Controller-in-the-Middle: Attacks on Software Defined Networks in Industrial Control Systems

Joseph Gardiner
Bristol Cyber Security Group,
University of Bristol
Bristol, UK
joe.gardiner@bristol.ac.uk

Adam Eiffert
University of Bristol
Bristol, UK
zg18997@bristol.ac.uk

Peter Garraghan
Computing and Communications,
Lancaster University
Lancaster, UK
p.garraghan@lancaster.ac.uk

Nicholas Race
Computing and Communications,
Lancaster University
Lancaster, UK
n.race@lancaster.ac.uk

Shishir Nagaraja
Computing and Information Sciences,
University of Strathclyde
Glasgow, UK
shishir.nagaraja@strath.ac.uk

Awais Rashid
Bristol Cyber Security Group,
University of Bristol
Bristol, UK
awais.rashid@bristol.ac.uk

ABSTRACT

Programmable networks are an area of increasing research activity and real-world usage. The most common example of programmable networks is software defined networking (SDN), in which the control and data planes are separated, with switches only acting as forwarding devices, controlled by software in the form of an SDN controller. As well as routing, this controller can perform other network functions such as load balancing and firewalls. There is an increasing amount of work proposing the use of SDN in industrial control systems (ICS) environments. The ability of SDN to dynamically control the network provides many potential benefits, including to security, utilising the dynamic orchestration of security controls. However, the centralisation of network control results in a single point of failure within the system, and thus potentially a major target of attack. An attacker who is capable of controlling the SDN controller gains near full control of the network. In this paper, we describe and analyse this very scenario. We demonstrate a number of simple, yet highly effective, attacks from a compromised SDN controller within an ICS environment which can break the real-time properties of industrial protocols, and potentially interfere with the operation of physical processes.

CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; **Maintainability and maintenance**; • **Security and privacy** → **Embedded systems security**.

KEYWORDS

Industrial Control Systems; ICS SecurityCritical infrastructure; SCADA; Operational Technology; Cyber physical security;

CPSIOTSEC'21, November 15, 2021, Seoul, South Korea

ACM Reference Format:
Joseph Gardiner, Adam Eiffert, Peter Garraghan, Nicholas Race, Shishir Nagaraja, and Awais Rashid. 2021. Controller-in-the-Middle: Attacks on Software Defined Networks in Industrial Control Systems. In *2021 Joint Workshop on CPS & IoT Security and Privacy (CPSIOTSEC'21), November 15, 2021.* ACM, Seoul, South Korea, 6 pages.

1 INTRODUCTION

In recent years, the notion of programmable networks has become a widely researched area. Whilst the control and data planes traditionally reside within the same device and were relatively static and deterministic, the control plane has increasingly shifted into software, allowing for real-time control of network flows. A well-known programmable network model is software-defined networking (SDN), wherein switches are purely forwarding devices which operate using a rule-based flow table populated through the use of a controller. An SDN controller can run multiple applications, providing specific functions past normal routing, including load balancing, firewalls and traffic monitoring. SDN most commonly refers to the specific Openflow [9] protocol for switch-controller communication, and is already in use in major organisations [7].

The additional control provided by SDN has led to it being proposed as a useful tool in the industrial control system (ICS) and supervisory control and data acquisition (SCADA) setting [6, 13, 17, 18]. Whilst SDN networks are usually largely static in nature, SDN can provide benefits such as redundancy, and also be able to assist in security through dynamic firewalls and the ability to react to attacks .

Whilst there are clear benefits to the use of SDN in ICSs, there is however, a downside through the introduction of a single point of failure in the SDN controller [1, 11, 15, 16]. Whilst it is generally accepted that SDN provides a potential vulnerability with a large impact, there is little experimental work demonstrating the impact of such attacks in a real-world setting. Whereas in a traditional network gaining control of the network is an intensive process requiring the compromise of individual switches, or attacking network protocols such as ARP or DNS, within an SDN the controller has almost total control over network routing. This can also potentially include other previously separate network functions such as firewalls. If this controller were to become compromised, the

attacker could gain control over the underlying network, potentially allowing them to directly attack devices through routing, for example by causing denial-of-service through dropping packets, or facilitate attacks such as person-in-the-middle (PITM) attacks. Increasingly SCADA systems are moving from serial connections to TCP-based connections over Ethernet for communication, including both for relaying data to SCADA back ends (which can operate on a slower connection) and for distributed IO between devices (which often use real-time protocols such as Profinet). With an increase in Ethernet-based networking the potential for major impact from a compromised SDN controller is massively increased.

In this paper, we present an analysis of the security of SDN within an ICS setting. We do this by first providing a model of the attacker, and then demonstrating a number of attacks against an ICS network that can be introduced through a compromised SDN controller. To our knowledge, this is the first demonstration of attacks caused by an compromised SDN controller in an industrial control systems setting.

2 BACKGROUND AND RELATED WORK

2.1 Software Defined Networking

In an SDN network, switches act as simple forwarding devices, with the control plane separated from the data plane into an SDN controller. This controller is responsible primarily for making routing decisions for the switch. The most common protocol used between switches and controllers is Openflow [9]. In an OpenFlow SDN network, switches maintain a table of flow rules, which match network flows to actions. For example, if a packet destined for mac address x is seen, then output the packet on port y. On observing a packet for a new flow not in the table, a switch will forward the packet to the controller through the use of a PacketIN message. The controller, on processing the request, will usually respond with a PacketOUT message, which pushes the packet out of a port (or set of ports), and a FlowMod, which contains a new flow rule to be installed within the switches flow table. A flow rule consists of a match field, a set of actions and usually a time-to-live to remove rules that have not been used for so many seconds. This is referred to as the "reactive" mode of control. Controllers can also be "proactive", installing flow rules automatically, or through human input, without requiring a PacketIN. Most networks will use a combination of both approaches. The proactive model is particularly well suited to an ICS environment where the network remains largely static, with relatively few new devices appearing.

There are number of OpenFlow SDN controllers built on different programming languages, for example Ryu (Python) and Floodlight, Opendaylight and ONOS (Java). These are typically deployed on general purpose hardware running standard operating systems.

One of the key aspect of SDN is the use of applications, which are the core components that actually control how the network operates. The controller interacts with applications through the use of a northbound interface, such as a RESTful API. These applications are either packed as part of the controller instance, or could potentially be remote services themselves.

2.2 Proposed Uses of SDN in ICS

There have been a number of proposed uses of SDN in an ICS environment. These most often use SDN as a tool to enable dynamic defences against attack. We discuss a selection of these here.

Silva et al. propose a multipath routing mechanism built using SDN as a method for mitigating eavesdropping attacks [17]. In this approach, shortest paths are computed between pairs of devices (according to Dijkstra's algorithm), and chosen. After a short timeout, the cost of the used path is increased, and the shortest path is recalculated, with the new shortest path then used. This means that the flow changes path frequently making it harder to eavesdrop on a flow for a continuous period.

Another use for SDN as a security mechanism is as a network intrusion detection system (NIDS). Silva et al. propose a one-class NIDS in which the SDN controller collects snapshots of Openflow statistics from switches which are sent to a data historian and then used to detect attacks, relying on the generally static nature of ICS networks [18].

Derhab et al. propose an SDN-WAN based architecture for IDS, which migrates the control layer to the cloud [6], along with a intrusion detection system to detect forged commands to ICS devices, and a blockchain-based integrity checking system for identifying attacks which modify switch flow rules.

One particular use case that has been proposed for SDN in an ICS environment is within smart grids. Rehmani et al, provide a detailed survey of SDN use within smart grids, including the security and privacy scheme within such architectures [13].

3 ATTACKER

We assume a targeted, well resourced attacker focused on a particular network. It is highly unlikely that a "script-kiddie" type attacker using pre-made malware and scripts will focus efforts on the SDN architecture of a network. It is expected that the attacker would have already gained access to the industrial network.

3.0.1 Attacker Goals. The compromise of the SDN controller can both be used to directly launch attacks, as well as to assist in performing traditional host-based attacks within the network. As the attacker gains a large amount of control over the routing of the network, as well as potentially other networking functions such as firewalls, gaining control over the SDN controller could become a key target for an attacker. Some of the specific attacker goals include the following:

Denial-of-Service The attacker wishes to prevent hosts from communicating.

Eavesdropping The attacker wishes to collect the traffic of either a single host (targeted) or group of hosts (indiscriminate), without affecting the availability of the network service.

Data Tampering The attacker wishes to change the contents of packets for a particular flow in order to carry out a person-in-the-middle attack.

Service Degradation The attacker wishes to degrade the performance of the network for a single host (targeted) or all hosts (indiscriminate) in order to make the network unusable, or to introduce errors in external applications that rely on high speed communications.

Attack Augmentation The attacker uses an attack on the SDN network to assist a secondary attack. This could include redirecting flows to assist in person-in-the-middle attacks, or to open holes in the SDN based firewall.

3.1 Attack Vector

We envision the attacker compromising the controller in one of the following ways:

Compromised Administrative Interface Most SDN controllers provide functionality for remote administration of the SDN controller through APIs or admin interfaces (such as web-based pages). If a host is compromised that is authenticated to these interfaces then the attacker can gain access to the control provided by the interface. Typically this will include changing the set of installed applications or manually installing/modifying flow rules on switches.

Compromised Northbound Interface The attacker compromises a northbound interface in order to interact with the controller, for example, by performing a person-in-the-middle attack on a RESTful interface.

Compromised Switch Controller Connection The attacker compromises the openflow connection between the switch and controller, for example by performing a person-in-the-middle attack, and modifies requests and responses.

Malicious application. A malicious application is installed on the controller by either a malicious admin, through admin error or through the compromise of a existing application.

Compromised host. In the most severe case, the host on which the controller resides is compromised, resulting in full attacker control over the controller runtime.

In this paper we focus on the malicious application/compromised host example, where the attacker can gain control over the core routing functionality of the SDN controller.

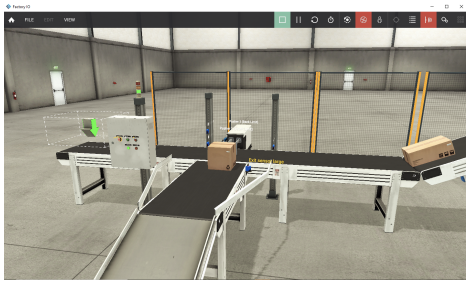
4 ATTACKS

In this section we propose a number of attacks, and where appropriate use a small scale demonstration setup using four difference industrial protocols to demonstrate the attacks.

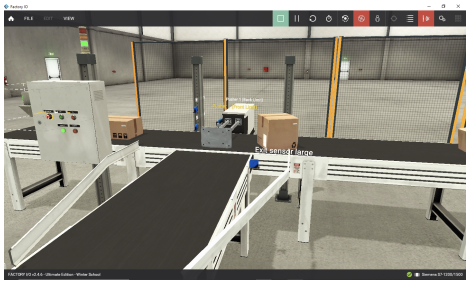
4.1 Setup

4.1.1 "Physical" Process. In order to measure the effect on a physical process, we make use of FactoryIO from RealGames [1]. FactoryIO is designed to be used for learning PLC programming, with the ability to build large scale factory simulations which are controlled by real-world devices. The software talks to PLCs using an Ethernet connection and is able to utilise a number of protocols, including S7Comm, Ethernet/IP and Modbus. This allows us to easily connect it up to different devices over our SDN network and measure the impact of the attacks on the different protocols. The scene we test with is a simple sorting scene, in which small and large boxes are moved down a conveyor belt and measured. The larger, taller boxes are pushed onto a secondary conveyor, whilst the shorter boxes are not. An example of this scene can be seen in Figure 1a.

[1] https://factoryio.com



(a) FactoryIO scene used for testing. The larger, cube-shaped boxes are pushed off the main conveyor, the rectangular boxes are not



(b) FactoryIO when under the flow rule blocking attack

4.1.2 SDN. Networking is provided by a Dell EMC PowerSwitch S3048-ON 1000BASE-T 48-port 1GbE top-of-rack (ToR) switch, which features support for SDN using Openflow (versions 1.0 and 1.3), and can operate with 3rd party controllers and operating systems (the "ON" portion of the model number represents "Open Networking"). The switch is running Dell EMC Networking OS9 (specifically 9.13), and has been configured to use Openflow 1.0. We use the Floodlight controller [12] to provide network control.

4.1.3 Devices and Protocols. We make use of four common ICS protocols in our testing:

Profinet - Profinet is a real-time protocol commonly used by Siemens devices. In particular, it is the protocol in use when using remote IO on Siemens devices. We make use of two S2-1200 PLCs running firmware version 4.2. One operates as the controller, and the second as the remote IO device. The laptop running FactoryIO is connected to the remote IO device over Ethernet, using S7Comm to communicate. The SDN switch sits between the two PLCs for these tests.

S7Comm - The S7Comm protocol is the primary protocol used for workstations and SCADA systems to interact with Siemens PLCs. As well as downloading programs to the device, it can be used to read and write memory addresses to the device. As an example, it is common for software such as data historians to use S7Comm to read values from devices. S7Comm is unencrypted, though newer devices use S7CommPlus which do use encryption (though this has been shown to be insecure [5]). S7Comm is used between FactoryIO and the remote IO PLC, with the SDN switch moved between the two. The connection between the two PLCs is instead through a standard Mikrotik switch.

Ethernet/IP - Ethernet/IP is an open protocol, most commonly used by Allen Bradley devices for communication, including remote IO, although unlike Profinet it is not a real-time protocol. We use an Allen Bradley Micro850 PLC, connected to FactoryIO over the SDN switch.

Modbus/TCP - Finally, we use the common Modbus/TCP protocol. We use OpenPLC[2] installed on a Raspberry Pi 4 Model B (4Gb Ram). OpenPLC is an open source PLC commonly used for research projects using the modbus protocol [4]. To use FactoryIO with OpenPLC, FactoryIO runs a Modbus server, which OpenPLC treats as a slave device. The SDN switch sits between OpenPLC and FactoryIO.

The topologies for each protocol are shown in Figure 2. In the diagrams, the labels on lines indicate the protocols in use. Note that the SDN controller is not shown, however it is running on a blade server directly connected to the switches management port. Where two devices are not shown to be connected using a switch, they are connected through a direct Ethernet connection.

4.1.4 Implementing the Attacks. All of the proposed attacks are implemented by simply taking a copy of the default Floodlight "Forwarding" routing application, and then modifying the operation of this application to perform the attack. In most cases this involves the portion of the application that produces the FlowMod packets to send to the switch. For the majority of the proposed attacks, these modifications only require the addition (or in some cases removal) of a few lines of code into the application. Floodlight supports multiple applications to be run at once, with applications able to specify where in the processing order they sit, and so we force our malicious application to be the first to process requests. Requests for non-targeted flows are processed as normal.

4.2 Denial-Of-Service

In this attack, the malicious application clears the flow table of the switch, and then proceeds to re-build the switches flow table. For any flow rules destined for a target device, the controller installs a flow rule with no action fields, which means that any matching packets are dropped by the switch, performing a denial-of-service.

4.3 Flow Rule Blocking

In this attack, the malicious application takes over the routing for flows for target devices, while ignoring other flows and allowing them to be processed as usual. Whereas in the normal case the routing application, on receiving a PacketIN, will send both a PacketOUT and a FlowMod to install a flow rule into the table, in the attack case the malicious application only sends a PacketOUT. This means that the switch needs to send all packets of the target flow to the controller for processing, producing a large amount of additional latency. The malicious application can add arbitrary amounts of delay to increase this latency further.

Demonstration. We apply this attack to all four protocols in our test setup. The attack is targeted to the PLC device; any other flows will be unnafected. On first seeing a request to the target device, the malicious application will allow any handshakes to complete, and after 20 second clear the flow table of the switch and then apply the attack. We apply flow rule blocking with no additional delay introduced by the controller (except the delay of contacting the controller itself, which is 3-4ms), and also with artificial additional delay on sending responses from the controller to measure how much extra latency is enough to cause the physical process to no longer sort blocks.

[2]https://www.openplcproject.com

Protocol	Real-time?	Operates with blocking?	Physical process affected?	Additional delay for failure
Profinet	Yes	No	Yes	-
S7Comm	No	Yes	No	40ms
Ethernet/IP	No	Yes	Yes	10ms
Modbus/TCP	No	Yes	Yes	10ms

Table 1: Results of flow-rule blocking attack.

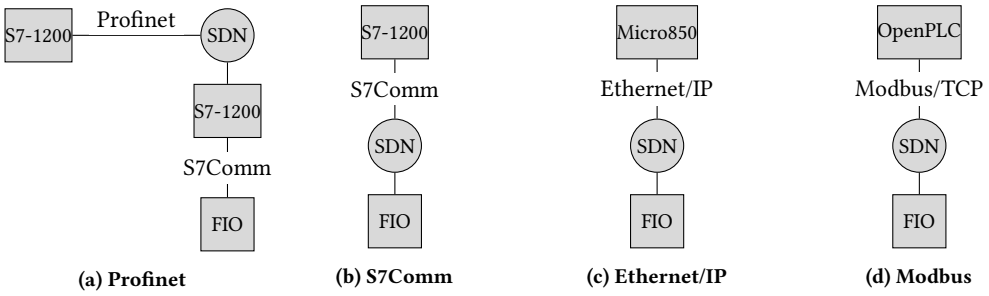
We presents the results of this attack on the four protocols in Table 1. As can be seen in the third column, all of the protocols, other that Profinet, were able to operate whilst under this attack. Profinet, the only real-time protocol, quickly raises an alarm once the attack begins. On inspecting the requests to the controller, Profinet messages are sent at a sufficient rate that the controller is unable to process requests and forward packets quick enough and a backlog forms. This, along with the additional latency, breaks the real-time properties of the protocol and causes Profinet to fail.

For those protocols that still managed to operate when under attack, only S7Comm showed no obvious impact on the physical process. Both Ethernet/IP and Modbus/TCP, when faced with the additional latency of passing packets through the SDN controller, both exhibited a noticeable lag in the pusher operation in the physical process, pushing blocks late and in some cases late enough that the block remain on the main conveyor. This clearly shows that there is a potential safety impact from this attack, as the reaction time of the process for any aspects which rely on this network communication are impaired.

Finally, we measure how much additional latency is required to cause the pusher to completely miss blocks, preventing sorting. S7Comm requires 40ms of additional delay, whilst Ethernet/IP and Modbus/TCP both only required 10ms additional controller delay to fail. This small value indicates that both Ethernet/IP and Modbus/TCP could be vulnerable to other SDN attacks that increase latency, for example through increasing path lengths. This effect can be seen in Figure 1b.

4.4 Controller Packet Tampering

We extend the previous attack to perform a controller based packet tampering attack. Depending on the particular switch, packets are either buffered on the switch, with only packet headers forwarded to the controller for processing, or the full packet, including data, is sent to the controller for processing. In cases where the full packet is sent to the controller, and the protocol is unencrypted (as S7Comm, Ethernet/IP and Modbus/TCP all are by default) we can make arbitrary modifications to the packets to perform a person-in-the-middle attack. The advantage of this attack is that no new routes are created – the switch already communicates with, and forwards packets to,the controller. The primary effect on the network is a large increase in the volume of packets sent from the switch to controller. In this example, we wish to overwrite the values relating to the pusher in our example process, to prevent blocks from being pushed off. Packets are modified by modifying the packet data when the PacketOUT message is created by the



(a) Profinet **(b) S7Comm** **(c) Ethernet/IP** **(d) Modbus**

Figure 2: Topologies used for testing. SDN = SDN Switch. FIO = FactoryIO host.

malicious forwarding application. Note that we cannot apply this to the Profinet connection, as the connection fails if packets are routed through the controller.

One small issues arose when developing this attack. If the data is simply changed, the TCP checksum is then incorrect and FactoryIO disconnects. This means that our malicious application needs to deconstruct the TCP layer of the packet, modify the TCP payload, and then rebuild the TCP header. This adds a very small amount of additional latency to the connection.

Demonstration. For S7Comm, FactoryIO uses a ReadVar S7Comm packet to read the state of the output addresses on the PLC. Within the response, the final 2 bytes of the packet, containing the read values, are returned, which are “B100” when the pusher is not being pushed, and "B900" when the pusher is active. Therefore, for any packets from the target device which contain "B900" as the final 2 bytes, these are replaced by "B100", successfully preventing the pusher from operating.

In the Modbus/TCP setup, OpenPLC updates the coil state to FactoryIO using the “Write Multiple Coils” request type, sending 2 bytes of data over. When the pusher is inactive, these 2 bytes are “B300”, changing to“BB00” when the attack is active. Similarly to S7Comm, these are the final 2 bytes of the packet. Again, the attack successfully blocks the pusher from operating.

The attack is a little more complicated in the case of Ethernet/IP. Whereas S7Comm and Modbus/TCP both request/write all of the output registers in a single request, within Ethernet/IP each individual output register is requested individually. This means that we need to observe the request from FactoryIO to the PLC for the register corresponding to the pusher (in this case “BOOL_OUT_3”, and then only modify the following response, replacing the final 3 bytes of the packet - “C10001” with “C10000”. As with this other two protocols, however, the attack is successful.

In this demonstration we prevent the pusher from pushing, however the same attack could be applied to, for example, overwrite values being sent to a SCADA server, potentially preventing alarms from being raised.

4.5 Controller Eavesdropping

A second extension to the Flow Rule Blocking attack is the ability of the controller to eavesdrop on packets that are sent to the controller, in particular if packets are not buffered by the switch and full packet contents are forwarded to the controller. For an attacker there are two obvious benefits to this. First, it is useful in asset discovery, as it reveals IPs, MAC addresses and protocols in use to the controller. Further, in some protocols, such as S7Comm, device specific details such as model numbers are sent as part of the protocol which can also aid in asset discovery. The second benefit is that if the protocol in operation can be observed, then the attacker can potentially learn about the behaviour of the underlying process by observing transmitted data. For example, Ethernet/IP reads and writes each individual register, including the register name, as individual requests. If these are used for remote IO or a data historian, an attack could monitor these over a period to learn which registers could be tampered with to affect the process.

4.6 Third-party Eavesdropping

A flow rule can have multiple action fields. If a flow rule has multiple action fields instructing the switch to forward packets to multiple ports, then the switch will mirror packets to those ports. This can allow an attacker to mirror traffic to a collection point to perform passive asset discovery and eavesdropping, without requiring capture in promiscuous mode. This does require the attacker to know the port number of the switch to which their eavesdropping machine is connected. This is information stored by the SDN controller itself and would be visible on any administrative interface.

Demonstration. Our malicious forwarding application, on creating a flow rule, adds an additional action field to output packets to the port where our attack laptop is connected, on which we run Wireshark, As soon as a new target flow is setup, then Wireshark will start capturing all traffic on the flow. S7Comm packets only appear into the network capture when the attack is started, as the laptop does not usually have visibility onto those packets, even when running in promiscuous mode.

It is important to note that for established flows, the simple version of this attack will not work as the controller will not be contacted to install the flow rule for the target device. The controller will have to direct the switch to delete the existing flow rule, and then install an updated flow rule proactively. The controller can do this in one command, which will prevent any disruption to the existing network.

4.7 SDN Enabled Person-in-The-Middle

In this attack, we use the routing capability of the compromised SDN controller in order to allow a attacker controlled device on

the network to perform a PITM attack. This attack clearly demonstrates the ability of an SDN attacker to modify the routing with an SDN. Traditionally, a technique such as ARP spoofing is required to redirect flows to an attack controller machine to perform a PITM attack. Instead, the attacker, who controls a machine within the target network, redirects a target flow through the use of malicious flow rules installed on switches. This has the benefit that the attacker machine itself only needs to receive and send on packets, and not launch the attack itself through a technique such as ARP spoofing, which is easily detectable and already covered well by intrusion detection systems.

Demonstration. We demonstrate this attack using FactoryIO with a Siemens S7-1200 PLC using S7Comm. We introduce a third device into the network, a laptop running Ubuntu 18.04. The laptop has two Ethernet connections to the switch, and on the laptop a virtual network switch is deployed using OpenVSwitch (OVS), with both physical adapters added as ports to the virtual switch. The OVS switch is controlled by its own Floodlight instance, which simply directs packets out of the adapter which the traffic did not come in on, effectively making the virtual switch a proxy. The OVS controller instance also has our controller packet tampering application installed, and so will intercept all target packets and modify them in the same way as our previous attack, however this behaviour is contained to the attacker laptop. On the core SDN switch, flow rules are manually installed through a malicious application to direct target flows through the laptop.

Through this attack, we are successfully able to perform a person-in-the-middle attack without the use of traditional techniques such as ARP spoofing. Within the data plane there are no unusual packets, only flows taking an unusual route. As with the controller packet tampering attack, as well as interfere with the physical process this attack could be used to modify packets sent back to SCADA systems to prevent monitoring and alarms.

4.8 Attack Mitigation

By installing monitoring on the control network between switches and controllers, and applying anomaly detection, attacks such as the flow rule blocking and its derived attacks (controller packet tampering and controller eavesdropping) could be detected. When these attacks are applied, the switches will generate a large volume of requests to the controller, far higher than the number expected in a relatively static ICS network. This anomalous behaviour can be easily identified.

For less obvious attacks that modify routing, such as the SDN enabled person-in-the-middle attack, identifying when these have occurred is a more difficult problem, as the network operation whilst under attack is not too dissimilar to normal network operation. There are a number of works that propose systems for detecting malicious flows in SDN, such as [2, 3, 8, 10, 14].

If SDN were to be deployed into an ICS network, it is important that the host on which the controller resides is kept up to date and secured. As SDN controllers generally run on top of desktop operating systems, any host vulnerabilities, including poor configuration such as insecure passwords, can lead to controller compromise, therefore steps to mitigate these vulnerabilities can go some way to preventing these SDN attacks.

5 CONCLUSION

With the increase attention on programmable networks, there have been a number of proposed uses for software defined networking in ICS environments. Whilst the use of SDN can provide a large number of benefits, in particular when used to improve security, the use of SDN itself introduces a potential vulnerability by centralising network control into a single point of failure. If the SDN controller were to become compromised, then an attacker could gain a control over the operation of the network, allowing them to either directly attack the ICS from the SDN controller, facilitate further host-based attacks. In this paper, we modelled an attacker who would target the SDN component of an industrial network, and provided, to our knowledge, the first demonstration of a number of attacks on an SDN-based ICS network. We hope that this work leads to further consideration of the security of SDN when used in ICS environments.

REFERENCES

[1] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov. Security in software defined networks: A survey. IEEE Communications Surveys Tutorials, 17(4):2317–2346, 2015.

[2] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani. Securing software defined networks: taxonomy, requirements, and open issues. IEEE Communications Magazine, 53(4):36–44, 2015.

[3] E. Al-Shaer and S. Al-Haj. Flowchecker: Configuration analysis and verification of federated openflow infrastructures. In ACM Workshop on Assurable and Usable Security Configuration, SafeConfig '10, page 37–44, New York, NY, USA, 2010. Association for Computing Machinery.

[4] T. R. Alves, M. Buratto, F. M. de Souza, and T. V. Rodrigues. Openplc: An open source alternative to automation. In IEEE Global Humanitarian Technology Conference (GHTC 2014), pages 585–589, 2014.

[5] E. Biham, S. Bitan, A. Carmel, A. Dankner, U. Malin, and A. Wool. Rogue7: Rogue engineering-station attacks on s7 simatic plcs. In Black Hat 2019, 2019.

[6] A. Derhab, M. Guerroumi, A. Gumaei, L. Maglaras, M. A. Ferrag, M. Mukherjee, and F. A. Khan. Blockchain and random subspace learning-based ids for sdn-enabled industrial iot security. Sensors, 19(14), 2019.

[7] S. Jain et al. B4: Experience with a globally-deployed software defined wan. SIGCOMM Comput. Commun. Rev., 43(4):3–14, Aug. 2013.

[8] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey. Veriflow: Verifying network-wide invariants in real time. In Workshop on Hot Topics in Software Defined Networks, HotSDN '12, page 49–54, New York, NY, USA, 2012. Association for Computing Machinery.

[9] N. McKeown et al. OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev., 38(2):69–74, mar 2008.

[10] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu. A security enforcement kernel for openflow networks. In Hot Topics in Software Defined Networks, HotSDN '12, page 121–126, New York, NY, USA, 2012. Association for Computing Machinery.

[11] A. S. Prasad, D. Koll, and X. Fu. On the security of software-defined networks. In 2015 Fourth European Workshop on Software Defined Networks, pages 105–106, 2015.

[12] Project Floodlight. Floodlight. http://www.projectfloodlight.org/floodlight/.

[13] M. H. Rehmani, A. Davy, B. Jennings, and C. Assi. Software defined networks-based smart grid communication: A comprehensive survey. IEEE Communications Surveys Tutorials, 21(3):2637–2670, 2019.

[14] C. Röpke and T. Holz. Preventing malicious sdn applications from hiding adverse network manipulations. In Workshop on Security in Softwarized Networks: Prospects and Challenges, SecSoN '18, page 40–45, New York, NY, USA, 2018. Association for Computing Machinery.

[15] S. Scott-Hayward, G. O'Callaghan, and S. Sezer. SDN security: A survey, pages 1–7. Institute of Electrical and Electronics Engineers (IEEE), 2013.

[16] Z. Shu, J. Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran. Security in software-defined networking: Threats and countermeasures. Mob. Netw. Appl., 21(5):764–776, Oct. 2016.

[17] E. G. d. Silva, L. A. D. Knob, J. A. Wickboldt, L. P. Gaspary, L. Z. Granville, and A. E. Schaeffer-Filho. Capitalizing on sdn-based scada systems: An anti-eavesdropping case-study. 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 165–173, 2015.

[18] E. G. d. Silva, A. Silva, J. Wickboldt, P. Smith, L. Granville, and A. Schaeffer-Filho. A one-class nids for sdn-based scada systems. pages 303–312, 06 2016.