

© 2022 Emily Belovich

AUTOMATIC VERIFICATION OF CYBERSECURITY BEST PRACTICES
FOR OPERATIONAL TECHNOLOGY NETWORKS

BY

EMILY BELOVICH

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Adviser:

Professor David M. Nicol

Abstract

Operational technology (OT) refers to the hardware and software systems used to monitor and control physical processes. OT systems are used extensively in industries such as electric, water and wastewater, oil and natural gas, transportation, and manufacturing. Reliable OT systems are necessary to maintain access to critical infrastructures. As such, it is essential they are protected from cyberattacks.

In the past, there was clear separation between an organization's OT systems and information technology (IT) systems. Today, however, OT networks are increasingly connected to IT networks. This introduces new cybersecurity concerns.

In this thesis, we outline cybersecurity best practices for OT networks. These best practices are recommended by respected organizations, including the U.S. Department of Homeland Security (DHS), the International Society of Automation (ISA), and the National Institute of Standards and Technology (NIST). Then, we discuss the development of a tool that can be used to analyze a given OT network and automatically verify (or disprove) the correct implementation of cybersecurity best practices. Finally, we present a case study where we use data from an example system to illustrate the implementation of our verification tool.

To my family, whose support made this possible.

Table of contents

List of Abbreviations	vi
Chapter 1 Introduction	1
Chapter 2 Background	3
Chapter 3 Motivation	5
Chapter 4 Cybersecurity Best Practices for OT Networks	7
4.1 DHS Seven Steps to Effectively Defend Industrial Control Systems	8
4.2 DHS Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies	9
4.3 ISA/IEC 62443	11
4.4 NIST Guidelines on Firewalls and Firewall Policy	15
4.5 NIST Guide to Industrial Control Systems (ICS) Security . . .	17
4.6 Selection of Best Practices	19
Chapter 5 Information Required for Verification	21
5.1 Physical and Logical System Topology	21
5.2 Port Scan Data	23
5.3 Switch Configuration Data	25
5.4 Information Documented by the System Administrator	26
Chapter 6 Best Practices Library	33
6.1 Best Practices for System Segmentation	34
6.2 Best Practices for Firewall Policies	37
6.3 Best Practices for Information Flows	43
6.4 Best Practices for Ports	44
6.5 Best Practices for Network Switches	45
Chapter 7 Verification Tool	49
7.1 Policy Creation	49
7.2 Input	51
7.3 Evaluation	51

7.4	Output	52
Chapter 8	Case Study	54
8.1	System Information	54
8.2	System Verification	60
Chapter 9	Conclusion	67
References		68

List of Abbreviations

2FA	Two-Factor Authentication
ACL	Access Control List
CIDR	Classless Inter-Domain Routing
CLI	Command Line Interface
DCS	Distributed Control System
DHS	(U.S.) Department of Homeland Security
DMZ	Demilitarized Zone
DTP	Dynamic Trunking Protocol
FBI	Federal Bureau of Investigation
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IACS	Industrial Automation and Control Systems
ICMP	Internet Control Message Protocol
ICS	Industrial Control System
ICS-CERT	Industrial Control Systems Cyber Emergency Response Team
IEC	International Electrotechnical Commission
IoT	Internet of Things

IP	Internet Protocol
IPv4	Internet Protocol version 4
ISA	International Society of Automation
IT	Information Technology
LAN	Local Area Network
MAC	Media Access Control
NCCIC	National Cybersecurity and Communications Integration Center
NIST	National Institute of Standards and Technology
Nmap	Network Mapper
NSA	National Security Agency
OT	Operational Technology
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SL	Security Level
SSH	Secure Shell Protocol
STO	Security Through Obscurity
SUC	System Under Consideration
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VPN	Virtual Private Network

Chapter 1

Introduction

In May 2021, a cyberattack resulted in the shutdown of one of the largest fuel pipelines in the United States. The cyberattack hit Colonial Pipeline, which carries gasoline, diesel, and jet fuel from Texas to New York [1]. The shutdown lasted for several days, leading to gas panic-buying, shortages, and price spikes in several states. The FBI later confirmed DarkSide, a group based in Russia, was responsible for the attack [2].

Cyberattacks on critical infrastructures, like the oil and gas industry, are particularly devastating because they pose a direct threat to human life. DarkSide is financially motivated and does not aim to cause real-world disruption [3], however, other groups might not be so “nice.” Events like the Colonial Pipeline attack shine a glaring light on the fragility of our critical infrastructures.

Cyberattacks on critical infrastructures are becoming more common due to the convergence of operational technology (OT) and information technology (IT). In June 2021, the CEO of Colonial Pipeline told Congress that the attackers were able to access the system by stealing a single password and using a legacy VPN that did not have multifactor authentication enabled [4]. DarkSide gained access to Colonial’s IT network, corrupted files on an internal drive, and left a ransom note. Upon discovering the ransom note, the company made the decision to shut down the entire pipeline. This action was an attempt to contain the attack and prevent the malware from spreading from the internal IT network to the OT network that controlled the pipeline operations

[5]. Colonial worried that the hardware and software systems responsible for controlling the pipeline operations (OT) were not securely isolated from the company's data systems (IT).

It is now more important than ever to protect our critical infrastructures by verifying the security of the OT systems that control them. This thesis makes the following contributions: (1) We discuss well-established cybersecurity best practices for OT networks, (2) We develop a tool that can be used to automatically verify the correct implementation of cybersecurity best practices for a given OT network, and (3) We present a case study that illustrates how our verification tool works using data from an example OT system.

This thesis is organized as follows:

- Chapter 2 provides background information on OT networks, including how they compare to IT networks.
- Chapter 3 motivates the need for our verification tool by discussing the importance of secure OT networks.
- Chapter 4 summarizes five documents that outline cybersecurity best practices for OT networks.
- Chapter 5 describes the relevant information about an OT network that may be used as input to the verification tool.
- Chapter 6 discusses the library of best practices. The library includes each best practice the verification tool has the ability to evaluate.
- Chapter 7 explains how the verification tool works.
- Chapter 8 presents a case study. In this case study we use data from an example system to illustrate the implementation of the verification tool.
- Chapter 9 concludes the thesis.

Chapter 2

Background

Operational technology (OT) is a classification of hardware and software systems that are used to monitor and control physical processes. As an example, consider a water treatment plant. At a high level, a water treatment plant has sensors that measure temperatures, pressures, and the quantity of various chemicals in the water. The sensors send this data to programmable logic controllers (PLCs) that analyze the data and output control commands. In response to these control commands, actuators may open or close valves to adjust the flow rate of various chemicals, a warning light may flash in the facility, or an alarm may sound to indicate an unsafe operational state. A water treatment plant would also have a human-machine interface, or HMI, that allows human operators to interact with the water treatment process. In addition, there would likely be a data historian, which is a computer used to record data about the plant's operation. All these devices, and the software required to operate them, are part of an OT network.

OT includes industrial control systems (ICSs). ICS is a general term used to describe the devices, systems, and networks used to control or automate industrial processes. Supervisory control and data acquisition (SCADA) systems and distributed control systems (DCS) are two common types of ICSs [6]. SCADA systems allow organizations to control industrial processes, collect and process data in real-time, and interact directly with system devices, like sensors and motors, through an HMI [7]. Related to this, the term DCS refers

to control achieved by intelligence distributed throughout the process to be controlled, rather than by a single, centrally located unit [8].

ICSs are typically used in industries such as electric, water and wastewater, oil and natural gas, transportation, chemical, pharmaceutical, pulp and paper, food and beverage, and discrete manufacturing [6]. Dependable ICSs are essential for reliable access to U.S. critical infrastructures.

In the past, there was a clear separation between OT systems and IT systems. In general, IT systems were concerned with the creation and transmission of electronic information while OT systems focused on the control of physical processes. OT systems implemented control procedures that were tailored to meet a specific business need and often used specialized hardware and software. Physical access to the control system was often controlled through security measures like fences, locks, and guards. In addition, OT systems were isolated, and there was no connection between an organization's OT network and IT network.

Today, however, the line between OT and IT networks is becoming blurred as OT networks are increasingly connected to IT networks [9]. The emergence of technologies such as reliable wireless communication, remote access capabilities, and Internet of Things (IoT) devices have made it feasible and desirable to integrate OT and IT networks. This integration has made it easier for organizations to monitor, control, and analyze the operation of their OT networks. However, the merging of OT and IT networks introduces new cybersecurity concerns.

Chapter 3

Motivation

Cybersecurity is important in both OT and IT systems, but in the past the approaches to securing the two systems were different.

In an IT system, protecting personal and proprietary data is typically the primary concern. Consequently, securing IT systems involved configuring firewalls, installing anti-virus software, and educating system users about the importance of creating strong passwords and how to recognize phishing attempts.

On the other hand, for an OT system such as an ICS, safety is usually of utmost importance. This includes human health and safety as well as fault tolerance to prevent loss of life or endangerment of public health or confidence, regulatory compliance, loss of equipment, loss of intellectual property, or lost or damaged products [6]. In an ICS, any security measure that compromises safety is unacceptable.

The principle of security through obscurity (STO) was commonly used to secure OT systems. STO aims to secure a system by deliberately concealing its internal operations, thereby concealing its vulnerabilities [10]. If there is no public information available about a system's operations malicious actors will find the system very difficult to penetrate. Using this strategy was acceptable in the past because ICSs were isolated, physically secure systems protected by fences, locks, and security guards. In addition, they were not connected to the internet and thus much less susceptible to hacking or viruses.

Today, OT and IT networks are becoming increasingly connected. Allowing information flow between the two networks means that STO is no longer sufficient to secure an OT system because the system is no longer isolated. Even if the system remains physically secure, communication paths between IT and OT make it possible for attackers to leverage a vulnerability in an IT network to compromise an OT network. Such an attack could have catastrophic consequences, enabling an attacker to shut down the electric grid, contaminate drinking water, or sabotage manufacturing processes.

Many of the cybersecurity best practices for IT networks, such as requiring strong user passwords, installing anti-virus software, patching known software vulnerabilities, and updating operating systems, aren't applicable to OT networks like ICSs [6].

For example, ICS hardware and software is typically older and used for a longer period of time than that of IT devices. This technology may be rarely updated or patched because legacy technology is no longer supported by the manufacturer. Also, installing updates may require the plant to cease operation and shut down for a period of time, which would interfere with critical business operations. In addition, ICS hardware is typically very specialized and constrained, making it impossible to install anti-virus software that requires additional computing power or increases device latency. Furthermore, it may not be feasible to require strong passwords or 2FA. Legacy devices may not have the ability to enforce strong password requirements, the same account may be used by multiple people, and enforcing 2FA mechanisms may be dangerous in an emergency situation when immediate action is required.

To summarize, cybersecurity is incredibly important in OT networks because a successful cyber attack could have devastating consequences. However, the approach to securing OT networks is very different than the approach to securing IT networks. The creation of a tool to analyze a given OT network and automatically verify the correct implementation of cybersecurity best practices would greatly benefit organizations, particularly those responsible for critical infrastructures. This thesis outlines the development and implementation of such a tool.

Chapter 4

Cybersecurity Best Practices for OT Networks

This chapter briefly summarizes five documents that outline cybersecurity best practices for OT networks and ICSs. The cybersecurity best practices included in the library were selected from these documents. This chapter does not attempt to describe which best practices were selected or their verification algorithms. That level of detail is contained in [Chapter 6](#).

It should be noted that some of the summaries in this chapter are incomplete. For brevity, sections of the documents that are not relevant to this thesis have been omitted from the summaries. On the other hand, if a relevant concept is discussed in more than one document, it is included in the summary of each document in which it is discussed. This is intentional because we want to emphasize the best practices that are well established and agreed upon by multiple respected organizations. In addition, one document may mention a particular security concept at a high level, while another document discusses the same concept in more detail. We have attempted to organize this chapter so the reader is first exposed to best practices at a high level, and then the details are discussed.

Finally, we would like to take a moment to define the terms “port” and “switch port” as both are used throughout this thesis. The term “port” refers to one of the 65,535 virtual TCP and UDP ports that correspond to an IP address.

Ports are used to distinguish between communication channels. In contrast, the term “switch port” is used to identify a physical port on a network switch. A network switch is a piece of hardware used to connect devices on a network.

4.1 DHS Seven Steps to Effectively Defend Industrial Control Systems

The document “Seven Steps to Effectively Defend Industrial Control Systems” [11] was published by the U.S. Department of Homeland Security (DHS) in collaboration with the Federal Bureau of Investigation (FBI) and the National Security Agency (NSA). It outlines seven strategies to effectively defend ICSs against cyber intrusions. The two strategies most relevant to this thesis are discussed below.

4.1.1 Reduce Your Attack Surface Area

An attack surface is the set of possible entry points on the boundary of a system. An attacker’s goal is to find and exploit a vulnerability in one of the entry points and gain access to the system. Reducing the area of a system’s attack surface decreases the number of possible entry points to the system, making it more difficult for attackers to penetrate the system.

The document provides several recommendations for reducing the attack surface area of a system. They include:

- Isolate ICS networks from untrusted networks, particularly the internet.
- Lock down all unused ports.
- Turn off all unused services.
- Use one-way communication when possible.
- If bidirectional communication is required, use a single open port over a restricted network path.

4.1.2 Build a Defendable Environment

A defendable environment is easier to protect in the event an attacker gains access to the system. Implementing network segmentation and restricting host-to-host communication paths are two ways to make an ICS more defendable.

Network segmentation helps to limit possible damage from perimeter breaches. Even if an adversary compromises one network segment, they will be unable to leverage the compromised segment to gain access other segments. A similar advantage is gained by restricting host-to-host communication paths. Even if a host is compromised, restricted communication paths make it less likely for an adversary to be able to jump from host to host and inflict additional damage.

The suggestions outlined in this document are described at a very high level. The following documents explain in more detail how to properly segment a network and the policies that should be implemented to sufficiently restrict host-to-host communication paths.

4.2 DHS Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies

The document “Improving Industrial Control Systems Cybersecurity with Defense-in-Depth Strategies” [9] was developed by the Department of Homeland Security (DHS)’s National Cybersecurity and Communications Integration Center (NCCIC) and Industrial Control Systems Cyber Emergency Response Team (ICS-CERT). It presents a holistic approach, referred to as Defense-in-Depth, to defending ICSs against cybersecurity threats.

Restricting the flow of information within a network is the primary strategy discussed in the document. This can be accomplished using a three-step process which is outlined below.

4.2.1 Assign Asset Criticalities

The first step is to assign each asset in the system a criticality based on its function and importance to plant operations. An accurate inventory that organizes all ICS assets according to their criticality provides a solid foundation for applying defense measures and helps ensure asset owners leave no critical devices unprotected.

4.2.2 Define Zones and Conduits

After assigning criticalities, the next step involves grouping the assets into zones and conduits. Each zone and conduit has a different security focus. Assets are typically partitioned based on their criticalities but other asset characteristics, such as device function, managing department, and necessary communication paths, can also be taken into account.

4.2.3 Establish Secure VLANs

The final step in this process is the actual segmentation of the system into the previously defined zones and conduits. This can be done using virtual local area networks, or VLANs, which divide physical networks into smaller logical networks. The document presents the following guidelines for establishing secure VLANs.

- The VLAN architecture needs to ensure the disabling of the Dynamic Trunking Protocol (DTP).
- Configure the switch ports as static access ports.
- Enable switch port security.
- Limit the number of media access control (MAC) addresses used by a single switch port.
- Disable unused switch ports.

4.2.4 Additional Best Practices

Segmenting the control system via zones and conduits is an effective way to restrict information flows within a network. For additional security, the document encourages organizations to implement the best practices listed below.

- Establish a demilitarized zone (DMZ) between the corporate and control networks to eliminate direct communication paths between the corporate and control networks.
- Ensure firewalls are configured to deny all network traffic by default and permit network traffic only by exception.
- Use data diodes to prevent information flow from lower security network zones to higher security network zones.
- Disable unneeded services, ports, and protocols to the fullest extent possible.

4.3 ISA/IEC 62443

The ISA/IEC 62443 series is the product of a joint effort between the International Automation Society (ISA) and the International Electrotechnical Commission (IEC) to address the security of industrial automation and control systems (IACS) throughout their lifecycle. The series is divided into four groups of documents organized by primary focus and intended audience.

4.3.1 Group 1

The first group of documents provides general information and introduces terminology and models that are used throughout the series. In particular, the concept of security zones and the Zone and Conduit Model are discussed [12].

Security zones are used to partition a system or network based on security requirements. A security zone is a logical grouping of physical, informational,

and application assets sharing common security requirements. Physical security zones are defined by grouping assets according to physical location. Organizations can also establish logical security zones in which assets are organized by criticality or other characteristics rather than their physical location. Each security zone is assigned a security level (SL), and all assets within a security zone must be protected according to the zone’s security level. The concept of security levels is very flexible, allowing organizations to decide the number of security levels to establish and define what each security level represents. A suggestion for establishing security levels is provided in Section 4.3.3.

The Zone and Conduit Model is used to describe the partitioning of assets within an ICS. All assets are assigned to a security zone or a conduit. Conduits are security zones that apply to specific communication processes. They can be thought of as “pipes” that connect security zones, and typically consist of the wiring, switches, routers, or other devices that make up communication paths in a network. A zone and conduit model makes it easier for an organization to assess common threats and vulnerabilities and then determine the security measures necessary to protect each zone and conduit.

4.3.2 Group 2

Documents in the second group describe policies and procedures associated with IACS security. Specifically, six security elements that are part of a well-design cybersecurity management system are discussed [13].

The first element is personnel security. Personnel security involves looking at new and current personnel to determine whether they will maintain the security of the IACS throughout the tenure of their employment. This involves background checks prior to hiring new employees and regular security training for current employees.

The second element involves physical and environmental security. Physical security measures include mechanisms like fences, locks, and guards to prevent unauthorized access and damage to IACS assets. Environmental security measures ensure that IACS assets aren’t exposed to conditions that would

make them unusable or damage the information they contain.

The third element is network segmentation. Network segmentation is the idea of separating IACS assets into zones to manage security risk. Different security countermeasures can be applied based on the zone's security level. It is also possible to limit information flow between zones of different security levels.

The final three security elements involve different types of access control: account administration, authentication, and authorization. Account administration is the process of granting and revoking account permissions. Ideally, personnel access is limited to the facilities, devices, and systems necessary to perform their job duties. The actions of authentication and authorization go hand in hand. Authentication involves positively identifying system users and is immediately followed by authorization, where system users are granted the account privileges they have been assigned.

4.3.3 Group 3

The third group of documents addresses security requirements at the system level. Most relevant to this thesis are the requirements for partitioning the system under consideration (SUC) into zones and conduits [14].

Notable zone and conduit requirements include:

- Group IACS assets into zones or conduits based on risk.
- IACS assets should be separated from enterprise assets.
- Safety related IACS assets should be separated from non-safety related IACS assets.
- Devices permitted to make temporary connections to the SUC should be separated from assets that are intended to be permanently connected to the IACS.
- Wireless devices should be separated from wired devices.

- Devices permitted to make connections to the SUC via networks external to the SUC should be grouped into a separate zone or zones.

Segmenting the control system via zones and conduits helps limit unnecessary data flow.

Section 4.3.1 mentioned that each zone and conduit is assigned a security level (SL) that indicates the level to which the zone or conduit must be protected. We use five different SLs (0, 1, 2, 3, 4), each with an increasing level of security [15]. Their definitions are provided below.

- SL 0: No specific requirements or security protection necessary
- SL 1: Protection against casual or coincidental violation
- SL 2: Protection against intentional violation using simple means with low resources, generic skills, and low motivation
- SL 3: Protection against intentional violation using sophisticated means with moderate resources, IACS specific skills, and moderate motivation
- SL 4: Protection against intentional violation using sophisticated means with extended resources, IACS specific skills, and high motivation

4.3.4 Group 4

The fourth and final group of documents addresses security requirements at the component level. These documents outline the capabilities embedded devices, host devices, and network devices must possess in order to achieve the IACS security requirements discussed in Sections 4.3.2 and 4.3.3. For example, one requirement specifies that components must have the ability to authenticate and authorize human users. This is a direct reference to Section 4.3.2 which discussed the concept of access control.

Notable component requirements [16] include:

- Components shall support a segmented network to limit the unnecessary flow of data.

- Components shall provide the capability to be configured according to recommended security configurations.
- Components shall provide the capability to specifically restrict the use of unnecessary functions, ports, protocols, and/or services.
- A network device at a zone boundary shall provide the capability to monitor and control communications at zone boundaries to enforce the compartmentalization defined in the risk-based zones and conduits model.

4.4 NIST Guidelines on Firewalls and Firewall Policy

The National Institute of Standards and Technology (NIST) published “Guidelines on Firewalls and Firewall Policy” [17]. This document provides an overview of firewall technologies, discusses options for placing firewalls within networks, and makes recommendations for establishing firewall policies.

4.4.1 Firewall Technologies

Firewalls are devices or programs used to monitor and control the flow of traffic between hosts and networks. Firewalls are a security measure implemented to prevent unauthorized access to a system and its resources.

The most basic function of a firewall is as a packet filter. As the name suggests, a firewall monitors packets traveling through a network and permits or prevents access based on a packet’s characteristics. When a packet reaches a firewall, the firewall examines its characteristics, like direction (inbound or outbound), source IP address and port, destination IP address and port, and the transport protocol being used to communicate. The firewall evaluates this information against an access control list (ACL) and decides if the packet is allowed to pass through the firewall or if it should be blocked.

There are many different kinds of firewalls with functionality that ranges from basic to advanced. The firewall described above is a very basic, stateless

packet inspection firewall. It does not keep track of the state of each packet passing through the firewall. In contrast, stateful packet inspection improves upon the performance of stateless firewalls by tracking the state of connections, in addition to packet information, and blocking packets that deviate from the expected state. The use of application firewalls, which deny or allow access based on the activity of an application, is also common.

4.4.2 Firewalls and System Architectures

Firewalls are used to separate networks with different security requirements. As such, firewalls should be placed between internal networks with different security requirements as well as at interfaces where internal networks meet external networks, like the internet.

Organizations may establish a demilitarized zone (DMZ) between a more secure ICS control network and a less secure corporate network in order to prevent direct access between the control network and the corporate network. If this architecture is used, one firewall should be placed between the control network and the DMZ, and another firewall should be placed between the DMZ and the corporate network.

4.4.3 Firewall Policies

A firewall policy describes how firewalls should handle network traffic. It is a set of rules that specify which packets should be permitted and which should be blocked, based on packet characteristics like direction of travel, source IP address and port, destination IP address and port, and the transport protocol used. Organizations develop unique firewall policies based on their security goals. In addition, NIST provides some recommended best practices for firewall policies. These best practices are listed below.

- Firewalls should block all inbound and outbound traffic that has not been expressly permitted by the firewall policy.

- Firewall policies should only allow necessary IP protocols through. Examples of commonly used IP protocols are ICMP, TCP, and UDP.
- Traffic with invalid source or destination addresses should always be blocked, regardless of the firewall location.
- Traffic with an invalid source address for incoming traffic or destination address for outgoing traffic should be blocked at the network perimeter.
- Traffic with a private destination address for incoming traffic or source address for outgoing traffic should be blocked at the network perimeter.
- Outbound traffic with invalid source addresses should be blocked.
- Incoming traffic with a destination address of the firewall itself should be blocked.

4.5 NIST Guide to Industrial Control Systems (ICS) Security

NIST also published a report that provides guidance for establishing secure ICSs [6]. Most relevant to this thesis, the report provides recommendations for integrating security into network architectures typically found in ICSs. This includes the concepts of network segmentation and segregation, boundary protection, and firewall policies.

4.5.1 Network Segmentation and Segregation

Network segmentation and segregation is one of the most effective concepts an organization can employ to protect its ICS. Network segmentation involves dividing the network into smaller networks based on factors such as department in charge, asset criticality, and information flow between assets.

The goal of segmentation is to limit access to sensitive information. In the event an attacker does exploit a vulnerability and gains access to the system,

proper network segmentation can limit the attacker’s view of the system and prevent them from gaining access to additional network segments.

Logical network segmentation can be achieved by using virtual local area networks (VLANs). In addition, unidirectional gateways, also known as data diodes, restrict information flows between connections to a single direction, implicitly segmenting the network.

Network segregation involves creating and enforcing a set of rules to control which communications are permitted through network segment boundaries. When creating a ruleset, NIST recommends implementing whitelisting instead of blacklisting. That is, grant access to the known good, rather than deny access to the known bad.

4.5.2 Boundary Protection

Boundary protection devices like gateways, routers, and firewalls are used to implement network segregation. These devices evaluate packets against a ruleset and determine if they are permitted to pass through the network boundary. This document provides some recommended best practices for allowing or blocking information flows. They include:

- Deny all communication traffic by default and only allow communication traffic by exception. This ensures that only approved communications are permitted.
- Only allow communication between authorized source and destination address pairs.
- Implement one-way data flow when possible, especially between different security domains.

4.5.3 Firewall Policies for ICSs

Firewalls are a very common boundary protection device. A thorough discussion of firewalls was provided in Section [4.4.1](#), and is omitted here to avoid

redundancy. However, like Section 4.4.3, this document presents some best practices for firewall policies, which are given below:

- Firewalls should be configured to deny all traffic except that which is absolutely required for business needs.
- Ports and services should be enabled and permissions granted on a specific case-by-case basis.
- All “permit” rules should be both IP address and TCP/UDP port specific.
- Traffic should be prevented from transiting directly from the control network to the corporate network.
- Outbound packets should be allowed only if those packets have a correct source IP address that corresponds to a device in the network.
- Devices in the control network should not be allowed to access the internet.

4.6 Selection of Best Practices

We did not include every best practice covered in the above documents in the best practices library. We chose to include, or omit, a best practice based on two criteria.

First, is there a well-known tool we can use to obtain the system information required to evaluate the best practice? If so, then the first criterion is satisfied. For example, port scan data for a system is required to evaluate certain best practices. Nmap, which stands for Network Mapper, is an open source tool used for network exploration and security auditing. Nmap can be used to obtain port scan data about a system of interest. Additional examples will be provided later.

Second, do we expect the information required to evaluate the best practice to be documented by a system administrator or other authority? If so, the second criterion is satisfied. We expect certain system information to be documented by a system administrator for configuration, security, or general

information purposes. For example, we expect the system administrator to possess records regarding the network switches present in the system hardware and the expected configuration of their switch ports. Additional examples will be provided later.

If a best practice did not satisfy either criteria, it was not included in the library. Chapter [6](#) discusses each best practice included in the library.

Chapter 5

Information Required for Verification

This chapter describes the information about an OT network that may be input to the verification tool. The information required for verification depends on which best practices the user chooses to evaluate. The required information must be contained in properly formatted .json files. Figures showing properly formatted example data are provided throughout the chapter.

In this chapter, we will refer to an OT network of interest as a “system” or “system under consideration (SUC).” This is done to avoid ambiguity when using the term “network,” because an OT network can itself contain multiple networks. The terms “system” or SUC will refer to an OT network, while the term “network” will refer to a network contained in the system.

5.1 Physical and Logical System Topology

The verification tool often requires information about a system’s physical and logical topology. Physical topology is concerned with the existence and placement of system components while logical topology documents how data flows within the system.

Several companies advertise tools with the ability to analyze a system and produce topology information including Tufin [\[18\]](#), RedEye Network Solutions

[19], Skybox Security [20], and Network Perception [21].

For this thesis we used the software tool NP-View, developed by Network Perception. Users can load configuration files from firewalls, routers, and switches into NP-View. Then, NP-View uses these files to perform a comprehensive analysis of the system. The results of the analysis can be exported as .json files.

NP-View provides a variety of system information. The items listed below may be used as input to the verification tool. For each item we also direct the reader to a figure that shows a short example of the information as it appears in a .json file.

- Information about the the zones/conduits present in the system, their assigned SL, and the entities assigned to each zone/conduit. Refer to Figure 5.1.
- Information about each device present in the system, including characteristics such as IP address, device type, assigned zone/conduit, and assigned criticality. Refer to Figure 5.2.
- Information about each (hardware) firewall present in the system, including its ACL(s), interface(s), and if it is a perimeter firewall. Note that firewalls are a special kind of device, as mentioned in the previous bullet. Refer to Figure 5.3.
- Information about each network present in the system, including characteristics such as CIDR, assigned zone/conduit, and assigned criticality. Refer to Figure 5.4.
- Information about the permitted information flows, including source and destination IP addresses and ports, transport protocol(s), and the intermediate steps along the flow path. Refer to Figure 5.5.

```

1  "groups": {
2    "Zone_GroupA_SL4": [
3      "10.2.3.0/24",
4      "FirewallA"
5    ],
6    "Zone_GroupB_SL3": [
7      "209.166.201.0/27"
8    ]
9  }

```

Figure 5.1: Example Zones/Conduits

```

1  "devices": {
2    "192.169.2.2": {
3      "deviceId": 9,
4      "deviceType": "RTU",
5      "interface": [
11     "ipName": "192.169.2.2",
12     "name": "192.169.2.2",
13     "group": "Zone_GroupC_SL1",
14     "criticality": "untrusted"
15   ]
16 }

```

Figure 5.2: Example Device

5.2 Port Scan Data

The verification tool may require port scan data for a system of interest. Port scanning is the action of remotely testing various TCP and UDP ports on a host to determine what state they are in. Popular port scanning tools include Unicornscan, Angry IP Scanner, SolarWinds Port Scanner, and Nmap. The information below is based on the Nmap port scanner, but all port scanners function similarly.

During a port scan ports are classified into three main states: Open, Closed, and Filtered. An Open port means an application is actively accepting TCP connections or UDP packets on the port [22]. Identifying open ports is often the primary goal of port scanning because each open port is a potential attack vector. Cyber adversaries attempt to exploit open ports to gain access to a host, while administrators try to protect open ports using firewalls. A Closed port is accessible, but there is no application listening on it at the present time [22]. However, closed ports may become open at some point in the future.

```

1  "devices": {
2      "FirewallA": {
3          "acl": [
4              {
5                  "action": "permit",
6                  "direction": "in",
7                  "simple_rules": [
8                      {
9                          "dstIPhigh": "255.255.255.255",
10                         "dstIPlow": "0.0.0.0",
11                         "dstPthigh": "80",
12                         "dstPtlow": "80",
13                         "protocol": "6",
14                         "srcIPhigh": "255.255.255.255",
15                         "srcIPlow": "0.0.0.0",
16                         "srcPthigh": "65535",
17                         "srcPtlow": "0"
18                     }
19                 ]
20             }
21         ],
22         "deviceId": 2,
23         "deviceType": "firewall",
24         "interface": [
25             {
26                 "IP": "10.2.3.1",
27                 "facing": "10.2.3.0/24",
28                 "name": "inside"
29             }
30         ],
31         "name": "FirewallA",
32         "group": "Zone_GroupA_SL4",
33         "criticality": "high",
34         "perimeter": 1
35     }
36 }

```

Figure 5.3: Example Firewall

Finally, a port is classified as Filtered if the scanner’s probes were prevented from reaching the port [22]. This generally means a firewall or other boundary protection device is blocking access to the port.

In the future, the lowercase terms, “open,” “closed,” and “filtered” refer to their general definitions, whereas the capitalized versions, “Open,” “Closed,” and “Filtered” refer to the state of a specific port.

For each IP address scanned, the port scanner constructs a table of ports. This table lists the port number, protocol, and state for each port [22]. An example of a properly formatted .json file containing port scan data is shown in Figure 5.6.

```

1  "networks": {
2    "10.2.3.0/24": {
3      "CIDR": "10.2.3.0/24",
4      "ips": [],
5      "netId": 989,
6      "group": "Zone_GroupA_SL4",
7      "criticality": "high"
8    }
9  }

```

Figure 5.4: Example Network

5.3 Switch Configuration Data

Cisco network switches can be accessed and configured through the command line interface (CLI). The user must access the CLI through an SSH client. PuTTY [23], [24], Solar-PuTTY, Bitwise SSH Client, and KiTTY are commonly used SSH clients. After configuring a switch, the “show” commands can be used to verify its operation [25]–[27].

Switch ports can be configured as access ports or trunk ports. An access port is a switch port that is a member of only one VLAN. A trunk port is a switch port that carries traffic for multiple VLANs.

The “show interfaces switchport” command provides information about each interface (switch port) on the switch. It tells the user the configured mode and operating mode of the interface and its participation in dynamic trunking (DTP). The “show port-security” command provides information about the security settings of an interface. Finally, the “show ip interface brief” command allows the user to quickly see the status of all the interfaces on the switch.

The information provided by the “show” commands is necessary to evaluate best practices relating to switches and VLANs. Figure 5.7 shows an example of a .json file containing properly formatted switch configuration data.

```

1  "flows": [
2    {
3      "application": "*",
4      "devices": [
5        {
6          "dstIP": "192.169.2.2-192.169.2.2",
7          "dstIncludes": [
8            {
9              "dstNet": "192.169.2.0/24",
10             "dstPt": "5631-5631",
11             "pathNumber": 1,
12             "protocol": "tcp",
13             "service": "pcanywhere-data",
14             "srcIP": "209.166.201.30-209.166.201.30",
15             "srcIncludes": [
16               {
17                 "srcNet": "209.166.201.0/27",
18                 "srcPt": "0-65535",
19                 "user": "*"
20               }
21             ]
22           }
23         }
24       ],
25     },
26     "flowsComplete": [
27       {
28         "pathNumber": 1,
29         "pathPoints": [
30           "209.166.201.0/27",
31           "FirewallB",
32           "192.169.2.0/24"
33         ]
34       }
35     ]
36   ]

```

Figure 5.5: Example Flow

5.4 Information Documented by the System Administrator

We expect the following information about the SUC to be documented by the system administrator or other authority. The verification tool may require this information to evaluate certain best practices. For each item we reference a figure that shows example information in a .json file.

5.4.1 System Whitelist

Whitelisting is a cybersecurity strategy that attempts to protect a private network from outside attacks. A whitelist is a list of administrator-approved entities. Only entities included on the whitelist are permitted to access the system. An entity not on the whitelist is automatically blocked.



Figure 5.6: Example Port Scan Data

We expect the system administrator to create a whitelist for the SUC that is divided into two parts: the source whitelist and the destination whitelist. The source whitelist contains the IP addresses of entities that are permitted to send packets. The destination whitelist contains the entities that are permitted to receive packets. The destination whitelist is organized by IP address, and then by TCP and UDP ports. An example whitelist is shown in Figure 5.8.

5.4.2 Whitelisted Ports and Blacklisted Ports

In addition to a system whitelist, we also expect the administrator to record the whitelisted and blacklisted ports for the SUC. The list of whitelisted ports contains TCP and UDP ports that have been approved. Network access to these ports is permitted. A business justification should be provided for each port on the whitelist.

In contrast to a whitelist, a blacklist is a list of entities that should be blocked. Entities on the blacklist are blocked from accessing the system, but all other entities are permitted.

```

1  "switch-config-data": {
2      "switchX": {
3          "show-interfaces-switchport": {
4              "0":{
5                  "Administrative Mode": "static access",
6                  "Negotiation of Trunking": "Off"
7              },
8              "1":{
9                  "Administrative Mode": "static access",
10                 "Negotiation of Trunking": "On"
11             }
12         },
13         "show-port-security": {
14             "0":{
15                 "Port Security": "Enabled",
16                 "Maximum MAC Addresses": 10
17             },
18             "1":{
19                 "Port Security": "Disabled",
20                 "Maximum MAC Addresses": 1
21             }
22         },
23         "show-ip-interface-brief": {
24             "0":{
25                 "Status": "up"
26             },
27             "1":{
28                 "Status": "up"
29             }
30         }
31     }
32 }

```

Figure 5.7: Example Switch Configuration Data

The list of blacklisted ports contains TCP and UDP ports with known vulnerabilities. For example, a 2019 study by the security vendor Alert Logic found that 65% of cyberattacks that target ports focus on TCP ports 22 (SSH), 80 (HTTP), and 443 (HTTPS) [28]. Consequently, a system administrator may choose to add these ports to the port blacklist if the services they provide are not used by the devices in the system.

An example port whitelist and port blacklist is shown in Figure 5.9.

5.4.3 Necessary IP Protocols

When IP packets are transmitted between hosts the protocol specified in the header section of the packet indicates the format of the data contained in the data section of the packet. The two most common protocols, with their



Figure 5.8: Example System Whitelist

protocol numbers, are TCP (6) and UDP (17).

TCP and UDP are both transport layer protocols. The transport layer provides logical communication between processes running on different hosts. Transmission Control Protocol (TCP) is a connection-oriented protocol. It ensures that packets are correctly transmitted from the source host to the destination host. TCP is often used for transmitting information like data files and web pages [29]. In contrast, User Datagram Protocol (UDP) is a simpler, connectionless protocol. It is faster than TCP, but does not attempt to correct or recover errors in IP packets. UDP is often used in applications where speed is prioritized over accuracy, such as in real-time video streaming [29].

We expect the system administrator to create a list of necessary IP protocols. This list identifies each protocol that may to be used to transmit IP packets in the SUC. An example is shown in Figure 5.10.

```

1  {"ports-whitelist": {
2      "ports": [
3          {
4              "port": "5631",
5              "protocol": "6",
6              "justification": "blah"
7          },
8          {
9              "port": "5632",
10             "protocol": "17",
11             "justification": "blah"
12         }
13     ]
14 },
15 "ports-blacklist": {
16     "ports": [
17         {
18             "port": "22",
19             "protocol": "6"
20         },
21         {
22             "port": "80",
23             "protocol": "6"
24         }
25     ]
26 }

```

Figure 5.9: Example Port Whitelist/Blacklist

5.4.4 Invalid IP Addresses and Private IP Addresses

We expect the system administrator to create two lists, one containing invalid IP addresses and the other containing private IP addresses. RFC 5737 [30] specifies IPv4 address blocks that are reserved for use in examples in specifications and other documents. These are referred to as invalid IP addresses. RFC 1918 [31] specifies IPv4 address blocks that are commonly used for local area networks (LANs) in residential and office environments. These are referred to as private IP addresses.

The system administrator may modify or add to the standard IP addresses to better tailor the lists to the SUC. For each IP address in the lists, Boolean values should be used to indicate whether an IP address is specified as a single address, in CIDR format, or as a range. An example is shown in Figure 5.11.

```

1  {
2    "necessary-IP-protocols": {
3      "protocols": [
4        {
5          "protocol": "TCP",
6          "number": 6
7        },
8        {
9          "protocol": "UDP",
10         "number": 17
11       }
12     ]
13   }
14 }

```

Figure 5.10: Example of Necessary IP Protocols

```

1  {
2    "invalid-IP-addresses": {
3      "IP-addresses": [
4        {
5          "IP-address": "192.0.2.0/24",
6          "single": 0,
7          "CIDR": 1,
8          "range": 0
9        }
10     ]
11   },
12   "private-IP-addresses": {
13     "IP-addresses": [
14       {
15         "IP-address": "10.0.0.0/8",
16         "single": 0,
17         "CIDR": 1,
18         "range": 0
19       }
20     ]
21   }
22 }

```

Figure 5.11: Example of Invalid/Private IP Addresses

5.4.5 Switch Specification Data

We expect the administrator to create a list that contains information about the network switches present in the SUC. For each network switch, each interface (switch port) should be identified. Boolean values should be used to identify an interface as Unused, an Access port, or a Trunk port. An example is shown in Figure 5.12.

In the future, the lowercase terms “unused,” “access,” and “trunk” refer to their general definitions, whereas the capitalized versions, “Unused,” “Access,” and “Trunk” refer to the status of an interface.

```

1  "switch-spec-data": {
2    "switchX": {
3      "switchX": {
4        "0": {
5          "Interface": "0",
6          "Unused": 1,
7          "Access": 0,
8          "Trunk": 0
9        },
10       "1": {
11         "Interface": "1",
12         "Unused": 0,
13         "Access": 1,
14         "Trunk": 0
15       }
16     }
17   }
18 }

```

Figure 5.12: Example Switch Specification Data

Chapter 6

Best Practices Library

This chapter describes the library of cybsersecurity best practices. The library includes each best practice the verification tool has the ability to evaluate. Just as a library patron selects a few books (but not every book) to check out from the library, so too may a user choose to evaluate some best practices but not others.

Each best practice in the library is encapsulated in a function. At least one piece of information about the SUC is provided to each function and is used to evaluate whether the best practice rule is followed. All information mentioned in this chapter was discussed in [Chapter 5](#).

The best practices have been divided into five categories and are discussed below. For some best practices the translation from documentation to verifiable statement isn't immediately obvious. In these cases we justify our translation.

For each best practice, we note the document(s) in which it is referenced and explain the function's algorithm. To facilitate understanding we include pseudocode for at least one best practice in each category. We do not include pseudocode for every best practice because many (especially those in the same category) have similar pseudocode.

6.1 Best Practices for System Segmentation

We begin by discussing the best practices for system segmentation. Segmentation is the process by which a system is partitioned into smaller pieces. This helps to limit possible damage from perimeter breaches. Even if an adversary compromises one segment, they will be unable to leverage the compromised segment to gain access to other segments.

6.1.1 Ensure each device and network is assigned a criticality.

To secure an OT system, “An organization must first identify the systems and components they consider business or mission critical. Then they must determine the criticality of each asset based on its function and importance to business operations,” [9], [14].

Each device and network present in the system is considered separately when evaluating this best practice. The function shown in Algorithm 1 is called as many times as there are devices and networks in the system. The function input is information about one device or network present in the system. The function returns “True” if the device/network is assigned a criticality. Otherwise, the function returns “False.” The system obeys the best practice if and only if the function returns “True” each time it is called.

Algorithm 1 input: device/network

```
1: if device/network criticality is unassigned then  
2:   return False  
3: end if  
4: return True
```

6.1.2 Ensure each device and network is assigned to a zone or conduit.

In addition to assigning asset criticalities, “The organization shall group IACS and related assets into zones and conduits as determined by risk,” [9], [11], [14].

Each device and network present in the system is considered separately when evaluating this best practice. The function is called as many times as there are devices and networks in the system. The function input is information about one device or network present in the system. The function returns “True” if the device/network belongs to a zone or conduit. Otherwise, the function returns “False.” The system obeys the best practice if and only if the function returns “True” each time it is called.

6.1.3 Ensure each zone and conduit has been assigned a valid security level (SL0 through SL4).

To categorize system segments, “A security level (SL) should be assigned to each zone and conduit,” [9], [12]. For this best practice the valid security levels are those introduced in Section 4.3.3, SL0 through SL4.

The function input is the information about the zones and conduits present in the system. The function iterates through each zone and conduit and returns “True” if all are assigned a valid SL. Otherwise, the function returns “False.”

6.1.4 Ensure each device/network in a zone/conduit has a criticality that is less than or equal to the security level of the zone/conduit.

As a general rule, “An asset with a low risk level may be assigned to a higher risk security zone, but assets with a high risk level should not be placed into a lower risk security zone. To do so would raise the risk of an unacceptable consequence in the event of a cyber security incident,” [13].

We translated this concept into a verifiable best practice by defining standard options for asset criticalities. An asset’s criticality may be Untrusted, Low, Medium, or High. Then we established a ruleset that states which criticalities are permitted in each SL.

- SL 0: Untrusted assets permitted
- SL 1: Untrusted and Low assets permitted
- SL 2: Untrusted and Low assets permitted
- SL 3: Untrusted, Low, and Medium assets permitted
- SL 4: Untrusted, Low, Medium, and High assets permitted

The function input is information about one device or network present in the system. The algorithm identifies its criticality, assigned zone/conduit, and the zone’s/conduit’s SL. The function returns “True” if the asset’s criticality and SL of the zone/conduit it belongs to obey the above ruleset. Otherwise, the function returns “False.”

6.1.5 Ensure information flows don’t pass from a lower security level zone to a higher security level zone.

Several document recommend implementing one-way data flow between different security domains [6], [9], [13]. We are most concerned with eliminating data flow from lower-security zones to higher-security zones. Permitting this behavior would enable an attacker to leverage a vulnerability in a lower-security zone to gain access to a higher-security zone. This defeats the purpose of system segmentation.

The function input is information about one permitted information flow, including its source and destination IP addresses and ports, the intermediate steps along the flow path, and the transport protocol. This information can be obtained using a tool to analyze a system’s logical topology. Intermediate steps along the flow path refer to the devices and/or networks the flow encounters

between its source and destination. Thus, information about all devices and networks present in the system is also required to evaluate this best practice.

The pseudocode for this function is provided in Algorithm 2. Given a flow, the function first determines the flow’s length, that is, the total number of points on the flow path. It then considers each point on the flow path, starting with the source. The algorithm gets the SL of the device/network associated with the *current* point on the flow path and the SL of the device/network associated with the *next* point on the flow path. The function returns “False” if the SL of the *next* point on the path is greater than the SL of the *current* point on the path. Otherwise, the algorithm returns to the top of the loop and the *next* point becomes the *current* point. This process continues until the *current* point is the second to last point on the flow path (and the *next* point is the flow destination). The function returns “True” if the flow does not pass from a lower-security zone to a higher-security zone at any point along its path.

Algorithm 2 input: flow, devices, networks

```

1: length_flow = number of points on flow path
2: for  $i = 1$  to length_flow-1 do:
3:   get current_point_SL                                ▷ SL of point  $i$  of flow
4:   get next_point_SL                                  ▷ SL of point  $(i + 1)$  of flow
5:   if next_point_SL > current_point_SL then
6:     return False
7:   end if
8: end for
9: return True

```

6.2 Best Practices for Firewall Policies

This section discusses best practices for firewall policies. Each firewall device has a policy associated with it. Firewall policies are used to control the flow of traffic between hosts and networks. These best practices outline effective ways to prevent unauthorized access to a system and its resources.

A firewall policy is a group of one or more access control lists (ACLs). ACLs are a sequence of Allow rules, which describe traffic that is permitted to pass through the firewall, and Deny rules, which describe traffic that should be blocked at the firewall.

Firewall Allow rules can be further divided into Allow In rules and Allow Out rules. As the name suggests, Allow In rules are used for ingress filtering and control the traffic attempting to enter a host/network. Allow Out rules are used for egress filtering and control the traffic attempting to leave a host/network.

6.2.1 Ensure there is a Deny All rule at the end of each firewall ACL.

A general best practice says firewalls should be configured to deny all traffic by default and only permit traffic by exception [6], [9], [13], [16], [17]. This ensures only approved communications are permitted. We determined that most vendors implement this best practice on their firewall devices by default. Therefore, this best practice is not included in the library.

6.2.2 Ensure no destination ports in the firewall Allow rules are on the port blacklist.

Ports frequently used as attack vectors should not be opened through the firewall [13]. Open ports are actively listening for and accepting packets. The goal is to prevent packets directed at known vulnerable ports from passing through the firewall. Thus, we limit our consideration to destination ports.

The function input is information about one firewall present in the system and the list of blacklisted ports. The algorithm iterates through each Allow rule and gets the range of destination ports and the transport protocol. The function returns “True” if no port/protocol pair on the port blacklist is contained in the range of destination ports and matches the transport protocol. Otherwise, the function returns “False.” The pseudocode for this function is shown in Algorithm 3.

Algorithm 3 input: firewall, port_blacklist

```
1: for each Allow rule in firewall policy do:
2:   get min_dst_pt
3:   get max_dst_pt
4:   get protocol
5:   for each (pt_bl, protocol_bl) on port_blacklist do
6:     if min_dst_pt  $\leq$  pt_bl  $\leq$  max_dst_pt & protocol_bl == protocol then
7:       return False
8:     end if
9:   end for
10: end for
11: return True
```

6.2.3 Ensure all destination ports in the firewall Allow rules are on the port whitelist.

Ports should be enabled and permissions granted on a specific case-by-case basis [6]. Only packets directed at approved ports should be permitted to pass through the firewall. As in the previous best practice, we limit our consideration to destination ports.

The function input is information about one firewall present in the system and the list of whitelisted ports. The algorithm iterates through each Allow rule and gets the range of destination ports and the transport protocol. The function returns “True” if all port/protocol pairs are on the port whitelist. Otherwise, the function returns “False.”

6.2.4 Ensure firewall rules only allow necessary protocols through.

As a general rule, firewall policies should only allow necessary IP protocols through [9], [17].

The function input is information about one firewall present in the system and the list of necessary IP protocols. The algorithm iterates through each Allow rule and determines the rule’s protocol. Then it checks if the protocol is on the list of necessary protocols. The function returns “True” if all firewall

rules comply. Otherwise, the function returns “False.” The pseudocode for this function is shown in Algorithm 4.

Algorithm 4 input: firewall, necessary_protocols

```

1: for each Allow rule in firewall policy do:
2:   get protocol
3:   if protocol not in necessary_protocols then
4:     return False
5:   end if
6: end for
7: return True

```

6.2.5 Ensure traffic with an invalid source or destination IP address is blocked in the firewall rules.

It is recommended that traffic with invalid source or destination IP addresses always be blocked [17].

The function input is information about one firewall present in the system and the list of invalid IP addresses. The algorithm iterates through each Allow rule and gets the range of permitted source and destination IP addresses. The function returns “True” if there is no overlap between a rule’s permitted IP addresses and the invalid IP addresses. Otherwise, the function returns “False.”

6.2.6 Ensure traffic with a private destination address for incoming traffic or source address for outgoing traffic is blocked at the network perimeter.

It is recommended that traffic with a private destination IP address for incoming traffic or source IP address for outgoing traffic be blocked at the network perimeter [17].

The function input is the information about one firewall present in the system and the list of private IP addresses. The algorithm iterates through each Allow rule. For each Allow In rule the algorithm checks for overlap between the

rule's destination IP addresses and the private IP addresses. For each Allow Out rule the algorithm checks for overlap between rule's source IP addresses and the private IP addresses. The function returns "True" if there is no overlap. Otherwise, the function returns "False."

6.2.7 Ensure incoming traffic with a destination address of the firewall itself is blocked.

It is recommended that incoming traffic with a destination IP address of the firewall itself be blocked [17].

The function input is the information about one firewall present in the system. The boundary where a firewall meets and exchanges information with another device or network is called an interface. A firewall can have multiple interfaces, and each one is assigned a unique IP address.

The algorithm iterates through each Allow In rule and gets the range of destination IP addresses. Then the algorithm checks if any of the firewall's interface IP addresses are contained in the range of destination IP addresses. The function returns "True" if there is no overlap between the destination IP addresses and the interface IP addresses. Otherwise, the function returns "False."

6.2.8 Ensure the source IP address in each firewall Allow Out rule corresponds to an IP address of a device in the system.

An organization should only permit outbound traffic with source IP addresses in use by the organization [6], [17].

The function input is information about one firewall and all devices present in the system. The algorithm iterates through each Allow Out rule and gets the range of source IP addresses. The function returns "True" if each source IP address in the range corresponds to one of the devices in the system. Otherwise, the function returns "False."

6.2.9 Ensure there is no firewall Allow rule that includes all source or destination IP addresses.

It is recommended that all firewall “permit” rules be both IP address and port specific [6]. This best practice focuses on the “IP address” part of the statement.

Each IPv4 address consists of 32 bits. IPv4 addresses are most often written in dot-decimal notation. This notation organizes the 32 bits into four octets, separated by periods, and expresses each octet as a decimal number. Using this notation, all possible IPv4 addresses are included in the range 0.0.0.0 to 255.255.255.255.

The function input is information about one firewall present in the system. The algorithm iterates through each Allow rule and gets the range of source and destination IP addresses. Then it checks if the range is 0.0.0.0 through 255.255.255.255. The function returns “True” if no Allow rule permits all possible IP addresses. Otherwise, the function returns “False.”

6.2.10 Ensure there is no firewall Allow rule that includes all destination ports.

It is recommended that all firewall “permit” rules be both IP address and port specific [6]. This best practice focuses on the “port” part of the statement.

IPv4 addresses are used to identify a device, while ports are used to identify processes or applications running on a device. Ports are referenced using a numerical value ranging from 0 to 65,535. We limit our consideration to destination ports, the ports listening for and receiving packets, because they are most vulnerable to attacks.

The function input is information about one firewall present in the system. The algorithm iterates through each Allow rule and gets the range of destination ports. Then it checks if the range is 0 through 65,535. The function returns “True” if no Allow rule permits all possible destination ports. Otherwise, the function returns “False.”

6.3 Best Practices for Information Flows

The best practices in this section are all different translations of the following statement: “Boundary protection devices should only allow communication between authorized source and destination address pairs,” [6]. We consider two different translations of the phrase “authorized source and destination address pairs.” The first translation requires that the source and destination IP addresses of each information flow correspond to known devices in the system. The second translation is more strict. It requires that the source IP address and destination IP address/port of each information flow be on the system’s source whitelist and destination whitelist, respectively.

6.3.1 The source IP address of each flow must correspond to an IP address of a device in the system.

The function input is information about one flow and all the devices present in the system. The algorithm gets the flow’s range of permitted source IP addresses. The function returns “True” if each source IP address in the range corresponds to a known device in the system. Otherwise, the function returns “False.” The pseudocode for this function is shown in Algorithm 5.

Algorithm 5 input: flow, devices

```
1: get flow_source_IP_min
2: get flow_source_IP_max
3: for  $i = \text{flow\_source\_IP\_min}$  to  $\text{flow\_source\_IP\_max}$  do
4:   if  $i$  not in devices then
5:     return False
6:   end if
7: end for
8: return True
```

6.3.2 The destination IP address of each flow must correspond to an IP address of a device in the system.

This function is identical to that in Section 6.3.1, except it checks destination IP addresses rather than source IP addresses.

6.3.3 The source IP address of each flow must be on the source whitelist.

The function input is information about one flow and the system's source whitelist. The algorithm gets the flow's range of permitted source IP addresses. The function returns "True" if each source IP address in the range is on the system's source whitelist. Otherwise, the function returns "False."

6.3.4 The destination IP address and port of each flow must be on the destination whitelist.

This function is identical to that in Section 6.3.3, except it checks flow destination IP addresses and ports against the system's destination whitelist.

6.4 Best Practices for Ports

The best practices in this section are two different translations of the statement, "Lock down all unused ports," [6], [9], [11], [16]. We define "used" ports as those that are Open or Closed. We consider all Open and Closed ports when evaluating this best practice because ports with these states are either actively listening for and accepting packets or may do so in the future.

6.4.1 Ensure all Open and Closed ports on the port scan are on the destination whitelist.

The function input is the port scan data for one IP address and the system's destination whitelist. The algorithm iterates through each port in the port

scan data. If a port’s state is Open or Closed, it checks for the IP address and port on the system’s destination whitelist. If all Open and Closed ports are on the system’s destination whitelist the function returns “True.” Otherwise, the function returns “False.” The pseudocode for this function is shown in Algorithm 6.

Algorithm 6 input: port_scan_data, destination_whitelist

```

1: get IP_address of port_scan_data
2: for each port in port_scan_data do
3:   if port is Open or Closed then
4:     if (IP_address, port) not on destination_whitelist then
5:       return False
6:     end if
7:   end if
8: end for
9: return True

```

6.4.2 Ensure all Open and Closed ports on the port scan correspond to a flow destination.

The function input is the port scan data for one IP address and the information about all permitted flows. The algorithm iterates through each port in the port scan data. If a port’s state is Open or Closed, it checks the IP address and port against each of the flows. The function returns “True” if all Open and Closed ports correspond to a flow destination. Otherwise, the function returns “False.”

6.5 Best Practices for Network Switches

This section discusses best practices for network switches. The input to each function is the specification data for one of the switches in the system and its corresponding configuration data.

6.5.1 Ensure the Dynamic Trunking Protocol (DTP) is disabled on each switch port.

To increase the security of a system’s VLANs, the Dynamic Trunking Protocol (DTP) should be disabled on each switch port [9]. The DTP makes network switches more user-friendly. If two switches are connected, the DTP enables the switches to automatically determine whether their interface should function as an access port or a trunk port [25]. This is a cybersecurity concern because it gives one side of the interface permission to modify the other side of the interface.

The algorithm iterates through each switch port (interface) in the specification data and references the configuration data. The function returns “True” if the configuration data indicates that “Negotiation of Trunking” is “Off” for each switch port. Otherwise, the function returns “False.” The pseudocode for this function is shown in Algorithm 7.

Algorithm 7 input: switch_SPEC_data, switch_CONFIG_data

```
1: for each interface in switch_SPEC_data do
2:   interface_CONFIG_data = switch_CONFIG_data[interface]
3:   if interface_CONFIG_data[‘Negotiation of Trunking’] != ‘Off’ then
4:     return False
5:   end if
6: end for
7: return True
```

6.5.2 Ensure each switch port limits the number of allowed MAC addresses to 1.

Administrators are advised to limit the number of media access control (MAC) addresses used by a single switch port [9]. A MAC address is a hardware address that uniquely identifies each device on a network. It is similar to an IP address, although MAC addresses are permanent while many IP addresses are dynamic [32]. We can prevent unauthorized devices from being connected to a switch port by limiting the number of allowed MAC addresses.

The algorithm iterates through each switch port (interface) in the specification data and references the configuration data. The function returns “True” if the configuration data shows that “Maximum MAC Addresses” is “1” for each switch port. Otherwise, the function returns “False.”

6.5.3 Ensure all switch ports not identified as trunk ports are configured as static access ports.

All switch ports should be configured as static access ports by default [9]. A switch port should only be configured as a trunk port if an administrator deems it necessary. When a switch port is configured as a static access port it cannot become a trunk port and cannot be used to send traffic to other VLANs.

The algorithm iterates through each switch port (interface) in the specification data and references the configuration data. The function returns “True” if the configuration data indicates that “Administrative Mode” is “static access” for each switch port specified as Unused or an Access port. Otherwise, the function returns “False.”

6.5.4 Ensure all switch ports not identified as trunk ports have port security enabled.

Enabling port security on switch ports reduces the risk associated with VLANs [9]. However, we note that port security cannot be enabled on trunk ports [33].

The algorithm iterates through each switch port (interface) in the specification data and references the configuration data. The function returns “True” if the configuration data indicates that “Port Security” is “Enabled” for each switch port specified as Unused or an Access port. Otherwise, the function returns “False.”

6.5.5 Ensure all unused switch ports are disabled.

All unused switch ports should be disabled [9]. Disabling unused switch ports renders them unavailable for any connectivity. Even if an unauthorized device

is connected to a switch port, if the port is disabled, the device will be unable to send or receive data.

The algorithm iterates through each switch port (interface) in the specification data and references the configuration data. The function returns “True” if the configuration data indicates that “Status” is “administratively down” for each switch port specified as Unused. Otherwise, the function returns “False.”

Chapter 7

Verification Tool

We created a tool that can analyze a given OT network and automatically verify (or disprove) the correct implementation of cybersecurity best practices. This chapter explains how to use the verification tool. It describes how to create a policy, the required input to the tool, the analysis performed by the tool, and how to interpret the output of the tool.

7.1 Policy Creation

The user must create a policy prior to using the verification tool. A policy is a customizable list of one or more cybersecurity best practices that will be evaluated against the SUC. The user selects best practices from the library to include in their policy.

The policy is created in an .xml file. Figure [7.1](#) provides an example of a simple policy. Each best practice in the policy is contained in a **PolicyRule** element. From now on, the terms “best practice” and “policy rule” are used interchangeably. Each **PolicyRule** element has three attributes (**name**, **quantifier**, and **evaluate**) and a child element, **Description**, which describes the best practice in words. The **name** attribute simply assigns an integer identification number to the best practice. The best practices in a policy are typically numbered sequentially. The **quantifier** attribute assigns a quantifier to the best practice. The quantifier will be discussed later in more detail.

The `evaluate` attribute assigns a function to the best practice. The function will also be discussed later in more detail. For now, just note that one or more parameters are passed to the function. These parameters specify the information about the SUC required to evaluate the best practice.

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <PolicyRuleSet dtdVersion="0.1" >
4    <ObjectList name="devices" imported="true" />
5    <ObjectList name="networks" imported="true" />
6
7    <PolicyRule name="1" quantifier="for_all" evaluate="assetCriticality(@devices)" >
8      <Description>Ensure each device is assigned a criticality.</Description>
9    </PolicyRule>
10
11   <PolicyRule name="2" quantifier="for_all" evaluate="assetCriticality(@networks)" >
12     <Description>Ensure each network is assigned a criticality.</Description>
13   </PolicyRule>
14 </PolicyRuleSet>
```

Figure 7.1: Example Policy

Each parameter passed to a function is listed in an `ObjectList` element at the top of the policy. Each `ObjectList` element has two attributes. The `name` attribute identifies the necessary system information. The `imported` attribute specifies if the information is imported by the user. (In some cases the necessary information is not directly imported by the user, but is instead computed within the tool.)

The policy in Figure 7.1 includes two best practices. The first best practice ensures each device in the system is assigned a criticality. The second best practice ensures each network in the system is assigned a criticality. Information about each device and network in the system is required to evaluate these policy rules. The `ObjectList` element `devices` is a list containing information about each device in the system. Similarly, The `ObjectList` element `networks` is a list containing information about each network in the system. Both must be imported by the user.

7.2 Input

First and foremost, an .xml policy file is a required input to the verification tool. Additional information about the SUC is also required and is dependent on the `ObjectList` elements in the policy. Each `ObjectList` element with the attribute `imported="true"` is a required input to the tool. This information is contained in .json files and was discussed in Chapter 5. The verification tool parses the information contained in properly formatted .xml and .json files prior to evaluation.

7.3 Evaluation

In this step the verification tool evaluates each rule in the .xml policy file and determines if the rule passes or fails. When we say a policy rule passes we mean the SUC obeys the best practice. Similarly, when we say a policy rule fails we mean the SUC does not obey the best practice.

The verification tool determines whether a rule passes or fails by calling the function assigned to the rule and evaluating the “True”/“False” value(s) returned by the function against the policy rule’s quantifier. As an example, the function `assetCriticality` is assigned to the first policy rule in Figure 7.1. Thus, to evaluate this rule the function `assetCriticality`, shown in Figure 7.2, is called.

```
def assetCriticality(funcArgs):  
    asset = funcArgs[0]  
    if 'criticality' not in asset or asset['criticality'] == 'unassigned':  
        return False  
    return True
```

Figure 7.2: Example Function

As discussed in Chapter 6, a function returns “True” or “False” each time it is called. The parameter(s) passed to the function determine how many times the function is called. An @ symbol in front of a function parameter means

the function is called multiple times, once for each entity in the group. For example, the function parameter “@devices,” seen in the first policy rule of Figure 7.1, means each device is evaluated separately and the function is called as many times as there are devices in the system.

The “True”/“False” value(s) returned by the function are evaluated against the policy rule’s quantifier to determine if the rule passes or fails. Currently, all the best practices in the library use the quantifier “for_all.” However, the verification tool framework also permits the quantifiers “not_forall,” “exists,” and “not_exists.” This allows the library to possibly be extended in the future.

A policy rule using the “for_all” quantifier passes if and only if its function returns “True” each time it is called. As an example, consider the first policy rule in Figure 7.1. Its description says, “Ensure each device is assigned a criticality.” Clearly, this rule passes if and only if every device in the system is assigned a criticality. Hence, the function parameter is “@devices” (because each device is evaluated separately) and the “for_all” quantifier is used.

As the verification tool evaluates each policy rule it creates an internal dataset. The dataset is organized by policy rule and contains an identifying piece of information for each entity that causes the policy rule to fail. The piece of information recorded depends on the policy rule. A rule that evaluates the devices in the SUC records device IP addresses in the dataset. On the other hand, a rule that evaluates the flows in the SUC records flow ID numbers in the dataset. This dataset becomes relevant when we discuss the output of the verification tool.

7.4 Output

Finally, the verification tool produces an output. The output communicates the results of the evaluation to the user.

If every rule in the policy passes, the tool simply outputs, “Policy ruleset passes.” This informs the user the SUC is compliant with each best practice in the policy.

For each policy rule that fails the tool outputs, “Policy rule X fails on

inputs...” and then lists each entity that caused the policy rule to fail. “X” is the policy rule’s integer identification number specified in the .xml policy file. Information about each entity that caused the best practice to fail is collected from the tool’s internal dataset. This informs the user which best practices in the policy the SUC fails to follow and directs them to the problematic entities.

Chapter 8

Case Study

This chapter presents a case study. We use an example system to show the implementation of the verification tool. However, for brevity we have significantly simplified the system. In addition, some characteristics of the system were slightly modified in order to provide examples of entities that comply with the best practices as well as entities that do not comply with the best practices. For completeness, the policy for this case study includes every best practice in the library and requires each type of system information described in [Chapter 5](#).

The policy is contained in an `.xml` file and the system information is contained in several `.json` files. However, these files are quite long and can be difficult to parse for readers who are not familiar with their structure. For this case study we do not present `.xml` or `.json` files. Rather than including the `.json` files, we have created a topology graphic and tables to convey the information from the `.json` files in a more reader-friendly way. Similarly, when the verification results are discussed we refer to each best practice by its description instead of referencing an `.xml` file.

8.1 System Information

In this section we discuss the relevant system information.

The system topology is shown in [Figure 8.1](#). The system consists of four

host devices (identified with black circle icons), two firewall devices (identified with brick wall icons), and three networks (identified with cloud icons). The figure also shows the IP address, firewall name, and CIDR address of each host, firewall, and network, respectively. The black lines indicate connections between components. The system is divided into three zones, which are identified by colored boxes. The security level (SL) of each zone is shown in the upper left-hand corner of the box.

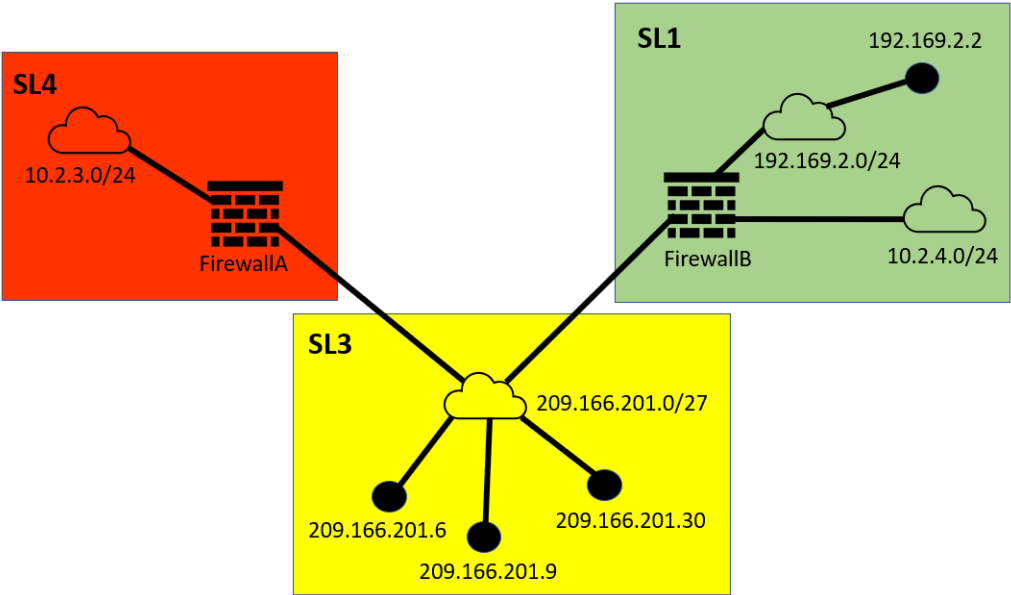


Figure 8.1: System Topology

Table 8.1 provides additional information about each host and firewall present in the system. Some of the information, like IP address and zone SL, is also provided in the topology diagram, but other information, such as device type and criticality, is unique to this table. The table on the right provides more details about the two firewalls. It notes that both are perimeter firewalls and lists the IP addresses of their interfaces.

Table 8.2 shows the firewall policies for FirewallA and FirewallB. Both firewalls have very simple policies. Each policy consists of two Allow In rules. For each rule we provide the source and destination IP addresses and ports, as well as the transport protocol. Some IP addresses and ports are specified as

Table 8.1: Devices and Firewalls

Devices				Firewalls		
IP Address	Device Type	Criticality	Zone SL	Name	Perimeter	Interface(s)
192.169.2.2	RTU	Untrusted	1	FirewallA	Yes	209.166.201.4
209.166.201.6	RTU	Medium	3			10.2.3.1
209.166.201.9	RTU	Low	3	FirewallB	Yes	209.166.201.5
209.166.201.30	RTU	Medium	3			10.2.4.1
FirewallA	Firewall	High	4			192.169.2.1
FirewallB	Firewall	Untrusted	1			

a range. This means the rule applies to any IP address and port within the range.

Table 8.2: Firewall Policies

Firewall Policies						
Firewall	Action	Source IP	Source Port	Destination IP	Destination Port	Protocol
FirewallA	Allow In	0.0.0.0-255.255.255.255	0-65535	0.0.0.0-255.255.255.255	80	TCP
FirewallA	Allow In	0.0.0.0-255.255.255.255	0-65535	0.0.0.0-255.255.255.255	443	TCP
FirewallB	Allow In	209.166.201.30	0-65535	192.169.2.2	5631	TCP
FirewallB	Allow In	209.166.201.30	0-65535	192.169.2.2	5632	UDP

Table 8.3 provides additional information about each network in the system. Once again, some of this information, like CIDR address and zone SL, is provided in the topology diagram, but this table also includes the criticality that has been assigned to each network.

Table 8.3: Networks

Networks		
CIDR	Criticality	Zone SL
10.2.3.0/24	High	4
10.2.4.0/24	High	1
192.169.2.0/24	Untrusted	1
209.166.2.0/27	Medium	3

Table 8.4 describes each information flow permitted in the system. Our example system has five flows, and each is assigned an identification number. For each flow, the table includes the source IP address and port, each of the

intermediate points along the flow path, the destination IP address and port, and the transport protocol.

Like the rules in the firewall policies, some source and destination IP addresses and ports are specified as a range. This means that every packet sent from a source IP address and port in the range to a destination IP address and port in the range, using the transport protocol specified, will travel through the system in the exact same way. For example, consider Flow #3. According to this flow specification, IP address 10.2.3.10 port 345 is permitted to send a packet to IP address 209.166.201.15 port 80 using TCP. Additionally, IP address 10.2.3.200 port 5678 is also permitted to send a packet to IP address 209.166.201.2 port 80 using TCP. Both packets would encounter the same intermediate points on the path from source to destination.

Table 8.4: Flows

Flows						
ID #	Source IP	Source Port	Intermediate Points	Destination IP	Destination Port	Protocol
1	209.166.201.30	0-65535	209.166.201.0/27	192.169.2.2	5631	TCP
			FirewallB			
			192.169.2.0/24			
2	209.166.201.30	0-65535	209.166.201.0/27	192.169.2.2	5632	UDP
			FirewallB			
			192.169.2.0/24			
3	10.2.3.0-10.2.3.255	0-65535	10.2.3.0/24	209.166.201.0-209.166.201.31	80	TCP
			FirewallA			
			209.166.201.0/27			
4	10.2.3.0-10.2.3.255	0-65535	10.2.3.0/24	209.166.201.0-209.166.201.31	443	TCP
			FirewallA			
			209.166.201.0/27			
5	192.166.2.2	0-65535	192.169.2.0/24	209.166.201.0-209.166.201.31	0-65535	Any
			FirewallB			
			209.166.201.0/27			

Table 8.5 shows the sample port scan data for the system. For each IP address scanned, we list the ports, protocols, and port states. Our table only includes ports that are Open or Closed because those are the states we are most interested in. We assume all other ports were categorized as Filtered, meaning they are blocked by a firewall. For brevity, the table does not include Filtered ports.

The topology figure shows that the system does not contain any network switches. However, in order to evaluate the best practices for network switches,

Table 8.5: Port Scan Data

Port Scan Data			
IP Address	Port	Protocol	State
192.169.2.2	80	TCP	Open
	5631	TCP	Open
	5632	UDP	Closed
209.166.201.6	80	TCP	Closed
	443	TCP	Open
209.166.201.9	80	TCP	Closed
	443	TCP	Open
209.166.201.30	80	TCP	Closed
	443	TCP	Open

we created hypothetical data for two network switches, SwitchX and SwitchY. Each switch has four switch ports, or interfaces, that are identified with the values 0 through 3. Table 8.6 shows the hypothetical switch configuration data for SwitchX and SwitchY.

Table 8.6: Switch Configuration Data

Switch Configuration Data						
Switch	Interface	Negotiation of Trunking	Maximum MAC Addresses	Administrative Mode	Port Security	Status
SwitchX	0	Off	10	static access	Enabled	up
	1	On	1	static access	Disabled	up
	2	Off	1	static access	Enabled	up
	3	Off	1	trunk	Disabled	up
SwitchY	0	Off	1	trunk	Disabled	up
	1	Off	1	static access	Enabled	administratively down
	2	Off	1	static access	Enabled	up
	3	Off	1	static access	Enabled	up

The remaining tables in this section contain data we expect the system administrator to have in his/her possession. We created the hypothetical data that is used for this case study. Table 8.7 shows the system whitelist. The left side of the table identifies the IP addresses on the source whitelist. The right side of the table identifies the destination whitelist. The destination whitelist contains the permitted destination IP addresses, ports, and transport protocols.

Table 8.8 shows the port whitelist and port blacklist. The left side of the

Table 8.7: System Whitelist

System Whitelist			
Source Whitelist		Destination Whitelist	
IP Address	IP Address	Port	Protocol
192.169.2.2	192.169.2.2	5631	TCP
209.166.201.30		5632	UDP
	209.166.201.6	80	TCP
		443	TCP
	209.166.201.9	80	TCP
		443	TCP
	209.166.201.30	80	TCP
		443	TCP

Table 8.8: Port Whitelist/Port Blacklist

Port Whitelist		Port Blacklist	
Port	Protocol	Port	Protocol
5631	TCP	22	TCP
5632	UDP	80	TCP
		443	TCP

table lists the ports and protocols that are on the port whitelist. The right side of the table lists the ports and protocols that are on the port blacklist. Notice that TCP port 80 and TCP port 443 are on the port blacklist and the destination whitelist, which is a clear conflict. This conflict should not occur in a real-world system. However, it was intentionally created for this case study to better illustrate both best practices that are followed and those that are not.

Table 8.9 shows the transport protocols that are permitted to be used in the system, as well as their protocol numbers. This system only permits the TCP and UDP protocols.

Table 8.10 lists the invalid IP addresses on the left and the private IP addresses on the right. For this case study we simply included the standard

Table 8.9: Necessary Protocols

Necessary Protocols	
Protocol	Protocol Number
TCP	6
UDP	17

Table 8.10: Invalid/Private IP Addresses

Invalid IP Addresses	Private IP Addresses
192.0.2.0/24	10.0.0.0/8
198.51.100.0/24	172.16.0.0/12
203.0.113.0/24	192.168.0.0/16

Table 8.11: Switch Specification Data

Switch Specification Data		
Switch	Interface	Interface Status
SwitchX	0	Unused
	1	Access
	2	Access
	3	Trunk
SwitchY	0	Trunk
	1	Unused
	2	Access
	3	Access

invalid IP addresses mentioned in [30] and the standard private IP addresses mentioned in [31].

Table 8.11 specifies the status of each interface of SwitchX and SwitchY. We identify each interface as Unused, Access, or Trunk.

8.2 System Verification

The policy and system information were provided as input to the verification tool. In this section we discuss the verification tool output. For each policy rule in the library we state the verification result and then explain why the rule passed or failed.

8.2.1 Best Practices for System Segmentation

Ensure each device and network is assigned a criticality.

Each device and network in the system complies with this best practice. As can be seen in Tables 8.1 and 8.3, each asset is assigned a criticality.

Ensure each device and network is assigned to a zone or conduit.

Each device and network in the system complies with this best practice. As can be seen in Figure 8.1 each asset in the system belongs to one of three zones.

Ensure each zone and conduit has been assigned a valid security level (SL0 through SL4).

The system complies with this best practice. As can be seen in Figure 8.1 the three zones in the system have been assigned SL4, SL3, and SL1, which are all valid security levels.

Ensure each device/network in a zone/conduit has a criticality that is less than or equal to the security level of the zone/conduit.

The network 10.2.4.0/24 does not comply with this best practice. The network criticality is High, but it is assigned to an SL1 zone. This is a violation of the rule defined in Section 6.1.4.

Ensure information flows don't pass from a lower security level zone to a higher security level zone.

Flow #5 does not comply with this best practice. Refer to Table 8.4 and Figure 8.1. Flow #5 has a source IP address of 192.166.2.2, and this host device is in the SL1 zone. The intermediate path points are the network 192.169.2.0/24, FirewallB, and the network 209.166.201.0/27. The violation occurs where the flow passes from FirewallB, which is in the SL1 zone, to the network 209.166.201.0/27, which is in the SL3 zone.

8.2.2 Best Practices for Firewall Policies

The verification tool uses the firewall policy data, in Table 8.2, to evaluate the following best practices.

Ensure no destination ports in the firewall Allow rules are on the port blacklist.

FirewallA does not comply with this best practice. The policy for FirewallA allows traffic to reach TCP port 80 and TCP port 443. Both of these ports are on the port blacklist in Table [8.8](#).

Ensure all destination ports in the firewall Allow rules are on the port whitelist.

Once again, FirewallA does not comply with this best practice. TCP port 80 and TCP port 443 are not on the port whitelist.

Ensure firewall rules only allow necessary protocols through.

Both firewalls comply with this best practice. They permit the TCP and UDP protocols, which are both on the list of necessary protocols in Table [8.9](#).

Ensure traffic with an invalid source or destination IP address is blocked in the firewall rules.

FirewallA does not comply with this best practice. The policy rules permit all possible source and destination IP addresses, including the invalid IP addresses specified in Table [8.10](#).

Ensure traffic with a private destination address for incoming traffic or source address for outgoing traffic is blocked at the network perimeter.

FirewallA does not comply with this best practice. We first note that FirewallA is a perimeter firewall. This is specified in Table [8.1](#). The policy rules govern incoming traffic and permit all possible destination IP addresses, including the private IP addresses specified in Table [8.10](#).

Ensure incoming traffic with a destination address of the firewall itself is blocked.

FirewallA does not comply with this best practice. The policy rules permit all possible destination IP addresses, including FirewallA's interface IP addresses specified in Table [8.1](#).

Ensure the source IP address in each firewall Allow Out rule corresponds to an IP address of a device in the system.

Both firewalls automatically comply with this best practice because their policies do not include any Allow Out rules.

Ensure there is no firewall Allow rule that includes all source or destination IP addresses.

FirewallA does not comply with this best practice. As discussed previously, its policy rules permit all possible source and destination IP addresses.

Ensure there is no firewall Allow rule that includes all destination ports.

Both firewalls comply with this best practice. Each policy rule limits the destination port to a single port.

8.2.3 Best Practices for Information Flows

The verification tool evaluates the following best practices by referring to the flow data contained in Table [8.4](#).

The source IP address of each flow must correspond to an IP address of a device in the system.

Flows #3 and #4 do not comply with this best practice. Both flows permit all source IP addresses in the range 10.2.3.0 through 10.2.3.255. However, none

of these IP addresses correspond to devices in the system, as can be seen in Table 8.1.

The destination IP address of each flow must correspond to an IP address of a device in the system.

Flows #3, #4, and #5 do not comply with this best practice. These flows permit all destination IP addresses in the range 209.166.201.0 through 209.166.201.31. This is 32 possible IP addresses. However, only three IP addresses in this range correspond to devices in the system: 209.166.201.6, 209.166.201.9, and 209.166.201.30.

The source IP address of each flow must be on the source whitelist.

Flows #3 and #4 do not comply with this best practice. Both flows permit all source IP addresses in the range 10.2.3.0 through 10.2.3.255. However, none of these IP addresses are on the source whitelist in Table 8.7.

The destination IP address and port of each flow must be on the destination whitelist.

Flows #3, #4, and #5 do not comply with this best practice. These flows permit all destination IP addresses in the range 209.166.201.0 through 209.166.201.31. This is 32 possible IP addresses. However, only three IP addresses in this range are on the destination whitelist: 209.166.201.6, 209.166.201.9, and 209.166.201.30. In addition, flow #5 permits all destination ports and any protocol, which does not agree with the destination whitelist, either.

8.2.4 Best Practices for Ports

The verification tool evaluates the following best practices by comparing the port scan data (Table 8.5) against the destination whitelist (Table 8.7) and flow data (Table 8.4).

Ensure all Open and Closed ports on the port scan are on the destination whitelist.

The host device with IP address 192.169.2.2 does not comply with this best practice. The port scan data says TCP port 80 is Open on this device. However, IP address 192.169.2.2 TCP port 80 is not on the destination whitelist.

Ensure all Open and Closed ports on the port scan correspond to a flow destination.

The host device with IP address 192.169.2.2 does not comply with this best practice. The port scan data says TCP port 80 is Open on this device. However, IP address 192.169.2.2 TCP port 80 does not correspond to a flow destination.

8.2.5 Best Practices for Network Switches

The verification tool evaluates the following best practices using the switch specification data (Table 8.11) and the switch configuration data (Table 8.6).

Ensure the Dynamic Trunking Protocol (DTP) is disabled on each switch port.

SwitchX does not comply with this best practice. It can be seen in Table 8.6 that “Negotiation of Trunking” is “On” for SwitchX Interface 1, which means the DTP is enabled.

Ensure each switch port limits the number of allowed MAC addresses to 1.

SwitchX does not comply with this best practice. It can be seen in Table 8.6 that 10 MAC addresses are permitted on SwitchX Interface 0.

Ensure all switch ports not identified as trunk ports are configured as static access ports.

Both switches comply with this best practice. It can be seen in Table 8.11 that SwitchX Interface 3 and SwitchY Interface 0 are the only two interfaces specified as trunk ports. In Table 8.6 the “Administrative Mode” of all other interfaces is set to “static access.”

Ensure all switch ports not identified as trunk ports have port security enabled.

SwitchX does not comply with this best practice. Table 8.11 notes that SwitchX Interface 1 is an Access port, however, it can be seen in Table 8.6 that “Port Security” is “Disabled” on this interface.

Ensure all unused switch ports are disabled.

SwitchX does not comply with this best practice. Table 8.11 notes that SwitchX Interface 0 is Unused, however, it can be seen in Table 8.6 that its “Status” is “up.” If the switch port was disabled, its “Status” would be “administratively down.”

Chapter 9

Conclusion

Cyberattacks on critical infrastructures are becoming more common, due in large part to the convergence of organizations' OT systems and IT systems. These attacks are particularly concerning because they pose a direct threat to human and environmental health and safety. As such, it is now more important than ever to protect critical infrastructures by verifying the security of the OT systems that control them.

The main contribution of this thesis is a tool that can be used to automatically verify the correct implementation of cybersecurity best practices for a given OT network. In this thesis, we first described OT networks and then used the importance of secure OT networks to motivate the need for our verification tool. Next, we summarized several documents that outline cybersecurity best practices for OT networks. Ultimately, we detailed the development of the verification tool. We discussed the system information required for verification, the library of best practices, and how the verification tool works. Finally, we presented a case study and used an example OT system to illustrate the implementation of the verification tool.

References

- [1] M. Peñaloza. “Ransomware attack shuts down a top U.S. gasoline pipeline.” (May 8, 2021), [Online]. Available: <https://www.npr.org/2021/05/08/995040240/cybersecurity-attack-shuts-down-a-top-u-s-gasoline-pipeline>.
- [2] Z. Cohen, G. Sands, and M. Egan. “What we know about the pipeline ransomware attack: How it happened, who is responsible and more.” (May 10, 2021), [Online]. Available: <https://edition.cnn.com/2021/05/10/politics/colonial-ransomware-attack-explainer/index.html>.
- [3] S. Morrison. “How a major oil pipeline got held for ransom.” (Jun. 8, 2021), [Online]. Available: <https://www.vox.com/recode/22428774/ransomware-pipeline-colonial-darkside-gas-prices>.
- [4] S. Kelly and J. Resnick-ault. “One password allowed hackers to disrupt Colonial Pipeline, CEO tells senators.” (Jun. 8, 2021), [Online]. Available: <https://www.reuters.com/business/colonial-pipeline-ceo-tells-senate-cyber-defenses-were-compromised-ahead-hack-2021-06-08/>.
- [5] C. Wilkie. “Colonial Pipeline paid \$5 million ransom one day after cyberattack, CEO tells Senate.” (Jun. 8, 2021), [Online]. Available: <https://www.cnbc.com/2021/06/08/colonial-pipeline-ceo-testifies-on-first-hours-of-ransomware-attack.html>.

- [6] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, “Guide to industrial control systems (ICS) security,” National Institute of Standards and Technology, Tech. Rep. NIST.SP.800-82r2, May 2015.
- [7] D. Krambeck. “An introduction to SCADA systems.” (Aug. 31, 2015), [Online]. Available: <https://www.allaboutcircuits.com/technical-articles/an-introduction-to-scada-systems/>.
- [8] “What is a distributed control system?” (Apr. 24, 2018), [Online]. Available: <https://controlstation.com/blog/what-is-a-distributed-control-system/>.
- [9] “Recommended practice: Improving industrial control system cybersecurity with defense-in-depth strategies,” U.S. Department of Homeland Security, Tech. Rep., Sep. 2016.
- [10] L. Lilly. “Security through obscurity: The good, the bad, the ugly.” (May 27, 2020), [Online]. Available: <https://thecyberpatch.com/security-through-obscurity-the-good-the-bad-the-ugly/>.
- [11] “Seven steps to effectively defend industrial control systems,” U.S. Department of Homeland Security, Tech. Rep.
- [12] “Security for industrial automation and control systems, Part 1-1: Terminology, concepts, and models,” International Society of Automation, Tech. Rep. ANSI/ISA-62443-1-1-2007, Oct. 2007.
- [13] “Security for industrial automation and control systems, Part 2-1: Establishing an industrial automation and control systems security program,” International Society of Automation, Tech. Rep. ANSI/ISA-62443-2-1-2009, Jan. 2009.
- [14] “Security for industrial automation and control systems, Part 3-2: Security risk assessment for system design,” International Society of Automation, Tech. Rep. ANSI/ISA-62443-3-2-2020, Aug. 2020.
- [15] “Security for industrial automation and control systems, Part 3-3: System security requirements and security levels,” International Society of Automation, Tech. Rep. ANSI/ISA-62443-3-3-2013, Aug. 2013.

- [16] “Security for industrial automation and control systems, Part 4-2: Technical security requirements for IACS components,” International Society of Automation, Tech. Rep. ANSI/ISA-62443-4-2-2018, Aug. 2018.
- [17] K. Scarfone and P. Hoffman, “Guidelines on firewalls and firewall policy,” National Institute of Standards and Technology, Tech. Rep. NIST.SP.800-41r1, Sep. 2009.
- [18] *Tufin*. [Online]. Available: <https://www.tufin.com/>.
- [19] *Redeye Network Solutions*. [Online]. Available: <https://www.redeyenetWORKS.com/>.
- [20] *Skybox Security*. [Online]. Available: <https://www.skyboxsecurity.com/>.
- [21] *Network Perception*. [Online]. Available: <https://network-perception.com/>.
- [22] G. Lyon. “Nmap network scanning.” (2021), [Online]. Available: <http://nmap.org/book/>.
- [23] *Access the CLI via PuTTY using a console connection on 300 and 500 series managed switches*, Cisco, Feb. 2020.
- [24] T. Keary. “How to configure Cisco switch: A step-by-step guide with commands.” (Apr. 7, 2021), [Online]. Available: <https://www.comparitech.com/net-admin/configure-cisco-switches/>.
- [25] E. Harmoush. “Configuring VLANs on Cisco switches.” (2021), [Online]. Available: <https://www.practicalnetworking.net/stand-alone/configuring-vlans/>.
- [26] J. Parker. “Guide to the most important Cisco IOS commands.” (Jul. 12, 2021), [Online]. Available: <https://www.pcwldd.com/cisco-commands-guide>.
- [27] “How to configure port-security on Cisco switch.” (2022), [Online]. Available: <https://networklessons.com/switching/how-to-configure-port-security-on-cisco-switch>.

- [28] T. Bradley. “65% of port vulnerabilities target just three ports.” (Sep. 16, 2019), [Online]. Available: <https://www.alertlogic.com/blog/65-of-port-vulnerabilities-target-just-three-ports/>.
- [29] M. Cook. “TCP vs. UDP: What’s the difference?” (Oct. 24, 2017), [Online]. Available: <https://www.lifesize.com/en/blog/tcp-vs-udp/>.
- [30] J. Arkko, M. Cotton, and L. Vegoda, “IPv4 address blocks reserved for documentation,” RFC Editor, RFC 5737, Jan. 2010. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc5737>.
- [31] Y. Rekhter, R. G. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, “Address allocation for private internets,” RFC Editor, BCP 5, Feb. 1996. [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1918.txt>.
- [32] S. Batt. “Understanding IP and MAC addresses: What are they good for?” (Oct. 27, 2020), [Online]. Available: <https://www.makeuseof.com/tag/ip-mac-address/>.
- [33] “Configuring port security,” in *Catalyst 4500 Series Switch Cisco IOS Software Configuration Guide*. San Jose, CA: Cisco Systems, Inc., 2020, ch. 50, pp. 50.1–50.34.