Validating the Security Policies in SDN Switch Across ICS Networks

Sandeep Gogineni Ravindrababu
sandeepg@uidaho.edu
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID

Jim Alves-Foss
jimaf@uidaho.edu
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID

ABSTRACT

The deployment of security on a network infrastructure requires the specification and enforcement of security policies that specify the allowed communication between devices on that network. However, there is a distinction between security policies and the technologies that implement those policies. There is also often a distinction between intended policy and deployed or configured policy. Therefore there is a need to confirm compliance between policy and reality in a network. This is especially true in industrial control systems where there is a lot of network infrastructure and special purpose devices which can not be scanned or analyzed using traditional cybersecurity tools. This paper outlines a method for determining whether the security policy applied in the control rules of an SDN switch deployed in an industrial control system network is compliant with the organization's high-level policies.

KEYWORDS

Policy, industrial control systems, software defined networking

1 INTRODUCTION

Industrial Control Systems (ICS) are recommended to follow standard procedures, protocols, or guidelines developed by the government or organizations to protect against cyber-attacks. The National Institute of Standards and Technology (NIST) has provided guidance for securing ICS networks consisting of supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and programmable logic controllers (PLC) [10]. This guide has clearly stated that system vulnerabilities in ICS can occur in hardware or software due to the misconfiguration and poor maintenance of devices deployed across the complex ICS networks.

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

Misconfigurations often contribute to the disclosure of confidential data, which aids adversaries in compromising the network. Misconfiguration problems can be mitigated in several ways, one of which is to adopt software-defined networking (SDN) technology. Several researchers have proven that SDN technology can increase the operational performance and efficiency of OT systems. Cruz et al. [4] has illustrated how interoperability and performance using SDN can be achieved in real-time networks. Mark et al. [5] have summarized the advantages of SDN on control systems. Pascal et al. [11] proposed SDN-based centralized methods for global optimizations on Internet of Things(IoT). And a survey on SDN for IoT has been conducted by Nikos et al. [1] emphasizing the impacts of SDN on 5G networks. A framework for network resilience using SDN has been designed by Amir et al. [6], and Clements et al. [3] have proposed solutions by evaluating SDN to reduce the digital attack surface of nuclear security systems.

Due to the versatility of SDN networking, which includes dynamic reconfiguration, unified policy management, flow control, fault tolerance, programmability, and more, it is easy to configure SDN devices in complex environments [7]. However, Chica et al. [2] has shown that SDN is vulnerable to specific attacks because of its architecture and Varsha et al., [12] specifies that poor configuration using SDN technology will still result in insecure networks. Considering the rapid evolution of SDN technology for implementing OT systems, the technology is far from being considered secure and dependable. In order to address the security misconfiguration issue in the ICS network using SDN technology, there is a need to develop a formal model to check whether the implemented policy is in line with the organization's specified policy, and this is our primary research objective. In this paper, we are considering enforced policies in an SDN switch as Implemented Security Policies (ISP) and the policies defined or specified by the organizations as Corporate Security Policy (CSP).

So, the main goal of this paper is to develop a tool that can take configuration files and high-level policy documents as input and perform a systematic review that tests the implemented security policy (ISP) for compliance with the defined security policy (CSP) as well as contradictions and inconsistencies. To achieve this, we must first extract the policies that have been enforced from the software-defined networking switch. This has been accomplished in our previous work [8]. In this paper, our focus is to:

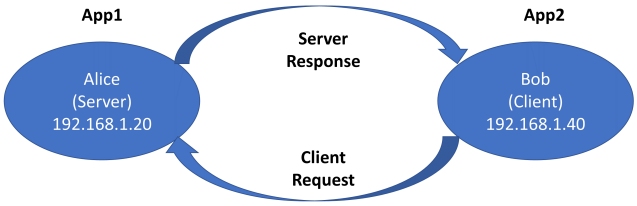

Figure 1: ICS Server Application

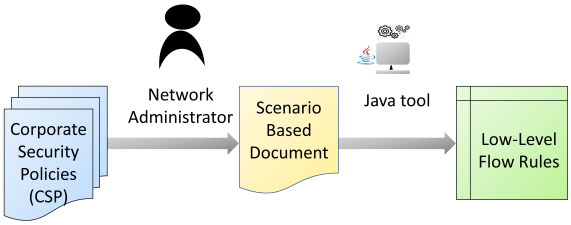

Figure 2: Conversion of CSP to Low-level flow rules

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

The remainder of the paper is organized in the following manner. Section 2 expands on the scenario-based method for converting high-level policies to low-level flow rules, while Section 3 outlines how to structure an ISP-to-CSP mapping model using sets and relations principles with an example. Section 4 delves into the analysis of the output generated by the previous section to see whether an SDN switch in an ICS network is configured correctly. Finally, Section 5 concludes the paper.

2 CONVERSION OF CSP TO LOW-LEVEL FLOW RULES

In our previous work [8], we developed a tool to automatically detect the security policy as implemented in the control rules of an SDN switch deployed in an industrial control system network. Using a scenario-based approach, this paper expands the tool's capabilities to automatically convert the CSP to low-level flow rules. The scenario-based programming approach allows network operators to specify security policies using example behaviors [13]. The network operator sets the required security policy using scenarios consisting of packet traces and the actions taken for each packet. The tool takes these scenarios as input and generates the flow table. The process used for converting the CSP to low-level flow rules is shown in Fig 2.

First and foremost, the network administrator must describe the security policy's behavior in the document. Following that, the tool takes this document as input and translates the scenarios into flow rules. Finally, these flow rules are stored in a database and translated into authorization policies for further investigation. To illustrate the process, we consider the example of a sample CSP defined for a simple ICS server application. App1 receives and responds to request from a client application App2 (Fig 1) defines the rules at high-level as:

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

Converting CSP into scenarios is done in three steps,

- **Establishing the domain**: We consider App1 and App2 as a set of domains, where each consists of multiple devices associated with unique IP and MAC addresses. This implies that no other device apart from App1 and App2 should communicate across the network, as shown in Fig 3(a).
  The scenario diagram for this is shown in Fig 4 (Scenario 1). We are defining a single tuple to denote the packet, with fields defining the domain. M1 and M2 specify the messages concerning the specified domains.
- **Defining Ports**: In this step, we define the ports for each device as shown in Fig 3(b). This implies that the communication between any device in with the domain must happen through the defined ports. The scenario diagram for this is shown in Fig 4(Scenario2). We use two tuples to denote the packet for this step, with the fields defining the domain and a set of ports.This scenario ensures that both domains App1 and App2 have a set of ports devoted to them, with each port designated for a specific device. This prevents overlapping and establishes a secure channel.
- **Defining Scenarios** : Based on the CSP provided, the network administrator can define a multiple scenarios for the devices associated within the domain. For example, consider that communication is allowed between the end-devices Alice and Bob. Alice is in the App1 domain, associated with IP address IP1 and Port P1. Bob is in the App2 domain associated with IP address IP5 and Port P2. The scenario for this is shown in Fig 4 (Scenario3). We use 4 tuples to denote a packet for this step, with the fields becoming device name,

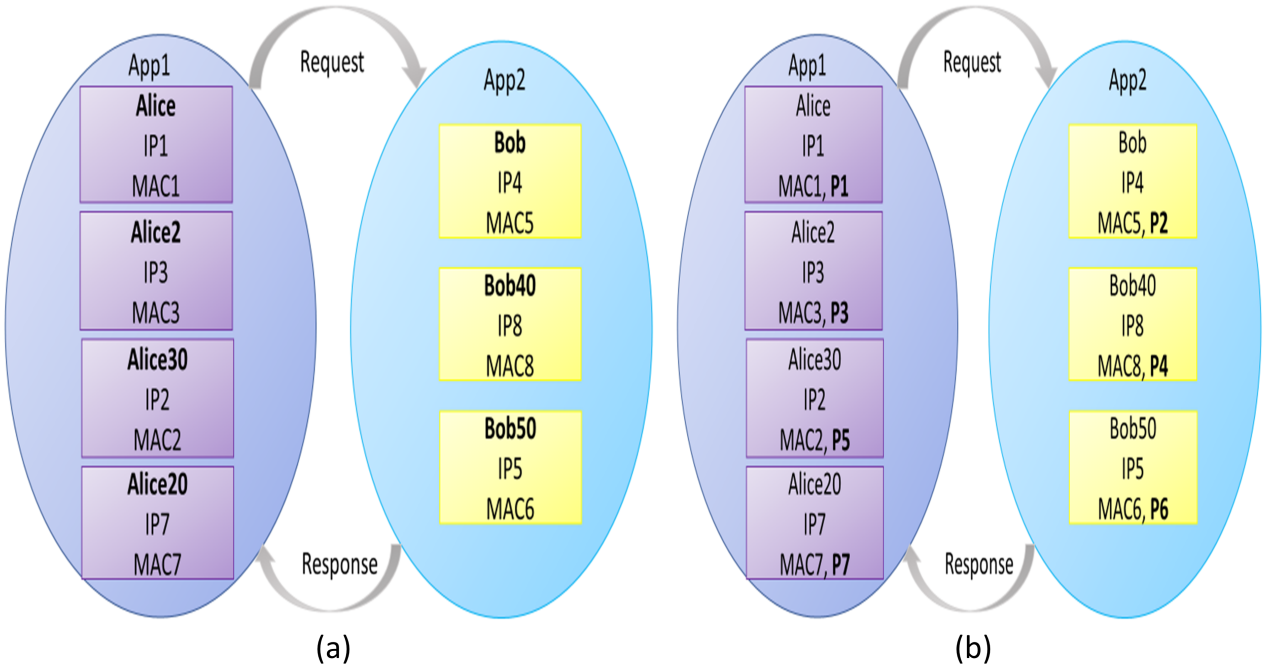

Figure 3: Configuring Devices and Ports for domains App1 and App2

Table 1: Flow table generated based on the scenarios file.

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

MAC address, IP address, and port used for sending and receiving packets.

For this paper, we have defined several scenarios as outlined in Fig 5. This scenario document is given as input to the tool which automatically converts the scenarios into low-level flow rules as show in Table 1. These flow rules are documented and stored in a database. The flow rules are then converted to ponder policies for further investigation. Our prior paper [8] discusses the importance of expressing flow rules as ponder policies in detail. The conversion of CSP to low-level flow rules is necessary because it allows network administrators to map flow rules without having to implement them in SDN switches, allowing for a comparison with existing configuration and quick analysis.

3 FORMAL APPROACH FOR DERIVING CONSISTENCY

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

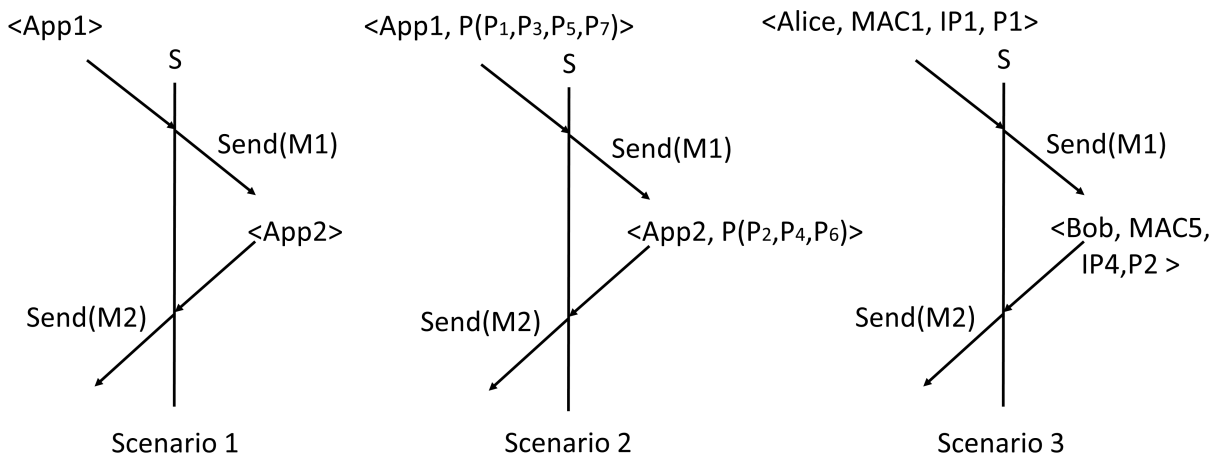

Figure 4: Step by Step process for coverting CSP to scenarios

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

Figure 5: Sample scenario document

devices. The following definitions are provided to formally specify the mapping process,

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

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

Definition2: We define CSP as a finite non-empty set consisting of elements CSP = $\{c_1, c_2, c_3, c_4 \ldots. c_m\}$, where for all c in CSP represents the low-level implementation of CSP such that c = $<S_c, T_c, R_q>$. S_c is subject, T_c is Target, and R_q is requirements that both subject and target must follow to communicate. To illustrate this with an example, let us consider a sample high-level CSP as follows:

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

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

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

For the above sample CSP, the low-level ISP is represented as $i_1 = <$ Bob, Alice, $\{S_m, R_m\}$, {ARP, B_1}$>$. Bob and Alice indicate device names, S_m and R_m refer to the specific operations that the source Bob can perform on Target. In

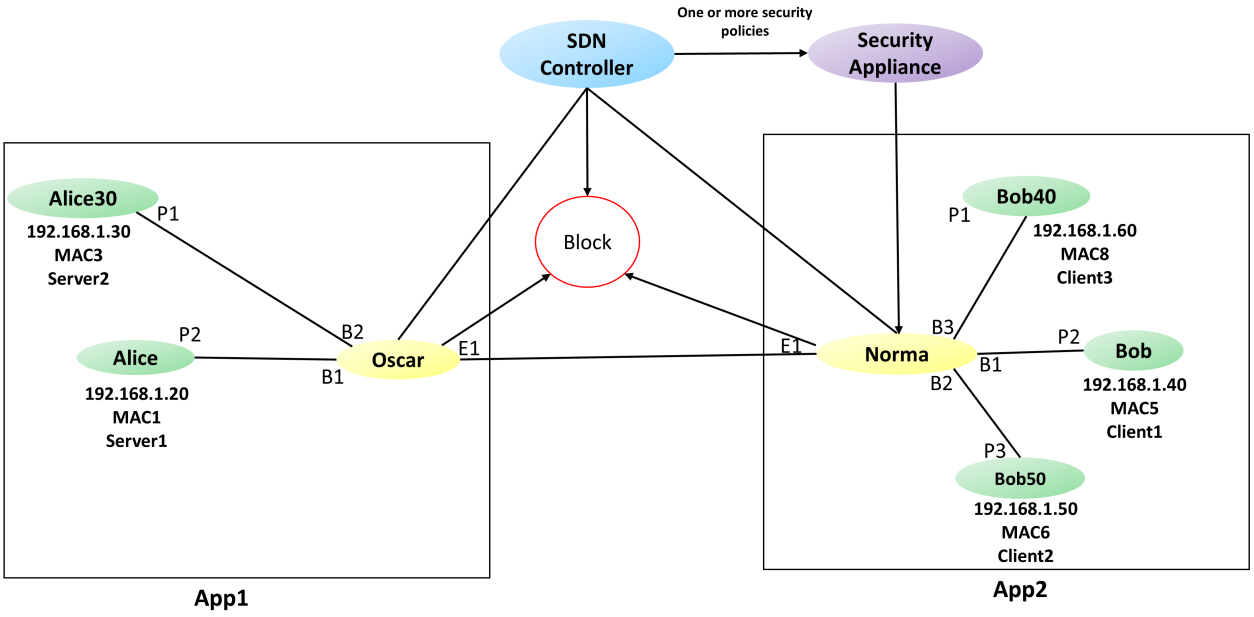

Figure 6: Low-Level network configuration before policy implementation

this case, Bob is allowed to send or receive messages from target Alice using ARP as a communication protocol through port B_1.

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

To understand how the mapping process works at network level, let us consider Flow Num 1 from Table 1. This flow rule clearly states that Alice can communicate with Bob using web-based protocols. Figure 6 depicts the low-level network configuration before policy implementation. Figure 7 depicts the low-level network configuration after the policy implementation. From the Fig 7, we can see that communication at low-level is allowed only from Alice to Bob through the switches Oscar and Norma. The SDN switches block other devices trying to communicate. So we conclude that the flow rule implemented on the network complies with the flow rule specified in Table 1. Hence we consider this as a valid mapping relation. In general, a network is said to be consistent if all flow rules are applied in accordance with the organization's policies. In our context, we define consistency as behavioral equality among two states, in this case it is between ISP and CSP. This equality between any two states or two processes can be determined using the bisimilarity property. Bisimilarity is a behavioral equivalency notion that is frequently used on processes. It is also known as robust equality because it solely considers the interaction that the processes may or may not have [9].

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

The M is considered as bisimilar, iff

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

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

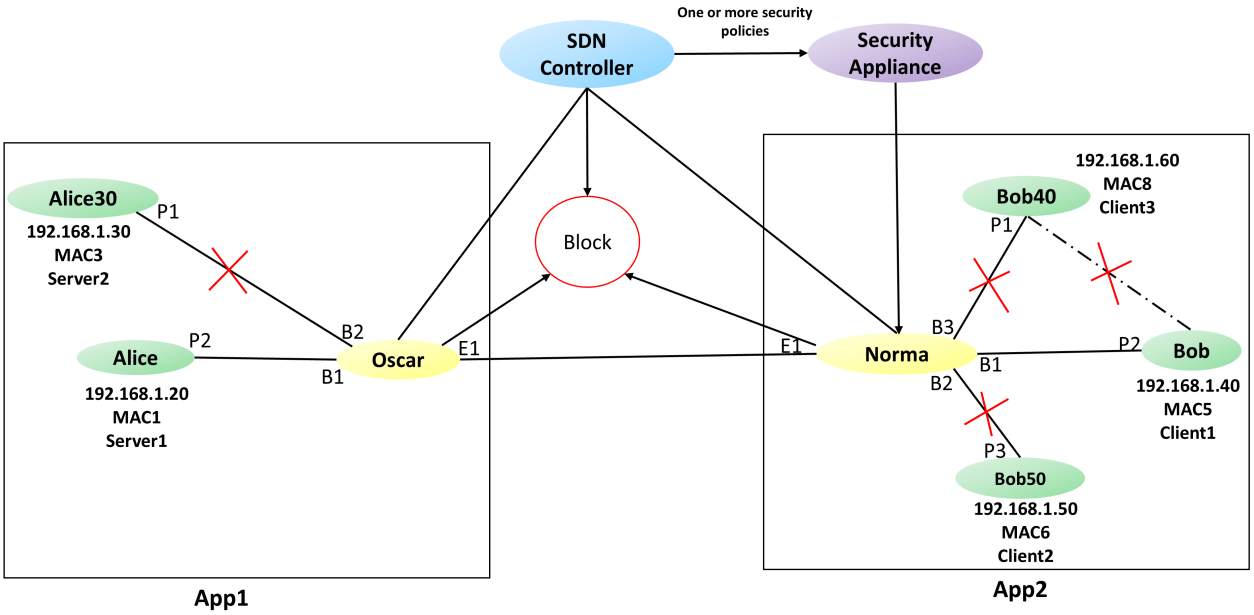

Figure 7: Low-Level network configuration after implementation of CSP

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

Figure 8: Verbose output

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

Figure 9: Matrix output

4 ANALYSIS OF CONFIGURATION FILE GENERATED BY FLOW CONTROLLER TO DETERMINE CONSISTENCY

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

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

Table 2: Flow table generated based on the configuration file.

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

now configured. The tool could be used to perform a formal analysis of the correspondence or to query and test the setup.

We used the SEL-5056 Flow Controller and SEL SDN switch to establish connection between the client and the server in order to create a backup file. We also defined several routes or paths to describe the communication flow between the client and the server. This sort of communication is supported by most OpenFlow switches, and the configuration backup file generated is vendor-specific. The simple ICS server application we configured is shown in Fig 1. We have two applications, App1 that receives and responds to request from a client application App2, running on separate machines. This high-level representation of authorized communication can be refined by mapping the abstract machine name to the network address of the device hosting the application and the transport of the communication [8]. For this example, we used TCP/IP communication; however, MODBUS, DNP3, NTP, GOOSE, etc. can be used. The communication between App1 and App2 can be defined as a Flow across the ICS network. This is one of the several flow rules we defined for the communication between App1 and App2.

Table 2 depicts the output of the configuration file, which contains information about flow rules generated by the SDN controller for our example ICS network. This summarizes the overall configuration of the network at a high level by displaying information like Source Name, Destination Name, Source MAC, Destination MAC, Source IP, and Destination IP, Flow type, and Policy Type. Next using Table 1 as reference, the tool maps the flow rules from Table 2. This mapping was created using a Java prototype that integrated with a SQLite database. To start, we queried the database to collect the flow rules for both ISP and CSP. In ISP flow table, the rows and columns specify the actual details applied across networking devices, while in CSP flow table, the low-level version of high-level rules is presented. The query's results are then saved separately as Java objects. Following that, Java Data Structures are used to map these entities. The tool produced verbose output for mapping between ISP and CSP (outlined in Fig 8), as well as matrix output for determining how many policies from ISP are in line with CSP (presented in Fig 9). We can see from the output that there are additional policies enforced in addition to the ones listed and some policies specified are not enforced, implying that the configuration file is inconsistent. The network administrator can use this tool to identify any additional policies that have been applied and dynamically remove them due to the programmability of SDN switches and safeguard the network. Now, the network/security administrator will be able to monitor the policies and check if they are implemented as defined.

5 CONCLUSION

SDNs have benefits and drawbacks; when combining them with OT systems it creates a productive and efficient environment for managing complex systems via policies. In this study we automated the consistency check process by analyzing the backup configuration file and high-level policy document. As a result, the overall process for evaluating the configuration file generated by SDN controller to ensure that it complies with the organization's policies is described. First, the tool takes the controller's configuration file as input and extracts the low-level flow rules, and stores them in a database. Next, it takes the scenario document as input and translates the high-level policies into low-level flow rules. Lastly, it maps ISP and CSP using the bisimilarity property to decide whether they are consistent or not. This work forms the basis for network administrators to monitor for security misconfigurations across networking devices. Although we only looked at a single controller's backup configuration file, we want to develop a formal methodology for evaluating the consistency of numerous backup configuration files generated by different controllers in the future. This will aid network administrators in detecting and mitigating misconfigurations throughout the complex ICS network at high-level.

ACKNOWLEDGMENTS

This work was supported in part by donations from Schweitzer Engineering Laboratory.

REFERENCES

[1] Nikos Bizanis and Fernando A Kuipers. 2016. SDN and virtualization solutions for the Internet of Things: A survey. *IEEE Access* 4 (2016), 5591–5606.

[2] Juan Camilo Correa Chica, Jenny Cuatindioy Imbachi, and Juan Felipe Botero. 2020. Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications* (2020), 102595.

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

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

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

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

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

[8] Sandeep Gogineni Ravindrababu and Jim Alves-Foss. 2020. Automated Detection of Configured SDN Security Policies for ICS Networks. In *Sixth Annual Industrial Control System Security (ICSS) Workshop*. 31–38.

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

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

[11] Pascal Thubert, Maria Rita Palattella, and Thomas Engel. 2015. 6TiSCH centralized scheduling: When SDN meet IoT. In *2015 IEEE conference on standards for communications and networking (CSCN)*. IEEE, 42–47.

[12] Varsha Venugopal, Jim Alves-Foss, and Sandeep Gogineni Ravindrababu. 2019. Use of an SDN Switch in Support of NIST ICS Security Recommendations and Least Privilege Networking. In *Proceedings of the Fifth Annual Industrial Control System Security (ICSS) Workshop*. 11–20.

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