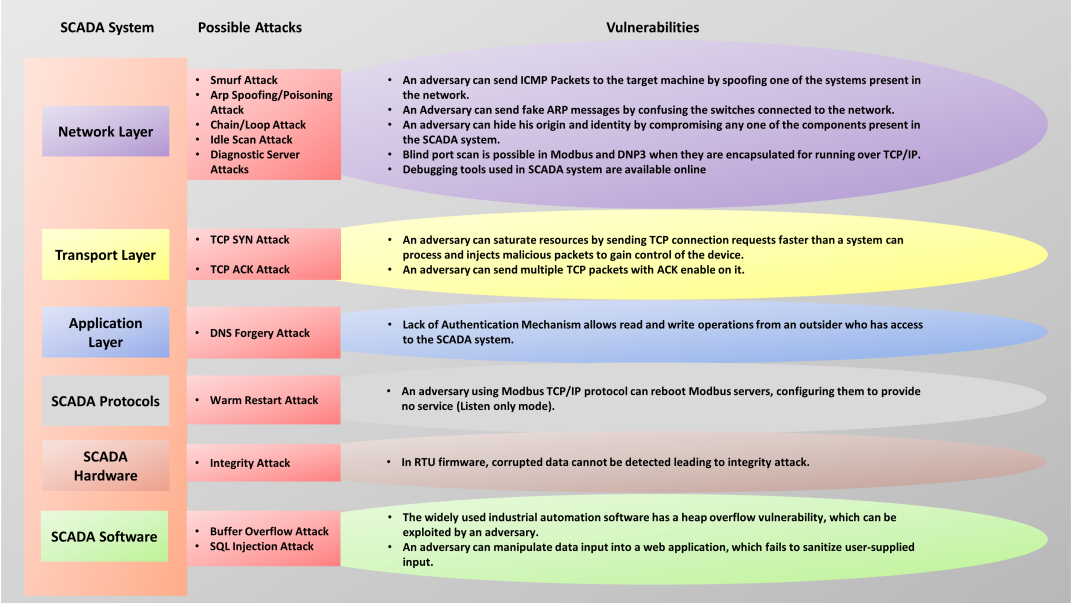

Figure 4.1: Attacks On SCADA Systems

4.2 Comparison of SDN Networks vs Traditional Networks for SCADA Systems

This chapter categorizes the security properties of SCADA systems based on the confidentiality, integrity, availability, authentication, authorization, accountability, scalability, visibility, flexibility, consistency, performance, robustness, and threat management system. These security criteria are compared with SDN switches and modern industrial control ethernet switches used in traditional networks as shown in Table 4.2, and from this comparison, one can easily derive the better technology to use for SCADA Systems.

Confidentiality refers to the prevention of disclosing information to unauthorized entities. Confidentiality in terms of SCADA systems is maintained by limiting the access to the information computed by the SCADA interfaces to the outside world. Only SCADA administrators are allowed to access the data. When the SCADA networks uses SDN, two standard methods, encryption and access control ensure confidentiality, [SAB14]. Modern based industrial standard ethernet switches also provide the features for secure access to networking devices while communicating with the SCADA components.

In Table 4.1, under the SCADA software section, there is a brief description of Buffer Overflow and SQL injection attacks on SCADA systems. The impact of confidentiality of SCADA systems with SDN networks and traditional networks is shown in Table 4.2. There are several mechanisms for SDN to encrypt network communication channels and to apply

Table 4.1: SCADA System Vulnerabilities And Attacks

SCADA System	Vulnerabilities
Network Layer	Debugging Tools used in SCADA systems are available online Debug service which runs on UDP port is enabled by default. RPC based protocol that uses UDP can be explored on the internet. Adversary need not have any coding knowledge to exploit the vulnerabilities. Blind port scan is possible in Modbus and DNP3 when they are encapsulated for running over TCP/IP. An adversary can send ICMP Packets to the target machine by spoofing one of the systems present in the network. An Adversary can send fake ARP messages by confusing the switches connected to the network. An adversary can hide his origin and identity by compromising any one of the components present in the SCADA system.
Transport Layer	An adversary can saturate resources by sending TCP connection requests faster than a system can process and injects malicious packets to gain control of the device. An adversary can send multiple TCP packets with ACK enableed on it.
Application Layer	Lack of authentication mechanism allows read and write operations from an outsider who has access to the SCADA system.
SCADA Software	The widely used industrial automation software has a heap overflow vulnerability, which can be exploited by an adversary. An adversary can manipulate data input into a web application, which fails to sanitize user-supplied input. Fixed memory allocation time requirement vulnerability in field devices can be exploited.
SCADA Hardware	In RTU firmware, corrupted data cannot be detected. An adversary may gain unauthorized remote access to devices and change their data set points. Another vulnerability is that attacker may change the operator display values to turn off the alarm after gaining unauthorized access. Attacks targeting at MAC level cause flooding attacks.
Modbus TCP/IP protocol	An adversary using Modbus TCP/IP protocol can reboot Modbus servers, configuring them to provide no service (Listen only mode). An adversary can send large-sized requests resulting in the crashing of servers. Modbus does not have any encryption or security measures, and an adversary can exploit weakness on the function code level. Unauthorized writes can be accomplished using 0x06 and 0x06. An adversary can gather information about remote devices like controller descriptions using the 0x11 function code. Using function code0x05 and 0x0F, an attacker can turn on and off the remote device, which results in sending the false results to SCADA HMI. Slave port can be restarted using the function code 0x08 with 0x01 and clear out the communication.
SCADA using DNP3 protocol	During reinitialization to default values, many devices will clear all queues, and this can be exploited by an attacker to cause a delay before they start accepting the requests again. Using function code 0x13, an attacker can manipulate the remote devices and suppress outputs or create false alarms. Function code 0x0D forces to reset and reconfigure DNP3 outstations.

Table 4.2: Security Properties Of SCADA Network Using SDN And Traditional Networks.
✓Indicates the feature is enabled or Available
—Indicates Not Applicable

Security Properties	Features	SDN	Traditional
Confidentiality	Data Encryption	✓	✓
	Access Control for SCADA Interfaces	✓	
Integrity	WRT Field Devices, RTU, and HMI		
Availability	SCADA Devices	✓	
	Network Devices	✓	✓
Authentication	SCADA Network Devices	✓	
	SDN Controllers		—
	Application		—
Authorization	Unauthorized Remote Access	✓	
Accountability	SCADA system Software users	✓	
Scalability	Adding new devices	✓	
Visibility	Centralized view	✓	
Flexibility	Maintenance and Reconfiguration	✓	
Consistency	Change in Configuration	✗	
Performance	Processing information and communication in SCADA Systems	✓	
Robustness	Single point of Failure	✓	✓
Threat Management System	Detection/Mitigation	✓	

access control mechanisms. In contrast, SCADA systems using traditional networks are limited, because the protocols designed on the traditional networks were not considered to be built with security mechanisms. Even if we upgrade the traditional networking protocols with a security feature, there is no guarantee that the SCADA system will support due to the internal dependency of SCADA components. Modern based industrial standard ethernet switches also provide the features for secure access to networking devices while communicating with the SCADA components.

Integrity refers to the correctness and trustworthiness of data. In SCADA systems, modifying the information while computing has to be restricted. Both SDN and traditional networks do not provide any countermeasures for this feature. In Table 4.1, there is a brief description of integrity attacks on SCADA systems. The attacks in SCADA systems show that the attackers first tries to modify the input or data that has been computed between the devices and sent across SCADA components to exploit the vulnerability present in them. If using SDN, integrity of flow rules and messages transferred between the layers has to be ensured [SAB14].

Availability is the property to access data, devices, and services whenever they are needed [SAB14]. In SCADA systems, access to SCADA and networking devices is critical because of the tightly coupled nature of the systems. The warm restart attack in Table 4.1, occurs when the adversary sends large size requests to crash the servers. SCADA systems using SDN provide

several solutions to mitigate the DoS attacks outlined by [KKS13]. In traditional networks, the availability is limited to networking devices, and to achieve this, network flows must be predefined and unchanged.

Authentication describes the property that entities are who claim to be [SAB14]. The authentication mechanism for SCADA networking devices is not provided; hence attacks like DNS forgery mentioned in Table 4.1 are possible. In SCADA systems, using a traditional network gives access to everyone, whereas in SDN, we can restrict it to the SCADA networking devices.

Authorization property describes the users or admins having authorized access to perform specific operations in SCADA systems. SDN networks providing authentication services, by default, ensures the authorization service to overcome the outsider attacks.

Accountability describes the actions performed by the authorized users. If there is any change in the expected behavior of the network, it lead to detect the possible attacks by constantly monitoring the log files. SCADA systems using SDN networks can keep track of every device present in the network by continuously analyzing the log files. In traditional networks, because of no proper authentication mechanisms, accountability property is not affected since the adversary modifies the Log files to hide there identity.

Scalability is a property where the networks or systems should be adaptable for changing needs. This property applies to the entire system, which consists of different components like SCADA software, SCADA hardware, networking devices, field devices, HMI, etc. Even though there are several publications concerning scalable SCADA systems, in this dissertation, we measure this property in terms of ease. Because of the programmability nature in SDN networks, restricting to flows, SDN networks are straightforward to configure and are more adaptable to meet the traffic demands. In traditional networks, this option is limited.

Visibility ensures the process of data collection, aggregation, and distribution across the SCADA system to monitor and interpret everything happening on a network. Visibility encompasses device discovery, network performance, security issues, and topology mapping. No matter how large and complicated the SCADA system gets, full visibility is provided by the SDN networks because of the centralized view approach described in Chapter 3. Flooding Attacks mentioned in Table 4.1 can be overcome if we use SDN networks for SCADA systems.

Consistency in SDN can be defined in three aspects:

- a consistent network view and control decisions made by the control plane.
- consistency and efficiency during data plane update.
- maintaining constant flow tables in the data plane. The fundamental causes of consistency issues, the potential inconsistency problems, and the state-of-art solutions are discussed in detail by [SWL19].

The SCADA system using traditional network has a considerable disadvantage concerning consistency since it lacks features like traffic programmability, network automation, and so on.

Performance can be measured in different aspects, and in this scenario, we consider information processing between SCADA devices and networking devices. In SDN, because of its design, i.e., the separation between the control plane and data plane, the SDN switch has to communicate with the controller whenever it receives the packet from an unknown source. This requires additional time to process incoming packets. This does not impact SCADA systems' performance because of their static configuration and pre-defined flow rules across the network.

Robustness of SCADA systems refers to the ability to withstand failures. In general, Information Technology (IT) systems using SDN networks are prone to a single point of failure attack because they rely on a centralized view with full network visibility. The controller maintains all the information about the entire network, which allows the adversary to influence any part of the system [CRS09b]. In traditional networks, adversaries can only tamper with the topology by convincing a set of switches or routers of a specific topology. Integrating SDN with SCADA systems, the controller plays a little or no part across the network. This gains the ability to withstand the single point of failure attack.

Threat Management System, SCADA system with SDN networks are supported by various attack detection and mitigation technologies, and some of the network threat defense mechanisms are mentioned in [SAB14]. Since this dissertation is concerned in presenting the security features, we are not describing the different attack detection and mitigation systems that are available both in SDN and Traditional networks.

4.3 Summary

This chapter presents a detail review of vulnerabilities present in SCADA system. Also, the comparison of SDN Networks vs Traditional Networks for SCADA Systems is analyzed. This analysis helps in two ways. First, it theoretically justifies that SDN network vulnerabilities do not significantly impact SCADA systems compared with traditional networks. Second, configuring the SDN networks by integrating the security property features improves the overall SCADA system security and performance.

Chapter 5

High-Level Policies

5.1 Policy-Based Network Management

In order to mitigate cyber attacks in Operational Technology Systems, we choose a policy-driven approach or Policy-Based Network Management (PBNM). Policy-based network management simplifies the administration of a large scale system. Traditional network management requires configuring many networking elements, and any problems present in the network should be solved manually, and policies are hardcoded. PBNM has been able to change the contents of the manual system dynamically without modifying the implementation of the managed system. The general PBNM is derived from the IETF policy framework and consists of four elements [BF14b],

- *Policy Management Tool (PMT):* This is a graphical tool to define which policies on the network are applied.
- *Policy Repository (PR):* Policies are stored in the form of a relational database or directory using Policy Repositories.
- *Policy Decision Point (PDP):* It parses the policy, checks the authorization and validity before communicating them to PEP.
- *Policy Enforcement Point (PEP):* Applies and executes different policies into the networking devices.

Advantages of a policy-driven approach over traditional management are,

- Policies are predefined and stored in the Policy Repository.
- PBNM approach implements automated analysis and verification of policies ensuring consistency in the majority of the policy languages.
- Policies can be changed dynamically at run time without changing the underlying implementation due to its abstract nature.

There are several policy languages like PDL, Ponder, Rei, XACML, EPAL, P3P/APPEL, ASL, and HERMES for specifying policies. This chapter describes and compares three policy languages Ponder, XACML, and HERMES by presenting a comparison table. The reason we choose these three languages out of others is because of their different natures. Ponder is based on object-oriented language; XACML, which stands for Extensible Access Control Markup Language, derived from XML; and HERMES is based on YAML.

Some of the terms used in when discussing policies are:

Policies A Policy is a rule that defines the choice in the behavior of the system [DDLS01a].

Policy Conflicts Conflicts may arise when the system administrator specifies the policies with errors and conflicting requirements [BF14b].

Subject A subject refers to users, principals, or automated manager components with management responsibility.

Target A subject accesses target objects (resources or service providers) by invoking methods visible on the target's interface.

Domain The Domain provides a means of grouping objects to which policies apply and can be used to partition the objects in an extensive system according to geographical boundaries, object type, responsibility, and authority or for convenience of managers.

Action Actions are activities which a subject can carry out [YLL02].

5.2 Ponder

Ponder is declarative, strongly-typed, and object-oriented, which makes the language flexible, extensible, and adaptable to a wide range of management requirements [DDLS01b].

Ponder is suitable for specifying many different types of policies [DDLS01a, BF14b, YLL02] including the following. Our ultimate project goal is to classify which of these are appropriate for ICS system, and to utilize them in our analysis. For now, we primarily focus on authorization poliices.

Authorization policies define the activities that a member of the subject domain can perform on the set of objects in the target domain are known as authorization policies.

Information Filtering Policies are used to transform the input or output information parameters.

Delegation Policies transfer the temporary access rights of a control system.

Refrain Policies define the actions that subjects must refrain from performing on target subjects even though they may be permitted to act.

Obligation Policies define what a manager must or must not do and hence guide the decision-making process under changing circumstances.

Privacy Policies define the privacy of an operation that subjects perform and confidentiality of information that subjects share. They can specify the privacy for the subject to create a new policy and decide the privacy of the information delivered on the internet.

Collaborative Policies are written using composite policy type and are used to group and inter-relate policies together to model the management structure within collaborative systems. There are four types of composite or collaborative policies.

Role provides a semantic grouping of policies, and can also specify the policies that apply to a mechanical component acting as a subject in the system.

Relationships are a group of policies that define the right and duties of roles towards each other.

Management Structure is a group of roles and relationship policies, which define the policy hierarchy of organization structure.

Group Policies are constructs that permit policymakers to group related policies together to instantiate all the policies at the same time, to share standard definitions between the policies, or to apply constraints in the set of policies.

There are few limitations of Ponder mentioned in the literature [HL12,CO08,PHS+08], some of them are:

- Lack of generality.
- Ponder only partially supports business-oriented specifications (Business process logic). Business-oriented specification come under Service Oriented Architecture (SOA)-specific requirements, in which the services are identified and implemented as primary business activities, and applications are formed based on the organization's core business processes.
- Ponder uses a low-level approach.
- There are no well-defined semantics for Ponder. A policy language is said to well-defined if that policy language written is independent of the particular information of that language.

5.3 Extensible Access Control Markup Language (XACML)

XACML [Oo19] is an OASIS standard that describes both a policy language and an access control decision request or response language. The Policy language is used to describe general access control requirements and has standard extension points for defining new functions, data types, combining logic, etc. The request-response language leads to farm a query to ask whether or not given action should be allowed and interpret the result. The response always includes

the answer about whether the request should be enabled using one of the four values: permit, deny, indeterminate, or not applicable.

In addition to providing request-response and policy languages, XACML also provides other pieces of information like a relationship, namely finding a policy that applies to a given request and evaluating the request against the policy to come up with yes or no answer. In an application deployed with XACML based access control, before a principal can perform an action on a particular resource, a Policy Enforcement Point (PEP) sends a request formulated in XACML to the Policy Decision Point (PDP) that stores principal specific XACML ACP rules. The PDP determines whether the request should be permitted or denied by evaluating the policies whose subject, action, and resource elements match the request. Finally, the PDP formulates its decision in the XACML response language and sends it to the PEP, which enforces the decision. [KEV05] Some of the terms to define with respect to XACML are [MBCS06], [MXT06], [Oo19],

- *Policy set:* Policy set is a container that can hold other policies or policy sets as well as references to policies found in remote locations.
- *Rule:* This is the fundamental element of the policy and identifies a complete and creates atomic authorization constraint that exists wrt policy.
- *Target:* A target element contains a set of constraints on the resources and actions of the subject. A target specifies what kind of requests a policy can be applied, and if the request does not satisfy the constraints in the target, then the whole policy element can be skipped without further examining its risks.
- *Condition:* A condition is associated with the rule which evaluates, and if evaluation returns true, then the rule can be permitted or denied, or else returns Not Applicable.
- *Attributes:* are the characteristics of the subject, resources, action, or environment.
- *Attribute Values:* Values associated with the attributes.
- *Policy-Combining algorithm:* The policy combining algorithm specifies the procedure by which the results of evaluating the component policies combine when evaluating the policy set, i.e., the Decision value placed in the response context by the PDP is the result of the evaluating the policy set, as defined by the policy-combining algorithm. A policy set may have combining parameters that affect the operation of the policy-combining algorithm.
- *Obligations:* There can be obligations in the policy set, and when PDP evaluates a policy set containing obligations, it returns certain of those obligations to the PEP in its response context.

XACML language is used in any environment without trying to provide access control for a particular environment or a specific kind of resource. In other terms, one policy can be written and used in different applications where policy management becomes much more

accessible. This language has been reviewed by a large community of users, and it will be easier to interoperate with other applications using the same standard language. In XACML, a policy can be written, which in turn refers to other policies kept in arbitrary locations known as a distributed policy. XACML can correctly combine the results from the different policies generated by different groups into one decision.

Limitations of XACML are [DKW07], [KEV05], [CO08], [SS10],

- XACML does not have the control to manage security in large distributed systems: Considering if a two policies can integrate, because the combination policies specified by XACML are not enough to integrate the policies specified by two parties. This is because XACML assumes all Points of Enforcement (POE's) are available to enforce any policy set by any party in the enterprise, and there are always enterprise administrators who can solve conflicts.
- XACML fails to incorporate built-in-trust and privacy-enhancing mechanisms: The possible attacks identified that could potentially breach the security of a system using XACML are:
  - *Unauthorized Disclosure*
  - *Message replay*
  - *Message insertion*
  - *Message Deletion and Message Modification.*
- XACML specifications lack the necessary syntax to specify exclusive access to resources, and the XACML policy framework does not enforce them. For example, services are built based on the content distribution network on a P2P- network in which Data Owner (DO) can store the data and Resource Owner(RO) can host the data. Therefore both DO and RO can specify there own XACML Policies.
- The problem of Conflicting polices faced by the administrator when XACML policies are specified manually or automatically using tools. XACML does not have a well-defined semantics to resolve these.

5.4 High Fidelity Policy: High-level, Easily Reconfigurable Machine Environment Specification (HERMES)

HERMES was developed by Jillepalli et al. [JCdL16] for implementing high-level browser security policy specifications, and it requires a language parser. HERMES is a High-level (Human-friendly) application policy specification language and supports, domain information like groups, sub-groups, users, roles, and devices specifying policy information as a parent, target, status, field, and rationale.

Table 5.1 provides a comparison between Ponder, XACML, and HERMES with there features and descriptions [And19], [HL12], [CO08], [PHS+08].

Table 5.1: Comparison Of Ponder, XACML, And HERMES

Features	Description	Ponder	XACML	HERMES
Language Specification				
XML	XML refers to whether the representation of the language is encoded in the XML format.	✗	✓	✗
Formalization	Formalization refers to whether a policy language can directly express in terms of policy rules in formulas of some logic (first-order logic) or translations to formal language	✓	✓	N/A
Policy Specification	Policy language is said to be complete if it has policy specification language, policy deployment model, policy analysis mechanisms, policy verification, and monitoring, conflict detection and a resolution mechanism to ensure the correctness and quality of the policies specified.	✓	✓	✓
Extensibility	A policy language should be easily extensible to support new types of policy expressions.	Partial	✓	✓
High-Level Specification	The high-level specification refers to the policy system which needs to stay at a reasonable level of abstraction so that precise policy requirements can be specified independently of the implementation details.	✓	✓	✓
Well-Defined Semantics	Well-Defined Semantics is based on rich formalisms such as description logic, or event calculus, which make it easier to perform automatic policy analysis than those that are not.			✓
Mutual Exclusiveness	Prohibiting to disclose valuable information to the same party is referred to as mutual exclusiveness.	✓		N/A
Supporting Types				

Obligation	Obligation refers to whether the policy language can trigger tasks that must perform when some Event occurs, and meets the related Condition.	✓	✓	✓
Domains	Domain refers to the support of the grouping of elements in a managed system so that policy can be systematically applied to them. The support for domain allows for the hierarchical grouping of objects.	✓	✓	✓
Meta-Policies	Meta policy allows rules and constraints to be placed on policies themselves thereby facilitating policy analysis and conflict resolution.	✓	✓	
Methods of Operation				
ECA	Event-Condition-Action paradigm refers to whether a language follows the ON (Event) IF (Condition) THEN (Action) to express its policies. If the language is not ECA, then it is based on the Condition–Action paradigm.	✓		✓
Index	Index refers to whether there exists a particular item for a policy engine to retrieve the required policies more efficiently.		✓	✓
RBAC	Role-Based Access Control used in policy-based access control management uses roles to bridge subjects and permissions. A subject can be assigned roles, and roles can be assigned permissions.	✓		✓

Dependability				
Minimal information disclosure	Minimal information disclosure in the context of policies refers to choosing privacy while disclosing the information.	✓		✓
Protect sensitive policies	These allow policies to be accessed by only network administrators.	✓	✓	✓
Possible Attacks found	XACML is prone to following attacks: Unauthorized Disclosure, Message replay, Message insertion, Message Deletion and Message Modification	N/A	✓	N/A
Structure				
Distributed Policy Enforcement	Distributed policy enforcement increases the flexibility and scalability of the framework	✓		✓
Change support	It refers to changes in the policies themselves based on updates in rules and regulations and also supports the change in business logic.			✓
Push Control	Push control anticipates possible deadlocks due to sensitive policies.	✓		N/A
Usage Control	Usage control refers to limiting the operations like specifying restrictions on the information provided to the target system.	✓		N/A
Hierarchy	Hierarchy refers to where parents consist of the collection of their children, and where prohibitions are inherited upward and downward, but permissions are inherited only downward.	✓		✓
Others				
Platform independent	Platform independent refers to support for executing policies in multiple systems.	✓	✓	✓

Policy Derivation	Policy Derivation is the ability to automatically derive individual policies from the policy of service orchestration and vice versa by saving the policy editors from codifying policies, which is time-consuming and manually error-prone.
Negotiation	Negotiation is considered to happen when, 1. Both peers are allowed to define a policy. 2. Both policies are taken into account when processing a request. 3. The evaluation of the request must be distributed.

5.5 Summary

This chapter presents a review of three different policy languages. First, it outlines the pro's and con's of ponder, XACML, and HERMES. Next, comparison is made between three to describe the differences between them and based on this comparison, i choose ponder policy language as a primary source to specify security policies.

Chapter 6

Automated Detection of Configured SDN Security Policies for ICS Networks

The material in this chapter was originally published as "Automated Detection of Configured SDN Security Policies for ICS Networks." [GRAF20], and has been slightly edited for content and presentation.

The best thing in SDN is the flexibility of managing High-level policies with ease and implementing the policies as logical flows. The policy-based network management approach simplifies the task of the network administrator. In SDN, it is further simplified in a way that the network operator can control and monitor network traffic through controller software. Because of this, the flow of traffic can be controlled and changed based on traffic demands. Even though the primary objective of this research is to retrieve the implemented logical flows from the SDN configuration files, the next aim is to convert the logical flows into policies using a policy specification language. This can help the network administrator to analyze the traffic flow of a network using the logic of the specification language. There are several policy languages like PDL, Ponder, Rei, XACML, EPAL, P3P/APPEL, ASL, and HERMES for specifying policies. For this research, we chose the Ponder policy language for two reasons. First, Ponder is declarative, strongly-typed, and object-oriented policy language. Since SDN switches are programmable, we can express the polices retrieved from the SDN switch using Ponder in an abstract and hierarchical model. Second, there is already an existing tool called PonderFlow [BF14a], which is derived from the Ponder policy language and is used to specify the policies for OpenFlow networks.

6.1 Analysis of Configuration Data File Generated By Flow Controller

As mentioned in Section 3.1.2 configuration data can be stored in a backup file. This file is essential because it saves all of the details of the current running configuration (Policies) of the existing network. This research focuses on analyzing this configuration backup file and retrieving the low-level information stored and mapping it to the network as configured in an ICS system. The goal is to eventually create a tool that can compare the system security policy to the system, as configured. The tool could be a formal analysis of the correspondence or used to query and test the configuration.

To generate a backup file, we established communication between client and server using the SEL-5056 Flow Controller and SEL SDN switch. We also defined the flow of communication between client and server using different routes or paths. Most OpenFlow switches allow this type of communication, and the configuration backup file generated is vendor-specific. The simple ICS server application we deployed is shown in Fig 6.1. We have two applications, App1 that receives and responds to request from a client application App2, running on separate machines. This high-level representation of autohorized communication can be refined by mapping the abstract machine name to the network address of the device hosting the application and the transport of the communication [VAFR19b]. For this example, we used TCP/IP communication; however, MODBUS, DNP3, NTP, GOOSE, etc. can be used. The communication between App1 and App2 can be defined as a Flow across the ICS network shown in Fig 3. This is one of the several flow rules we defined for the communication between App1 and App2. To further understand this analysis, it is essential to define the following terms:

Node A Node in the backup file is defined as hosts or end devices or SDN switches where the communication happens from one end to another.

Source Node This is defined as the node that initiates the communication and sends it to the other nodes.

Destination Node This is defined as the node that receives the information and does not transmit it further.

Ports In general, SDN switches contain physical network ports through which the flow of data is carried from the source node to the destination node.

Flow Rules or Paths These specify the authorized flow or path from source to the destination using ports.

The overall process for retrieving low-level information from the backup file is depicted in Fig 6.3. The process starts by taking the backup file as input and retrieves the total number of active flow rules. The active flow rules refer to the paths established for the communication

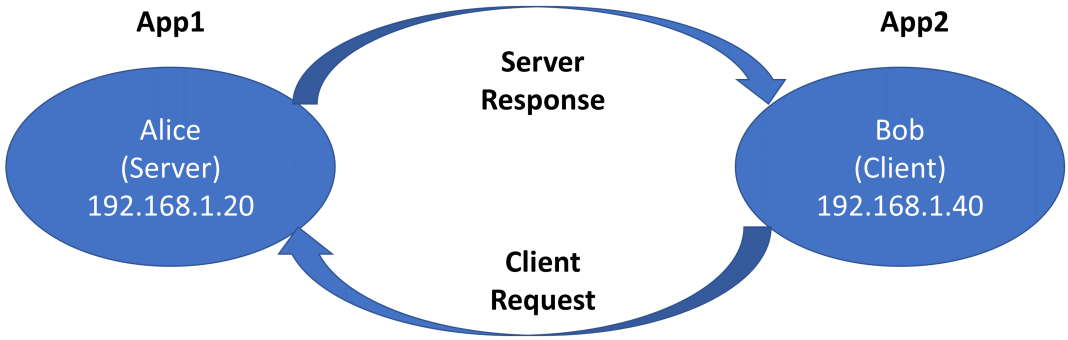

Figure 6.1: Application-level View Of Communication

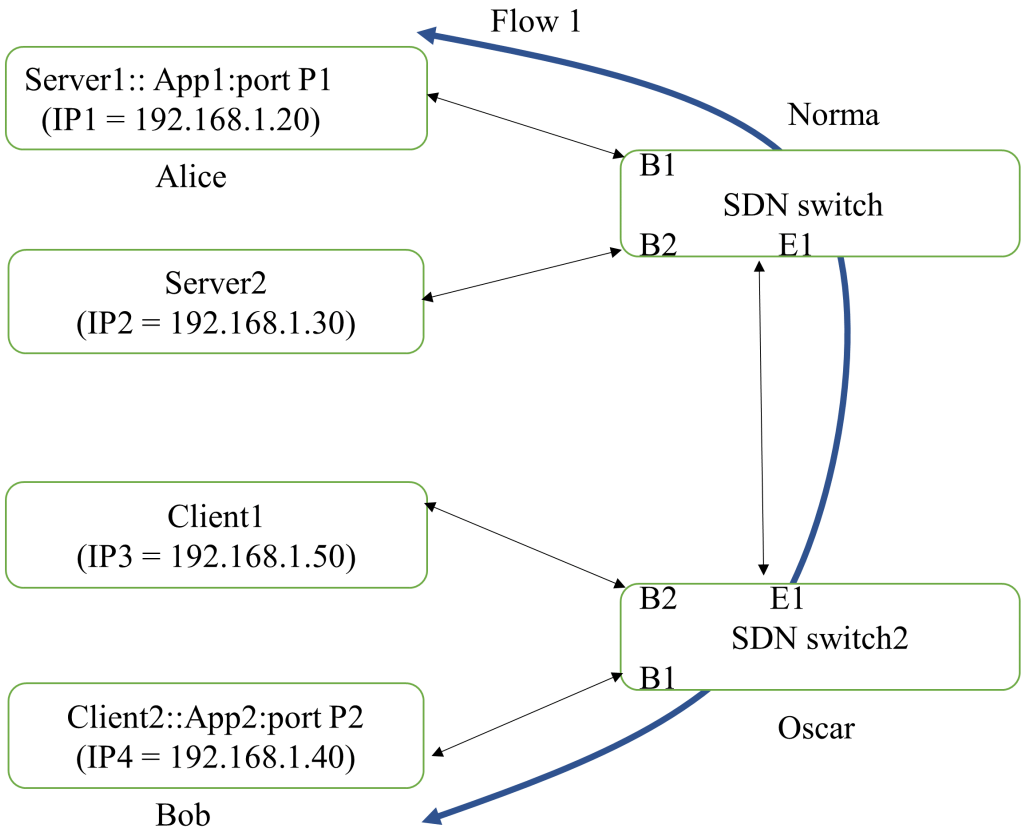

Figure 6.2: ICS Network-Level View Of Communication

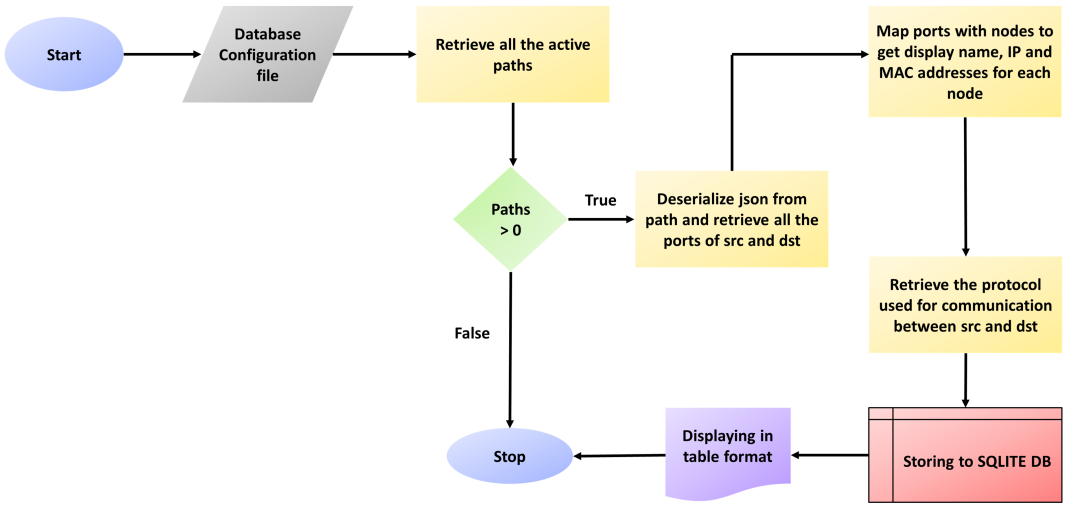

Figure 6.3: Process To Retrieve Low-Level Information From Backup File.

between App1 and App2. Next, for each flow rule, we retrieve the information about the source node, destination node, and the port information like through which port number the communication should occur. The next steps retrieve the high-level details like display name, IP address, MAC address, and communication protocol used for the source node and destination node. This information helps us analyze the low-level flow rules at a higher level with precise configuration details. The results are stored in an SQLite database and displayed in a tabular format.

The process described here was implemented in a prototype Java program, integrating with an SQLite Database. Firstly, we retrieved the information from the backup file by querying. The backup file, backup.db (and SQL Database), consists of 107 rows, where each row consists of precise details about the network. Then, results from this query are stored in JSON format. These are then deserialized into plain old java object (POJO) classes. From these POJO class objects, we retrieve the Source Port, Destination Port, Ingress Port, Egress Port for each flow. Using the Port values, we extract the nodes corresponding to the ports and additional node information, such as the display name of each node. With this information, we retrieved the IP address and MAC address by mapping the configNode_configtree_element and operationalnetworknode_operationaltree_element. The communication protocol used between the source and destination node is retrieved by mapping the flow_configtree_element and Logical connection id from the plannedpathsets_configtree_element. In short, the tool takes the backup file as input and dynamically generates the retrieved flow details in a tabular form, stores the information in SQLite DB for further analysis, and generates Ponder p2 policy files by converting flows into Ponder policies. The goal is to help the network/security administrator to monitor the policies and check if they are implemented as defined.

Table 6.1: Flow Table Generated From The Backup File.

Flow Num.	Source	Dest.	Protocol Used	Source MAC	Destination MAC	Source IP	Destination IP	Flow Type	Policy Type
1	Alice2	Bob	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
2	Alice2	Bob	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
3	Alice	Bob	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
4	Alice	Bob	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
5	Alice	Bob	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
6	Alice30	Bob50		080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
7	Bob50	Alice30		080027E460D6	080027887106	192.168.1.40	192.168.1.20	UniDirectional	Auth
8	Alice2	Bob	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.40	UniDirectional	Auth
9	Bob	Alice2	IPv4	080027E460D6	080027887106	192.168.1.40	192.168.1.20	UniDirectional	Auth
10	Alice2	Bob	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
11	Alice30	Bob50		080027887106	080027E460D6	192.168.1.20	192.168.1.40	UniDirectional	Auth
12	Alice30	Bob50	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.40	UniDirectional	Auth
13	Alice30	Bob50	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
14	Bob50	Alice30	IPv4	080027E460D6	080027887106	192.168.1.40	192.168.1.20	UniDirectional	Auth

Table 6.2: Additional Information For Active Polices Of Flow 4.

Policy / Flow No.	Ingress Port	Egress Port	Policy Type
4	Bob	OscarB1	Auth
4	OscarE1	NormaE1	Auth
4	NormalB1	Alice	Auth
4	Alice	NormaB1	Auth
4	NormalE1	OscarE1	Auth
4	OscarE1	Bob	Auth

6.1.1 Analysis of Flow Rules Retrieved From Configuration Backup File using tabular format

Table 6.1 depicts the output of the configuration backup file, which contains information about flow rules generated by the SDN controller for our example ICS network. This summarizes the overall configuration of the network at a high level by displaying information like Source Name, Destination Name, Source MAC, Destination MAC, Source IP, and Destination IP, Flow type, and Policy Type. For each flow rule, additional details like information about the ports and direction of travel are shown in Table 6.2.

The flow rule defined in Fig 6.2 can be mapped to flow 4 in Tables 6.1. The detailed information with respect to this flow rule is generated in Table 6.2. Communication between client and server occurs through the SDN switch named Norma and Oscar using ports B1 and E1 (from Table I). This flow rule is considered to be bidirectional since the communication can happen from both sides. The communication protocol used for this flow rule is Address Resolution Protocol (ARP). The next section describes the process for converting the flow rules retrieved from the backup file to Ponder authorization policies.

6.2 Converting Flows into Ponder policies

For this chapter, we use the Ponder Policy language, a declarative, strongly-typed, and object-oriented language, which makes it flexible, extensible, and adaptable to a wide range of management requirements [DDLS01a]. Ponder is the most expressive language for addressing user privacy requirements. It can implement all scenarios. However, Ponder does not support a full-fledged ontology for specifying resources. Conditions and actions used by Ponder are applied in an imperative language (e.g., C or Java). The policy is thus closely interwoven with the code controlling the resource [DHS07].

6.2.1 Generating Ponder Policies

For our purposes, flow rules retrieved from the configuration backup file are converted into Ponder authorization policies. This helps the network/security administrator foresee issues

```
inst ( auth+ | auth- ) policyName    "{"
   subject  [<type>]    domain-Scope-Expression ;
   target   [<type>]    domain-Scope-Expression ;
   action               action-list ;
   [ when               constraint-Expression ; ]    "}"
```

Figure 6.4: Syntax Of Ponder Authorization Policies.

```
domain := root/factory/domain.

root at: "Node" put: domain create.

addNode := [ :name :wards |
    Node := root/Node.
    Node := root/Node at: name put: domain create.

    // Create the switches and ports domain
    Node at: "switch" put: domain create.
    Node at: "port" put: domain create.

    wards do: [ :name |
        switch := Node/switch at: name put: domain create.
        switch at: "link" put: domain create.
        switch at: "port" put: domain create.
    ].
].
root at: "addNode" put: addNode.appended
```

Figure 6.5: Creating An SDN Domain To Store Authorization Policies.

```
103 root/tauthdom at: "h13" put:
104                 (newauthpol subject: root/Node/Bob/switch
105                             action: "passMessage"
106                             target: root/Node/OscarB1/switch
107                             focus: "st").
108 root/tauthdom/h13 active: true.
109
110 root/tauthdom at: "h14" put:
111                 (newauthpol subject: root/Node/OscarE1/switch
112                             action: "passMessage"
113                             target: root/Node/NOrmaE1/switch
114                             focus: "st").
115 root/tauthdom/h14 active: true.
116
117 root/tauthdom at: "h15" put:
118                 (newauthpol subject: root/Node/NormaB1/switch
119                             action: "passMessage"
120                             target: root/Node/Alice/switch
121                             focus: "st").
122 root/tauthdom/h15 active: true.
123
124 root/tauthdom at: "h16" put:
125                 (newauthpol subject: root/Node/Alice/switch
126                             action: "passMessage"
127                             target: root/Node/NormaB1/switch
128                             focus: "st").
129 root/tauthdom/h16 active: true.
130
131 root/tauthdom at: "h17" put:
132                 (newauthpol subject: root/Node/NOrmaE1/switch
133                             action: "passMessage"
134                             target: root/Node/OscarE1/switch
135                             focus: "st").
136 root/tauthdom/h17 active: true.
137
138 root/tauthdom at: "h18" put:
139                 (newauthpol subject: root/Node/OscarB1/switch
140                             action: "passMessage"
141                             target: root/Node/Bob/switch
142                             focus: "st").
143 root/tauthdom/h18 active: true.
```

Figure 6.6: Authorization Policy generated for a single flow from the flow table.

```
root/addNode value: "Alice" value: #( "Alice").

root/addNode value: "NormaB1" value: #( "NormaB1").

root/addNode value: "NormaE1" value: #( "NormaE1").

root/addNode value: "OscarE1" value: #( "OscarE1").

root/addNode value: "OscarB1" value: #( "OscarB1").

root/addNode value: "Bob" value: #( "Bob").
```

Figure 6.7: Dynamic Addition Of Nodes For Generating Authorization Policy.

by examining the traffic flows across the network. There are some simple tools that allow visualization of all authorized actions using Ponder and will be beenfiical in integrating into a complete system. The general syntax of a Ponder authorization policy is given in Figure 6.4. We have mapped SDN hosts to entities and SDN flows to actions. Any flow rule that is not specified as an action is assumed to be disallowed.

To generate any Ponder policy file, we first set up a domain using domain generation code (Figure 6.5). The domain element defines the path or the environment where the policies are stored. Here the node is considered the root element, and flows are established for each node through switches and ports. The SDN switches and end-devices, which act as nodes, are in a separate file (Figure 6.7). The number of nodes we retrieved from the file has been generated automatically by the tool to define the authorization policies. The policy generation code generated is stored in Nodeauth.p2 (Fig 6.6). The p2 files are executed using an ant build file.

Finally, we create an authorization policy for each flow, and the goal here is to define actions as flow rules for the current network. The authorization policies are set to active by default. We can make them inactive while the Ponder files are in process. The Sample authorization policy generated by the tool is depicted in Figures 6.5-6.6 for the ICS system show in Fig 6.2. The authorization policy in Fig 6.6 is generated by specifying the actions for the subject and target. The focus keyword specifies if the policy should be applicable for the subject (s) or target (t). In this case, we have enabled for both subject and target. This authorization policy allows the packets to flow between end devices using the SDN switch, as mentioned in Section 6.1.1. The authorization policy status can be set to true or false at run time. This serves the purpose of controlling the traffic flow based on the demand by the network administrator.

6.2.2 Using Ponder Policies

We can now query the Ponder authorization policies and graphically display what is allowed and not allowed. The next step of this work is to set up queries of the Ponder system based on a high-level security policy. The goal to find out how under what circumstances different nodes can communicate with each other, which protocols are allowed and how those maps to ports.

6.3 Summary

This chapter describes a process to retrieve the flow rules from the configuration backup file generated by an SDN flow controller. The overall process involves retrieving the policies which are stored as logical flows in the database, adding them to an SQliteDB for further analysis, and converting them to the Ponder policies for verification. The process is implemented as a tool and results are presented. In the next chapter, we present a process to compare the retrieved flow rules from the backup file with the policies defined by the organization to determine the consistency. This will help the network administrators to check for security misconfigurations across the network.

Chapter 7

Validating Security Policies in an SDN Switch Across ICS Networks

The material in this chapter was originally published as "Validating the Security Policies in SDN Switch Across ICS Networks." [GRAF21], and has been slightly edited for content and presentation.

Industrial Control Systems (ICS) are recommended to follow standard procedures, protocols, or guidelines developed by the government or organizations to protect against cyber-attacks. The National Institute of Standards and Technology (NIST) has provided guidance for securing ICS networks consisting of supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and programmable logic controllers (PLC) [SFS11]. This guide has clearly stated that system vulnerabilities in ICS can occur in hardware or software due to the misconfiguration and poor maintenance of devices deployed across the complex ICS networks.

In ICS networks, security misconfigurations are security policies that are incorrectly configured across networking devices. A security policy is a collection of rules that are permitted or prohibited in the system. In general, organizations develop policies to achieve their objectives and defend themselves from cyber threats. The network administrator enforces these policies through the organization's networking equipment, such as switches and routers. As a result, how well these policies are enforced within the networking devices determines the security of an ICS network.

Misconfigurations often contribute to the disclosure of confidential data, which aids adversaries in compromising the network. Misconfiguration problems can be mitigated in several ways, one of which is to adopt software-defined networking (SDN) technology.

The main aim of this work is to develop a tool that can take configuration files and high-level policy documents as input and perform a systematic review that tests the implemented security policy (ISP) for compliance with the defined security policy (CSP) as well as contradictions

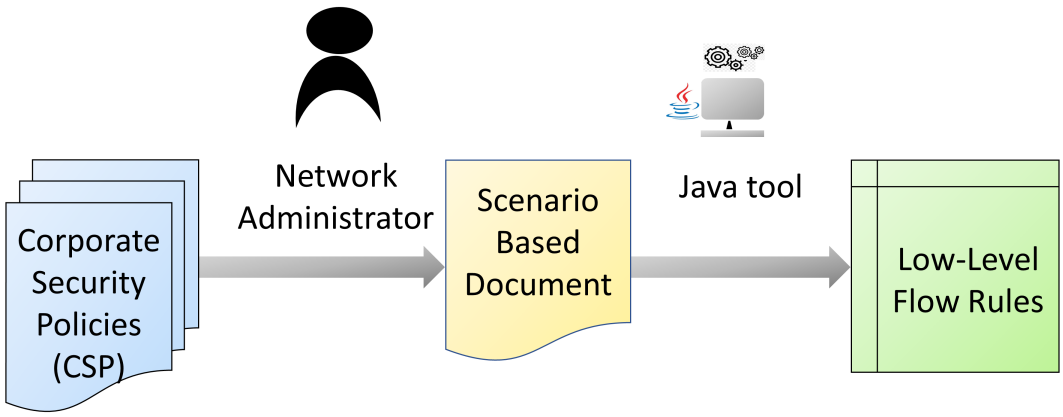

Figure 7.1: Conversion Of CSP to Low-Level Flow Rules

and inconsistencies. To achieve this, we must first extract the policies that have been enforced from the software-defined networking switch, as described in Chapter 6. In this chapter, our focus is to:

- Design a tool to convert the high-level policies defined at the organization level to the low-level flow rules using a scenario-based approach.
- Develop a formal model to map the ISP with the policies specified by the organization using mathematical and relational operations.
- And to analyze if ISP in an SDN switch is consistent with respect to CSP.

7.1 Conversion of CSP to Low-level FLow Rules

In Chapter 6, we developed a tool to automatically detect the security policy as implemented in the control rules of an SDN switch deployed in an industrial control system network. Using a scenario-based approach, this chapter expands the tool's capabilities to automatically convert the CSP to low-level flow rules. The scenario-based programming approach allows network operators to specify security policies using example behaviors [YLAL15]. The network operator sets the required security policy using scenarios consisting of packet traces and the actions taken for each packet. The tool takes these scenarios as input and generates the flow table. The process used for converting the CSP to low-level flow rules is shown in Fig 7.1.

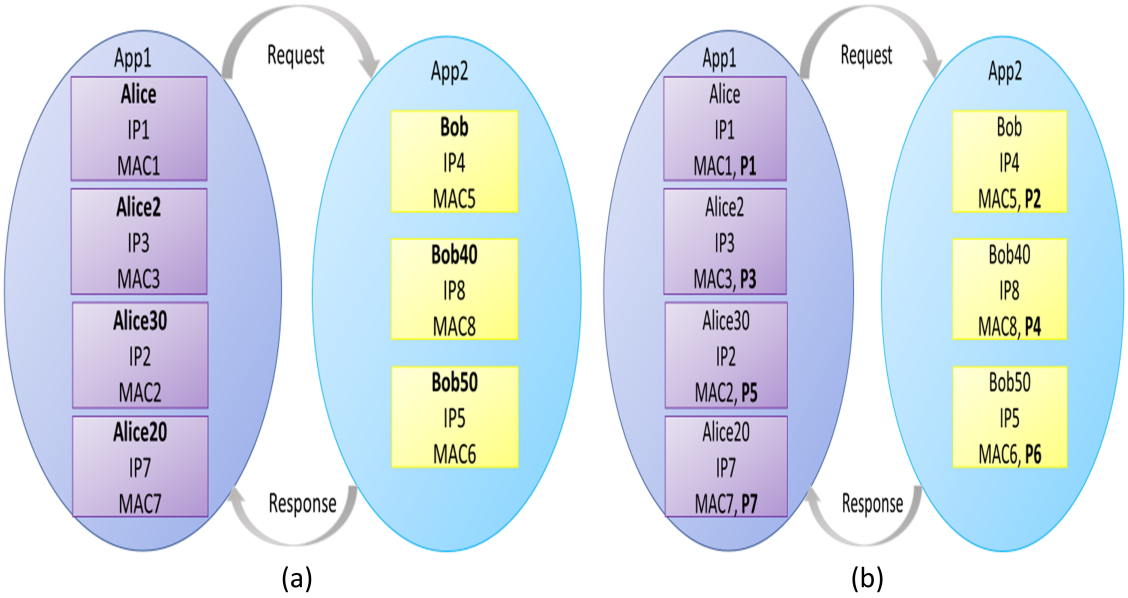

Figure 7.2: Configuring Devices And Ports For Domains App1, App2

First and foremost, the network administrator must describe the security policy's behavior in the document. Following that, the tool takes this document as input and translates the scenarios into flow rules. Finally, these flow rules are stored in a database and translated into authorization policies for further investigation. To illustrate the process, we consider the example of a sample CSP defined for a simple ICS server application. App1 receives and responds to request from a client application App2 (Fig 6.2) defines the rules at high-level as,

- Communication between App1 and App2 should consist only of the set of authorized messages between the applications.
- App1 is identified by a specific transport layer port (in this case, TCP port 1062).
- App1 and App2 run on authorized machines (as defined by their IP addresses and potentially MAC addresses).
- No other device on the network is authorized to participate in the Flow (no reading or writing of messages along with the flow).

Converting CSP into scenarios is done in three steps,

- **Establishing the domain**: We consider App1 and App2 as a set of domains, where each consists of multiple devices associated with unique IP and MAC addresses. This implies that no other device apart from App1 and App2 should communicate across the network, as shown in Fig 7.2(a).

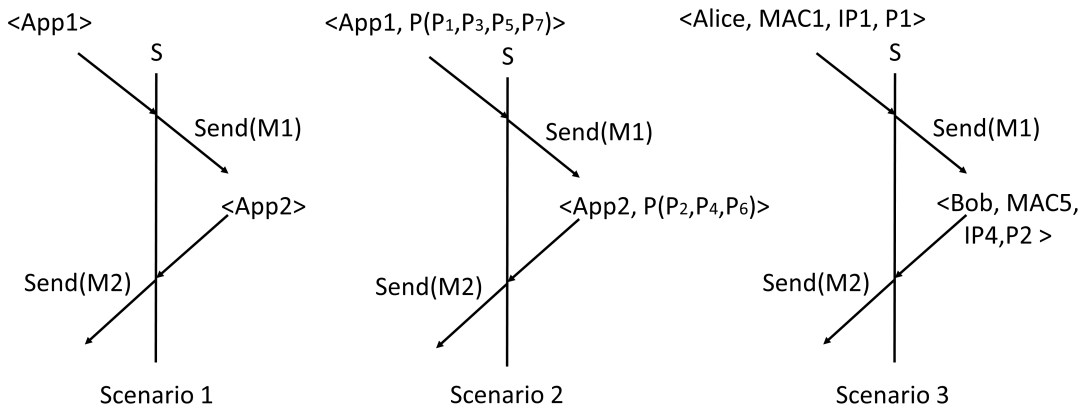

Figure 7.3: Step By Step Process For Coverting CSP To Scenarios

The scenario diagram for this is shown in Fig 7.3 (Scenario 1). We are defining a single tuple to denote the packet, with fields defining the domain. M1 and M2 specify the messages concerning the specified domains.

- **Defining Ports**: In this step, we define the ports for each device as shown in Fig 7.2(b). This implies that the communication between any device in with the domain must happen through the defined ports. The scenario diagram for this is shown in Fig 7.3 (Scenario2). We use two tuples to denote the packet for this step, with the fields defining the domain and a set of ports.This scenario ensures that both domains App1 and App2 have a set of ports devoted to them, with each port designated for a specific device. This prevents

Table 7.1: Flow Table Generated Based On The Scenarios File.

Flow Num	Source	Destination	Protocol Used	Source MAC	Destination MAC	Source IP	Policy Type
1	Alice	Bob	Web-based	MAC1	MAC5	IP1	Auth
2	Bob	Alice	Web-based	MAC5	MAC1	IP4	Auth
3	Bob50	Alice30	Web-based	MAC6	MAC2	IP5	Auth
4	Alice30	Bob50	Web-based	MAC2	MAC6	IP2	Auth
5	Alice2	Bob	Web-based	MAC3	MAC5	IP3	Auth
6	Bob	Alice2	Web-based	MAC5	MAC3	IP4	Auth
7	Alice20	Bob40	Web-based	MAC7	MAC8	IP7	Auth
8	Bob40	Alice20	Web-based	MAC8	MAC7	IP8	Auth

Alice, MAC1, IP1, P1, Bob -> Send(M1)
Bob, MAC5, IP4, P2, Alice -> Send(M2)
Bob50, MAC6, IP5, P4, Alice30 -> Send(M3)
Alice30, MAC2, IP2, P1, Bob50 -> Send(M4)
Alice2, MAC3, IP3, P1, Bob -> Send(M5)
Bob, MAC5, IP4, P2, Alice2 -> Send(M6)
Alice20, MAC7, IP7, P1, Bob40 -> Send(M7)
Bob40, MAC8, IP8, P2, Alice20 -> Send(M6)

Figure 7.4: Sample Scenario Document

overlapping and establishes a secure channel.

- **Defining Scenarios** : Based on the CSP provided, the network administrator can define a multiple scenarios for the devices associated within the domain. For example, consider that communication is allowed between the end-devices Alice and Bob. Alice is in the App1 domain, associated with IP address IP1 and Port P1. Bob is in the App2 domain associated with IP address IP5 and Port P2. The scenario for this is shown in Fig 7.3(Scenario3). We use 4 tuples to denote a packet for this step, with the fields becoming device name, MAC address, IP address, and port used for sending and receiving packets.

For this research, we have defined several scenarios as outlined in Fig 7.4. This scenario document is given as input to the tool which automatically converts the scenarios into low-level flow rules as show in Table 7.1. These flow rules are documented and stored in a database. The flow rules are then converted to ponder policies for further investigation. Our prior paper [RAF20] discusses the importance of expressing flow rules as ponder policies in detail. The conversion of CSP to low-level flow rules is necessary because it allows network administrators to map flow rules without having to implement them in SDN switches, allowing for a comparison with existing configuration and quick analysis.

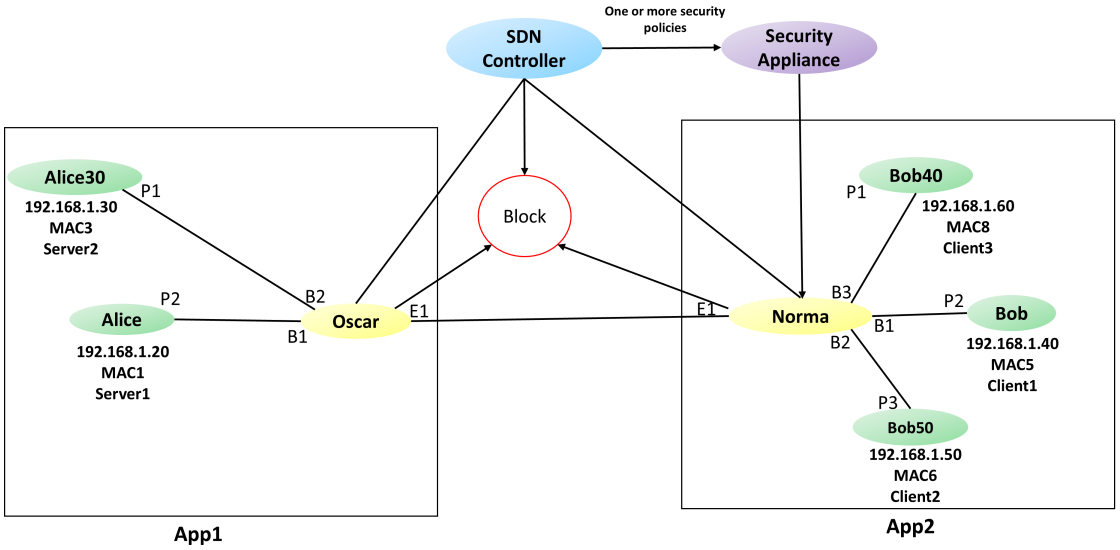

Figure 7.5: Low-Level Network Configuration Before Policy Implementation

7.2 Formal Approach for Deriving Consistency

Our main purpose in this research is to see if the ISP is consistent with the CSP. To do this, we transformed the high-level CSP into low-level flow rules in the previous section. In this section, we'll map the CSP and ISP to demonstrate that they have a valid relationship. This mapping is crucial because it allows network administrators to see if the high-level policies are being properly enforced on the low-level devices. The following definitions are provided to formally specify the mapping process,

Definition1: A policy is a function that maps the subject and target to a set of authorized operations. It is represented as P:S×T→P(R), where P refers to policies, S refers to the subject, T refers to the target, and P(R) is considered as the power set of operations. For each policy, there are multiple low-level flow rules. These rules are defined by the network administrator while implementing the policies on the networking devices. The low-level flow rule is represented as $P_L = < S_L, T_L, O, R >$, Where P_L refers to the low-level flow rules, S_L refers to the subject, T_L refers to the target, O refers to operations performed by the subject on target (Request, Send, and Receive messages). R refers to rules to be applied for each operation.

Based on Definition1, we present the following definitions for Corporate Security Policies (CSP) and Implemented Security Policies (ISP).

Definition2: We define CSP as a finite non-empty set consisting of elements CSP = $\{c_1$, c_2, c_3 ,c_4 .... c_m $\}$, where for all c in CSP represents the low-level implementation of CSP such

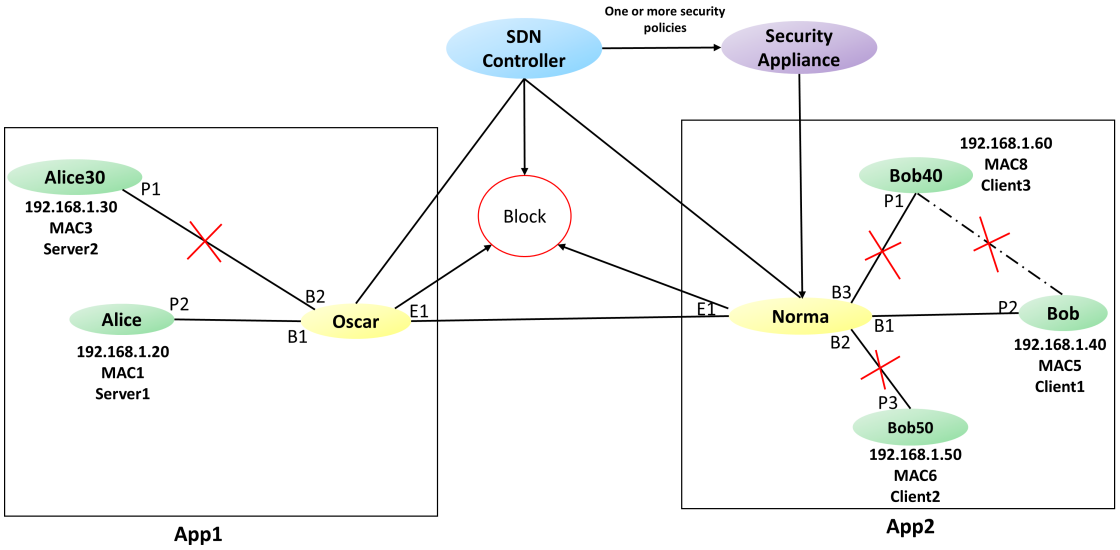

Figure 7.6: Low-Level Network Configuration After Implementation Of CSP

that c = $< S_c, T_c, R_q >$. S_c is subject, T_c is Target, and R_q is requirements that both subject and target must follow to communicate. To illustrate this with an example, let us consider a sample high-level CSP as follows:

'Communication between two devices is achieved using specific protocols and designated ports'.

The low-level implementation of the above policy is represented as c1 = {Device1, Device2, (Web-based Protocols and Specified Ports)}. This means that Device1 and Device2 can communicate with each other using web-based protocols through a set of ports.

Definition3: We define ISP as a finite non-empty set consisting of elements ISP = {i_1, i_2, i_3 ,i_4 .... i_n}, where for all i in ISP represents the low-level implementation of ISP such that i = $< S_i, T_i, O, R >$. S_i is subject, T_i is Target, O is operations performed by subject on target, and R is the rules that must be defined for each operation.

For the above sample CSP, the low-level ISP is represented as $i_1 = < Bob, Alice, \{S_m, R_m\}, \{ARP, B_1\} >$. Bob and Alice indicate device names, S_m and R_m refer to the specific operations that the source Bob can perform on Target. In this case, Bob is allowed to send or receive messages from target Alice using ARP as a communication protocol through port B_1.

From these definitions, we derive mapping relation as,

Definition4: The relation M: ISP→CSP is consistent mapping if m = (i,c) ∈ M ⇒ $i_{<S_i,T_i,O,R>}$ satisfies $c_{<S_c,T_c,R_q>}$ and ∀ i ∈ ISP ∃ c ∈ CSP such that (i,c) ∈ M. And, ∀ c ∈ CSP ∃ i ∈ ISP, (i,c) ∈ M.

This definition states that the requirements specified by the organization must stricty be implemented across networking devices. If this condition is satisfied then we are considering as both the CSP and ISP are consistent. In other words, the rules and operations implemented on networking devices must be derived from the requirements specifed in CSP. Although, this might not be possible in complex environments, but one should strictly implement the policies to avoid misconfigurations and enhance security. If the requirement specified by the organization cannot be enforced or applied, then there must be an alternative while configuring appliances. Also, there are chances of network being partially consistent, i.e., some policies are skipped during implementation for various reasons like lack of resources or dependency issues, but there are not any additional policies implemented that does not exists with CSP. These will not effect the system in short term but cannot be termed as secure.

To understand how the mapping process works at network level, let us consider Flow Num 1 from Table 7.1. This flow rule clearly states that Alice can communicate with Bob using web-based protocols. Figure 7.5 depicts the low-level network configuration before policy implementation. Figure 7.6 depicts the low-level network configuration after the policy implementation. From the Fig 7.6, we can see that communication at low-level is allowed only from Alice to Bob through the switches Oscar and Norma. The SDN switches block other devices trying to communicate. So we conclude that the flow rule implemented on the network complies with the flow rule specified in Table 7.1. Hence we consider this as a valid mapping relation. In general, a network is said to be consistent if all flow rules are applied in accordance with the organization's policies. In our context, we define consistency as behavioral equality among two states, in this case it is between ISP and CSP. This equality between any two states or two processes can be determined using the bisimilarity property. Bisimilarity is a behavioral equivalency notion that is frequently used on processes. It is also known as robust equality because it solely considers the interaction that the processes may or may not have [San11].

Let us define a sample set ISP consisting of elements from i_1 to i_7 and set CSP consisting of elements from c_1 to c_5. According to Definition4, M is a mapping relation consisting of matching elements from both ISP and CSP. i.e., M = {(i_1, c_1), (i_2, c_1), (i_3, c_2), (i_4, c_3), (i_5, c_4), (i_6, c_5), (i_7, c_5)}

The M is considered as bisimilar, iff

- For all i ∈ ISP, there must be atleast one c such that c ∈ CSP,
- The converse, on the rules from ISP.

This implies that for all implemented flow rules i should be matched to the specified policy c and if a pair of flow rules (i, c) is true for the above conditions, then it satisfies the bisimilarity conditions on single flow rule as a single flow rule is considered here. As a result, if all of the flow rules implemented at low-level networking devices fulfill the organization's standards, the mapping relation M is bisimilar. Inconsistency and security misconfiguration arise if any improper flow rules are implemented on these devices that were not defined by the organization

```
ISP Flow rule 1 Matches with CSP Flow rule of 5
ISP Flow rule 2 Matches with CSP Flow rule of 5
ISP Flow rule 3 Matches with CSP Flow rule of 1
ISP Flow rule 4 Matches with CSP Flow rule of 1
ISP Flow rule 5 Matches with CSP Flow rule of 1
ISP Flow rule 6 Matches with CSP Flow rule of 6
ISP Flow rule 7 Matches with CSP Flow rule of 6
ISP Flow rule 8 Matches with CSP Flow rule of 3
ISP Flow rule 9 Matches with CSP Flow rule of 4
ISP Flow rule 10 Matches with CSP Flow rule of 4
ISP Flow rule 11 Matches with CSP Flow rule of 3
ISP Flow rule 12 Does not Matches with any CSP Flow rule
ISP Flow rule 13 Does not Matches with any CSP Flow rule
ISP Flow rule 14 Does not Matches with any CSP Flow rule
```

Figure 7.7: Verbose Output Generated Based On ISP And CSP

or if the specified flow rule is not implemented.

7.3 Analysis of Configuration File Generated by Flow Controller to Determine Consistency

A flow controller is a software that constitutes a control plane and configures all the network appliances on how to forward data packets. It also keeps track of all the rules for the network components and controls them in an internal control flow table. This will usually require the use of many network switches. Each SDN switch receives the proper subset of rules from the controller. To authenticate this switch programming, a secure network will use digital signatures and other encryption technology.

Configuration data is typically saved in a backup file generated by the SDN flow controller. This file is necessary because it maintains all of the details of the existing network's current operational configuration (Policies). This study examines this configuration backup file, retrieving the low-level information contained inside it, and mapping it to the network as configured in an ICS system to see how consistent it is. The main objective is to extend the tool's capabilities so that it can compare the system security policy to the system as it is now configured. The tool could be used to perform a formal analysis of the correspondence or to query and test the setup.

We used the SEL-5056 Flow Controller and SEL SDN switch to establish connection between the client and the server in order to create a backup file. We also defined several routes or

CSP

ISP	1	2	3	4	5	6	7	8
1	0	0	0	0	1	0	0	0
2	0	0	0	0	1	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0
6	0	0	0	0	0	1	0	0
7	0	0	0	0	0	1	0	0
8	0	0	1	0	0	0	0	0
9	0	0	0	1	0	0	0	0
10	0	0	0	1	0	0	0	0
11	0	0	1	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0

Figure 7.8: Matrix Output Generated Based On ISP And CSP

paths to describe the communication flow between the client and the server. This sort of communication is supported by most OpenFlow switches, and the configuration backup file generated is vendor-specific. The simple ICS server application we configured is shown in Fig 6.2. We have two applications, App1 that receives and responds to request from a client application App2, running on separate machines. This high-level representation of authorized communication can be refined by mapping the abstract machine name to the network address of the device hosting the application and the transport of the communication [RAF20]. For this example, we used TCP/IP communication; however, MODBUS, DNP3, NTP, GOOSE, etc. can be used. The communication between App1 and App2 can be defined as a Flow across the ICS network. This is one of the several flow rules we defined for the communication between App1 and App2.

Table 6.1 depicts the output of the configuration file, which contains information about flow rules generated by the SDN controller for our example ICS network. This summarizes the overall configuration of the network at a high level by displaying information like Source Name, Destination Name, Source MAC, Destination MAC, Source IP, and Destination IP, Flow type, and Policy Type. Next using Table 7.1 as reference, the tool maps the flow rules from Table 6.1. This mapping was created using a Java prototype that integrated with a SQLite database. To start, we queried the database to collect the flow rules for both ISP and CSP. In ISP flow table, the rows and columns specify the actual details applied across networking devices, while in CSP flow table, the low-level version of high-level rules is presented. The query's results are then saved separately as Java objects. Following that, Java Data Structures

are used to map these entities. The tool produced verbose output for mapping between ISP and CSP (outlined in Fig 7.7), as well as matrix output for determining how many policies from ISP are in line with CSP (presented in Fig 7.8). We can see from the output that there are additional policies enforced in addition to the ones listed and some policies specified are not enforced, implying that the configuration file is inconsistent. The network administrator can use this tool to identify any additional policies that have been applied and dynamically remove them due to the programmability of SDN switches and safeguard the network. Now, the network/security administrator will be able to monitor the policies and check if they are implemented as defined.

7.4 Summary

This chapter describes the consistency check process by analyzing the backup configuration file and high-level policy document. As a result, the overall process for evaluating the configuration file generated by SDN controller to ensure that it complies with the organization's policies is presented. First, the tool takes the scenario document as input and translates the high-level policies into low-level flow rules using scenario based approach is outlined. Second, a formal approach for ISP and CSP using the bisimilarity property to determine the consistency between ISP and CSP is achieved. This work forms the basis for network administrators to monitor for security misconfigurations across networking devices.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The development of applications with security features in SDN has focused on either security as an afterthought or security as a decentralized requirement of a large part of the control plane. Successful development of highly secure critical infrastructure or ICS systems must occur from the beginning of the design by configuring the policies based on organization needs. However, standard ICS system design requires that all security enforcing parts of the system enforce the full security policies. This requires a tremendous amount of time and effort to verify that the policy is correct and usually requires analysis of the system as a whole.

This dissertation applies the well-known concept of bisimilarity to verifying policies deployed on SDN switches. The work presented here addresses the validation of security policies specified by the organization to policies deployed in the network by the network administrator. In this work, I have presented the list of attacks on different sub-systems of OT and analyzed the cyber incidents reported in the CVE database to determine the trends. This list will inspire and help the research community to develop countermeasures to the weaknesses present in these systems. I have also outlined the vulnerabilities of SCADA Systems and SDN networks. This vulnerability assessment helps in two ways. First, it theoretically justifies that SDN network vulnerabilities do not significantly impact the performance of SCADA systems compared with traditional networks. Second, configuring the SDN networks by integrating the security property features improves the overall SCADA system security.

To check if the policies deployed or implemented on an SDN switch are consistent with the high-level policies defined by the organization, there is a need to develop formal models. To achieve this, I have described a process and developed a tool to retrieve the flow rules from the configuration backup file generated by an SDN flow controller. The overall process involves retrieving the policies stored as logical flows in the database, adding them to an SQliteDB for further analysis, and converting them to the Ponder policies for verification.

Next, I have automated the consistency check process by analyzing the backup configuration file and high-level policy document. As a result, the overall process for evaluating the

configuration file generated by the SDN controller to ensure that it complies with the organization's policies is described. First, the tool takes the controller's configuration file as input, extracts the low-level flow rules, and stores them in a database. Next, it inputs the scenario document and translates the high-level policies into low-level flow rules. Lastly, it maps ISP and CSP using the bisimilarity property to decide whether they are consistent or not. Analysis of multiple backup configuration files were tested and the details are given in Appendix A.

The work presented in this dissertation leads to a practical security tool or framework, which will benefit network administrators by providing policies deployment guidance and reducing overall system misconfigurations.

8.2 Future Work

The previous section has summarized the results of this dissertation. However, as in every research effort, the results have indicated some areas where further work could lead to other important conclusions. For example, I have looked at a single controller's backup configuration file in this work. However, more than one controller will be in complex ICS system design. It is essential to develop a formal methodology for evaluating the consistency of numerous backup configuration files generated by different controllers. This will aid network administrators in detecting and mitigating misconfigurations throughout the complex ICS network at a high-level.

Using NIST principles, it is also essential to create a testbed for each OT subsystem that integrates with software-defined networking technologies. And try to exploit all the vulnerabilities and attacks on SCADA Systems by incorporating security features outlined in this research work to analyze them in terms of performance, safety, security, and reliability properties.

The research problem in this dissertation work focused on is specific to SEL SDN switches and flow controllers. In the market, there are various vendor-specific SDN switches. Comparison of different vendor-specific SDN switches can be a research topic in and of itself, leading to an examination of the various security features used as well as an analysis of the backup configuration file which determines the security of ICS systems.

In conclusion, securing critical infrastructure and OT systems is a field loaded with intriguing challenges. We will continue our efforts to secure the OT and its sub-systems.

Bibliography

[Abb16] Ali Abbasi. Ghost in the plc: Designing an undetectable programmable logic controller rootkit via pin control attack. *University of Twente Research Information*, pages 1–35, 2016.

[ACP+17] Adel Alshamrani, Ankur Chowdhary, Sandeep Pisharody, Duo Lu, and Dijiang Huang. A defense system for defeating ddos attacks in sdn based networks. In *Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access*, pages 83–92, 2017.

[AGH14] Carlos Aguayo Gonzalez and Alan Hinton. Detecting malicious software execution in programmable logic controllers using power fingerprinting. Technical report, 2014.

[Alo12] Aloul,Fadi and Al-Ali, A R and Al-Dalky, Rami and Al-Mardini, Mamoun and El-Hajj, Wassim. International journal of smart grid and clean energy smart grid security: Threats, vulnerabilities and solutions. Technical report, 2012.

[And19] Anne Anderson. A comparison of two privacy policy languages: Epal and xacml. 2019.

[APMR18] Sajjad Amini, Fabio Pasqualetti, and Hamed Mohsenian-Rad. Dynamic load altering attacks against power system stability: Attack models and protection schemes. *IEEE Transactions on Smart Grid*, 9(4):2862–2872, jul 2018.

[BA01] M. Bond and R. Anderson. Api-level attacks on embedded systems. *Computer*, 34(10):67–75, 2001.

[BF14a] Bruno Lopes Alcantara Batista and Marcial P Fernandez. Ponderflow: A policy specification language for openflow networks. In *Proc. ICN*, page 215, 2014.

[BF14b] Bruno Lopes Alcantara Batista and Marcial Porto Fernandez. Ponderflow: A new policy specification language to sdn openflow-based networks. *International Journal on Advances in Networks and Services*, 7(3 & 4), 2014.

[BIBC+15a] Yaniv Ben-Itzhak, Katherine Barabash, Rami Cohen, Anna Levin, and Eran Raichstein. Enforsdn: Network policies enforcement with sdn. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, may 2015.

[BIBC+15b] Yaniv Ben-Itzhak, Katherine Barabash, Rami Cohen, Anna Levin, and Eran Raichstein. Enforsdn: Network policies enforcement with sdn. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 80–88. IEEE, 2015.

[BK16] Nikos Bizanis and Fernando A Kuipers. Sdn and virtualization solutions for the internet of things: A survey. *IEEE Access*, 4:5591–5606, 2016.

[Boo] *Cyber Security for Industrial Automation and Control Systems (IACS) EDITION 2 Open Government status Open.*

[CCS13] Ang Cui, Michael Costello, and Salvatore Stolfo. When firmware modifications attack: A case study of embedded exploitation. 2013.

[CDF+06] Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner, and Alfonso Valdes. Using model-based intrusion detection for scada networks. pages 209–237, 2006.

[CDF+07] Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner, and Alfonso Valdes. Using model-based intrusion detection for scada networks. In *Proceedings of the SCADA security scientific symposium*, volume 46, pages 1–12. Citeseer, 2007.

[CJ03] Mihai Christodorescu and Somesh Jha. Static analysis of executables to detect malicious patterns. In *12th USENIX Security Symposium (USENIX Security 03)*, 2003.

[CO08] Juri Luca De Coi and Daniel Olmedilla. A review of trust management, security and privacy policy languages. In *SECRYPT*, 2008.

[CQSM16] Tiago Cruz, Rui Queiroz, Paulo Simões, and Edmundo Monteiro. Security implications of scada ics virtualization: survey and future trends. In *ECCWS2016-Proceedings fo the 15th European Conference on Cyber Warfare and Security*, page 74. Academic Conferences and publishing limited, 2016.

[CRS09a] Alvaro A. Cardenas, Tanya Roosta, and Shankar Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in scada systems. *Ad Hoc Networks*, 7(8):1434–1447, November 2009.

[CRS09b] Alvaro A Cardenas, Tanya Roosta, and Shankar Sastry. Rethinking security properties, threat models, and the design space in sensor networks: A case study in scada systems. *Ad Hoc Networks*, 7(8):1434–1447, 2009.

[CSNN20] Samuel L Clements, Cameron T Smith, William K Nickless, and Charles Nickerson. Evaluating software defined networking solutions to reduce the digital attack surface of nuclear security systems. Technical report, Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2020.

[CVE] CVE-2014-0160. Available from MITRE.".

[Cze16] Robert Czechowski. Security policy and good practice for implementation of smart grid solutions. *Przeglad Elektrotechniczny*, 92(3):177–181, 2016.

[DDLS01a] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The Ponder policy specification language. In *Proc. of the International Workshop on Policies for Distributed Systems and Networks*, POLICY '01, page 18–38, 2001.

[DDLS01b] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. *Lecture Notes in Computer Science*, pages 18–38, 2001.

[DHS07] C. Duma, A. Herzog, and N. Shahmehri. Privacy in the semantic web: What policy languages have to offer. In *Eighth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'07)*, pages 109–118, 2007.

[DKW07] Vijayant Dhankhar, Saket Kaushik, and Duminda Wijesekera. Xacml policies for exclusive resource usage. Technical report, 2007.

[DMSK16] Neelam Dayal, Prasenjit Maity, Shashank Srivastava, and Rahamatullah Khondoker. Research trends in security and ddos in sdn. *Security and Communication Networks*, 9(18):6386–6411, 2016.

[DSMF08] Giovanna Dondossola, Judit Szanto, Marcelo Masera, and Igor Nai Fovino. Effects of intentional threats to power substation control systems. *International Journal of Critical Infrastructures*, 4(1/2):129, 2008.

[EI16] E-ISAC11. Analysis of the cyber attack on the ukrainian power grid. 2016.

[FCMT09] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. An experimental investigation of malware attacks on scada systems. *International Journal of Critical Infrastructure Protection*, 2(4):139–145, 2009.

[FP18] Andry Putra Fajar and Tito Waluyo Purboyo. A survey paper of distributed denial-of-service attack in software defined networking (sdn). *International Journal of Applied Engineering Research*, 13(1):476–482, 2018.

[FPC09] Aurelien Francillon, Daniele Perito, and Claude Castelluccia. Defending embedded systems against control flow attacks. 2009.

[FWD19] Barbara Filkins, Doug Wylie, and AJ Dely. Sans 2019 state of ot/ics cybersecurity survey. *SANS Technology Institute*, 2019.

[GKR+08] Annarita Giani, Gabor Karsai, Tanya Roosta, Aakash Shah, Bruno Sinopoli, and Jon Wiley. A testbed for secure and robust scada systems. *ACM SIGBED Review*, 5(2):1–4, jul 2008.

[GRAF20] Sandeep Gogineni Ravindrababu and Jim Alves-Foss. Automated detection of configured sdn security policies for ics networks. In *Sixth Annual Industrial Control System Security (ICSS) Workshop*, pages 31–38, 2020.

[GRAF21] Sandeep Gogineni Ravindrababu and Jim Alves-Foss. Validating the security policies in sdn switch across ics networks. In *Seventh Annual Industrial Control System Security (ICSS) Workshop*, 2021.

[GRAF22] Sandeep Gogineni Ravindrababu and Jim Alves-Foss. Analysis of vulnerability trends and attacks in ot systems. In *Proceedings of Seventh International Congress on Information and Communication Technology*. Springer, 2022.

[Gue10] M. D. Guel. A short primer for developing security policies. 2010.

[HCPS08] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, dec 2008.

[HF+18] Kevin E Hemsley, E Fisher, et al. History of industrial control system cyber incidents. Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States), 2018.

[HG11] Adam Hahn and Manimaran Govindarasu. Cyber attack exposure evaluation framework for the smart grid. *IEEE Transactions on Smart Grid*, 2(4):835–843, dec 2011.

[HL12] Weili Han and Chang Lei. Survey paper: A survey on policy languages in network and security management. *Comput. Netw.*, 56(1):477–489, jan 2012.

[HLC17] Jin-bing Hou, Tong Li, and Cheng Chang. Research for vulnerability detection of embedded system firmware. *Procedia Computer Science*, 107:814–818, 2017.

[HNS17] Mark Hadley, David Nicol, and Rhett Smith. Software-defined networking redefines performance for ethernet control systems. In *Power and Energy Automation Conference*, 2017.

[JCdL16] Ananth A. Jillepalli and Daniel Conte de Leon. An architecture for a policy-oriented web browser management system: Hifipol: Browser. Technical report, 2016.

[JNY11a] Dong Jin, David M. Nicol, and Guanhua Yan. An event buffer flooding attack in dnp3 controlled scada systems. 2011.

[JNY11b] Dong Jin, David M Nicol, and Guanhua Yan. An event buffer flooding attack in dnp3 controlled scada systems. In *Proceedings of the 2011 Winter Simulation Conference (WSC)*, pages 2614–2626. IEEE, 2011.

[KAL+18] Mutaz HH Khairi, Sharifah HS Ariffin, NM Abdul Latiff, AS Abdullah, and MK Hassan. A review of anomaly detection techniques and distributed denial of service (ddos) on software defined network (sdn). *Engineering, Technology & Applied Science Research*, 8(2):2724–2730, 2018.

[KEV05] Yared Keleta, J Eloff, and H Venter. Proposing a secure xacml architecture ensuring privacy and trust. Technical report, 2005.

[KKS13] Rowan Klöti, Vasileios Kotronis, and Paul Smith. Openflow: A security analysis. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2013.

[KLKP09a] Dong-Joo Kang, Jong-Joo Lee, Seog-Joo Kim, and Jong-Hyuk Park. Analysis on cyber threats to scada systems. *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, October 2009.

[KLKP09b] Dong-Joo Kang, Jong-Joo Lee, Seog-Joo Kim, and Jong-Hyuk Park. Analysis on cyber threats to scada systems. In *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, pages 1–4. IEEE, 2009.

[KLMR04] Paul Kocher, Ruby Lee, Gary McGraw, and Anand Raghunathan. Security as a new dimension in embedded system design. In *Proceedings of the 41st annual Design Automation Conference*, pages 753–760, 2004.

[LNR11] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 14(1):1–33, may 2011.

[LR14] Adrian Lara and Byrav Ramamurthy. Opensec: A framework for implementing security policies using openflow. *2014 IEEE Global Communications Conference*, dec 2014.

[LZL+17] Gaoqi Liang, Junhua Zhao, Fengji Luo, Steven R. Weller, and Zhao Yang Dong. A review of false data injection attacks against modern power systems. *IEEE Transactions on Smart Grid*, 8(4):1630–1638, jul 2017.

[MANC+11a] Malaz Mallouhi, Youssif Al-Nashif, Don Cox, Tejaswini Chadaga, and Salim Hariri. A testbed for analyzing security of scada control systems (tasscs). *ISGT 2011*, jan 2011.

[MANC+11b] Malaz Mallouhi, Youssif Al-Nashif, Don Cox, Tejaswini Chadaga, and Salim Hariri. A testbed for analyzing security of scada control systems (tasscs). In *ISGT 2011*, pages 1–7. IEEE, 2011.

[MBCS06] P. Mazzoleni, E. Bertino, B. Crispo, and S. Sivasubramanian. Xacml policy integration algorithms. *Proceedings of the eleventh ACM symposium on Access control models and technologies - SACMAT '06*, 2006.

[MC16] Robert Mitchell and Ing-Ray Chen. Modeling and analysis of attacks and counter defense mechanisms for cyber physical systems. *IEEE Transactions on Reliability*, pages 350–358, 2016.

[MCHL14] Kebina Manandhar, Xiaojun Cao, Fei Hu, and Yao Liu. Detection of faults and attacks including false data injection attack in smart grid using kalman filter. *IEEE Transactions on Control of Network Systems*, 1(4):370–379, dec 2014.

[McL11] Stephen McLaughlin. On dynamic malware payloads aimed at programmable logic controllers. *Proceedings of the 6th USENIX conference on Hot topics in security*, pages 10–10, 2011.

[ME10] Anthony R. Metke and Randy Ekl. Security technology for smart grid networks. *IEEE Transactions on Smart Grid*, 1(1):99–107, june 2010.

[MG13] Thomas H Morris and Wei Gao. Industrial control system cyber attacks. In *1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013) 1*, pages 22–29, 2013.

[MGS17] Amir Modarresi, Siddharth Gangadhar, and James PG Sterbenz. A framework for improving network resilience using sdn and fog nodes. In *2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pages 1–7. IEEE, 2017.

[MGSFW14a] Cristian Cleder Machado, Lisandro Zambenedetti Granville, Alberto Schaeffer-Filho, and Juliano Araujo Wickboldt. Towards sla policy refinement for qos management in software-defined networking. *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, may 2014.

[MGSFW14b] Cristian Cleder Machado, Lisandro Zambenedetti Granville, Alberto Schaeffer-Filho, and Juliano Araujo Wickboldt. Towards sla policy refinement for qos management in software-defined networking. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 397–404. IEEE, 2014.

[Mil11] Charlie Miller. Battery firmware hacking. *Black Hat USA*, pages 3–4, 2011.

[MRLG11] Amir-Hamed Mohsenian-Rad and Alberto Leon-Garcia. Distributed internet-based load altering attacks against smart power grids. *IEEE Transactions on Smart Grid*, 2(4):667–674, dec 2011.

[MXT06] Evan Martin, Tao Xie, and Yu Ting. Defining and measuring policy coverage in testing access control policies. *Information and Communications Security*, pages 139–158, 2006.

[NFCMT09] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. An experimental investigation of malware attacks on scada systems. *International Journal of Critical Infrastructure Protection*, 2(4):139–145, dec 2009.

[NFGMS11] Igor Nai Fovino, Luca Guidi, Marcelo Masera, and Alberto Stefanini. Cyber security assessment of a power plant. *Electric Power Systems Research*, 81(2):518–526, feb 2011.

[NS17] Gorby Kabasele Ndonda and Ramin Sadre. A low-delay sdn-based countermeasure to eavesdropping attacks in industrial control systems. In *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7. IEEE, 2017.

[Oo19] Oasis-open.org. A brief introduction to xacml. 2019.

[PHS+08] T. Phan, J. Han, J. Schneider, T. Ebringer, and T. Rogers. A survey of policy-based management approaches for service oriented systems. In *19th Australian Conference on Software Engineering (aswec 2008)*, pages 392–401, 2008.

[PMB15] Dorottya Papp, Zhendong Ma, and Levente Buttyan. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 145–152. ieee, 2015.

[RAF20] Sandeep Gogineni Ravindrababu and Jim Alves-Foss. Automated detection of configured sdn security policies for ics networks. In *Sixth Annual Industrial Control System Security (ICSS) Workshop*, pages 31–38, 2020.

[RLSS10] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems. *Proceedings of the 47th Design Automation Conference on - DAC '10*, 2010.

[RS16] V KRISHNA Reddy and D Sreenivasulu. Software-defined networking with ddos attacks in cloud computing. *International Journal of innovative Technologies (IJIT)*, 4(19):3779–3783, 2016.

[SAB14] Lisa Schehlmann, Sebastian Abt, and Harald Baier. Blessing or curse? revisiting security aspects of software-defined networking. In *10th International Conference*

on Network and Service Management (CNSM) and Workshop, pages 382–387. IEEE, 2014.

[San11] Davide Sangiorgi. *Introduction to bisimulation and coinduction.* Cambridge University Press, 2011.

[Sch19] Schweitzer Engineering Lab. *SEL-2740S Software-Defined Network (SDN) Switch SEL-5056 SDN Flow Controller*, 2019.

[SFS11] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[SRO13] G Sandaruwan, P. H., P. Ranaweera, S., and Vladimir Oleshchuk, A. Plc security and critical infrastructure protection. *2013 IEEE 8th International Conference on Industrial and Informat*, 2013.

[SS10] Kamalbir Singh and Sarbjeet Singh. Design and evaluation of xacml conflict policies detection mechanism. *Core.ac.uk*, 2010.

[SWL19] Jinshu Su, Wen Wang, and Cong Liu. A survey of control consistency in software-defined networking. *CCF Transactions on Networking*, 2(3-4):137–152, 2019.

[TLM08a] Chee-Wooi Ten, Chen-Ching Liu, and Govindarasu Manimaran. Vulnerability assessment of cybersecurity for scada systems. *IEEE Transactions on Power Systems*, 23(4):1836–1846, November 2008.

[TLM08b] Chee-Wooi Ten, Chen-Ching Liu, and Govindarasu Manimaran. Vulnerability assessment of cybersecurity for scada systems. *IEEE Transactions on Power Systems*, 23(4):1836–1846, 2008.

[TPE15] Pascal Thubert, Maria Rita Palattella, and Thomas Engel. 6tisch centralized scheduling: When sdn meet iot. In *2015 IEEE conference on standards for communications and networking (CSCN)*, pages 42–47. IEEE, 2015.

[VAFR19a] Varsha Venugopal, Jim Alves-Foss, and Sandeep Gogineni Ravindrababu. Use of an sdn switch in support of nist ics security recommendations and least privilege networking. In *Proceedings of the Fifth Annual Industrial Control System Security (ICSS) Workshop*, pages 11–20, 2019.

[VAFR19b] Varsha Venugopal, Jim Alves-Foss, and Sandeep Gogineni Ravindrababu. Use of an sdn switch in support of nist ics security recommendations and least privilege networking. In *Proceedings of the Fifth Annual Industrial Control System Security (ICSS) Workshop*, pages 11–20, 2019.

[YLA+18] Yifei Yuan, Dong Lin, Siri Anil, Harsh Verma, Anirudh Chelluri, Rajeev Alur, and Boon Thau Loo. Netegg: A scenario-based programming toolkit for sdn policies. *IEEE/ACM Transactions on Networking*, 26(5):2104–2117, 2018.

[YLAL15] Yifei Yuan, Dong Lin, Rajeev Alur, and Boon Thau Loo. Scenario-based programming for sdn policies. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pages 1–13, 2015.

[YLL02] Chunsheng Yang, Fuhua Oscar Lin, and Hong Lin. Policy-based privacy and security management for collaborative e-education systems. In *Proceedings of the 5th IASTED International Multi-Conference Computers and Advanced Technology in Education (CATE 2002)*, volume 1, pages 501–501. Citeseer, 2002.

[YLR11] Yanling Yuan, Zuyi Li, and Kui Ren. Modeling load redistribution attacks in power systems. *IEEE Transactions on Smart Grid*, 2(2):382–390, june 2011.

[YLS+11] Y. Yang, Tim Littler, S. Sezer, K. McLaughlin, and H. F. Wang. Impact of cyber-security issues on smart grid. *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, dec 2011.

[YQST12] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on cyber security for smart grid communications. *IEEE Communications Surveys and Tutorials*, 14(4):998–1010, 2012.

[ZJS11] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on scada systems. In *2011 International conference on internet of things and 4th international conference on cyber, physical and social computing*, pages 380–388. IEEE, 2011.

Appendix A

Case Studies

This section provides the analysis and validation of different backup configuration files generated by the SDN controller.

A.1 Case 1

Using the process described in Chapter 6, flow rules based on the backup configuartion file generated by the SDN controller are depicted in Table A.1. For the validation purpose, i have used mapping definition defined in the Chapter 7, section 7.2. The flow table outlined in Table 7.1 represents the low-level flow rules defined at the corporate level and is used for this comparison. The output for the validation of flow rules or policies between ISP and CSP is shown in Fig A.1 (Verbose output) and Fig A.2 (Matrix output). We can see from the output that there are some policies enforced are not actually specified at high-level (i.e., flow rules in ISP 1-5, 7, 8, 11, 12). Also the policies listed by the organization are not enforced (i.e., flow rules from CSP 1, 2, 5, 6, 7, 8), implying that the configuration file is inconsistent.

A.2 Case 2

Table A.2 depicts the output of another backup configuartion file generated by the SDN controller. For validation, same procedure is used. The output for the validation of flow rules between ISP and CSP is shown in Fig A.3 (Verbose output) and Fig A.4 (Matrix output). We can see from the output that there are some policies listed by the organization are not enforced (i.e., flow rules from CSP 2, 3, 6, 7, 8), but there are no additional policies implemented and we can consider this configuartion file as partial consistent.

Table A.1: Flow Table For 2nd Backup Configuration File Generated By SDN Controller.

Flow Num.	Source	Dest.	Protocol Used	Source MAC	Destination MAC	Source IP	Destination IP	Flow Type	Policy Type
1	Alice2	Bob40	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.4	Bi-Directional	Auth
2	Alice2	Bob40	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.4	Bi-Directional	Auth
3	Alice	Bob40	ARP	080027887106	080027E460D6	192.168.1.10	192.168.1.4	Bi-Directional	Auth
4	Alice	Bob40		080027887106	080027E460D6	192.168.1.10	192.168.1.4	Bi-Directional	Auth
5	Alice	Bob40		080027887106	080027E460D6	192.168.1.10	192.168.1.4	Bi-Directional	Auth
6	Bob50	Alice30	IPv4	080027E460D6	080027887106	192.168.1.50	192.168.1.30	Bi-Directional	Auth
7	Alice20	Bob40	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.4	Bi-Directional	Auth
8	Alice20	Bob40	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.4	Bi-Directional	Auth
9	Alice30	Bob50	ARP	080027887106	080027E460D6	192.168.1.30	192.168.1.50	Bi-Directional	Auth
10	Alice30	Bob50	IPV4	080027887106	080027E460D6	192.168.1.30	192.168.1.50	Bi-Directional	Auth
11	Alice20	Bob50	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.50	Bi-Directional	Auth
12	Alice20	Bob50	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.50	Bi-Directional	Auth

Table A.2: Flow Table For 3rd Backup Configuration File Generated By SDN Controller.

Flow Num.	Source	Dest.	Protocol Used	Source MAC	Destination MAC	Source IP	Destination IP	Flow Type	Policy Type
1	Alice2	Bob	IPv4	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
2	Alice2	Bob	ARP	080027887106	080027E460D6	192.168.1.20	192.168.1.40	Bi-Directional	Auth
3	Alice	Bob	ARP	080027887106	080027E460D6	192.168.1.10	192.168.1.40	Bi-Directional	Auth
4	Alice	Bob	ARP	080027887106	080027E460D6	192.168.1.10	192.168.1.40	Bi-Directional	Auth
5	Alice	Bob	IPv4	080027887106	080027E460D6	192.168.1.10	192.168.1.40	Bi-Directional	Auth
6	Alice2	Bob	IPv4	080027E460D6	080027887106	192.168.1.20	192.168.1.40	Bi-Directional	Auth
7	Alice2	Bob	ARP	080027E460D6	080027887106	192.168.1.20	192.168.1.40	Bi-Directional	Auth
8	Alice30	Bob50	IPV4	080027887106	080027E460D6	192.168.1.30	192.168.1.50	Bi-Directional	Auth
9	Alice30	Bob50	ARP	080027887106	080027E460D6	192.168.1.30	192.168.1.50	Bi-Directional	Auth

```
ISP Flow rule 1 Does not Matches with any CSP Flow rule
ISP Flow rule 2 Does not Matches with any CSP Flow rule
ISP Flow rule 3 Does not Matches with any CSP Flow rule
ISP Flow rule 4 Does not Matches with any CSP Flow rule
ISP Flow rule 5 Does not Matches with any CSP Flow rule
ISP Flow rule 6 Matches with CSP Flow rule of 3
ISP Flow rule 7 Does not Matches with any CSP Flow rule
ISP Flow rule 8 Does not Matches with any CSP Flow rule
ISP Flow rule 9 Matches with CSP Flow rule of 4
ISP Flow rule 10 Matches with CSP Flow rule of 4
ISP Flow rule 11 Does not Matches with any CSP Flow rule
ISP Flow rule 12 Does not Matches with any CSP Flow rule
```

Figure A.1: Verbose Output Generated Based On ISP And CSP

CSP

ISP		1	2	3	4	5	6	7	8
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0
	6	0	0	1	0	0	0	0	0
	7	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	0
	9	0	0	0	1	0	0	0	0
	10	0	0	0	1	0	0	0	0
	11	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0

Figure A.2: Matrix Output Generated Based On ISP And CSP

```
ISP Flow rule 1 Matches with CSP Flow rule of 5
ISP Flow rule 2 Matches with CSP Flow rule of 5
ISP Flow rule 3 Matches with CSP Flow rule of 1
ISP Flow rule 4 Matches with CSP Flow rule of 1
ISP Flow rule 5 Matches with CSP Flow rule of 1
ISP Flow rule 6 Matches with CSP Flow rule of 5
ISP Flow rule 7 Matches with CSP Flow rule of 5
ISP Flow rule 8 Matches with CSP Flow rule of 4
ISP Flow rule 9 Matches with CSP Flow rule of 4
```

Figure A.3: Verbose Output Generated Based On ISP And CSP

CSP

ISP	1	2	3	4	5	6	7	8
1	0	0	0	0	1	0	0	0
2	0	0	0	0	1	0	0	0
3	1	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0
7	0	0	0	0	1	0	0	0
8	0	0	0	1	0	0	0	0
9	0	0	0	1	0	0	0	0

Figure A.4: Matrix Output Generated Based On ISP And CSP