

Techniques for Enhancing Security in Industrial Control Systems

VIJAY VARADHARAJAN, Advanced Cyber Security Engineering Research Centre, The University of Newcastle, Australia
UDAY TUPAKULA, Advanced Cyber Security Engineering Research Centre, The University of Newcastle, Australia and
University of New England, Australia
KALLOL KRISHNA KARMAKAR, Advanced Cyber Security Engineering Research Centre, The University of Newcastle, Australia

Increasingly Industrial Control Systems (ICS) systems are being connected to the Internet to minimise the operational costs and provide additional flexibility. These control systems such as the ones used in power grids, manufacturing and utilities operate continually and have long lifespans measured in decades rather than years as in the case of Information Technology (IT) systems. Such industrial control systems require uninterrupted and safe operation. However, they can be vulnerable to a variety of attacks, as successful attacks on critical control infrastructures could have devastating consequences to the safety of human lives as well as a nation's security and prosperity. Furthermore, there can be a range of attacks that can target ICS and it is not easy to secure these systems against all known attacks let alone unknown ones. In this paper, we propose a software enabled security architecture using Software Defined Networking (SDN) and Network Function Virtualisation (NFV) that can enhance the capability to secure industrial control systems. We have designed such an SDN/NFV enabled security architecture and developed a Control System Security Application (CSSA) in SDN Controller for enhancing security in ICS by achieving real time situational awareness and dynamic policy-driven decision making across the network infrastructure. In particular, CSSA can be used for establishing secure path for end-to-end communication between devices and also deal against certain specific attacks namely denial of service attacks, from unpatched vulnerable control system components and securing the communication flows from the legacy devices that do not support any security functionality. We also discuss how CSSA provides reliable paths for safety critical messages in control systems. We discuss the prototype implementation of the proposed architecture and the results obtained from our analysis.

CCS Concepts: • **Security and privacy → Systems security**; • **Computer systems organization → Embedded and cyber-physical systems**; **Distributed architectures**; • **Networks → Cyber-physical networks;**

Additional Key Words and Phrases: Control system security, software defined networking, network function virtualisation, security architecture, security attacks

6

Authors' addresses: V. Varadharajan, Advanced Cyber Security Engineering Research Centre, The University of Newcastle, EAG03d, University Drive, Callaghan NSW 2308, Australia; e-mail: vijay.varadharajan@newcastle.edu.au; U. Tupakula, University of New England Elm Avenue, Armidale NSW 2351, Australia and University of Newcastle, ES228, University Drive, Callaghan NSW 2308, Australia; e-mails: uday.tupakula@une.edu.au, uday.tupakula@newcastle.edu.au; K. Krishna Karmakar, Advanced Cyber Security Engineering Research Centre, The University of Newcastle, ES231, University Drive, Callaghan NSW 2308, Australia; e-mail: KallolKrishna.Karmakar@newcastle.edu.au.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
2378-962X/2024/01-ART6 $15.00
https://doi.org/10.1145/3630103

ACM Reference format:
Vijay Varadharajan, Uday Tupakula, and Kallol Krishna Karmakar. 2024. Techniques for Enhancing Security in Industrial Control Systems. *ACM Trans. Cyber-Phys. Syst.* 8, 1, Article 6 (January 2024), 36 pages.
https://doi.org/10.1145/3630103

1 INTRODUCTION

Control systems are complex infrastructures consisting of several interconnected heterogeneous devices including sensors, actuators, master terminal units, remote terminal units, **programmable logic controllers (PLCs)**, and communication devices [6]. Control systems such as electricity generation and supply systems, gas supply systems, logistics, manufacturing and hospital systems are considered as critical national infrastructure. The operational technologies that support critical infrastructure depend heavily on IT systems for their monitoring and control. Traditionally, physical separation of critical control systems from enterprise IT systems was used as the primary means for achieving security.

Modern control system architectures as well as business needs and cost control measures have resulted in an increasing integration of control and enterprise IT architectures. Physical separation alone no longer provides a viable business option for managing, utilizing, or securing ICS. There has been an ever increasing threat to these **Industrial Control Systems (ICS)** systems as their infrastructure components can have a long service (preventing security updates) and the need to respond in real time, as well as their evolution from isolated into Internet connected environment.

Industrial Control Systems form a critical part of national infrastructures, and hence any compromise or disruption to such systems can lead to significant economic losses as well as putting human lives under threat. Attacks such as Stuxnet [31], Duqu [7] on nuclear power plants and ransomware attacks (such as NotPetya and WannaCry [1]) to hospital networks highlighted the vulnerability of the critical infrastructures to cyberattacks. The report in [21] presents an overview of different attacks targeting critical infrastructures from the year 1993 to 2017. Hence, such infrastructures need enhanced security and monitoring services to defend them from cyberthreats and malicious adversaries.

Currently there is a considerable research interest in this area and there have been many publications in different areas related to ICS security. For instance, some researchers have investigated threat and attack surface [18, 37, 50, 54] of critical infrastructures; some have focused on the analysis of attacks on specific critical control systems such as power plants [7, 31], healthcare [1] and autonomous vehicular networks [10]. Some researchers have highlighted the enforcement of specific security policies and mechanisms in ICS [3]. Several techniques such as [19, 25, 39, 45, 48] have been proposed for detecting attacks and for forensic analysis [60] in ICS. Recently, there has also been a considerable interest in the use of emerging technologies such as **Digital Twins (DT)** [2, 13, 17, 24, 35], Software Defined Networking [12, 14, 28, 49, 55], Network Function Virtualisation [11, 19, 46, 49], Cloud [45, 55], Edge Computing [55], Fog Computing [59] Machine learning [25, 29, 30, 48] and Blockchain [33] for enhancing security in ICS. However, the increasing number of attacks targeting ICS confirms the lack of availability of robust solutions to deal with cyber attacks to make ICS more resilient. This underlies the significant challenges faced in securing ICS systems whose infrastructure is often spread over large geographical areas, consisting of multiple domains managed by different administrative authorities, lacking visibility for real time situational awareness and for secure configuration of policies. In this regard, SDN, NFV and DT technologies provide specific mechanisms to secure critical infrastructures. For instance, the SDN Controller having an overall visibility over its network can help to achieve better situational awareness with respect to detecting attacks. NFV can provide dynamic enforcement of

security policies. DT enables representation of holistic critical environment using virtualisation technology for researchers to visualize and analyse the impact of attacks and use data analytics to predict the consequences of ongoing attacks. However, these technologies also have their own limitations. For instance, the overall network view available at the SDN Controller can itself be vulnerable to cyber threats. Issues such as real time updates and single point of failure can affect the availability of the SDN Controller. Dynamic placement of security functions using NFV can itself increase the security vulnerabilities, and DT can also result in attacks being transferred from DT to ICS networks. Hence, the focus of this paper is to develop a robust security architecture making a systematic use of these technologies for enhancing security in the ICS.

The main aim of this paper is to develop a secure software enabled architecture for protecting control systems from cyberattacks. Our architecture consists of a **Control System Security Application (CSSA)**, which makes use of SDN [15] and NFV [22] technologies for enhancing security in control systems. SDN enables programmable networks and provides separation of control and data planes, allowing management protocols to be separated from the data traffic. The control plane consists of a logically centralised Controller (which can be distributed in practice), and has native applications for management of network devices. Data plane consists of both physical and virtual network devices and implements protocols for forwarding traffic based on flow rules as well as protocols for (e.g., OpenFlow) for communicating with the control plane. NFV allows softwarisation of network functions enabling the provision of network services as **Virtual Network Functions (VNFs)**. Hence, NFV enables the critical network operators to deploy security services in a dynamic fashion.

Such a software enabled architecture enables secure dynamic policy-based decision making, which is particularly suitable for securing control systems. Control systems are autonomous decision-making agents making real-time decisions. Real time decision making requires the capability to make dynamic decisions to ensure higher availability of services which is an important requirement in control systems. The software enabled architecture proposed in this paper is able to define fine grained flow-based security policies for ensuring the availability of critical control subsystems. Another design issue is the inability to make updates to certain critical components which in turn can make them vulnerable. The proposed architecture can make such unpatched (vulnerable) components to be accessed in a secure manner, only via secure flows, and only from authorised users and devices, thereby protecting them from attacks. Furthermore, traditionally control systems were designed to operate in closed environments, which meant that often they do not possess much (or any) security functionality. However, with the increasing use of Internet protocols such as IP for the management of control system components, they are susceptible to various attacks. The proposed security architecture provides the capability to secure the flows from these legacy devices that do not support the required security functionalities.

The main contributions of our work are as follows:

— Design of SDN, NFV and DT based security architecture for enhancing the security in control systems
— Real time situational awareness mechanisms for enhancing the availability of SDN Controller and critical services in control systems
— Mechanisms for establishing secure and reliable communication paths in control systems
— Secure access to unpatched services and preventing access to control system from unauthorised devices
— Security functions for continuous monitoring providing the capability to detect attacks and for dynamic generation of policies for securing vulnerable services in control systems

— Proof of concept for demonstrating the performance of our security architecture against different attacks

The novelty in our approach comes from the secure integration and control and data planes providing dynamic control of paths and flows in the control system infrastructure.

For instance, this provides the capability to deliver packets related to critical operations even when there are failures in networks due to attacks. CSSA is designed to enhance the reliability of the end-to-end communication paths supporting critical operations such as alarms and circuit breaker operations in control systems. CSSA implements two proactively configured routes with the required **service level agreements (SLAs)** to support flows related to safety critical operations in a control system. By default, switches make use of the primary data path for forwarding the safety related flows between the devices. The secondary data paths are used as backup, in case of failure of any communication links in the primary path. Note that there is no need to alert the Controller since secondary paths are dynamically configured by the CSSA.

Now consider the case where there is a failure of the primary and backup data path at a switch. In this case, the switch forwards the complete safety critical message to the Controller. CSSA then determines the source and destination addresses related to this flow and uses the control plane to directly forward the message to the egress switch. Note that in this case there is no need to determine and establish the routes by initiating flow_mod messages to all the switches. Hence, our approach is able to meet the stringent time requirements for critical operations.

Another novel characteristic of our architecture is that it can be used to generate dynamically policy rules based on the attack analysis in a DT environment for securing vulnerable devices in the ICS.

The paper is organised as follows. Section 2 considers a multi domain ICS environment and presents the attacker model. Then we discuss the requirements for enhancing security in control systems. In Section 3 we propose a software enabled security architecture for control systems based on SDN and NFV technologies. We first present a high level overview of the CSSA and then describe the important components of CSSA. We also discuss the specific security functions that are invoked in the switches to deal with the attacks from end hosts. In Section 4, we discuss how our proposed architecture satisfies the design requirements described in Section 2. Section 5 presents the implementation of the CSSA and describes how it deals with different attack scenarios. Then we present the performance results associated with these scenarios. Section 6 considers some relevant related works, and finally Section 7 concludes the paper.

2 ICS SCENARIO AND ATTACKER MODEL

In this section, we provide an overview of the multi domain ICS scenario with heterogeneous devices and outline the attacker model. Then we discuss the requirements that have been considered in the design of our security architecture.

2.1 Multidomain ICS Scenario

We consider a multi domain ICS environment with a Domain Controller for maintaining security policies for the different components in the domain. The Domain Controller is used as **Policy Decision Point (PDP)** for making decisions related to the enforcement of security policies on heterogeneous devices in the multi domain ICS environment for counteracting different attacks. The **Policy Enforcement Points (PEP)** can be implemented in the Domain Controller or on the heterogeneous devices.

Our architecture makes use of SDN and NFV technologies for the enforcement of security policies in the ICS. Let us consider a simple scenario shown in Figure 1, where a single SDN Controller cluster is used for managing the ICS domains such as control station, multiple process domains

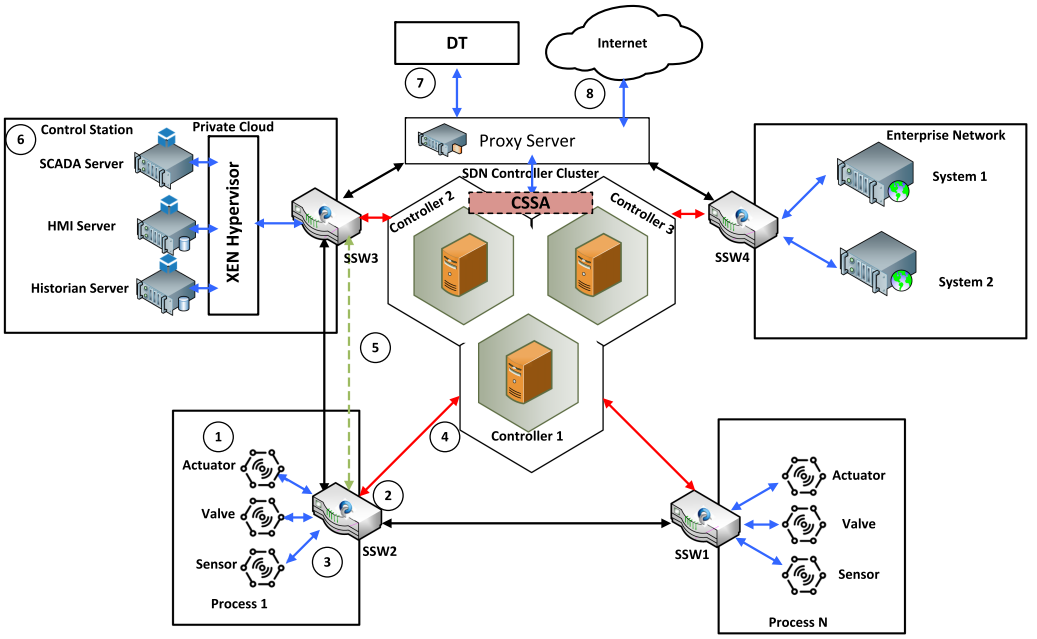


Fig. 1. ICS scenario.

and the enterprise domain. We also make use of DT for replicating critical components of the ICS in the virtual environment for attack analysis and dynamic generation of security policies. Synchronisation between the DT and the ICS domain is achieved via one way information flow from the ICS to the DT environment through a proxy. To minimise the risks of attacks from DT environment to the ICS domains [2], the information flow from the DT to ICS is only allowed via manual intervention by the SDN Controller administrator.

The control station is used for monitoring multiple processes in the control system. There can be different servers such as a SCADA server, HMI Server, Historian Server for monitoring and controlling different processes in the ICS. The services in control station domain are implemented in a private cloud to achieve higher availability of critical services. Each process can consist of several field devices such as MTU/RTU, actuators, valves and different sensors such as temperature, pressure, accelerometers and cameras. The enterprise network consists of several client and server machines. SDN switches are used for interconnecting different devices in each logical domain and a proxy (security gateway) is used for interconnecting different logical domains. As shown in the Figure 1, there is no direct communication link between the enterprise domain and the other ICS domains. The SDN administrator configures rules to permit selectively authorised users in the enterprise domain to access the systems and devices in the ICS domains. The only communication path between the systems in the enterprise domain and the other ICS domains is through the proxy.

The SDN Controller is used for managing the complete network. Our architecture supports the usage of a single Domain Controller for multiple domains as well as separate Domain Controller for each domain. For simplicity, in this paper, we have shown a single Domain Controller for managing multiple domains of the ICS.

2.2 Attacker Model

There can be a range of attacks exploiting vulnerabilities in heterogeneous devices, processes, services and control systems belonging to different ICS domains. For instance, there can be

different hardware, firmware and software related attacks targeting different domains in the ICS. The hardware of devices such as PLCs and RTUs can be exploited to generate different attacks such as fault injections, logic reverse engineering, intellectual property theft and backdoors. The firmware. which resides between the hardware and software of the devices (such as smart meters, access points and servers) can be maliciously altered or overwritten to change the ICS processes or cause service disruption. Software vulnerabilities such as buffer overflow, SQL injection or weakness in the authentication systems such as the use of default or hard-coded passwords can be exploited to inject malware or take control of the systems (e.g., for demanding ransom).

At the network layer, there can be different attacks on devices such as modems, routers and firewalls. There can also be attacks targeting the weaknesses in communication protocols such as clear text communications to perform malicious actions such as spoofing and eavesdropping, as well as scanning the network and generating flooding to interrupt time critical operations. Attackers can propagate the attacks to different ICS domains where there is a lack of segmentation or weakness in the isolation mechanisms between the domains. For instance, attackers can make use of phishing attacks to target the users in enterprise domain and then use these systems to spread the attacks into ICS domains. Also, there can be malicious users who can exploit BYODs to generate attacks in different domains. Finally, there can be different attacks targeting the process layer such as injecting incorrect information or spurious messages to degrade the performance of the process or to halt the process potentially resulting in safety breaches.

The ability of the attacker to generate sophisticated attacks depend on the attacker's knowledge of the system. The greater the knowledge of the attacker, the greater the attacker's ability and the greater the probability for the attacker to cause adverse impact. For instance, denial of service attacks jamming the networks and preventing data from reaching the intended destination can be carried out by an attacker without any prior system knowledge. However, these attacks can still have significant impact on the critical services being provided. Flooding attacks can delay communications related to time critical messages causing a significant impact on the operation of critical services. Similarly, replay attacks may not need system knowledge. On the other hand, the attackers need to have system knowledge for generating attacks such as false-data injection and ransomware. Then there are attacks where the attacker not only has the system knowledge but also has access to network channels leading to more sophisticated data injection attacks which can be difficult to detect. In terms of resources available to the attacker, if the attacker is able to gather sequences of data from the system, even though the physical system remains unaffected, they can be used to develop more sophisticated attacks, such as when and what to replay to cause damage. In terms of disrupting the system functions, the way a particular attack affects the system operation depends on both on the system itself (e.g., where the attack is happening) as well as on the nature of the attack. A physical attack directly affects the dynamics of the system whereas a cyberattack can affect the system due to the interconnection between the cyber and physical components. So there is a need to address attacks related to both cyber and physical components.

Let us now briefly summarise the various attacks addressed in our architecture.

— Disruption attacks: This is where the attackers compromise one or more devices in the control system and use them for flooding the network with junk messages causing delays to safety critical messages thereby potentially impacting the availability of critical services in the control system. For instance, the Slammer worm resulted in the shutdown of the nuclear power plant.
— Unauthorised devices: Attacks using unauthorised devices access causing damage to communications and other devices in the control system. For instance, employees using unauthorised BYOD to access the control system.

— False data injection attacks: These attacks alter the control parameters and sensor readings injecting false data that can result in safety breach events in the critical infrastructure.
— Malware attacks: The attackers can disable the security tools in the servers hosting critical services and install backdoors such as rootkits with hidden processes.

2.3 Security Requirements for Control Systems

In practice, there are many requirements when it comes to securing an ICS and it is beyond the scope of this work to take into consideration all of them. So without claiming completeness, we discuss the requirements that we have considered in the design of security architecture for enhancing security in control systems.

— **R1: Higher Availability Requirements for Control systems**
Control systems have higher availability requirements compared to systems in enterprise networks. Even without attacks, real-time data exchanged within networked control systems can suffer from communication constraints such as link failures and congestion. Attacks such as **Denial of Service (DoS)** can have significant impact on the availability of control systems. Hence, one of the critical security requirements is the need to ensure that control systems continue to function when under attack. Control system operators need continuous supervision and visualization of the entire network for operational monitoring and management. To achieve this, it is necessary to understand the system flows and the expected behavior, and identify when behaviors change, and have the tools and training to know what needs to be done to get the system back to normal operating conditions. Hence, there is a need for techniques that can make real time decisions for higher availability of services in the control systems. Also, the SDN Controller can be a single point of failure which can have a major impact on the availability of ICS. Hence, there is a need to protect SDN Controller from attacks.
— **R2: Restricted Access to Unpatched Systems to Minimise Risks**
There will be issues related to applying security patches to systems and devices in the control systems. In some cases, there can be several legacy devices and/or services and/or applications that do not have any support for patches by the vendors. Furthermore, some of the operators hesitate to promptly apply the patches as it may require several days of advanced planning and/or suspend some of the critical operations to apply the patches. There have been cases where applying security patches has impacted the operation in critical control systems. For instance, a nuclear power plant was accidentally shut down [27] as patching the system resulted in resetting the control system data by rebooting the system. This caused the safety systems to incorrectly interpret the data reset event as a drop in water levels of the reservoirs that cool the plant's radioactive nuclear fuel rods and shut down the power plant. However, the provisioning of services on unpatched servers open the door to different types of attacks. Hence, there is a need to ensure accessibility to the unpatched systems by authorised users while protecting them against attacks.
— **R3: Securing Flows between the End Hosts**
Initially, ICS protocols were designed and constructed to operate as a closed system. So physical security alone was considered as an important factor for securing control systems. Even basic security services like authentication and encryption for device communication was not considered as a design requirement. So the absence of security features and services in these protocols were overlooked due to closed deployment of ICS infrastructure. Raw ICS communication or unencrypted ICS flow communication is prone to **Man-in-The-Middle (MiTM)** attack where an adversary can eavesdrop and inject malformed information. This can lead to

total disruption of the whole system. Hence, techniques are needed to secure the flows from the legacy and resource-constrained devices that lack security mechanisms and services. Recently, we are experiencing slight improvements as ICS protocols are stacked into IP, empowering Internet-enabled management of ICS Controller [41]. Like any other communication over the Internet, ICS communication requires basic security mechanisms or services.

— **R4: Reliable Paths for Safety Critical Messages**
The ICS processes have a wide range of time-related requirements, including very high speed, consistency, regularity, and synchronization. Some systems may require the computation to be performed as close to the sensor and actuators as possible to reduce communication latency and perform necessary control actions on time. It is important to meet these time-related requirements when designing security for the ICS.

— **R5: Prevent Access to ICS Domains using Unauthorised Devices**
With the employees potentially using their own BYOD to access different systems and services, it can pose huge risks to the provision of critical services in ICS domains, as the users could have full administrative control on their devices which may not comply with the security policies of ICS domains. So there is a need to make the users aware of these risks and possibly train them for ensuring their BYOD compliant with the ICS security policies.

— **R6: Isolate Malicious Entities in the ICS Domain**
As the attack surface for ICS can be large, attackers can exploit vulnerabilities in the critical services to disable the security tools and perform different malicious actions and hide their state from the Controller. For instance, attackers can install backdoors such as rootkits with hidden processes, encrypting files and demanding ransom. Also, the attackers can alter the control parameters and override the safety features to operate the devices in unsafe manner. For instance, Stuxnet worm manipulated the parameters of a set of centrifuges used for uranium enrichment which resulted in the operation of centrifuges at dangerous speeds. So there is a need for techniques for detecting these attacks.

3 CONTROL SYSTEM SECURITY APPLICATION

In this section, we will first discuss the assumptions made in our architecture and then present its overview. We will describe the operation of the proposed architecture and the specific security functions to counteract attacks in the control systems.

— We assume that the Controller in the SDN domain is not compromised. This is a common assumption in SDN security architectures. However, in our approach, we have discussed the use of clustering techniques for enhancing the availability of the SDN Controller and securing the Controller against attacks by making use of NFV for enforcing policies to protect the SDN Controller form malicious end hosts.
— We use the term end host to represent a client or server or field device. We assume that end hosts are connected using SDN switches. Hence, any communication between the end hosts pass through the switches.
— A flow is used to represent communication between the end hosts. A flow can be a single packet or sequence of packets between the end hosts.

3.1 Overview

Control Systems are complex environments with several devices, processes, services and applications that are designed to operate in a vast variety of locations and situations including mobile conditions. We have developed a CSSA, which uses SDN and NFV technologies to achieve real-time

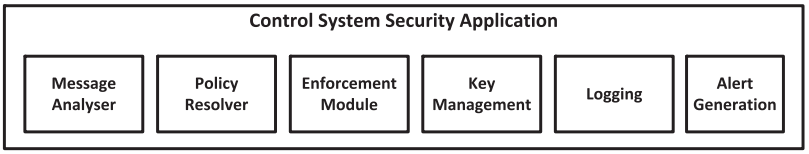


Fig. 2. CSSA components.

situational awareness to enhance security and also deal with attacks effectively and efficiently in critical control system infrastructures. The CSSA uses SDN technology to achieve robust situational awareness and dynamic policy-driven decision making across the network infrastructure. The SDN paradigm helps to achieve situational awareness due to two essential characteristics: an overall topology view of the network and the ability to create programmable data routing paths. However, attackers can perform different attacks to corrupt the overall topology view at the Controller. So there is a need to enforce different security mechanisms to ensure that the Controller topology view available at the SDN Controller can be trusted. Otherwise, it is possible for the attackers to corrupt this view. Hence, CSSA first makes use of NFV to enforce different security functions to prevent attackers from corrupting the overall view available at the Controller. Then CSSA makes use of the Controller topology view available at the Controller for establishing real time situational awareness in the control system network. This will enable the CSSA for secure monitoring and the ability to deal with attacks for enhancing the availability of control system networks. One of the main advantages with our approach is enforcing the security policies at the switching level, as the attack is detected and prevented nearest to the malicious adversary or end host (i.e., from where it originated).

We have designed and developed CSSA for enhancing security in control systems. Figure 2 shows the important components of the CSSA which is designed to run as an application on SDN Controller. The main components of the CSSA are: Message Analyser, Policy Resolver, Key Management, Policy Enforcement, Logging and Alert Generation. Message Analyser is used for analysing the incoming request to extract information such as source address, destination address and flow type and forwarding the information to policy resolver. Policy Resolver is used for storing the policies related to all the devices in the domain. It makes use of the information provided by the message analyser to query its database and determine the appropriate policies. The Enforcement Module derives the mechanisms for enforcing the policies determined by the Policy Resolver. The Key Management module is used for generating keys for secure communications. The Logging module is used for logging all the messages transferred between the Controller and the CSSA. Alert Generation is used to raise alerts to the Security Administrator when attacks are detected in the control system network.

We have used a modular approach in the design of CSSA to support distributed implementation of modules on multiple Controllers. NFV is used for implementation of specific security functions at the security gateway and/or switches. However, the specific security functions that can be implemented on the switches depends on the features available in the switches. If a switch is not suitable for enforcing specific security function, then it can be implemented in the other devices (such as other switches or security gateway) which support this functionality and the flows are dynamically routed to pass through these devices.

Figure 1 presents a programmable networked ICS network infrastructure. Here the routing plane consists of programmable **SDN Switches (SSW)**. The Controller can be single point of failure. There can be different attacks [57] targeting availability confidentiality and integrity of the Controller. Since support for TLS is mandatory for SDN Controller and the switches, this can help to deal with the confidentiality and integrity related attacks. Higher availability can be achieved

by implementing the Controller on a server cluster. Furthermore, techniques such as Primary-Secondary Strategy, Database Centric Failover, Election Based Consensus Systems, and Broker Based Controller Resiliency can be used [28] for making Controllers resilient against the attacks.

Our current implementation makes use of ONOS SDN Controller which has inbuilt functionality to support higher availability by implementing it as a cluster. We are utilising this feature of ONOS to deal with availability attacks at the Controller and Application levels. Also, we use fine-granular policies to deal with unauthorised/unauthenticated network asset usages, which helps to deal with attacks like improper exception handling and external dependency (and the like) type of attacks. Then there are vulnerabilities arising from other malicious applications in the Controller and also from different devices in the data plane. We assume that techniques such as Cross-App poisoning [52] are used for differentiating the applications based on RBAC policies to prevent tampering, injection and app/script/code manipulation attacks on the Controller by malicious applications. We assume that CSSA runs with highest role privileges on ONOS Controller. Furthermore, CSSA makes use of NFV to enforce preliminary policies at the switches to deal with the attacks from malicious end hosts targeting attacks on the Controller and also preventing topology poisoning attacks at the Controller.

We make use of SDN Controller implementation on a cluster as it helps to achieve load balancing and deal with the single point of failure problem. Also, the cluster based approach helps in creating security-critical network containers, network recovery, and support real time updates to the core network OS and northbound applications without stopping the Controller. For instance, the cluster implementation of SDN Controllers has an inbuilt feature to support real time updates without losing availability of their services. This update feature takes a SDN Controller into the maintenance mode by moving its services to another controller in the cluster. Now the updates are installed on this Controller in the maintenance mode. If the update requires restart, the Controller in the maintenance mode can also be restarted. Now the updated Controller can rejoin the cluster and its services can be restored. The process can be repeated to update other SDN Controllers in the cluster.

The SDN Controller Cluster controls the SDN switches using northbound applications. The Controller cluster sits behind a proxy server and hence is not reachable by the adversary from the internet. The DT [2] makes use of virtualisation to replicate critical services and network asset within the ICS network infrastructure. Unlike traditional digital twin, which logs the data and energy level of the devices (sensor and actuators), our DT abstraction focuses on logging the network assets' security status. Here, we have two types of DT representation, one specific to the Servers/Controllers and the other to the switches and processes in ICS. The Controller/Sever DT keep a log of its state at any point in time. The internal and external state of the asset represents the state. The internal state is presented by the operating systems state and the applications/processes running in it at any time. The external state represents the I/O and network communication with the external assets. Similarly, the ICS process and switches have internal and external states. Any changes in them would be reflected in the DT entities.

An Enterprise Network consists of the ICS employees' system. The Proxy server provides internet services to the Enterprise Network. There is default deny for communication between enterprise systems and devices in ICS. The administrator can permit selective communication for authorised users from the enterprise systems to the ICS through the proxy.

Now, we will present how the information flows within our programmable ICS network infrastructure between the devices in process domain and services in the control station domain.

— Step - 1: The ICS processes are provisioned in the ICS programmable network infrastructure. They can be automatically or manually provisioned. During the provisioning phase,

the safety operation and control parameters of the devices are determined from the device specification and ICS processes are checked in test-bed environment and we have created a clean state for them. The DT holds this information.

— Step - 2: CSSA is implemented as an application on the SDN Controller enforces preliminary security policies (such as source address validation and thresholds) at the switches using NFV to validate the flows generated by the devices before generating Packet-In message. The OpenFlow switches can be either physical or virtual. Our network setup consists of both of them. For instance, we have both Raspberry Pi and Zodiac FX based hardware switches. We are using OVS as the software switch. These SDN Switched (virtual/physical) are integrated into a software service that can be physically used on the ICS network infrastructure. They are also helpful in providing certain security services, such as flow isolation, IDS, IPS and so on. We have explained these features in Section 3.3.

— Step - 3: The field devices in process domain send updates of sensor reading to the servers in Control Station. The switches validate the data packets from the devices based on the preliminary policies.

— Step - 4: Valid data packets from the devices will result in generation of Packet-In message to the Controller which are subsequently forwarded to the CSSA. Invalid data packets such as with spoofed source address or packets exceeding threshold will be dropped and an alert is raised to the SDN Controller.

— Step - 5: CSSA analyse the Packet-In message and makes use of the overall topology view available at the Controller to extract the required information and determines the relevant policy for permitting or denying the flow. If CSSA makes a decision to permit the flow, it makes use of the forwarding application in the SDN Controller Cluster to determine the route for the data packets from the ICS processes to the Control Stations. Sometimes, the process's raw data are also logged into the Private Cloud. Now CSSA initiates a command to configure the switches to establish a secure communication path for sending the updates from process domain to the server in the Control Station domain.

— Step - 6: The Control Station holds various servers. The data packet is destined to the server with the destination address. The server analyses the data packet and sends control instructions to the ICS processes with the help of the SDN Controller Cluster core services.

— Step - 7: The Proxy Server acts as a firewall service, keeping the SDN Controller Cluster isolated from the Internet. This server uses VPN to separate the control instruction and data flow. In addition, this proxy server considers network asset state information as the control information and forwards them to the DT using a separate VPN network. This helps safeguard the network assets' state information from adversarial tampering.

3.2 Control System Security Application

We have developed the CSSA, which is hosted on an SDN Controller. The CSSA has security policies for all network devices in a given SDN based Control System domain. We use OpenFlow protocol for communications between the SDN Controller and the switches. We have also used JSON for the specification and communication of security policies between the CSSA and the switches. We have developed templates for the specification of security policies based on different parameters and conditions such as device type, time, location, source, destination, event and type of traffic. For instance, policies can specify access to services during specific time intervals and/or using certain devices from specific locations. Similarly, there can be different policies depending on the source address and/or destination address. Now let us discuss the operation of CSSA.

The CSSA configures preliminary security functions at the switches to protect infrastructure nodes from flooding attacks and also prevent malicious end nodes from corrupting the overall

topology view available at the Controller. This will enable CSSA to make use of the reliable overall topology view available at the Controller for developing robust situational awareness of the control system network.

When a switch receives a new flow from the field devices or end host, it first validates the flow request based on the preliminary security policies such as validating the source address of the packets and checks if the number of flow requests exceeds the threshold. If the flow meets the preliminary security policies, then the switch checks if the flow matches any of the flow rules to determine if there is an existing path for forwarding the flow to the destination node. If it matches with any of the flow rules, then the flow is forwarded based on flow rules. If the flow does not match with any of the flow rules, then the switch generates a Packet-In message to the SDN Controller which is subsequently forwarded to the CSSA.

The message analyser extracts information such as source address and destination address from the Packet-In message and forwards it to the Policy Resolver. The Policy Resolver uses the information provided by the message analyser to determine if the flow request is originating from a valid device and/or user and also if there are any specific policy requirements related to this flow. For instance, the flow can be using insecure protocol for communication with other devices or the flow may be related to safety critical message with specific operational requirements. Hence, the Policy Resolver queries its database to determine the related policy and forwards it to the Enforcement Module. The Enforcement Module is used for enforcement of the policies locally or dynamically invoking the specific functions at the switches in the data plane. For instance, there can be security policy which mandates secure communication if the field devices are making use of insecure protocols. In this case, enforcement modules initiate a request to the Key Management module to generate keys for secure communication and distribute the key to the corresponding switches (connected to the respective end hosts). Now the enforcement module configures the flow rules in the switches for forwarding the flow according to the security policy. The paths are established for the flows to pass through flow validation ( ) for continuous monitoring of the flows between the end hosts. For instance, the security policy can also mandate for continuous monitoring of the flows by matching with attack signatures and/or anomaly detection. The switches raise an alert to the CSSA if any of the end hosts start sending attack traffic after secure routes are established by the CSSA.

When CSSA receives an alert regarding a malicious end host, then the enforcement module can dynamically configure access control rules for isolating the malicious end host from the control system network and/or the alert generation component can raise an alert to the security administrator. The administrator can conduct an offline analysis to determine the seriousness of the event and decide to either isolate the suspicious end host from the network or restrict the end host communication depending on the availability requirements of the end host.

3.3 Security Functions for Switches

As already mentioned, the specific security functions that can be enforced on the switch depends on the resources and features available at the switch. In this section we will describe the important security functions that are implemented at the switches.

3.3.1 Local Store(). This **LS()** function is used for the temporary storage of information related to the end hosts that are connected to the switches. For instance, LS() is used for storing information such as end device fingerprints, attack signatures, whitelists, blacklists, behaviour profiles of the end hosts and audit records related to the incoming and outgoing traffic from the end hosts. If the end hosts are mobile devices, then CSSA updates this information to the access points during the handover process.

3.3.2 Flow Differentiator(). There is a need to differentiate between the flows to meet the specific operational requirements of the control systems. For instance, reactive path establishment is not suitable for safety critical message in control systems. This function is used to differentiate between the flows to enable secure and safe operation of the control systems.

3.3.3 Flow Validation(). **FV ()** is used for protecting the control system infrastructure elements such as switches and the Controller from attacks and it enables CSSA to establish robust situational awareness in the network. For instance, the end hosts can generate different attacks such as flooding the switches and/or the Controller and also poison the overall topology view available at the Controller. For instance, CSSA depends on the overall topology view available at the Controller for establishing situational awareness in the control system network. Attackers can make use of different techniques to poison the overall view available at the Controller. For instance, the attackers can poison the overall topology view available at the Controller by sending a malformed OFPT-Packet-In request to change the Host Tracking System Table which is used for storing the host and switch topological view at the Controller. Hence, FV() is used for enforcing preliminary policies such as validating the source address of the packets originating from end hosts and using thresholds to prevent end hosts from flooding the switches or SDN Controller.

The FV() is also used in the validation of incoming and outgoing flows from all the end hosts connected to the switches. It makes use of source address validation, signature and anomaly based techniques for detecting attacks. Validating the source address prevents an end host from injecting malicious messages and generating attack traffic with spoofed identities. The signature and anomaly based techniques are then used to deal with the attacks that are generated with the correct source address. The specific security functions that can be invoked in the switches depend on the functionality of the switch. For instance, source address validation, attack detection using signatures, whitelists and blacklists can be implemented on most of the switches. Anomaly detection using machine learning techniques depends on the resources available at the switch. If there is a need for using machine learning techniques for detecting attacks at the switches which do not have the required resources, then CSSA can configure the switches to replicate or redirect the flows through the security gateway. Currently, we are working on developing functions for making use of machine learning techniques for detecting attacks at the security gateway.

3.3.4 Flow Encryption (). In control systems there can be several legacy and resource constrained devices that do not support any security functionality or protocols. For instance, several protocols such as Modbus, Profibus, DNP3, and IEC 61850 exist for communication in ICS systems. However, Modbus which is inherently insecure and does not provide security against confidentiality integrity or availability of the data transmitted using the protocol is the most widely used protocol in ICS. Some of the reasons for wide deployment are due to openly published protocol, ease of use and speed of transmission. These characteristics led to the wide adoption of the protocol, which has inevitably created major security issues for these networks. Although TLS security extensions have been proposed for the Modbus, most of the control systems still use older versions of the protocols which do not make use of TLS security. Our recommendation is to make use of TLS for secure communication if it is supported in the ICS.

However, there are issues that limit the usage of secure Modbus in some ICS. For instance, secure Modbus operates over port 802 whereas Standard Modbus TCP operates over port 502. This port separation prevents older devices not using secure Modbus from confusing the two protocols when communicating. So there is a need for securing the communication in the ICS, which does not support secure Modbus protocol. Hence, in these cases, security administrators can make use of this FE() for securing the communication between the legacy devices which do not support TLS. The FE component is used for securing the communications between the end hosts. If a new

flow is destined to devices/hosts that do not support security functionality, then the CSSA can enforce policy to encrypt the flow at the switch (which is connected to the source host) and to decrypt the flow at the switch (which is connected to the destination host). If the flow security policy mandates secure communication between the end hosts, the key management module in the CSSA will generate the required symmetric key for securing the flows and distribute the key to the relevant switches.

3.3.5 State Validation(). This component is used for detecting the attacks exploiting safety data types and control parameters of physical devices [18] in critical infrastructures. For instance, attackers can generate sophisticated attacks in critical infrastructures using false data injections, altering the sensor reading, tampering the sensors or disabling the communication links between the devices. The CSSA makes use of state validation to detect these attacks and maintain the integrity of the industrial process. We assume that safety data types and control parameters are derived from the device specifications, safety processes, and from analysis in the test bed environments.

Our current work involves the detection of suspicious events based on the information available from design specifications of the devices and the testbed environments.

4 DISCUSSION

In this section we will discuss how the specific requirements are achieved with our security architecture. As the Controller has the visibility of its network domain topology and devices, the CSSA uses this information to make real time decisions on communication between the end hosts based on their availability requirements. The traffic flows initiated by the end hosts are subjected to security policies in the CSSA in the SDN Controller of that domain. The source host could be any client machine or field device. The initial packet header from the source host is sent by the switch (to which this host is connected) to the SDN Controller which is subsequently forwarded to the CSSA. The header contains all the usual network and service parameters such as the source address, destination address, and the packet type. The CSSA extracts the relevant parameters from the incoming packets and uses the Policy Resolver to determine whether the communication or flow is permitted according to the policies configured by the security administrator. If the flow is permitted by any of the policies, then the path is established to enable communication between the end hosts. If the flow is not permitted by any policy then the flow is dropped. If the end hosts still continues to generate excessive flow requests then an alert is raised to the security administrator. This is a significant advantage as it helps to deal with attacks such as the outbreak of worms in critical control systems. For instance, the Slammer worm which randomly scanned for vulnerable machines for spreading the attack created severe congestion in the network and as well as disabling a safety monitoring system in a nuclear power plant for over five hours [44]. Using the proposed architecture, CSSA drops all the randomly generated malicious flow requests which do not satisfy policies permitting the flows. As the malicious flows are dropped at the source end, the impact on the bandwidth is minimized. This can be significant in certain control systems.

4.1 Higher Availability Requirements for Control Systems

The SDN Controller is implemented as a cluster to achieve higher availability. Also, the services in control station domain are implemented in private cloud to deal with the dynamic variations in the load, achieving higher availability against DDoS attacks and quick restoration of services in case of attacks on these services. There can be multiple process domains in the ICS that are controlled by Control Station domain. In case of attacks originating in any of the process domains, then the specific process domain is dynamically isolated to minimise the impact on the operation of other process domains.

The number of connections that can be handled by the servers depends on the application or hardware requirements. Since the critical services are hosted in private cloud, the resources allocated to the specific services can be varied to adapt to the dynamic variations in the legitimate load on the servers. However, if there is an outbreak of worms in the ICS, then there is also a need to deal with the malicious flows in addition to dynamically increasing the resources allocated to the services. In our architecture, the SDN Controller is used to limit the number of flows to the server depending on the server's capacity. The flow control can be enforced on a per flow, per device, per domain or per location. For instance, the attackers can take control of many devices in the ICS and use them to generate flooding attacks with spoofed sources address. This can completely block the ICS network. In the default configuration, the CSSA enforces two thresholds for all the devices in the ICS. The first threshold limits the total number of flows that can be initiated to any other end host. The second threshold is used to limit the number of flows that can be initiated to servers running critical services.

Now assume that there is a specific server providing a critical service and consider how the CSSA can help to achieve higher availability for that critical service. Assume that some end hosts in the control system are compromised by an attacker and used to generate flooding attacks on the server that is hosting the critical service. Without the proposed architecture, the attacker can flood the critical service severely impacting the availability of the service. Using the proposed architecture, the CSSA uses the network domain visibility to make decisions, in real time, on communications between the end hosts based on their availability requirements. If the critical server is capable of handling x flows/sec, then the CSSA restricts the flows to the critical server to less than x flows/sec. The dynamic policy driven approach in the CSSA allows the flows to be restricted based on a variety of parameters such as maximum number of flows per end host, maximum number of flows per switch and maximum number of flows per control process or logical domain.

4.2 Restricted Access to Unpatched Systems to Minimise Risks

As already mentioned our recommendation is to patch the system with the latest updates where ever possible. However, there can be a case where patching cannot be readily applied and there is a need for the services to be running and accessed by some of the control system operators. Now let us discuss techniques for protecting the unpatched server from attacks.

In this case, the flows to the unpatched server can be permitted only from selective hosts. Furthermore, the authorised users can only access these devices using devices that are registered with the CSSA. The security administrator can configure CSSA to permit flows to the unpatched server only from the authorised users with registered devices.

If any of the end hosts are making a flow request to the unpatched server, then the ingress switch forwards the flow request to the CSSA. The CSSA enforces a default deny policy for all flows to these services and provides restricted access to authorised users that are required to access these services. Furthermore, all the flows from the permitted hosts can be monitored for attack traffic and the malicious end hosts can be isolated if they are attempting to exploit the vulnerability of the unpatched server.

4.3 Securing Communication between Entities

Let us consider two specific aspects for securing communication between different entities.

First, there is a need to support secure communication between the CSSA/SDN Controller and the switches with minimal overhead. Second, there is a need to enhance the security of the flows between the end hosts which are using insecure protocols for communication. Let us consider these cases in detail.

4.3.1 Secure Flow Rule and Policy Installation with Minimal Overhead. It is mandatory for the SDN Controller and switches to support TLS based secure communication. However, usage of TLS based secure communication is not mandatory between the SDN Controller and switches. Since there is need to support different operations between the SDN Controller and switches, most of the operators are hesitant to make use of TLS secure communication protocol due to the overhead involved in the secure connection establishment between them. For instance, there is need for several rounds of communication for different operations such as flow rule configuration and updates, handle route failures, enforce different security policies for monitoring the flows and provisioning the required SLA. So, most of the operators are using insecure communication between the SDN Controller and the switches. This opens the door for different attacks in the control system. For instance, there is a possibility for an attacker to modify or delete flow rules to disable communication between filed devices related to critical operations, insert malicious flows to replicate, reroute the flows from critical devices to the attacker. So, there is need to support secure communication between CSSA/SDN Controller and switches with minimal overhead. However, the recent TLS 1.3 version offers better performance and stronger security compared to TLS 1.2. For instance, TLS 1.2 requires two round trips to complete the handshake while TLS1.3 requires only one round trip to complete the handshake. Furthermore, TLS 1.3 introduced the **Zero Round Trip Time Resumption or (0-RTT)** for sending the data in the first message by completely eliminating the handshake process for previously visited websites. Furthermore, TLS 1.3 provides stronger security as it supports only the algorithms and cipher suite that do not have known vulnerability.

Alternatively, emerging protocols such as QUIC can be used for improving the performance of communication between the SDN Controller and the switches. QUIC is able to achieve high performance since it is built on top of UDP and there is no need for establishing the connection between the SDN Controller and switches. This will significantly minimise the time delays for communication between the Controller and switches. Hence, it is advantageous for application in securing critical infrastructures. However, note that communication with QUIC is unreliable since it is based on UDP. Also, it is not mandatory to support to QUIC based communication between the SDN Controller and switches. In our architecture, TLS 1.3 has been used in the establishment of symmetric key for secure communication between the CSSA/SDN Controller and the switches. The symmetric key is used for all further communication between the CSSA/SDN Controller and the switches.

In this section, we will present the steps and process for installation of secure flow rules and policies after a secure connection is established between the Controller and switches using standard TLS. Note that we will not be describing the secure connection establishment process using TLS since it is well known.

Steps 1-4 in Table 1 are standard in TLS based secured communication between the SDN Controller and switches. In Step 5, the Controller generates a pre-master secret and encrypts it with the switches public key. Finally, it creates a hash of it and signs it with its private key. Now, both parties use the pre-master secret and nonces to calculate the master key (a symmetric key). It is represented as Step 6 in Table 1. Switch acknowledges the Controller about the reception of the pre-master secret and the Controller responds by acknowledging the reception. It is represented as Step 7 and 8 in Table 1. Thus, the Controller uses the master key to install the flow rules and policies securely.

Notations:

— Version information of switch and Controller is represented as V_{SW_i} and V_C, respectively.
— Cipher suite advertisement by the switch is presented as $Suite_{SW_i}$.
— Controller has a pair of public and private keys ($K_{PU(C)}/K_{PR(C)}$).

Table 1. Modified TLS for Secure Flow Installation

STEP 1: $SW_i \rightarrow C :< SW_i, V_{SW_i}, Suite_{SW_i}, n_{SW_i} >$
STEP 2: $C \rightarrow SW_i :< V_C, Suite_C, n_{SW_i}, Cert_C > [h(V_C, Suite_C, n_{SW_i})]_{K_{PR(C)}}$
STEP 3: $C \rightarrow SW_i :< SW_i, n_C, req > [h(SW_i, n_C, req)]_{K_{PR(C)}}$
STEP 4: $SW_i \rightarrow C :< C, n_C, Cert_{SW_i} > [h(C, n_C)]_{K_{PR(SW_i)}}$
STEP 5: $C \rightarrow SW_i :< SW_i, n'_C, [P_{MS}]_{K_{PU(SW_i)}} > [h(SW_i, n'_C, [P_{MS}]_{K_{PU(SW_i)}})]_{K_{PR(C)}}$
STEP 6: Both parties calculate $f(n_{SW_i}, n_C, n'_C, P_{MS})$ to get KM_{C-SW_i}
STEP 7: $SW_i \rightarrow C :< C, n'_C, ack > [h(C, n'_C, ack)]_{KM_{C-SW_i}}$
STEP 8: $C \rightarrow SW_i :< SW_i, n'_C + 1, ack' > [h(SW_i, n'_C + 1, ack')]_{KM_{C-SW_i}}$

— Cipher suite selected by the Controller is $Suite_C$.
— Nonce generated by the switch n_{SW_i}.
— Nonce generated by the switch n_C, n'_C.
— KM_{C-SW_i} is the master key for any particular Controller (C) to switch (SW_i) communication.
— P_{MS} is pre-master secret.
— Controllers certificate is represented by $Cert_C$.
— Switches certificate is represented by $Cert_{SW_i}$.
— Hash function is represented as $h(.)$.
— A function to calculate the master key is represented as $f(.)$, which is a combination of MD5 and SHA.
— A certificate request is represented as req.
— An acknowledgement is represented as ack.

The CSSA security policies are termed as **Policy Expressions (PE)**, which specify whether packets and flows from field devices and end hosts follow a particular flow path in the ICS network. The PE specification syntax uses an enhanced version of RFC1102 [8]. We have developed a policy specification language using JSON for the specification of secure communication policies in SDN domain. A detail discussion on the policy specification language can be found in [53]. In this work we have extended policy expressions to support communication of field devices in control systems.

The policy expressions are fine-grained and specify a range of policies using various attributes of field devices and flows; for instance, these attributes include different types of devices (virtual or physical), source and destination attributes, flow attributes, and constraints, requested services, security services and security/trust labels. The attributes are **(a) Flow Attributes:** Flow ID, sequence of packets associated with the Flow, type of packets, Security Profile indicating the set of security services that are to be associated with the packets in the Flow; **(b) Device Attributes:** ID specific to **Field Devices (FD)** including sensor/actuator; **(c) Switch Attributes:** Identities of the Switches and Security Label of the Switches; **(d) Host Attributes:** Identities of Hosts such as Source/Destination Host ID; **(e) Flow and Domain Constraints:** Constraints such as Flow Constraints (FlowCons) associated with a specific device Flow; **(f) Services:** For which the PE applies (e.g., FTP storage access); **(g) Time Attributes:** The period for which the PE remains valid and time constraints for the flow; and **(h) Path:** Indicates a specific sequence of switches any particular flow from specific field devices/users should traverse.

The Flow Constraints are conditions that apply to specific flows from any field devices within a network. For instance, a constraint might specify the flows from a specific type of sensor device should only go through a set of switches that can provide guaranteed bandwidth. From a security point of view, a constraint could be that a flow should only go through virtual OpenFlow switches that have a particular Security Label. The PEs support wildcards for attributes, enabling it to set policies for a group of services. A simplified Policy Expression template is as follows:

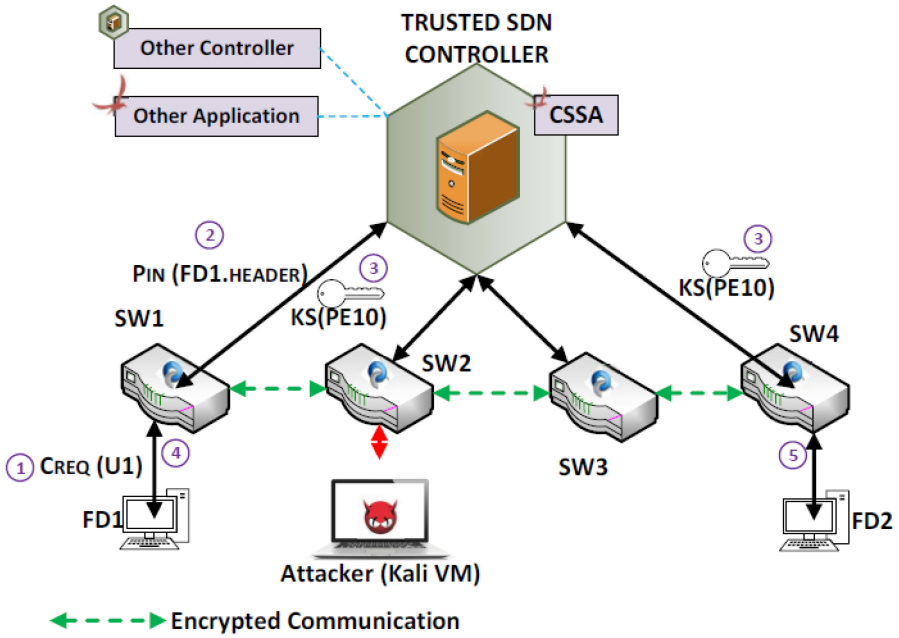


Fig. 3. Process of encrypting the payload; CSSA- SouthBound security application.

$PE_i = < FlowID, FDID, SourceAS, DestAS, SourceHostIP, DestHostIP, SourceMAC, DestMAC, User, FlowCons, DomCons, Services, Sec - Profile, Seq - Path >:< Actions >$
where i is the Policy Expression number.

4.3.2 Securing the Flow Information in Insecure Protocols. Currently, protocols such as Modbus are being extended to support TLS based secure communication in control system networks. In this section, we will present how the CSSA is used for securing the flows from the field devices using insecure protocols. As already mentioned, the end host flows are insecure in typical control system applications. So there is need to secure the flows during the transit. In the current SDN network, by default, none of the Controllers provides on-demand confidentiality service for user flows. Our CSSA provides confidentiality services to the users or devices if required. The administrator can specify the end hosts (with limited resources) that require secure communication. Now, we explain the mechanism in detail.

Let us consider, a scenario shown in Figure 3, where a Field Device $FD1$ connected with switch $SW1$ is trying to communicate with Field Device 2 $FD2$ in switch $SW2$ using modbus. In normal SDN operation, $FD1$ sends the packet to $SW1$. Since it is a new communication it will not have any flow rules and $SW1$ sends a *Packet-In* request to the Controller. The Controller knows the network topology and hence deploys the flow rules in the OpenFlow switches that will establish the communication between $FD1$ and $FD2$. However, the payload for all flows in such communications in SDN is un-encrypted.

In the CSSA, we are able to provide confidentiality service to the communicating devices. We have policy expressions that specify which of the communication entities would be provided with confidentiality services. The Controller generates symmetric keys for each of these communications according to the policy expressions. The key convention used for the policy expression is

$K_{S(PolicyExpressionID)}$. The switches use these symmetric keys to encrypt the flow payload during communication. The agents in the switch help in the encryption and decryption process of the payload. The distribution or sharing the symmetric key with the OpenFlow switches present within a path other than the source and destination switch is entirely dependent upon the Controller and the policy expressions present in CSSA. In exceptional cases, the intermediary OpenFlow switches can request for these symmetric keys for any particular flow. We have introduced new messages in the OpenFlow protocol for this operation, which is shown in Listing 1. In our scenario, we have a policy expression ($PE10$) which expresses that the communication between $FD1$ and $FD2$ should be provided with confidentiality service. When the communication request ($C_{REQ(FD1)}$) comes to the $SW1$ from $FD1$, it issues a *Packet-In* request ($P_{in(FD1.header)}$). The CSSA checks the policy and finds the confidentiality service needs to be provided for the communicating parties. Hence, it generates the key $K_{S(PE10)}$, deploys the flow rules to $SW1 - SW4$ and also sends the key to them. As shown in Figure 3, $SW1$ uses $K_{S(PE10)}$ to encrypt the payload and send it to the destination switch $SW4$, where it is decrypted and transferred to $FD2$. The Controller will not share the encryption key with the intermediary route switches unless authorised in the policy expression or any intermediary route switches requests for the key.

STEP 1: SW1 receives insecure flow initiated by FD1 to FD2
STEP 2: SW1 generates Packet-In to SDN Controller which is subsequently forwarded to CSSA
STEP 3: CSSA generates symmetric key and updates it to SW1 and SW2
STEP 4: SW1 encrypts the FD1 flow and forwards it to SW4
STEP 5: SW4 uses symmetric key to decrypt the flow and forwards it to FD2

Listing 1. Request message for flow specific symmetric keys

```
/* Request message (Symmetric-key). */
struct ofp_ks_reqmsg {
struct ofp_header header;      /* Type OFPT_EXPERIMENTER. */
uint32_t ksreqmsg;          /* Symmetric-key request
                             /* Reason of request. */
uint8_t ksreqmsg_reason[0];
};
enum onf_ksreqmsg{
REQUEST =1<<0 /* 1 indicates that it's requesting for Symmetric-key */
ADDITIONAL INFO=1<<0 /* 1 indicates that the request message contains
some reason. */
}
```

4.4 Reliable Paths for Safety Critical Messages

Let us first discuss some issues that can impact the flows related to critical operations in a control system.

For instance, in the case of IEC 61850 electric substation Automation Systems which runs on Fast Ethernet LAN, the communication system must support messages with strict timing requirements associated to protective relaying strategies. Typically, GOOSE and **Sampled Values (SV)** messages are time critical messages with tight deadlines of 3 ms. It is important to note that even for safety critical messages, there is a need to enforce policies for secure monitoring of the messages. Lack of security measures give an opportunity for the attacker to flood the network with malicious safety critical messages. Furthermore, if the attacker has several devices under his control, then s/he can make use of all the devices to generate malicious safety critical messages to flood the control

system network. Hence, it is important to validate the safety critical messages before routing them to the destination node. This has to be achieved while meeting the stringent timing requirements of the safety critical messages.

Without CSSA, the switches generate Packet-In message to the Controller if routes are not available or when any of the communication links for the existing path fail in the network. This can cause significant delays to the safety critical messages. Hence, there is a need to provide reliable paths for safety critical messages in the control systems.

The CSSA is designed to significantly enhance the reliability of the end-to-end communication path for supporting safety critical operations such as alarms, circuit breaker operations in control systems. Figure 4 shows the flow chart for the switch operation for routing of the safety critical messages.

We assume that the Security Administrator is aware of the devices that can initiate flows related to the safety critical operations. The CSSA implements two proactively configured routes with the required SLA to support flows related to safety critical operations in the control system. By default, switches make use of the primary data path for forwarding the safety related flows between the devices. The secondary data paths are used as backup, in case of failure of primary data path. Note that the backup data paths are used only in case of failure of the primary data paths and there is no need to alert the Controller since routes are already configured by the CSSA. Furthermore, the backup data paths are used only for routing of the safety critical messages. For instance, since the CSSA is aware of the devices that can initiate and receive safety critical messages, the routes are proactively configured into the switches by the CSSA. When the switch receives a new flow that is related to safety critical operation, the flow differentiator determines that the flow is related to safety critical operation by analysing the source and destination address of the packet. Now the switch checks to see if there are any active data paths to route the packet to the destination. Since CSSA has already pre-configured the routes, the switches choose the primary data path and forward the message through this link.

In the case of a failure of the primary data path, the switch uses the backup data path for forwarding this message to the end device. Also, note that for messages that are not related to safety critical operation, the switches follow routine process in case of failure of the primary communication paths. For instance, the switch will raise a link failure error message and also initiate Packet-In message to the Controller. Now the CSSA will make use of the overall topology view available at the Controller to determine the route and initiate flow_mod message to all the switches for establishing the flow route between the end devices.

Now let us consider the case where there is failure of links for the primary and backup data path at a switch. In this case, the switch forwards the complete safety critical message as a Packet-In message to the Controller. CSSA then determines the source and destination address related to this flow and uses the control plane to directly forward the message to the egress switch. Note that in this case there is no need to determine and establish the routes by initiating flow_mod messages to all the switches. Hence, our approach can meet the stringent time requirements for critical operations.

4.5 Preventing Access to ICS Domains using Unauthorised Devices

In this scenario, we assume that user Alice who belongs to ICS2 domain has registered to access the services in the domain using her personal laptop by providing details such as user name, hardware and software details of the laptop such as processor information, memory, storage, MAC address, OS and applications installed in the laptop. We assume that Alice is a benign user and has knowledge to ensure that her laptop is compliant with the ICS2 domain usage policies. The administrator configures the below policy which enables Alice to access services in ICS2 domain.

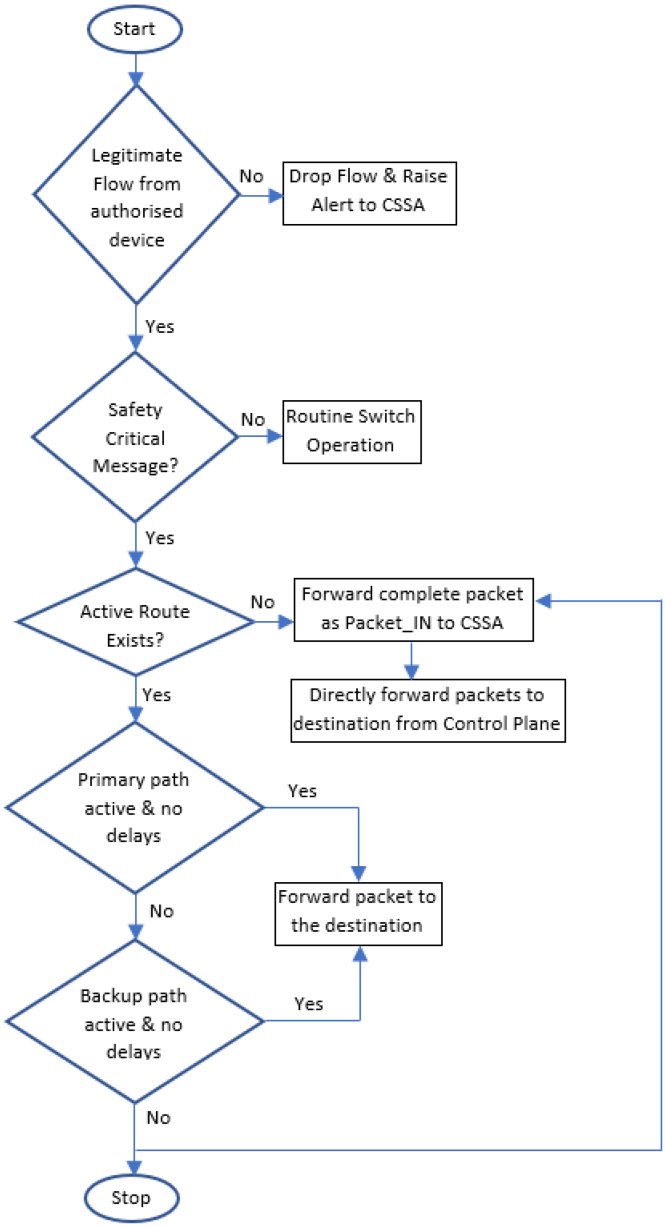


Fig. 4. Reliable path.

$PE_8^{ICS2} =< *, *, ICS2, *, (172.16.10.66), (79 : c8 : 82 : b2 : 7b : 1a), *, Alice, *, *, (80, 443), *, * >:< allow >$

Now consider Alice wanting to access web server in ICS2 domain. When the switch or access point receive the first packet, it will generate a Packet-In request to the SDN Controller which includes information such as the source IP, destination IP address, source and destination port numbers and MAC address of the laptop. The Packet-In message is subsequently forwarded to the CSSA. The message analyser extracts all the relevant information and forwards it to the policy resolver. Since the administrator has already configured policies to permit access to her laptop,

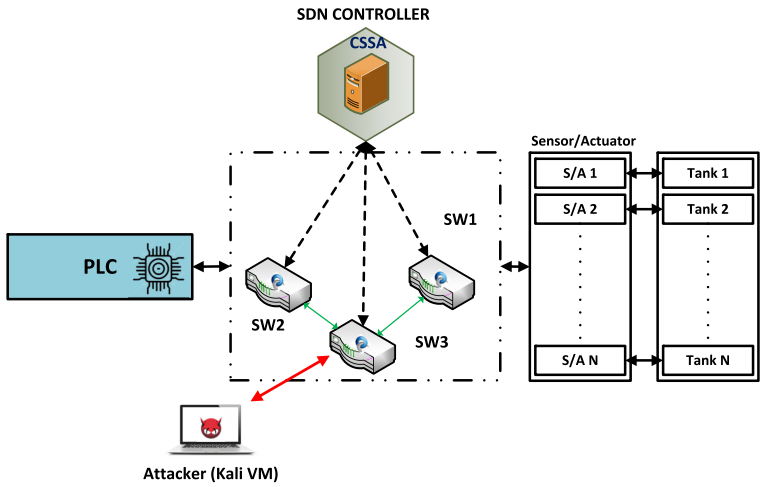


Fig. 5. CPS attack scenario.

the policy resolver makes a decision to permit access to web server in ICS2 domain according to the (PE_8^{ICS2}) policy expression. Now the enforcement module configures the relevant switches to establish a secure path. Also, the flows are configured to pass though different security functions to detect any suspicious behaviour after paths are established between her laptop and the web server.

Now consider that Alice has purchased a new laptop and is trying to access the web server in ICS2 domain. This will not be permitted by the CSSA since the new laptop is not registered to access services in ICS2 domain. Alice has to register her new laptop with CSSA to access services in ICS2 domain.

4.6 Detect Attacks and Dynamic Enforcement of Policies to Minimise the Impact of Attacks

Let us consider an example scenario to discuss how our architecture can counteract attacks exploiting safety data types and control parameters of physical devices and also cyber related attacks in critical infrastructures. Consider a scenario shown in Figure 5, where PLC's are used to monitor the oil levels in multiple tanks and different actuators are dynamically switched on and off for maintaining the oil levels in the tanks. If there is unsecure communication between the sensors and the PLC, then the attacker can alter the sensor readings when messages are transferred between the sensors and the Controller. Then there is a possibility for the failure of sensors or failure of communication links which can result in not detecting events such as overflow of the oil in the tanks which can result in serious hazards. The CSSA is able to address both these issues. The CSSA uses dynamic encryption for securing the communication between the sensors and the PLC. In this case, the sensors messages are encrypted at the ingress switch and decrypted at the egress switch and forwarded to the PLC. Note that our architecture supports any communication protocols between the devices and SDN is used as a wrapper for secure encapsulation of the messages at the ingress switch and decapsulating the messages at the egress switch.

The CSSA uses state-based analysis for detecting failure of devices and suspicious activity in the critical infrastructures. For instance, consider the case where the attackers tamper the sensor devices or disable the communication links to prevent switching of the actuators resulting in the overflow of the oil from the tanks which can be a serious hazard. In this case, the CSSA detects

Table 2. Comparison of CSSA with other Similar Security Solutions

Paper	Unpatched System Protection	Flow Security	Reliable Path for Critical Com.	Attack Detection and Mitigation				Mitigation Uses
				DDoS	UCT	RCE	Ransomware	
[3]	No	No	No	Yes	Yes	Yes	No	Policies (RBAC Only)
[17]	No	No	No	Yes	Yes	No	Yes	Digital Twin
[14]	No	No	No	Yes	Yes	Maybe	No	Packet Analysis
[45]	No	No	No	Yes	Yes	Maybe	No	ML (Communication Pattern)
[39]	No	No	No	Yes	No	Yes	No	ML (Packet Headers)
[35]	No	No	No	No	No	No	No	DT (Auth. Policies)
CSSA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Policies (Path, flow, IDS)

*UCT- Unauthorised Communication Tampering
*RCE- Remote Code Execution

these attacks using state-based analysis. For instance, the CSSA maintains the approximate time it takes to fill the tank to reach the maximum level. This information is based on the analysis of the operation in test-bed environment or by calculating the maximum time it takes for the pump with flow rate X litres/min to fill a tank with capacity of Y litres where Y>X. The CSSA keeps track of the time when the pump was switched on and predicts the time when the pump needs to be switched off for a specific tank. If the CSSA does not receive the maximum level reached message from the sensor within the predicted time, then it resets the pump and raises an alert to the administrator.

The CSSA uses NFV for dynamic enforcement of policies using flow validation ( ) for detecting different attacks in ICS. For instance, attacks which disable the host based security tools to hide the state from Controller are detected using the introspection based security functions. In Section 5.1.4, we demonstrate the usage of introspection security function for detecting ransomware attacks in DT environment and dynamically generate security policy rules for securing the vulnerable services. In the current implementation we have done enforcement of policy rule manually to avoid spreading of malware from the DT environment to the ICS environment [35]. However, this can be done automatically.

4.7 Comparison

Table 2 presents comparison of our architecture with relevant works. From the table it is clear that our architecture has unique advantages such as protection of unpatched system, flow security, provisioning reliable paths for time critical operations. Also, our architecture is able to deal with sophisticated malware such as ransomware and dynamically generate policies for securing the vulnerable system from attacks.

5 IMPLEMENTATION

In this section, we first present brief discussion on specific software modules developed for ONOS SDN Controller and Open virtual switches (see Figure 6). Then we describe the prototype implementation of our architecture and demonstrate how it deals with different attack case scenarios.

We have developed the CSSA for hosting over NorthBound Interface of the ONOS Controller which runs inside a Ubuntu Server VM. The components shown above the dotted line in Figure 6 are related to the ONOS SDN Controller. The modules shown in green boxes show the specific software extensions developed to support our architecture. The components shown below the dotted line in Figure 6 are related to the Open vSwitch. Open vSwitch aka OVS is an open-source multilayer virtual switch used for network virtualization. It helps to establish control over distributed physical servers using virtualized switching abstraction. It is implemented using C. However, it can be adapted in various environments. It supports and uses OpenFlow. It also supports other protocol like NetFlow, sFlow, and the like. In most cases, it is used in data centres. It is supported by most of the Linux oriented virtualization technologies, such as Xen/XenServer, KVM and Virtual

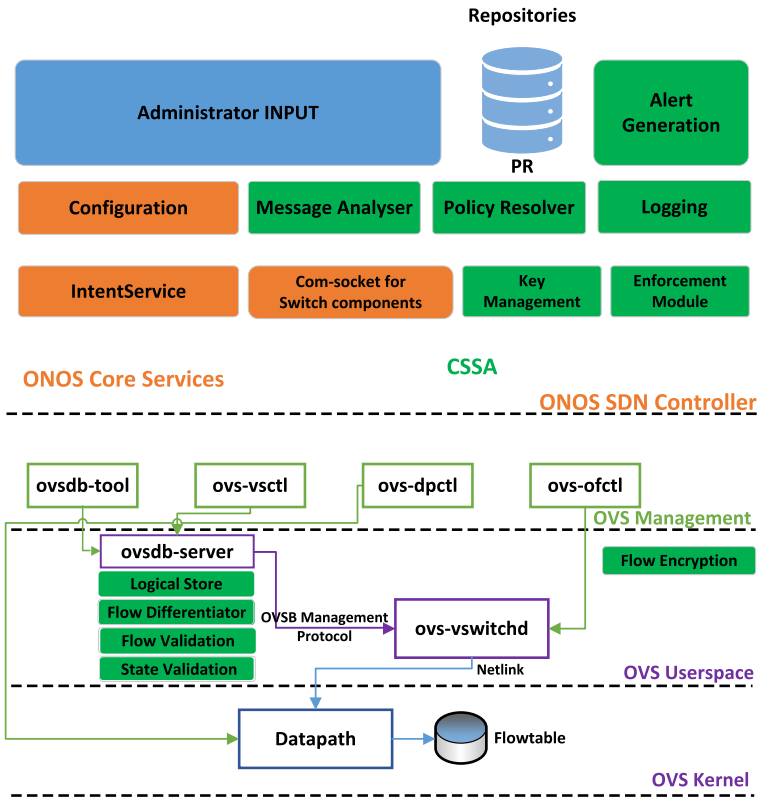


Fig. 6. Software module interaction with ONOS core services.

Box. As shown in the Figure 6, there are different components that are implemented in the kernel space and user space. The green boxes show the specific extensions we have developed to support implementation of our architecture.

5.1 Prototype

We have used ONOS as the SDN Controller and Open Virtual Switches to justify our CSSA. We are using ACSRC Cyber-Labs for the simulation. The Cyber-Labs environment uses VMWare cluster. We are using **vRA (vRealize Automation)** templates for dynamic deployment of the prototype environment. For this prototype environment, we are using a hybrid approach, where we have both physical (Raspberry PI/Zodiac FX) and virtual (OVS) OpenFlow switches. The ONOS cluster consists of a set of VMs. The ONOS Cluster is built upon the Atomix Framework [26]. Atomix is a pure distributed cloud application framework for decentralised cloud application development and management at runtime. By default, ONOS Cluster provides certain properties, such as fault tolerance, scalability, upgradability, and so on. In addition, ONOS Cluster uses the Gossip protocol to keep the cluster nodes in sync. ONOS Cluster nodes reside in three states with respect to the devices: Standby, Master, and None. To satisfy certain cluster properties, ONOS Cluster changes the node status. For instance, one specific update to the topology application puts one ONOS Cluster node to standby. After the upgrade, other cluster nodes synchronise with the upgraded node using the gossip protocol automatically. However, cluster implementation experiments have not been part of this paper. The experimental work on cluster implementation forms part of our

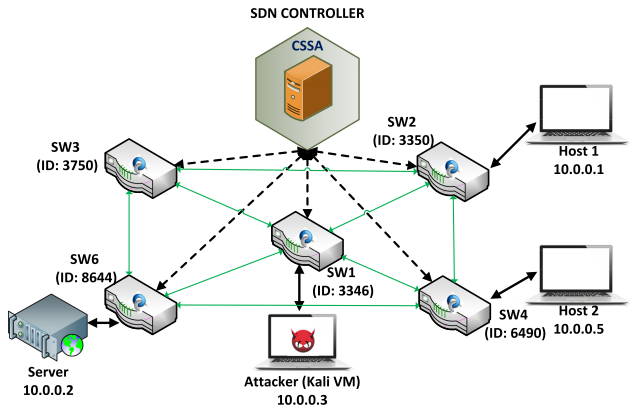


Fig. 7. Network setup.

work on **quality of service (QoS)**. This is a major issue in its own right and we intend to write a separate paper on this issue. Also, we created a private cloud for achieving high availability of critical services and to maintain our high-level business applications.

Figure 7 shows the network setup with RTU's, PLC and controllers implemented in virtual machines and the virtual machines are connected using SDN switches (virtual/physical). We have simplified the diagram to make the scenario easily understandable. Here, the SDN Controller is running in the SDN Cluster. The Server is running as part of the Control Station server and the attacker can be external or in the Enterprise network. The server has access to the SCADA device, and hence they are the target for the attacker. For example, this could correspond to an oil and gas company using an integrated control system to manage refining operations at plant sites [40], to remotely monitor the pressure and flow of gas pipelines, and control the flow and pathways of gas transmission. Similarly, a water utility company can remotely monitor well levels and control the well pumps; monitor flows, tank levels, or pressure in storage tanks; monitor pH, turbidity, and chlorine residual; and control the addition of chemicals to the water. In Figure 7, Host 1 is RTU/PLC slave machine (10.0.0.1) simulates the acquisition of parameters from 10 boiler water tanks and a main tank, and one valve actuator controlled from the PLC. Host 2 (10.0.0.5) is another slave device of this simulation network setup. Here, we have used ModbusPal (1.6) as the slave device (10.0.0.1 and 10.0.0.5), which mimics the actions of a real Modbus slave machine.

Modbus Server (10.0.0.2) runs the SCADA MTU/PLC MASTER which is based on a free version of Mbtget simulator. The MTU machine also runs the SCADA HMI to visualize the process.

In addition to the Modbus master and slave devices, we have connected an adversary 10.0.0.3. This can be a rogue Modbus slave device or a real external rogue device mimicking the IP and MAC of a slave device in the Modbus network.

In a legitimate case, an administrator can use Mbtget running in Modbus server to read and write ModbusPal register and coil values running in Modbus slave device. It can also be used to send control actions to the Modbus slave devices. For attack purpose, we use *modbusclient*(*auxiliary*/*scanner*/*scada*/*modbusclient*) Metasploit script. Attackers can use this tool to capture the data from the target and analyse the data before performing the attack. For instance, attackers may want to know which coil controls the pressure valve on the system and what value that pressure value has in its register that is the threshold for opening and closing it. If these values are changed, the entire plant could be placed at risk [40]. Now we will present three attack scenarios and how they can be mitigated using our security architecture:

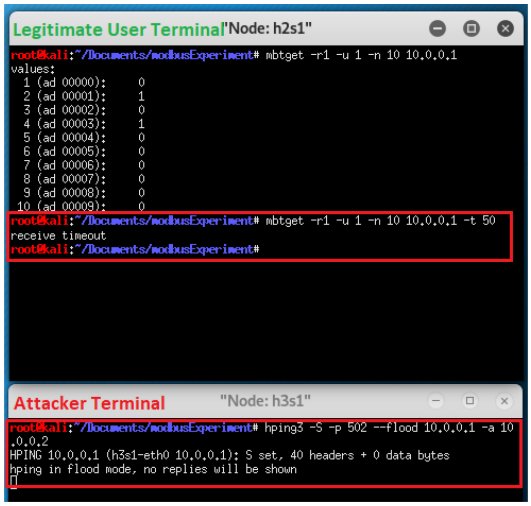


Fig. 8. DDoS attack on MobBus server.

5.1.1 DDoS in SCADA Infrastructure. In this scenario, we simulate a DoS attack, where a legitimate client is trying to read coil values from the Modbus server; however, due to DoS attack launched by an adversary completely stops the legitimate user from reading the values. In our setup, Mbtget sends read and write instructions to the Modbus Slave devices (in this case, ModbusPal). We are considering Mbtget as the legitimate client device and the ModbusPas as the server device.

In the legitimate case, Mbtget sends the coil read request to the ModbusPal and it can immediately read the coil values without any delay (as shown in the first portion of Figure 8).

After that, the adversary in 10.0.0.3 uses *hping3* and launches a DoS attack towards the ModbusPal server. The ModbusPal was active on port 502. Previously, the adversary intercepted some communication traffic between Mbtget and ModbusPal, from which he knows about the port. The adversary also masquerades the IP address of the legitimate client and launches the flooding attack. This is a SYN-Flooding attack and the hping3 is creating fake SYN packets targetted to the ModbusPal server. It takes a few minutes to overwhelm the packet processing capacity of the ModbusPal server. This causes a delay in communication and clogs any read and write operation to and from the client devices.

$$PE_{16} = < *, *, *, *, *, 10.0.0.1, *, *, *, *, *, Packet = 6, *, *, *, * >:< Deny >$$

We have set a timeout period of 50 seconds (-t 50) and send the same read operation to the server while the attack was in action. ModbusPal fails to process the request and the operation request times-out. The successful attack is highlighted (red) in Figure 8. We kept the attack running for more than seven minutes; after seven minutes, the attack completely freezes the VM.

In our architecture, when DoS attacks are detected in the domain, the CSSA makes use of this real time situational awareness for ensuring the availability of the servers that are hosting critical services in the domain. For instance, the CSSA configures the switches for reactive routing for the servers running critical services. In this mode of operation, any new flow initiated by the end devices will result in generation of Packet-In message to the Controller which is subsequently forwarded to the CSSA. Hence, the CSSA has real time situational awareness of the control system

domain and can determine the number of flows initiated by each end source to the servers running critical services. Depending on the service requirements, the CSSA can permit or deny the flows to the server or restrict the maximum number of flows that can be initiated from each end source. In this case, the CSSA configures the Ingress switch to drop all the flows from the malicious end source. PE_{16} shows that for 10.0.0.1 slave device packet count is set to 6. Hence, the CSSA drops the DoS attempt by the adversary.

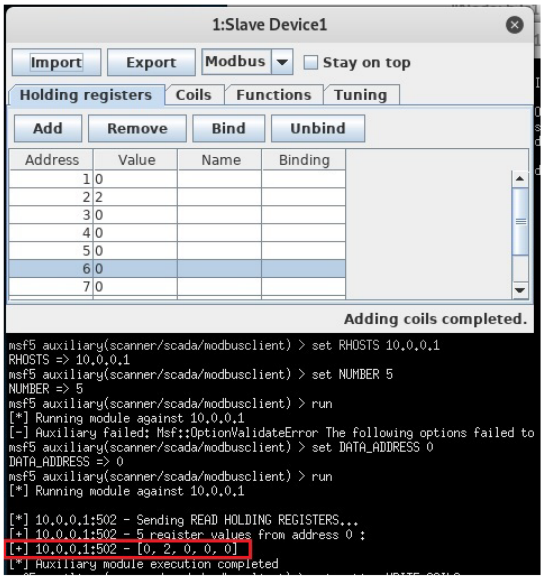
5.1.2 Unauthorised R/W on SCADA Infrastructure: In this attack, we assume that the adversary knows the SCADA network infrastructure and has access to one of the machines in the network. We also assume that the adversary used *modbusdetect*(*auxiliary*/*scanner*/*scada*/*modbusdetect*) melasploit script to detect the ModBus slave devices (IP: 10.0.0.1). Now, the adversary knows the IP address (10.0.0.1) of the ModBus slave device. At this stage, we will show two attacks, in the first attack, the adversary reads the contents of the slave devices register values and in the second one, the adversary writes the coil values of the slave devices.

The IP address of the adversary is 10.0.0.3 and from this IP he uses *modbusclient*(*auxiliary*/*scanner*/*scada*/*modbusclient*) Metasploit script. In the first, attack, he uses the script to maliciously read the values of the holding registers of the slave device1. Since the network infrastructure and modbusPal does not have any mechanism to check the authenticity of the of the read request, he successfully reads the values. Figure 9(a) shows a successful read operation. The console part is the attacker's window and the first part is the ModbusPal Slave device1 Holding register view. In the Slave device1's holding register values from address 1-5 is [0, 2, 0, 0, 0]. From the console view of the attacker, we can see that modbusclient read the value [0, 2, 0, 0, 0] for addresses ranging from 1-5 (Highlighted in red). Now, the adversary uses the same script to alter the coil values of the slave device1. The adversary sends a binary stream "1010101010" to overwrite the coil values from 1-10. As shown in the ModbusPal view of Figure 9(b), it is being successfully overwritten by the binary stream (Highlighted in red). The ModbusPal or the network controller does not check the authenticity of the requesting devices and also, the protocol uses a clear text, it becomes easy for the adversary to send fake requests.

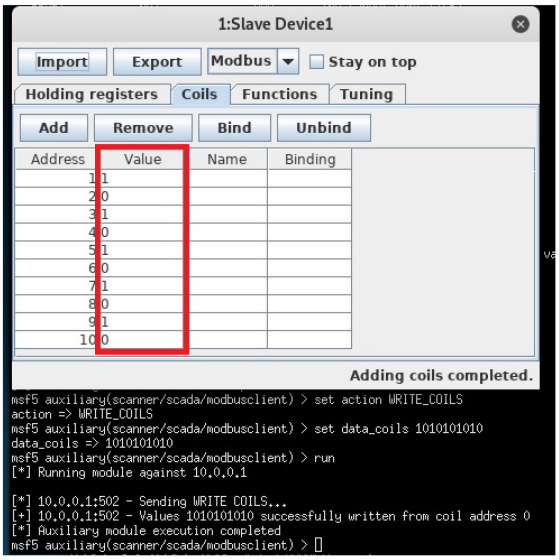
In our security architecture, these attacks are prevented as the switches validate the source address of the device originating the messages and also use AES based secure communication between the end hosts. The message is encrypted at the switch that is connected to the source devices and decrypted at the switch that is connected to the destination device. Figure 10 shows the end to end delays caused due to the encryption and decryption of the insecure flows. In this case. there are 100 devices connected to the ingress and egress switches. Each source is generating 10 flows/sec and only one of the source node is generating clear text traffic. In this case, the insecure flows from the device are encrypted at the ingress switch and decrypted at the egress switch and forwarded to the destination. It has to be noted that the additional delays are only for the insecure flows.

5.1.3 Malicious Command Execution in SCADA Infrastructure. In this scenario, we describe how we can execute malicious commands in the SCADA infrastructure. The remote code execution will provide the adversary shell access. Our network setup is similar to previous experiments. Here, ModbusPal is running on 10.0.0.1 and the adversary is using 10.0.0.3.

In this attack, first, we reserve-engineered the ModbusPal (version 1.6b) application using Re-caf a Java bytecode editor and noticed that the developers of ModbusPal left an empty execute () method in the program (as shown in Figure 11(a)). We have embedded a Netcat reverse shell (as shown in Figure 11(b)) inside the method and mapped the method to execute when ModbusPal is run (by clicking the Run button). The reverse shell syntax can be altered to execute any malicious instructions at the ICS system level. Such operations can damage the ICS infrastructure tremendously or can be used to launch further attacks.



(a)



(b)

Fig. 9. Modbus attacks: (a) unauthorised read (b) unauthorised write.

The adversary started a Netcat listener on port 4444 and the moment a legitimate user starts a Modbus network, the adversary receives the shell of the user. From there, the adversary can control the whole ICS machine. Figure 12 shows the successful attack.

With our security architecture, attack signatures based on deep packet inspection are applied at SW1 switch using NFV to prevent malicious client VMs from accessing the vulnerable execute () method on the server. Then when we launched the same attack on the Web Server VM, the attack

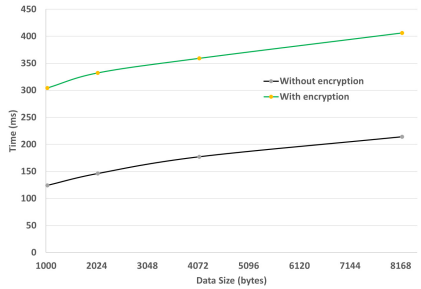


Fig. 10. Flow encryption-decryption time.

```
public void execute(ModbusRequest req) {
    throw new UnsupportedOperationException("Not supported yet.");
}
```

(a)

```
while (this.executeThread) {
    try {
        Process p = Runtime.getRuntime().exec("nc -e /bin/sh 10.0.0.3 4444");
```

(b)

Fig. 11. (a) empty execute method (b) embedded netcat shell code.

```
root@kali:~# nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.0.0.3] from (UNKNOWN) [10.0.0.1] 35782
```

Fig. 12. Successful attack.

```
[+] Remote Command Innection Mode
[+] Checking for vulnerability...
[X] Connection Error
[X] Connection Error
root@kali:~/Desktop#
File  Edit  View  Search  Terminal  Help
root@kali:~# nc -lvp 4444
listening on [any] 4444 ...
```

Fig. 13. Attack prevention at ingress switch.

was not successful and the malicious VM is isolated from the network. As shown in Figure 13, the malicious VM1 lost its connectivity.

5.1.4 Ransomware Attack on SCADA Infrastructure. This is a scenario specific to REvil / Sodinokibi (AKA Sodin) ransomware infecting control stations in the ICS domain which is emulated in DT environment. The attack is detected using **Virtual Machine Introspection (VMI)** which resides in the hypervisor [16, 34]. We assume that one of the Control Station servers was running a vulnerable version of the Oracle WebLogic server (v-12.1.3.0). A legitimate employee who was authorised to access services in ICS domain was targeted with a phishing attack and downloaded a malware injection kit in the Control Station servers. Now we observe what actions REvil did and how our CSSA can detect it. These malware actions are logged in the DT of the Control Server.



Fig. 14. REvil malicious process issuing commands.

We observe the changes as the current state of the DT differs from the previous stable snapshot of the DT.

The Historian server is the affected server and it's a virtual server. We have used introspection based security agent in this case to detect the attack. At first, the REvil ransomware issued a series of *NtDelayExecution* calls to delay the execution to evade detection be security agent. After that, it creates a registry with the path "\\REGISTRY \\MACHINE \\SOFTWARE \\WOW6432Node\\recfg" with six values: "sub_key", "sk_key", "pk_key", "0_key", "stat" and "rnd_ext". The first four values are binary, while "rnd_ext" contains the file extension that the encrypted files will have and part of the readme file name. Then the ransomware process started to scan A, B, E, F and so on up to Z driver. All these drives were opened for root folders, for example A:\, B:\and so on (the VM has a C drive as the boot disk and a D drive as a virtual CD\DVD drive). After this scan the "stat" registry key is initialised. The malware process issues command "cmd /c vssadmin.exe Delete Shadows /All /Quiet & bcdedit /set default recoveryenabled No & bcdedit /set default bootstatuspolicy ignoreallfailures" to delete the shadow copies. The malware process creates "d60dff40.lock" and "jcg34-readme.txt" file and starts to encrypt all the files. The ransom note was extracted as an image and stored in "C:\Users\Xen\AppData\Local\Temp\j23428o.bmp" location. To display the ransom note, the ransomware process edits the registry to change the wallpaper to point to the newly created image. As this malware is a **Ransomware as a Service (RaaS)** framework, it tries to connect to other domains.

The DT service logs all these changes periodically. Figure 14 presents the REvil ransomware process issuing the shadow copy delete command. This is part of the internal state information of the Historian Server. Our DT service application compares the internal state changes and raises alerts. Now CSSA dynamically generates a policy rule to block the propagation of the ransomware. Also, it helps to restore the historian server to the DT state default state. In the current implementation we have done enforcement of policy rule manually to avoid spreading of malware from the DT environment to the ICS environment. However, this can be done automatically.

5.1.5 Performance. We have conducted performance analysis of the CSSA using tools such as CBench and Jconsole. CBench is benchmarking tool for measuring the throughput of the SDN Controller by sending different types of traffic for varying number of switches. We have measured the flow rate with and without the CSSA for 20 switches where each switch is connected to 10 end hosts and the number of flow rules in the switches are increased from 100 flow rules to 400 flow rules. Figure 15 shows the throughput with and without the CSSA. The orange bars show the throughput obtained by activating default applications such as the ONOS core, drivers, proxy and the forwarding application. Blue bars show the throughput of the Controller after activating the CSSA together with the previously mentioned applications. As shown in the Figure 15, the throughput decreases with the CSSA and it also varies in accordance with the number of flow rules in the switches.

With a similar setup, we have used Jconsole to measure the CPU usages and Heap memory usages while ONOS is working with and without CSSA. Jconsole is a built-in API of JAVA to monitor the individual JAVA modules running within the JVM. Figure 16 shows the CPU usages,

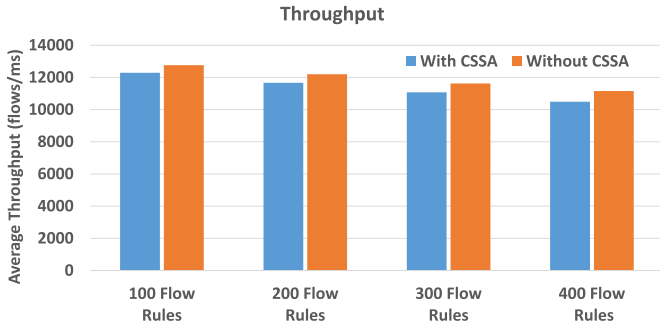


Fig. 15. Throughput.

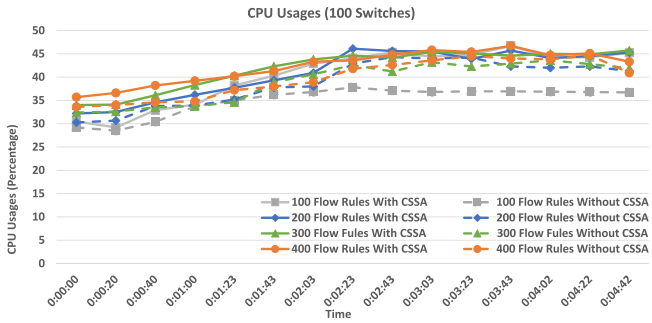


Fig. 16. CPU usages.

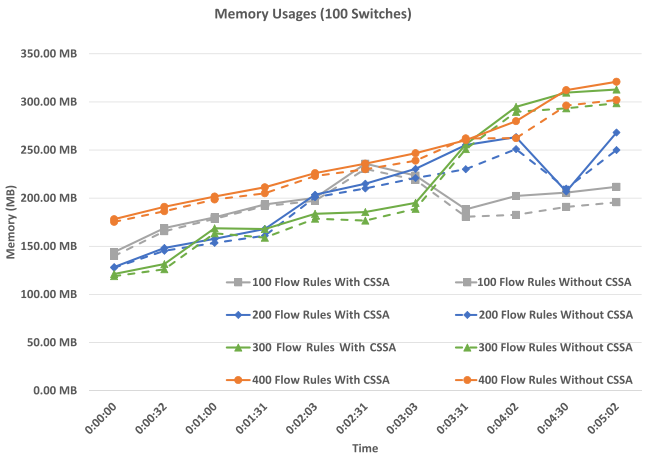


Fig. 17. Heap memory usages.

and Figure 17 shows the Heap memory usages without and with CSSA for varying number of flow rules. In both the cases, the overhead increases with the increase in the number of flow rules in the switches.

6 RELATED WORK

In this section we present some relevant related work and compare with our architecture.

6.1 Attacks on ICS

Some researchers have developed techniques for generating different types of attacks in CPS and analysis of the attacks. Peng et al. [43] analysed attacks in smart grid infrastructures and McLaughlin et al. [37] and Yaacoub et al. [54] presented a detailed study on the threat landscape of ICS. Nawrocki et al. [41] conducted a detailed study on the security impact of connecting ICS to Internet. The traffic analysis is correlated with data from honeypots and Internet-wide scans to separate industrial from non-industrial ICS traffic and the authors provide an in-depth view on Internet-wide ICS communication and identified several critical risks to the ICS and Internet. Barrère et al. [4] proposed security metric based on AND/OR graphs to represent cyber-physical dependencies among network components. The proposed metric is able to identify sets of critical cyberphysical components, with minimal cost for an attacker to make the control system into enter into a non-operational state. Niedermaier et al. [42] developed testbed for testing ICS against specific attack scenarios such as attacker having basic knowledge and local access to the control system and also for the the case of attackers having remote access to the network. Kus et al. [29] demonstrated that most of the machine learning techniques that have been proposed for attack detection in ICS are vulnerable to the zero day attacks. CANflict [10] demonstrated techniques to control the CAN traffic at the bit level and bypass the protocol's rules from remotely compromised ECU. A common attack in ICS is a control-logic attack, in which an adversary alters the ICS processors' control logic of handling physical parameters, thus enabling the adversary to control the ICS domain [50]. Various control-logic attacks on ICS focus on modifying the PLC program [23], [56]. However, such modifications can be detected by the ICS forensic investigator due to the ICS protocol [23], [56]. Zubair et al. in [60] introduced a new network-based control-logic attack on the ICS domain. In this attack, an adversary remotely injects a benign memory duplicator script in the functional PLC unit in an ICS domain which copies the local memory contents into a protocol-mapped address space. Later the adversary can read them over the network. Our work focused on analysing attacks that are possible in different ICS domains and making use of SDN, NFV and DT technologies for mitigating the attacks.

6.2 ICS Attack Detection

Some researchers have worked on the detection of attacks in CPS. CloudPAD [45] was proposed for off-premises deployment of anomaly detection in ICS using neural network based on the Long Short-Term Memory architecture. Also, [25, 48] proposed anomaly detection techniques for ICS. Hasan and Mohan [20] proposed techniques for making use of two SDN Controllers for securing the control plant against attacks. A High-Performance Controller is used for managing the control plant during normal operation and the management of plant is shifted to Safety Controller when some suspicious activity is detected in the control plant. Mittal and Mueller [38] proposed techniques to detect timing based attacks by analysing communication timing deviations of observed time from the expected time on end nodes. The idea is to extend Linux network stack by embedding timing information within TCP and UDP packets and they also presented an analysis on the susceptibility of real-time applications to malware attacks. Gehrmann and Gunnarsson in [17] presented a security architecture, which utilises a digital twin concept in a cloud infrastructure based ICS. The work argued for the need for security services in ICS and how they can help in mitigation. However, these techniques do not make use of SDN and NFV technologies for securing control systems. Flooding attacks are common in critical infrastructures like oil and gas [40]. Mohammed et al. presented a novel field flooding attack in [39]. Modbus packet memory structure is used to perform this DoS attack. It overflows the memory bank in PLCs, causing a successful DoS attack in critical infrastructures using the Modbus protocol. They have

proposed an XGBoost-based classifier to detect and mitigate the attack effectively [39]. We have presented Modbus-based DDoS attacks in this paper as well. As machine learning-based solutions require proper datasets and resources to detect such attacks, our solution focuses on policies to dynamically detect and mitigate DDoS attacks. In this paper, we have demonstrated how CSSA can be used to deal with cyber and physical attacks in ICS. In particular, we have demonstrated how our architecture can make use of real time situational awareness and flow validation security policies to deal with cyber related attacks. We have also discussed how state validation can be used for detecting attacks that exploit the safety and control parameters of the physical devices.

6.3 Security Services using Emerging Technologies

Currently there is considerable interest for making use of emerging technologies for addressing specific challenges and enhancing security in critical infrastructures [30, 33, 58]. For instance, [30] proposed how deep learning can be used to address specific challenges in robotics and [33] shows how blockchain can be used to enhance security in smart manufacturing. Surminski et al. [51] proposed software based remote attestation techniques which do not require custom hardware extensions or trusted computing components to verify the correct functionality of an untrusted remote devices in realtime systems. Alcaraz and Lopez [2] presented a detail survey on how digital twins are being used for analysing the threats in industry 4.0 and made recommendations for trustworthy use of digital twins. Alcaraz et al. has also proposed [3] techniques for dynamic enforcement of fine granular role based access control policies for securing smart grids. There are some survey works on DT which presents how they can be used to deal with security issues in ICS domain [13, 55]. Our work is focused on using SDN and NFV for addressing the specific challenges and enhancing the security in critical infrastructures. We also demonstrated how DT is used for analysing the impact of ransomware on the critical infrastructures and used CSSA for dynamic generation of policy rules to minimise the threat on the critical services. Furthermore, techniques such as [3] can be used for extending the architecture for fine granular enforcement of policies using RBAC.

There is also prior work [12, 14, 28, 49, 55] for using SDN for securing the critical control systems. Yang et al. in [55] have proposed a data-driven smart manufacturing infrastructure which uses SDN. The proposed architecture uses edge clouds for real-time collection, storage and processing of manufacturing systems data. After collection, it uses cloud applications and SDN to adjust the whole manufacturing infrastructure dynamically to better control and maximise production. However, it paves the way for converting physical systems using programmable networks and devices with the power of fine-grained collection of data and policies stored in the cloud. Lallo et al. [12] proposed to use spare bandwidth to replicate traffic to the IDS node for detecting attacks. However, there may not be any spare bandwidth during DoS attacks. Kurtz and Weitfeld [28] proposed to make use of multiple SDN Controllers for securing the critical infrastructures with broker based voting mechanism to detect faulty and malicious SDN Controllers. Erokhin et al. [14] proposed to analyse traffic statistics in the Controller application for detecting the attacks in critical information infrastructure.

However, it has to be noted that SDN also comes with specific security challenges. There is ongoing work [9, 32, 36, 47] for addressing security issues related to SDN technology. VNFs are dynamic and allow rapid deployment of services (such local/global policies, IDS, ML-based anomaly detection, etc.) over network domains. Recently, ML-based VNF services are used in IIoT, CPS, ICS, and edge networks to detect and mitigate attacks in such networks [5, 11, 19, 46, 59]. Oliveira et al. in [11] proposed how intelligently VNFs can be deployed in IIoT networks to mitigate DDoS attacks. Their work presents tree-based structures, XGBoost, Decision Tree and Random Forest, suitable for detecting DDoS attacks in the IIoT networks. However, the influencing factor for the ML model in their study was the number of scenarios for the models. They also suggested the

VNF placement closest to the attacker helps to mitigate the DDoS attacks effectively. We have proposed a software enabled architecture that is based on SDN and NFV technologies. We have also described how our architecture is able to address the specific security requirements of the control systems. In particular, we have demonstrated how our security architecture can be used for achieving real time situational awareness in control systems, establishing secure communication paths between devices, fine granular monitoring of the flows for detecting the attacks and isolation of malicious devices in control system network.

7 CONCLUDING REMARKS

In this paper, we have proposed a software enabled security architecture for securing control systems. Our security architecture uses SDN technology to achieve dynamic policy driven decision in critical control systems. Such an approach enables dynamic isolation of control system components and network devices that are vulnerable to security attacks. The security architecture uses NFV technology to provision dynamically security functions at the required places in the network infrastructure thereby enhancing the security capability of control system components. We have demonstrated that this combined SDN and NFV enabled security architecture can be used to protect control systems from attacks such as denial of service attacks, from unpatched vulnerable control system components as well as to secure the communication flows from the legacy devices that do not support any security functionality.

REFERENCES

[1] Maxat Akbanov, Vassilios G. Vassilakis, Ioannis D. Moscholios, and Michael D. Logothetis. 2018. Static and dynamic analysis of WannaCry ransomware. In *Proc. IEICE Inform. and Commun. Technol. Forum ICTF*, Vol. 2018.

[2] Cristina Alcaraz and Javier Lopez. 2022. Digital twin: A comprehensive survey of security threats. *IEEE Communications Surveys & Tutorials* (2022).

[3] Cristina Alcaraz, Javier Lopez, and Stephen Wolthusen. 2016. Policy enforcement system for secure interoperable control in distributed smart grid systems. *Journal of Network and Computer Applications* 59 (2016), 301–314.

[4] Martín Barrère, Chris Hankin, Nicolas Nicolaou, Demetrios G. Eliades, and Thomas Parisini. 2020. Measuring cyber-physical security in industrial control systems via minimum-effort attack strategies. *Journal of Information Security and Applications* 52 (2020), 102471.

[5] Abderrahmane Boudi, Ivan Farris, Miloud Bagaa, and Tarik Taleb. 2019. Assessing lightweight virtualization for security-as-a-service at the network edge. *IEICE Transactions on Communications* 102, 5 (2019), 970–977.

[6] Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry. 2008. Research challenges for the security of control systems. *HotSec* 5 (2008), 15.

[7] Eric Chien, Liam OMurchu, and Nicolas Falliere. 2012. W32. Duqu: The precursor to the next Stuxnet. In *5th USENIX Workshop on Large-Scale Exploits and Emergent Threats*.

[8] D. D. Clark. 1989. Policy routing in Internet protocols. Request for Comment RFC-1102. *Network Information Center* (1989).

[9] Marc C. Dacier, Hartmut König, Radoslaw Cwalinski, Frank Kargl, and Sven Dietrich. 2017. Security challenges and opportunities of software-defined networking. *IEEE Security Privacy* 15, 2 (March 2017), 96–100. https://doi.org/10.1109/MSP.2017.46

[10] Alvise de Faveri Tron, Stefano Longari, Michele Carminati, Mario Polino, and Stefano Zanero. 2022. CANflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks *(CCS'22)*. Association for Computing Machinery, New York, NY, USA, 711–723. https://doi.org/10.1145/3548606.3560618

[11] Guilherme Werneck de Oliveira, Michele Nogueira, Aldri Luiz dos Santos, and Daniel Macêdo Batista. 2023. Intelligent VNF placement to mitigate DDoS attacks on industrial IoT. *IEEE Transactions on Network and Service Management* (2023).

[12] Roberto Di Lallo, Federico Griscioli, Gabriele Lospoto, Habib Mostafaei, Maurizio Pizzonia, and Massimo Rimondini. 2017. Leveraging SDN to monitor critical infrastructure networks in a smarter way. In *2017 IFIP/IEEE IM*. IEEE.

[13] Marietheres Dietz and Gunther Pernul. 2020. Unleashing the digital twin's potential for ICS security. *IEEE Security & Privacy* 18, 4 (2020), 20–27.

[14] Sergey Erokhin, Andrey Petukhov, and Pavel Pilyugin. 2019. Critical information infrastructures monitoring based on software-defined networks. In *Proceedings of the 24th Conference of Open Innovations Association FRUCT*. FRUCT Oy, 83.

[15] Open Networking Foundation. 2012. *Software-Defined Networking: The New Norm for Networks*. Technical Report. https://http://opennetworking.wpengine.com/wp-content/uploads/2011/09/wp-sdn-newnorm.pdf
[16] Tal Garfinkel and Mendel Rosenblum. 2003. A virtual machine introspection based architecture for intrusion detection. In *NDSS*, Vol. 3. San Diego, CA, 191–206.
[17] Christian Gehrmann and Martin Gunnarsson. 2019. A digital twin based industrial automation and control system security architecture. *IEEE Transactions on Industrial Informatics* 16, 1 (2019), 669–680.
[18] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. 2018. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.
[19] Nadra Guizani and Arif Ghafoor. 2020. A network function virtualization system for detecting malware in large IoT based networks. *IEEE Journal on Selected Areas in Communications* 38, 6 (2020), 1218–1228.
[20] Monowar Hasan and Sibin Mohan. 2019. Protecting actuators in safety-critical IoT systems from control spoofing attacks. In *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*. 8–14.
[21] Kevin E. Hemsley and E. Fisher. 2018. *History of Industrial Control System Cyber Incidents*. Technical Report. Idaho National Lab. (INL), Idaho Falls, ID (United States).
[22] NFV ISG. 2013. *Network Functions Virtualisation (NFV)-Virtual Network Functions Architecture*. Technical Report. ETSI, Tech. Rep.
[23] Sushma Kalle, Nehal Ameen, Hyunguk Yoo, and Irfan Ahmed. 2019. CLIK on PLCs! attacking control logic with decompilation and virtual PLC. In *Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS)*. 1–12.
[24] Kallol Krishna Karmakar, Vijay Varadharajan, Pete Speirs, Michael Hitchens, and Aron Robertson. 2022. SDPM: A secure smart device provisioning and monitoring service architecture for smart network infrastructure. *IEEE Internet of Things Journal* 9, 24 (2022), 25037–25051. https://doi.org/10.1109/JIOT.2022.3195227
[25] Jonguk Kim, Jeong-Han Yun, and Hyoung Chun Kim. 2019. Anomaly detection for industrial control systems using sequence-to-sequence neural networks. *arXiv preprint arXiv:1911.04831* (2019).
[26] Ayaka Koshibe. 2016. ONOS cluster coordination. https://wiki.onosproject.org/display/ONOS/Cluster+Coordination Accessed 10 August 2022.
[27] Brian Krebs. 2008. Cyber incident blamed for nuclear power plant shutdown. *Washington Post, June* 5 (2008), 2008.
[28] Fabian Kurtz and Christian Wietfeld. 2017. Advanced controller resiliency in software-defined networking enabled critical infrastructure communications. In *2017 ICTC*. IEEE, 673–678.
[29] Dominik Kus, Eric Wagner, Jan Pennekamp, Konrad Wolsing, Ina Berenice Fink, Markus Dahlmanns, Klaus Wehrle, and Martin Henze. 2022. A false sense of security? Revisiting the state of machine learning-based industrial intrusion detection. In *Proceedings of the 8th ACM on Cyber-Physical System Security Workshop*. 73–84.
[30] Artúr István Károly, Péter Galambos, József Kuti, and Imre J. Rudas. 2021. Deep learning in robotics: Survey on model structures and training strategies. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 1 (2021), 266–279. https://doi.org/10.1109/TSMC.2020.3018325
[31] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9, 3 (2011), 49–51.
[32] Seungsoo Lee, Changhoon Yoon, Chanhee Lee, Seungwon Shin, Vinod Yegneswaran, and Phillip A. Porras. 2017. DELTA: A security assessment framework for software-defined networks. In *Proc. of NDSS*, Vol. 17.
[33] Jiewu Leng, Shide Ye, Man Zhou, J. Leon Zhao, Qiang Liu, Wei Guo, Wei Cao, and Leijie Fu. 2021. Blockchain-secured smart manufacturing in industry 4.0: A survey. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 1 (2021), 237–252. https://doi.org/10.1109/TSMC.2020.3040789
[34] Tamas K. Lengyel, Steve Maresca, Bryan D. Payne, George D. Webster, Sebastian Vogl, and Aggelos Kiayias. 2014. Scalability, fidelity and stealth in the DRAKVUF dynamic malware analysis system. In *Proceedings of the 30th Annual Computer Security Applications Conference*. 386–395.
[35] Javier Lopez, Juan E. Rubio, and Cristina Alcaraz. 2021. Digital twins for intelligent authorization in the B5G-enabled smart grid. *IEEE Wireless Communications* 28, 2 (2021), 48–55.
[36] Eduard Marin, Nicola Bucciol, and Mauro Conti. 2019. An in-depth look into SDN topology discovery mechanisms: Novel attacks and practical countermeasures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1101–1114.
[37] Stephen McLaughlin, Charalambos Konstantinou, Xueyang Wang, Lucas Davi, Ahmad-Reza Sadeghi, Michail Maniatakos, and Ramesh Karri. 2016. The cybersecurity landscape in industrial control systems. *Proc. IEEE* 104, 5 (2016), 1039–1057.
[38] Swastik Mittal and Frank Mueller. 2021. T-pack: Timed network security for real time systems. In *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 20–28.
[39] Abubakar Sadiq Mohammed, Eirini Anthi, Omer Rana, Neetesh Saxena, and Pete Burnap. 2023. Detection and mitigation of field flooding attacks on oil and gas critical infrastructure communication. *Computers & Security* 124 (2023), 103007.

[40] Abubakar Sadiq Mohammed, Philipp Reinecke, Pete Burnap, Omer Rana, and Eirini Anthi. 2022. Cybersecurity challenges in the offshore oil and gas industry: An industrial cyber-physical systems (ICPS) perspective. *arXiv preprint arXiv:2202.12179* (2022).
[41] Marcin Nawrocki, Thomas C. Schmidt, and Matthias Wählisch. 2019. Uncovering vulnerable industrial control systems from the internet core. *arXiv preprint arXiv:1901.04411* (2019).
[42] Matthias Niedermaier, Alexander von Bodisco, and Dominik Merli. 2019. CoRT: A communication robustness testbed for industrial control system components. *arXiv preprint arXiv:1904.04286* (2019).
[43] Chen Peng, Hongtao Sun, Mingjin Yang, and Yu-Long Wang. 2019. A survey on security communication and control for smart grids under malicious cyber attacks. *IEEE Transactions on Systems, Man, and Cybernetics* 49, 8 (2019), 1554–1569. https://doi.org/10.1109/TSMC.2018.2884952
[44] Kevin Poulsen. 2003. Slammer worm crashed Ohio nuke plant network. *http://www. securityfocus. com/news/6767* (2003).
[45] Sanjeev Rao, Majid Ghaderi, and Hongwen Zhang. 2022. CloudPAD: Managed anomaly detection for ICS. In *Proceedings of the 4th Workshop on CPS & IoT Security and Privacy* (Los Angeles, CA, USA) *(CPSIoTSec'22)*. Association for Computing Machinery, New York, NY, USA, 55–61. https://doi.org/10.1145/3560826.3563383
[46] Rishi Sairam, Suman Sankar Bhunia, Vijayanand Thangavelu, and Mohan Gurusamy. 2019. NETRA: Enhancing IoT security using NFV-based edge traffic analysis. *IEEE Sensors Journal* 19, 12 (2019), 4660–4671.
[47] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. 2013. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine* 51, 7 (2013), 36–43.
[48] Dmitry Shalyga, Pavel Filonov, and Andrey Lavrentyev. 2018. Anomaly detection for water treatment system based on neural network with automatic architecture optimization. *arXiv preprint arXiv:1807.07282* (2018).
[49] Mohammad Javad Shayegan and Amirreza Damghanian. 2023. A review of methods to prevent DDOS attacks using NFV and SDN. In *2023 9th International Conference on Web Research (ICWR)*. IEEE, 346–355.
[50] Ruimin Sun, Alejandro Mera, Long Lu, and David Choffnes. 2021. SoK: Attacks on industrial control logic and formal verification-based defenses. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 385–402.
[51] Sebastian Surminski, Christian Niesler, Ferdinand Brasser, Lucas Davi, and Ahmad-Reza Sadeghi. 2021. RealSWATT: Remote software-based attestation for embedded devices under realtime constraints. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2890–2905.
[52] Benjamin E. Ujcich, Samuel Jero, Anne Edmundson, Qi Wang, Richard Skowyra, James Landry, Adam Bates, William H. Sanders, Cristina Nita-Rotaru, and Hamed Okhravi. 2018. Cross-app poisoning in software-defined networking. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 648–663.
[53] Vijay Varadharajan, Kallol Karmakar, Uday Tupakula, and Michael Hitchens. 2018. A policy-based security architecture for software-defined networks. *IEEE Transactions on Information Forensics and Security* 14, 4 (2018), 897–912.
[54] Jean-Paul A. Yaacoub, Ola Salman, Hassan N. Noura, Nesrine Kaaniche, Ali Chehab, and Mohamad Malli. 2020. Cyber-physical systems security: Limitations, issues and future trends. *Microprocessors and Microsystems* 77 (2020), 103201.
[55] Chen Yang, Shulin Lan, Lihui Wang, Weiming Shen, and George G. Q. Huang. 2020. Big data driven edge-cloud collaboration architecture for cloud manufacturing: A software defined perspective. *IEEE Access* 8 (2020), 45938–45950.
[56] Hyunguk Yoo and Irfan Ahmed. 2019. Control logic injection attacks on industrial control systems. In *ICT Systems Security and Privacy Protection: 34th IFIP TC 11 International Conference, SEC 2019, Lisbon, Portugal, June 25-27, 2019, Proceedings 34*. Springer, 33–48.
[57] Changhoon Yoon, Seungsoo Lee, Heedo Kang, Taejune Park, Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. 2017. Flow wars: Systemizing the attack surface and defenses in software-defined networks. *IEEE/ACM Transactions on Networking (TON)* 25, 6 (2017), 3514–3530.
[58] W. Yuan, Z. Li, and C. Y. Su. 2021. Multisensor-based navigation and control of a mobile service robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 4 (2021), 2624–2634. https://doi.org/10.1109/TSMC.2019.2916932
[59] Luying Zhou, Huaqun Guo, and Gelei Deng. 2019. A fog computing based approach to DDoS mitigation in IIoT systems. *Computers & Security* 85 (2019), 51–62.
[60] Nauman Zubair, Adeen Ayub, Hyunguk Yoo, and Irfan Ahmed. 2022. PEM: Remote forensic acquisition of PLC memory in industrial control systems. *Forensic Science International: Digital Investigation* 40 (2022), 301336.

Received 14 November 2022; revised 12 October 2023; accepted 13 October 2023