These aren't the PLCs you're looking for: Obfuscating PLCs to mimic Honeypots

Sam Maesschalck, *Member, IEEE*, Will Fantom, Vasileios Giotsas, Nicholas Race, *Member, IEEE*

***Abstract*—Industry 4.0 and the trend of connecting legacy Industrial Control Systems (ICSs) to public networks have exposed these systems to various online threats. To combat these threats, honeypots have been widely used to provide proactive monitoring, detection and deception security capabilities. However, skilled attackers are now adept at fingerprinting and avoiding honeypots. Therefore, we take a fundamentally different approach in this paper. Instead of the honeypot representing a real system, we deploy it as a deterrent. Through obfuscation, the aim is to make an attacker believe the real system is a honeypot and collect threat intelligence data on the attacker. To achieve this, we introduce a new obfuscation technique that allows real ICSs to present themselves as honeypots. By taking advantage of honeypot fingerprinting techniques, we are able to deter attackers from interacting with the real Programmable Logic Controller (PLC) within the industrial network. The approach is implemented and evaluated using different penetration testing tools and an expert evaluation highlighting the benefits of obfuscation in that potential adversaries would be misled into assuming the PLC is a honeypot.**

***Index Terms*—Industrial Control Systems, ICS, Programmable Logic Controllers, PLC, Honeypots, Security, Software-Defined Networking.**

I. Introduction

OVER the past decades, adversaries targeting industrial control systems (ICSs) have changed from mainly insider attacks to also include a significant amount of nation-state attacks that leverage the interconnection of IT and OT environments [42, 32]. State-sponsored attacks are often highly sophisticated and meticulous [43], meaning that ICS security capabilities need to be improved in terms of intrusion detection and in terms of preventing exploitation of vulnerabilities. As nation-states have significant resources to attack a system, the focus has to be on identifying compromise before damage has been done. Security solutions must be deployed within the organisational network environment to achieve this.

Honeypots are an essential resource to obtain threat intelligence and identify anomalies within the network. These systems, deployed to attract adversaries, can provide valuable information to security operators [30] and have successfully detected exploitation attempts of zero-day vulnerabilities [37]. Nonetheless, attackers can use anti-honeypot techniques to identify the presence of honeypots and circumvent interacting with them [22, 48]. Therefore, the conventional use of honeypots may not suffice to defend from skilled attackers. We take a fundamentally different approach to honeypots, and instead of investing time to make them behave like a real system, we aim to benefit from the honeypot characteristics.

This paper explores the notion of security by obfuscation and misdirection by making operational Programmable Logic Controllers (PLCs) mimic mimicking honeypot characteristics, similar to how attackers try to obfuscate malicious code during an attack [35]. Obfuscation has been shown to be effective and used as a security measure on a regular basis [51]. For instance, Tor or VPN tunnels are successful in helping users avoid tracking of their traffic and circumvent geoblocking and IP blocklisting [28, 21]. Our system focuses on hiding the PLC by pretending it is a honeypot, and does not interact with the system itself. This means that the proposed technique does not interfere with any operations of the PLC but rather provides a facade of honeypot-like characteristics in front of the PLC, making an adversary who uses anti-honeypot techniques believe the system is a honeypot. Our proposed system does not aim to substitute other security systems but rather to complement them by trying to direct the adversaries' attention away from critical systems.

Due to the critical nature of the devices we are focusing on [13], their limited resources [9], and the impact on safety they can have, our system achieves those goals without interrupting operations. To achieve this, we leverage Software-Defined Networking (SDN) to dynamically route traffic to other devices, isolate devices and virtually move network interfaces within the network. SDN also allows us to monitor all traffic in the network and obtain a broader look at network activity to identify suspicious traffic.

In summary, this paper presents the following contributions:

- Presentation of a new security architecture to obfuscate PLCs as honeypots
- The implementation of our architecture within an ICS and SDN environment
- Evaluation of our proposed obfuscator system

After this introduction, Section 2 provides the necessary background information on software-defined networking and honeypots. Section 3 introduces the theoretical basis on which we have based our obfuscator system. Afterwards, Section 4 discusses the architecture of our proposed obfuscator, and a model of the system follows in Section 5. The next section, Section 6, presents a practical evaluation of the obfuscator within a real environment. Section 7 provides practical considerations when deploying an obfuscator within an ICS environment. Finally, Section 8 presents the conclusion of the paper and future work.

S. Maesschalck is with RHEA Group, Belgium and the School of Computing and Communications, Lancaster University, United Kingdom. W. Fantom, V. Giotsas and N. Race are with the School of Computing and Communications, Lancaster University, United Kingdom. (e-mail: s.maesschalck@rheagroup.com, w.fantom@lancaster.ac.uk, v.giotsas@lancaster.ac.uk, n.race@lancaster.ac.uk

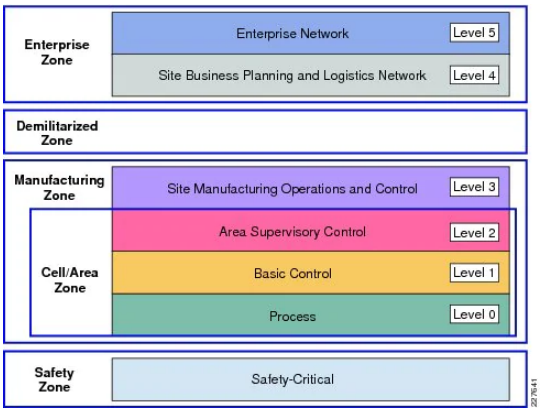

Fig. 1. Purdue Model [8]

II. Background

A. Industrial Control Systems

ICS generally perform control and automation tasks within critical infrastructure, such as the nuclear, water, oil, gas and electricity industries [13, 31]. ICSs are generally deployed to conform to the Purdue Model (Figure 1), which describes the different layers within the Operational Technology (OT) environment. This paper mainly focuses on levels 1 and 2, where human-machine interaction (HMI) devices and programmable logic controllers (PLC) are deployed.

Connecting ICS devices to the Internet enables more sophisticated and flexible processes [26] but introduces ICS to various online threats that these devices were not designed to face [1]. Attacks such as PCaaD [15], show how these devices can be exploited to obtain operational data or enable command and control back channels. Current ICS security solutions are designed to ensure the systems' availability by protecting the IT infrastructure connected to the OT environment [25], partly because any added latency or interference can adversely affect ICS operations [20]. Therefore, ICS security mechanisms need to guarantee minimal operational interference.

B. Software-Defined Networking (SDN)

SDN has enabled the design of programmable and reconfigurable networks, minimising the need for physical changes in cabling and topology. Although SDN has not yet been propagated throughout ICS networks, there are many opportunities for the OT environment to benefit from it. The ability to amalgamate a copy of all traffic on the network and route it to an Intrusion Detection System can significantly improve security within the OT network [24]. Additionally, SDN can allow automatic evaluation of network security policies, which can be both time-consuming and disruptive in a non-SDN environment [38]. However, the presence of SDN controllers may pose a single point of failure, which, if compromised, can provide adversaries with full control of the networks, particularly with the weak security of industrial protocols [11].

C. Honeypots

The approach taken by honeypots to improve security in the network differs from traditional security systems. Their value generally lies in being exploited [44] and using the adversary's actions against them instead of keeping the adversary at bay. However, the way they are deployed impacts the value of data obtained through them [29]. Previous research into the applicability of ICS honeypots for OT security [30] has shown that honeypots can address the requirements for critical infrastructure security, which range from technical requirements, i.e. detection of intrusion attempts, to regulatory and compliance requirements related to frameworks such as the UK Cyber Assessment Framework and the NIST Cybersecurity Framework. Other recent work has proposed semi-virtual honeypots for ICS to reduce the cost of honeypot deployments [53]. Or the development of a honeypot with the capability to support a wide variety of vendors, simulating a plethora of protocols to an advanced level [27]. However, a recent study [30] has shown that the current ICS honeypot deployments within the research literature are not always convincing and are constantly aiming to be more like a real system.

D. Previous Work on SDN Honeypots

There has been previous work in the area of honeypots and SDN. However, to our knowledge, we are the first ones to develop an obfuscator-type honeypot.

SDN has been proposed for many purposes, such as creating larger honeypot systems or honeynets, managing multiple honeypots centrally, or redirecting traffic within a network. For example, in [50], the authors propose an architecture using SDN in combination with a hybrid honeypot system to simulate the network topology and achieve attack traffic migration. The attack traffic migration focuses on the redirection of attacks based on their level of sophistication. Honeypots and SDN have also been used to defend against botnets [19], reducing the infection rate within a network by using SDN to forward traffic to the honeypots and block loaders. Another paper [23] describes using SDN to monitor internal network traffic between different honeypots. Allowing for multiple different honeypots to be centrally managed by HoneyProxy and transfer attacks to the relevant honeypots. The authors of [5] proposed using SDN to enable freedom in implementing their proposed honeypot, which simulates several industrial processes in its Mime Plant module. SDN has also been mentioned as a tool to extend honeypot functionalities to enable better data analytics, packet inspection, and detection[4]. A paper [10] related to our work focused on using honeypots in an SDN environment to defend against DDoS attacks and investigate an anti-honeypot attack that can identify honeypots in the network. The anti-honeypot technique focuses on the attacker recognising the existence of honeypots in the network and, subsequently, the types of honeypots. This is of interest to us as we rely on the attacker identifying the honeypot to protect the real system.

We take this one step further by utilising SDN to redirect traffic and turn a honeypot and real PLC into a singular system from a network perspective. Allowing operational traffic to flow freely to the target system and presenting more protocols and features to an attacker. Using the honeypot as a type of scarecrow where the scarecrow is also the crop, warning

adversaries that they might be interacting with a defensive system that monitors their activity when, in fact, they have identified a real system. Luring them away from the actual PLC.

III. Obfuscating Industrial Control Systems

This paper focuses on a component of ICSs, namely PLCs. These controllers control other devices within the industrial environment, such as a manufacturing process and output signals that change the state of physical devices, such as valves. What we aim to do with these systems is implement a different approach to using honeypots within the environment. Rather than deploying honeypots and making them look like a real system, we are making PLCs look like honeypots. This is what we define as obfuscation within this paper, obscuring the PLC by making it pretend to be a honeypot. Deploying a honeypot without any signatures, such as no active connections going to it and no actual data being on the system can be challenging. Instead of making everything look like a real system, why don't we go the other way and make real things look like honeypots?

Fake honeypots have been described in research [40] and present an interesting approach to secure the system and circumvent the issues in deploying realistic honeypots. These systems fulfil both defence and threat intelligence goals. This dual-purpose system has several essential characteristics of itself when looking into deployment. We must ask what is needed to make an adversary believe the system is a honeypot effectively and ensure the 'obfuscator' system does not hinder the system. Especially within industrial control system environments, this is an important aspect that must be considered. Availability is one of the primary focuses of these systems, as they can operate critical processes within a facility such as a nuclear power station.

A. Honeypot Characteristics

Contrary to asking what characteristics real systems have, we have to ask what a honeypot looks like. In our previous survey of ICS honeypots [30], many characteristics were found that could lead to a worse capability to fool adversaries into attacking the deployed honeypot. These ranged from deploying a PLC on Amazon's AWS to deploying the basic configuration of a honeypot tool such as Conpot. Low-interaction honeypot tools such as Conpot have many common signatures themselves when they are not properly deployed, which can lead to a difference in traffic to the system [29]. Another important indication a system might be a honeypot is when no traffic or suspicious traffic is going to the system. For example, a series of messages that keeps repeating itself. Many of these characteristics can be found in the initial state of the Cyber Kill Chain [52], reconnaissance when an attacker does asset discovery.

When evaluating the characteristics of honeypots, it must be noted that many differences between deployments exist. Broadly, these can be linked to their level of interaction (low-, medium- and high-interaction) [30], which are an indication of how much the system responds to the actions an adversary makes. The obfuscator honeypot itself could be viewed as a hybrid-interaction honeypot, with the services running on the obfuscator being low- to medium-interaction and the PLC behind not being an actual honeypot but providing similar functionality to a high-interaction PLC honeypot.

B. Ensuring Availability

As stated previously, availability is one of the most important characteristics of industrial control systems. Therefore, this has to be at the forefront of implementation when considering the use of an obfuscator. When using an obfuscator, the main concern is what would happen when the system went offline. This can happen for many reasons, such as the OS running into an error due to an adversary that gained access to the system or a general issue with the device hardware. Therefore, when implementing an obfuscator, we aim to ensure availability to the PLC in any case. For this, we have leveraged software-defined networking (SDN) to allow us to deploy the obfuscator physically detached from the PLC, but from a network perspective, they will still pose as the same device.

C. Increasing Security

This system's main goal must be to increase the security of the industrial control systems deployed within the network. To achieve this, there is no one solution. We expect the obfuscator to be deployed alongside other security measures and for it to be seen as an extension to those systems. The use of firewalls, secure programming practices and other good security practices has to remain. Especially within critical environments, a wealth of standards and guidelines need to be followed, and honeypots can fit into [30]. Due to the obfuscator behaving similarly to a honeypot, it can also fit within these, although the aim is to discourage adversaries from interacting with the system aside from basic reconnaissance. We aim to divert the attacker's attention from the industrial protocol to the honeypot services by deploying the honeypot around the real PLC. If an adversary were to interact with the obfuscator, there should be a link to other security systems, such as an IDS and alert the SOC.

IV. Obfuscator Architecture

As we are working in ICS, we have to keep the Purdue Model (Figure 1) in mind. This model describes how the OT environment is structured and how the interaction between devices situated on different layers happens. Within this paper, we assume we have a setup that conforms to the model and only focuses on the OT environment, specifically level 2 and level 1. This allows us to focus more on the concept, implementation and working of the obfuscator.

Within this section, we introduce the architecture of our obfuscator, followed by how the obfuscator would be viewed on the network. Finally, we describe how the obfuscator can be implemented within an SDN environment.

A. Obfuscator

The goal of the obfuscator is to have the system resemble itself as a honeypot effectively; in other words, it should make the system look less like an actual PLC. With this, we want to ensure that the system looks more like a honeypot during the asset discovery or reconnaissance phase. We can do this by looking at some existing ICS honeypots, such as Conpot and mimicking their configuration. This means our obfuscator can run services like HTTP, SSH and FTP. However, there is no limit to the services the obfuscator can run. In addition to this, there has to be a system connected to these services that interacts with these services in a suspicious manner. For example, a repeated series of the same messages will have an adversary question if the traffic is real or simulated. For every PLC, a forwarder also needs to be configured so that all communications aside from the industrial control port traffic get sent to the obfuscator instead of the PLC. Industrial traffic can happen at several ports, such as 502 for MODBUS or 102 for Siemens S7comm. Therefore, the obfuscator is twofold. One part is the honeypot, and the other part is the forwarder that enables the honeypot and the obfuscated system to present as one system.

The obfuscator services themselves can be deployed very flexibly. They can range from basic implementations deployed by importing libraries in a small script, such as seen on a low-interaction honeypot, to actual deployments of the services, such as seen with a high-interaction honeypot. However, as these systems are deployed to make attackers believe they are on a honeypot, they should be limited to how extensively these are implemented. The obfuscator should be a clearly worse deployed honeypot than the actual honeypots deployed within the environment. Additionally, all traffic captured by the obfuscator should be logged and sent to a device that is connected to the SIEM (Security Information and Event Management) and SOC (Security Operations Centre).

In terms of the implementation of the forwarder, we would recommend this to be done on the network infrastructure itself, such as through SDN, rather than on the obfuscator honeypot. This is mainly to ensure that if the obfuscator goes down, the PLC still receives all the industrial port traffic. This is important as we do not want to interfere with the device's actual operation. Additionally, the obfuscator's goal is not to prohibit any attacks on the PLC but to make adversaries less likely to carry them out. Therefore, the system does not prohibit an attacker or an engineer from performing any action on the OT protocols running on the PLC and must be accessible; this system needs to be deployed alongside other security measures. When deploying the forwarder, it is crucial that this goes both ways, and the traffic going back to the client/attacker is structured as if it comes from the actual PLC. The obfuscator honeypot should not interfere with the operation of the PLC and preferably be deployed separately to ensure the PLC remains operational when the honeypot is unresponsive.

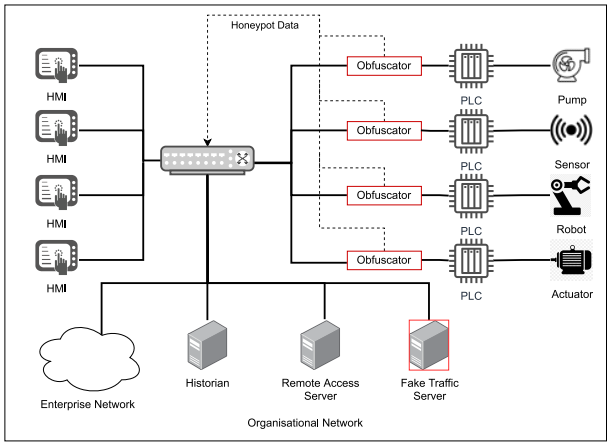

Fig. 2. Obfuscator OT Network Perspective. The dotted line represents data obtained on the obfuscator honeypot that feeds back into the network's SOC or other security measures.

B. Obfuscator from a network perspective

From a network perspective, the obfuscator has to be indistinguishable from the actual PLC. There should be no difference between the response of the obfuscator and the PLC. Therefore, from an attacker's perspective, it has to look like they are interacting with the PLC even though the obfuscator honeypot responds (Figure 2). This links in with asset discovery and reconnaissance, and an attacker at this stage should believe the system is more likely to be a honeypot and not see that services are running on a separate system. Any data obtained by the honeypot should feed back into the security systems within the network.

The network is a standard OT network, deployed according to the Purdue model we discussed previously. In our example network overview, we have several HMIs that are each configured to communicate with one of the PLCs. Both devices represent level 1 and level 2 of the Purdue model. We have included a historian and remote access server to represent level 3 of the model. Level 4 and level 5 are incorporated into the Enterprise Network. When we add some level 0 devices such as a pump, sensor, robot and actuator, we complete the Purdue model. This is an essential high-level representation of an OT environment within an organisation.

C. SDN Enabled Obfuscator

Leveraging SDN within this environment enables many more functionalities than when the obfuscator is deployed within a traditional network and also allows the forwarding of traffic across the network. This means that the issue related to the availability of the device we discussed earlier in the paper, as the forwarding of traffic to and from the obfuscator, can be handled on the network infrastructure/SDN controller itself.

Implementing SDN within the underlying architecture does not change the adversary's perspective but allows for more flexibility within the existing network architecture in terms of resilience and traffic engineering, such as forwarding traffic streams to an IDS [24] or as a flexible security feature [46]. An

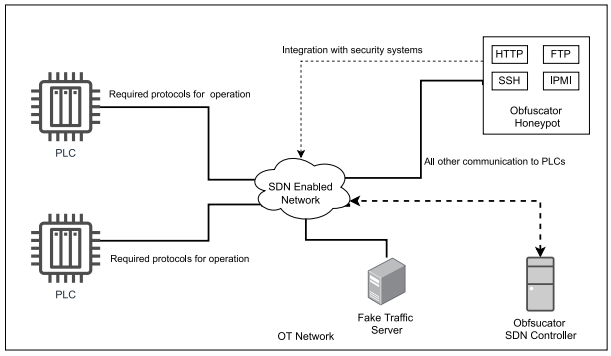

Fig. 3. Example of SDN Enabled Obfuscator Architecture. The SDN controller determines traffic flow to the PLC based on the loaded flow rules or redirects them to the honeypot.

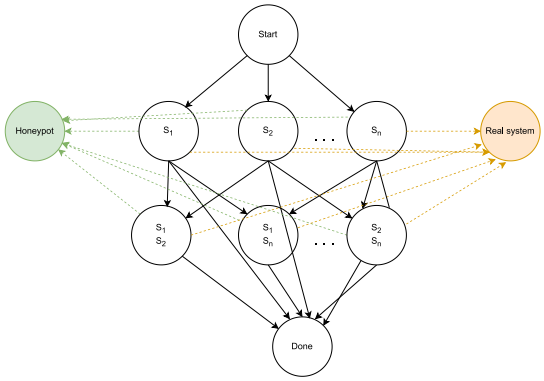

Fig. 4. Generic model of the obfuscator system. Depicts n services which each give the attack a belief the system is a honeypot or real system.

overview of an SDN-enabled architecture for the deployment of the obfuscator can be seen in Figure 3. Within the example architecture, OpenFlow is leveraged in combination with Ryu to provide the capabilities needed within the environment. Instead of these, SDN can also be implemented with solutions such as P4, Cisco ACI, and ONOS. The controller determines the traffic flow to the PLCs and to the obfuscator honeypot. Both PLCs are obfuscated by the honeypot and, therefore, receive traffic directed to both PLCs, but the SDN controller shows the traffic originating from the respective PLC instead of the honeypot, which can assist with network monitoring. On the network itself, this would not be noticeable. Similar to the non-SDN environment, the fake traffic server is on the network, providing traffic to the obfuscator honeypot to allow passive network scans to pick up the services.

V. Obfuscator Model

We have constructed a Hidden Markov chain model to obtain a theoretical evaluation of the obfuscator. Within this model, we assume the obfuscation successfully makes the obfuscator honeypot and the real PLC present as one system. In a real environment, this would depend on the implementation of the obfuscator forwarder. We evaluate our implementation of the forwarder later in the paper. This model represents one instance of the obfuscator and allows for flexibility in deploying multiple services. Due to the nature of the system, the model represents different states of which the next states depend on the states already visited. A service $(S_1 \rightarrow S_n)$ that looks more similar to one that might be deployed on a (poorly) deployed honeypot would result in the adversary estimating a higher probability (belief) that the system is a honeypot [30, 34, 17].If the adversary engages in multiple honeypot-like services, the total belief weight increases more significantly than the total belief weight that the system is a real system. The exact belief weight an adversary believes a system is a honeypot (h) or not (r) depends significantly on the type and configuration of the service deployed. Both belief weights have to be between 0 and 1, with the sum of both equating to 1. The adversary's probability of moving across services is random, as the attacker can move freely within the system. However, suppose the belief that the system is a honeypot and that the system is a real system is similar or low. In that case, an adversary is more likely to move to another service or return to the service to continue investigating, as can be seen with different honeypot deployments [29]. When this confidence is high, an adversary could leave the system if they determine it to be a honeypot, as they do not want to risk being exposed [17, 3], and might be likely to investigate the actual OT protocol more if they determine it is a real system. However, they would not be likely to move to another service if they believe the system is a honeypot and might not be a valuable target or if the risk of exposure is too great [34, 6]. The sum of the belief weights has to remain between 0 and the number of services the attacker has visited, as we assume the attacker has no belief before interacting with the system.

In order to calculate the probability of an adversary deciding whether the system is a honeypot or a real system, we need to use a formula that can encompass the adversary visiting multiple services (X). For each service visited, the attacker obtains a belief that the system is a honeypot (h) and a belief that the system is not a honeypot (r). For each extra service the attacker visits, the beliefs get added to the current belief the attacker has. The value of this at the end is the total belief that the attacker thinks the system is a real system (R) or a honeypot (H). To calculate if the attacker believes the system is more likely to be a honeypot (E[H]), we use the total belief weight that it is a honeypot (H) and divide this by the number of services the attacker visited (n(X)). Dividing H by the number of services visited normalises the value within a range of 0 to 1. This gives us the probability that the attacker believes the system is a honeypot. Subtracting this value (E[H]) from 1 gives us the probability that an attacker believes the system is a real system (E[RS]). This is also obtainable by replacing H with R in the formula used to calculate E[H]. If this value is greater than 0, the attacker believes the system is a honeypot rather than a real system. This has to be calculated when the attacker reaches the 'done' state of the model based on all the services they have visited. The following formulas can, therefore be derived:

$$H = \sum_{n \in X} h(S_n)$$

$$R = \sum_{n \in X} r(S_n)$$

Service	HP Belief Weight	RS Belief Weight
S_1	h_1	r_1
S_2	h_2	r_2
...	...	...
S_n	h_n	r_n

TABLE I
GENERIC MATRIX OF HONEYPOT AND REAL SYSTEM BELIEFS

Service	HP Belief Weight	RS Belief Weight
HTTP	0.95	0.05
FTP	0.91	0.09
MODBUS	0.08	0.92

TABLE II
EXAMPLE MATRIX OF HONEYPOT AND REAL SYSTEM BELIEFS

$$E[H] = \frac{H}{n(X)}$$

$$E[RS] = \frac{R}{n(X)}$$

Which can also be written as

$$E[H] = \frac{\sum_{n \in X} h(S_n)}{n(X)} \quad \text{and} \quad E[RS] = \frac{\sum_{n \in X} r(S_n)}{n(X)}$$

We have adapted the generic model of the obfuscator to represent and evaluate one possible configuration (Figure 5). This example represents a deployment of the obfuscator where three services are accessible: an emulated HTTP and FTP service and the real Modbus service running on the actual PLC. We assume the attacker has free choice to start with either service, and at the end state of the model, the attacker has assumed the system is either a honeypot or a real system. Beliefs within this model (Table II) are discussed and agreed on by 5, which should give us a confidence interval of 93.75% [18], ICS and/or penetration testing experts. We asked the experts the likelihood they would believe the system is real or a honeypot based on the services seen. We have used a Conpot honeypot (running HTTP and FTP) and an Allen-Bradley PLC (running MODBUS) as an example system. These numbers are specific to our deployment and configuration of these services. How these services are deployed, e.g. a dummy FTP service with no interaction or an HTTP service with a different website will impact the values. Therefore, these should be taken as an example for our system and might not necessarily be the same in another deployment. Mapping these beliefs to the previously described functions shows that an attacker is significantly more likely to believe the system is a honeypot rather than a real system. We have mapped these for three scenarios which the attacker could follow.

- Scenario 1: The attacker initially investigates the FTP service; from this, they move on to the Modbus service, which then leads to the HTTP service. The honeypot belief weight the attacker has is 1.94 (honeypot belief of FTP + Modbus + HTTP) compared to 1.06 (real system belief of FTP + Modbus + HTTP) it is a real system when they reach the final state. When we (calculate $E[H]$), the result is 0.647 and the expectation that the system is a real system (E[RS]) is 0.353. Therefore, the attacker determines there is a greater probability that the system is a honeypot than a real system.
- Scenario 2: The attacker initially investigates the HTTP service; from this, they leave the system. The probability the attacker thinks the system is a honeypot is 0.95 (honeypot belief of HTTP) compared to 0.05 (real system belief of HTTP) it is a real system when they reach the final state. When we (calculate $E[H]$), the result is 0.95 and the expectation that the system is a real system (E[RS]) is 0.05. Therefore, the attacker determines there is a greater probability that the system is a honeypot than a real system.
- Scenario 3: The Attacker initially investigates the Modbus service; from this, they move on to the HTTP service. Afterwards, they leave the system. The probability the attacker thinks the system is a honeypot is 1.03 (honeypot belief of Modbus + HTTP) compared to 0.97 (real system belief of Modbus + HTTP) it is a real system when they reach the final state. When we (calculate $E[H]$), the result is 0.515 and the expectation that the system is a real system (E[RS]) is 0.485. Therefore, the attacker determines there is a greater probability that the system is a honeypot than a real system.
- Scenario 4: The attacker initially investigates the Modbus service; afterwards, they leave the system. The probability the attacker thinks the system is a honeypot is 0.08 (honeypot belief of Modbus) compared to 0.92 (real system belief of Modbus) it is a real system when they reach the final state. When we (calculate $E[H]$), the result is 0.08 and the expectation that the system is a real system (E[RS]) is 0.92. Therefore the attacker determines there is a greater probability that the system is a real system than a honeypot.

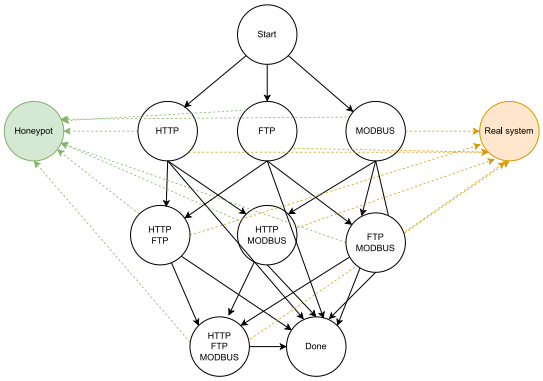

Fig. 5. Example model of the obfuscator system. It depicts three services, HTTP, FTP and MODBUS, which each give the attack a belief the system is a honeypot or real system.

From this, we can see that because the Modbus protocol is part of the real system, it reduces the chance an adversary would view the system as a honeypot compared to the other services. However, if we consider a more appropriate path for an attacker, investigating multiple parts of the system, the chance an attacker determines the system is a honeypot is significantly higher. It is improbable that an attacker would only

focus on one protocol running on the system. Additionally, once an attacker has interacted with services on the system, aside from the real service, they should have been noticed by the SOC as the honeypot aspect of the system should be linked in with other security systems.

VI. Evaluation

A. Setup

The evaluation setup focuses on the obfuscator [36] rather than the network around it. For this, we deploy one PLC with one Conpot obfuscator honeypot and a Ryu-based [45] obfuscator controller within our test network (Figure 8), which will be running within our ICS lab deployed according to the testbed proposed by Green et al. [14] and uses IPv4 as IPv6 has only recently been supported by Siemens PLCs [41]. The PLC deployed is a Siemens Simatic ET 200S and is connected to the obfuscator and attacker system through a Pica8 P-3297 switch deployed in OVS mode. The obfuscator is configured to redirect traffic to a Conpot honeypot (Ubuntu 20.04 LTS, i5-8265U, 8GB) which is set up in a default configuration emulating a Siemens Simatic ET 200S PLC. As this template also includes an S7Comm emulation, we have disabled this as the Siemens PLC provides this service. With this setup, the obfuscator presents itself as a Siemens Simatic ET 200S PLC, as the real device it is connected to, and it has all the default Conpot signatures. The obfuscator controller [**Github**], running on Raspberry Pi4, is designed to obfuscate traffic by redirecting traffic to the Conpot honeypot if any protocol other than S7Comm or SNMP is used. The honeypot itself is obfuscated as all packets directed to its IP address that are not redirected from the obfuscated PLC IP are dropped. We have also deployed a system providing fake traffic to the honeypot on a regular basis. This system connects to the Modbus, HTTP, Bacnet, IPMI, FTP and TFTP services by sending a basic request to those services sequentially on a rolling basis.

To establish a baseline for our obfuscation, we also run the tests on the same PLC. We do this by bypassing the SDN environment by directly connecting the PLC to the system we use as the attacker.

Conpot is running the following built-in protocols which are part of its Siemens S7-200 emulator: Modbus on TCP port 502, HTTP on PCP port 80, Bacnet on UDP port 47808, IMPI on UDP port 623, FTP on TCP port 21 and TFTP on UDP port 69.

The obfuscator itself uses the Ryu framework for OpenFlow controllers, where it acts as a firewall, a layer 2 learning switch, and an obfuscator. The functionality of this controller is split across five flow tables, as shown in figure 7, where all static logic is populated based on a provided configuration. The configuration 6 consists of a set of hidden nodes representing PLCs within the network and a set of obfuscation rules, in which a set of header values are provided along with a honeypot node that should handle any traffic of that type. From this, the controller ensures that all incoming traffic to the network destined for a node used as a honeypot is dropped by the firewall table. The controller then installs flow entries in the static obfuscation table to identify any traffic that should be obfuscated, associating any packet in messages with a unique cookie per obfuscation rule. Then any future packet that matches with a static obfuscation rule will trigger the controller to install the appropriate bi-directional dynamic rule. The result of this is that any packet addressed to a protected node of a traffic type specified in an obfuscation rule will have its destination MAC and IPv4 addresses rewritten with that of a honeypot, and any returned packets will have their source MAC and IPv4 rewritten to that of the original packet's destination.

```
#bypass the obfuscator
[[86fe486e7302002f.bypass]] #port 102
eth_type = 2048
ip_proto = 6
tcp_dst = 102

[[86fe486e7302002f.bypass]] #port 161
eth_type = 2048
ip_proto = 17
udp_dst = 161

#Redirect any IP traffic heading to any of the protected nodes redirect to the honeypot
#Higher priority comes first (e.g. 2 then 1)
[[86fe486e7302002f.rule]] #All IP traffic from PLC to honeypot
id = 1
name = "redirect ip"
priority = 2
[86fe486e7302002f.rule.match]
eth_type = 2048
[86fe486e7302002f.rule.honeypot]
mac_address = "98:fa:9b:53:c5:17"
ipv4_address = "172.21.1.190"

[[86fe486e7302002f.rule]] #HTTP to honeypot
id = 3
name = "redirect HTTP"
priority = 4
[86fe486e7302002f.rule.match]
eth_type = 2048
ip_proto = 6
tcp_dst = 80
[86fe486e7302002f.rule.honeypot]
mac_address = "98:fa:9b:53:c5:17"
ipv4_address = "172.21.1.190"
```

Fig. 6. Example of obfuscator Flow Rules. Shows two bypass rules for traffic that goes to the PLC, a redirection rule from the PLC to the honeypot for HTTP and a redirection rule for all other IP traffic destined for the PLC to the honeypot.

B. Methodology

To evaluate the success of the obfuscator in presenting PLCs as a honeypot within the test environment (figure 8, we have run several penetration testing and network evaluation tools to evaluate how the system presents itself to adversaries running these tools. An overview of the tools we have used within the environment can be found in Table III.

Tools such as Nmap and Wireshark are used for asset discovery, and Wireshark can also be used for passive discovery of devices on the network. We understand that adversaries do not commonly use Nmap within these environments due to the noise it generates on the network. However, our evaluation shows us how our system would behave when scanned. The tools we use in this evaluation generally fit with the MITRE ATT&CK for ICS Matrix [33, 2] discovery tactic and can be used for network sniffing, network connection enumeration and other techniques listed under this tactic. Our evaluation focuses on the reconnaissance stage of the cyber kill chain.

As availability is one of the most important aspects of critical infrastructure/OT environments, we have also deployed a test to evaluate the performance of our system compared to the PLC by itself to ensure the obfuscator does not prohibit its functions.

We have run each of these tools on both the obfuscated PLC and the non-obfuscated PLC IP address. This has been done to understand the impact the obfuscator has on network

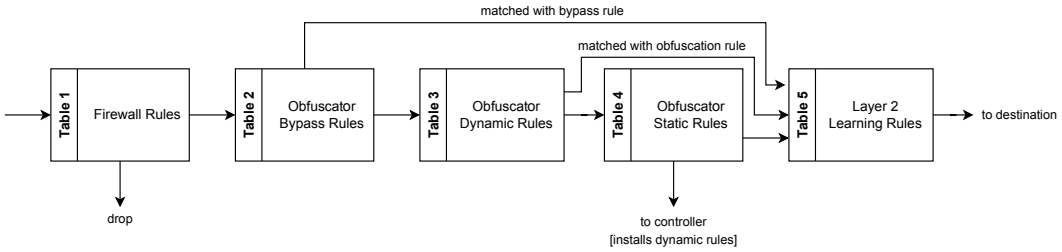

Fig. 7. Flow Tables used by the OpenFlow Obfuscator. Depicting how bypass and obfuscator rules are handled by the controller and how the traffic not matching any rule is handled.

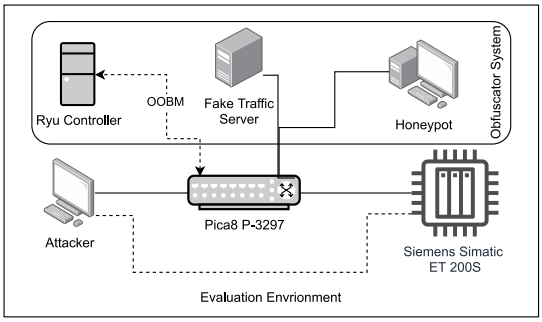

Fig. 8. Diagram of the evaluation scenario where the attacker attempts to perform an asset discovery on the network looking for PLCs. The dotted line to the Siemens PLC represents the baseline evaluation scenario. The dotted line to the controller represents the control plane connection.

Penetration testing	Networking
Nmap	Wireshark
Nessus	Snap7
PLCScan	Ping
S7Scan	httping
	hping3

TABLE III
PENETRATION TESTING AND NETWORKING TOOLS USED IN THE EVALUATION OF THE OBFUSCATOR EFFICACY

performance, how well the obfuscator is able to hide from being a separate system and most importantly, the capability to obfuscate the PLC as a honeypot from an attacker's perspective.

1) Obfuscation: To evaluate the obfuscation capabilities, we have deployed several tools on both the obfuscated and non-obfuscated PLC. In terms of penetration testing tools, we have deployed all tools listed within Table III. Nmap has been used to evaluate how the obfuscator responds to network scanners commonly used within penetration tests. This allows us to verify if the scanners reveal that the S7 protocol does not run on the same system as the services run by Conpot. Nessus, PLCScan and S7Scan focus on vulnerability scanning. Running these tools indicates which vulnerabilities can be found through the obfuscation, which cannot be found on the PLC. In addition to these penetration testing tools, we ran Wireshark, Ping, HTTPing and Snap7. The Snap7 script used runs the following commands: plc.get_connected, plc.get_cpu_info, plc.get_plc_datetime, plc.ab_read(1,5), and plc.upload(1). These commands try to connect to the PLC and perform several tasks, such as enquiring what CPU is running and downloading data from the device (upload).

Wireshark provides an overview of activity on the network and if network traffic to the obfuscated IP differs from the traffic sent to the non-obfuscated PLC. To further evaluate the network-level obfuscation, we deploy Ping to evaluate if there is any difference in delay between the host and the services running on Conpot and the host and S7Comm on the PLC. Finally, Snap7 is used to evaluate the response to S7comm requests through the obfuscated and the non-obfuscated PLC.

We have further evaluated the obfuscation capabilities by inviting two experts and tasking them with investigating the network. For the first test, we let them into the network with no prior knowledge and asked them to identify all systems on the network and evaluate if they were honeypots or real systems. Afterwards, we introduced them to the obfuscator and asked them to evaluate the system by investigating if they could spot that there are two devices behind the one IP address.

2) Performance: To evaluate the performance of the obfuscator, we have deployed three networking tools within our evaluation environment. We used Ping and HTTPing to evaluate the delay between the host, the obfuscated services and the S7comm service running on the PLC through the obfuscator. In addition to Ping, we have used a script to evaluate the amount of traffic our Ryu obfuscator controller and the non-obfuscated PLC can handle. The script leverages HPing3 to flood the environment with data and Ping to evaluate the performance of the network at the same time, starting at two packets per second (0.13 KBps) and increasing to 1 000 000 packets per second (64 MBps).

C. Results

1) Without Obfuscation: When running Nmap scans on the non-obfuscated PLC, Nmap detects only port 161/udp, which runs SNMP and port 102/tcp, running iso-tsap, which is the S7comm service. Running the Nmap built-in S7-info script, Nmap returns port 102/tcp as open and gives the correct information about the PLC (e.g. System name, MAC address and Serial Number). Nmap fails to detect the OS running on the machine (on all scans) and guesses it to be devices such as BorderGuard firewall, Xerox printer and Motorola wireless bridge. PLCScan and S7Scan both return the expected information of the PLC. Nessus returns one high-risk SNMP vulnerability. All Snap7 commands return the expected values

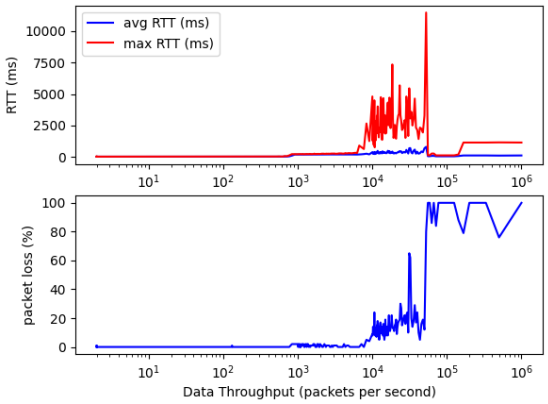

Fig. 9. PLC performance evaluation graph showing the packet loss and RTT during the stress test. The drop in RTT is a result of the high packet loss.

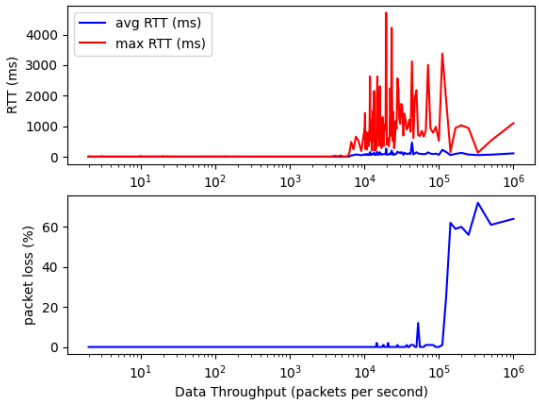

Fig. 10. Obfuscator system performance evaluation graph showing the packet loss and RTT during the stress test. The drop in RTT is a result of the high packet loss.

and do not return any errors. Wireshark does not show any odd behaviour, and all packets show to be sent from and to the correct machines. Ping returns an average RTT of 5ms, a maximum RTT of 9ms, and a minimum RTT of 3ms for 50 pings. The PLC starts to drop packets (¿10%) when sending 10,200 packets per second (0.65 MBps) (9) and starts to significantly drop packets (24%) at 10,600 packets (0.68 MBps). The highest packet loss (100%) is first reached at 55,500 packets per second (3.55 MBps). The maximum RTT (11,483 ms) observed during the test is at around 52,500 packets per second (3.36 MBps), the average RTT maximum throughout the test is 689 ms, and the average of the average RTT is 119 ms.

2) With Obfuscation: Running Nmap on the obfuscated PLC returns a wealth of information. Scanning TCP ports, Nmap returns FTP running on port 21, HTTP on port 80, iso-tsap (S7comm) on port 102 and mbap on port 502 (MODBUS). The s7-info script returns port 102/tcp as open and shows the correct information about the PLC. The five highest guesses Nmap made for the OS are Linux 2.6.32 - 3.10 (93%), Synology DiskStation Manager 5.2-5644 (92%), Linux 3.1 (91%), and Linux 3.2 (91%). Scanning UDP ports on Nmap only shows port 161/snmp being open, with the other services running (TFTP, IPMI (reported as asf-rmcp), and Bacnet) being reported as opened—filtered. Aggressive OS guesses are widely different and include Citrix Access Gateway VPN gateway (95%), Linksys WRT610Nv3 WAP (95%), and 3Com OfficeConnect 3CRWER100-75 wireless broadband router (94%). Both TCP and UDP scans return the MAC address as belonging to Siemens AG.

PLCScan and S7Scan show the expected information about the PLC, which is to be expected as these are only scanning the S7 protocol. Nessus reported no vulnerabilities on the device. When we ran our Snap7 script, it resulted in the expected return values from the PLC. Wireshark does not show any communication with the honeypot as the source, and the Ryu obfuscator successfully obfuscated the packets as if they were sent from the obfuscated PLC.

Ping returns an average RTT of 3ms, a maximum RTT of 4ms and a minimum RTT of 2ms after 50 pings (which are redirected to the honeypot). HTTPing returns a higher average RTT of 235ms, with a maximum response time of 358ms. Looking at the packet loss in our evaluation setup 10 we reach a maximum of 72% packet loss at around 330,000 packets per second during our test (up to 1,000,000 packets per second), which results in 21.3 MBps. Any significant packet loss (25%) starts at 125,000 packets per second (8 MBps). The first packet loss we see higher than 10% is at around 52,000 packets (12%) or 3.37 MBps. The maximum RTT (4718 ms) observed during the test was at 19607 packets per second (1.25 MBps), the average RTT maximum throughout the test was 276 ms, and the average of the average RTTs was 30 ms.

D. Expert Evaluation

As stated previously, we have invited two experts to evaluate the obfuscator system in the same evaluation environment as used previously without the PLC bypass. One expert we invited is a penetration tester holding an OSCP certification, an expert in OT security, and has been involved with ICS-specific penetration testing. The other expert has more than a decade of penetration testing experience and is currently a security engineer within an OT environment. As with the previous test, to make the test more realistic, we simulated basic traffic within the environment going to the PLC and the honeypot through the fake traffic server; the PLC is connected to a water tank on which we open and close several valves during the test. We provided both experts with IP addresses of systems that were not part of the evaluation, as they were used to emulate traffic.

Our first expert cautiously approached the system due to the warning that a honeypot could be running within the network. They started by running Wireshark in promiscuous mode to enable them to see all the network traffic without actively scanning the network. With this, they identified one system running on the network; the system was identified to be running HTTP, FTP, Modbus and S7Comm. The expert identified this as being odd and was reluctant to investigate the

system further, believing the system was potentially a honeypot. When probed to investigate further, they investigated some of the HTTP traffic, through which they identified the default Conpot website. At this point, they told us they believed the system to be a honeypot. After we explained the system and its goal, they explored the network more aggressively. Tools being used include NMAP (including the s7-info script and a custom script), S7Scan, Wireshark and a custom-developed tool. Using these tools, the expert aimed to find evidence of the honeypot and PLC being two systems hidden behind the same IP address. After a thorough examination, the expert reported no evidence of this, as all the data indicated this looked like it was one system. The expert also did not discover the IP address that was obfuscated. One note made was that the S7Comm implementation could be seen as too developed, and there seemed to be a disparity between the quality of emulation of this service (acknowledging this was a real service running on a real PLC) and the honeypot services. However, they believed this would not pose a problem as an adversary would be reluctant if there were any indication of the system being a honeypot.

The second expert also approached the system without us telling them there might be honeypots on the network. When probed, the expert indicated knowing of honeypots and told us they were always wary of honeypots within an environment. After the initial passive scan of the network, they indicated that they identified one system on the network. Contrary to the first expert, they investigated the Wireshark traces more in-depth from the start and told us they believed the system was a Conpot honeypot. This indicates that the expert has previous experience with this type of honeypot. At this point, we introduced the system to the expert and asked them to break the obfuscation. For this, the expert used tools like Nmap (including the Modbus-discover and s7-info script), Wireshark, S7Scan, PLCScan, OpenVAS, and Lansweeper. These tools investigated the IP address discovered earlier and actively scanned the network. The expert reported seeing no indication that the S7Comm service was hosted on a separate system from the other services, nor did they discover the obfuscated system's IP address. However, similar to the first expert, they noticed the S7Comm service was very advanced for a honeypot but noted that an adversary within the network would be very concerned with being stealthy and probably would not interact with this system.

E. Discussion

Even if the obfuscation ultimately failed, due to the attacker identifying the system as real, they would have interacted with the honeypot services. which would result in the SOC being alerted of activity on the honeypot, and actions could be taken. If this device was Internet-connected, Shodan would have identified the Conpot default signatures

Our previous work [29], where we evaluated the performance of several honeypot deployments on the Internet, has shown that the deployed Internet-facing ICS honeypots received a very low level of traffic on protocols such as S7Comm and Modbus. And they saw most traffic on IT protocols such as HTTP and FTP, protocols such as BACnet that are also used outside of industrial environments such as smart homes [49], and the Common Industrial Protocol [7]which is used to integrate with enterprise-level Ethernet networks. Furthermore, the study saw a lot of Internet scanners or sources that do not provide much information on the adversary targeting the system, such as Digital Ocean. During the first and second periods, 24% and 21%, respectively, of IP addresses seen belonged to Censys, and 26% and 20%, respectively, belonged to Digital Ocean. Therefore this evaluation focuses on how the system would be perceived by the adversary, which is why we constructed the evaluation around the behaviour of tools, a practical evaluation and an expert evaluation. Through the expert evaluation, we aimed to understand how expert penetration testers would approach and view the system.

We can see a positive outcome when looking at and comparing the results of the non-obfuscated and obfuscated PLC. Based on the data, there is no indication that our obfuscator impairs any of the functions of the PLC. This is very important in an ICS environment as availability is of utmost importance. The average RTT between the devices is nearly identical with only a 2ms difference, in favour of the obfuscated PLC, which we can consider a similar result. Further, the stress test shows that our SDN testing environment does not impact the PLC performance as our evaluation setup first sees any noteworthy packet loss (25%) at 125 000 packets per second (8 MBps), which is significantly higher than the first considerable packet loss (24%) we have observed on the PLC at around 10 600 packets per second (0.68 MBps). We also saw a nearly identical average RTT during the ping test, showing that the SDN environment does not add an extra delay but performs similarly or even better. During the stress test, the average and maximum RTT were also lower than the RTT when connected to the PLC.

Looking at the obfuscation capabilities, we can observe that both systems present as one from a network perspective, and all communication going to the attacker originates from one IP address (from the attacker's perspective). However, in our tests, it is important to note that the MAC address reports as belonging to a Siemens AG device. This could be mitigated by using the IP address of the honeypot as the target IP address and obfuscating the IP of the PLC. Further, we can see that all S7Comm-related scanners and communication reports are the same on both the obfuscated and non-obfuscated PLC. Therefore, the obfuscation does not impact the PLC's expected performance.

The OS guesses from Nmap show differences when scanning the TCP or UDP protocols. Scanning the TCP ports on the obfuscated PLC reports a high probability of the system being a Linux-based system, which is what the honeypot is running. When scanning all UDP ports, only SNMP returns as being open, and the OS guesses are similar to the guesses on the non-obfuscated PLC. One odd result can be seen on the Nessus scan. For the obfuscated PLC, Nessus does not report any vulnerability. In contrast, the non-obfuscated PLC returns one high vulnerability on the SNMP service, which bypasses the honeypot on the obfuscated PLC.

Our two expert evaluations also show that the obfuscator

behaves as expected and successfully makes the honeypot and PLC present themselves as the same system under one IP address. It was interesting that both of the experts indicated they saw a discrepancy between the real service and the honeypot services. However, they did state that given the priority of adversaries within these environments is to remain undetected, which aligns with research [3], an adversary would be wary of anything that might uncover their existence. Both experts also state that they believe a system like this could work in a real environment and thought our usage of honeypots was interesting. Given both experts work within the OT domain and have significant penetration experience, we deem this a success of our obfuscator system as we have successfully convinced two knowledgeable attackers to halt interactions with the real system. This expert evaluation enabled us to better understand the thought process behind the attacker by directly interacting with them, which would not be possible with other evaluation mechanisms.

Based on these results, we can conclude that the goal of the obfuscator has been achieved, as both systems show as one from a network perspective. As none of the tools we ran on the system indicated that two different systems are responding to the attacker, the average attacker would not notice this. However, a highly sophisticated attacker could potentially have specific tools they might have developed that could indicate that these are two separate systems. We deem this a possibility with every security measure, as there can always be ways highly knowledgeable attackers can circumvent a security measure. Furthermore, from running the tools, the attacker would see multiple services running on the device and spot the Conpot signatures, which can be found on the website. Because we are not using Conpot's S7Comm service, not all signatures can be seen on the system. Looking back, if the attacker ran Nmap on the PLC, it is highly likely they would have interacted with the honeypot services, which If an attacker were to interact with the honeypot, this would result in the SOC being notified of unusual activity, which is beneficial to the security of the PLC.

VII. Deployment Considerations

When implementing an obfuscator within an industrial network, attention has to be paid to the requirements of the network. As stated previously, it is vital to keep availability, reliability and safety in mind. To satisfy this, a risk assessment can be done on the implementation of the obfuscator and the impact it can have on the network and system it is connected with, and on the implementation of SDN if this will be implemented as well. There is also a need for additional resources to deploy this system, which does not need to be significant as one server can be used for multiple PLCs and has the same cost as other honeypots. The highest cost of the deployment is that the system uses SDN, which might already be available or be used for other purposes.

SDN can be used for purposes such as rerouting possible adversaries that are hitting the obfuscator to a high-interactive honeypot environment. This can be done by implementing ML algorithms or other practices, such as the system described in [50], that classify interactions with the obfuscator. Systems for intrusion detection in ICS environments, such as DEIDS [16] and Hamids [12], have already been developed that could be implemented in this system. This further has to be in line with other logging and monitoring systems deployed and the operational process in the SOC. Implementation of this has to be evaluated when the system is implemented.

Instead of deploying an actual system to function as an obfuscator honeypot, the honeypot services can also be virtualised by deploying bespoke Docker containers which can be extended by dynamically spinning up a Docker instance based on adversary behaviour. Doing this provides an individual environment for each adversary within the network and allows for flexibility in the deployment of the obfuscator. This avenue also can lower the cost of deployment considerably. When implementing the forwarder, it is important that an attacker cannot distinguish between a reply from the obfuscator and a reply from the actual PLC. Certain characteristics of the network stack might have fingerprintable information regarding the source system, such as the MAC address which we have copied in addition to the IP address. This further ties in with ensuring the obfuscator itself is not directly accessible by an attacker. Otherwise, they might be able to identify the system as an obfuscator. These issues can be more complex within an operational environment compared to the testbed used in our evaluation.

Within this paper, for evaluation purposes, we have deployed a limited Ryu version of the obfuscator controller in a bespoke environment combined with a Conpot honeypot. This has been done to allow us to evaluate the concept of obfuscating PLCs in a practical setting. Obfuscating a PLC within a real-world environment would require developing code specific to this environment and analysing the security of the obfuscator code before deployment.

Finally, another avenue for deployment would be within a moving target defence (MTD) environment. Implementing MTD within an operational network would require more evaluation and preparation but could be a potential avenue to improve upon the obfuscator. Depending on what MTD techniques that are going to be implemented assessments have to be done if MTD would interfere with the availability or any other aspect of the operations of the system. This should include possible delays that are introduced or how techniques such as IP hopping will be implemented.

VIII. Conclusion

This paper has introduced and evaluated the concept of obfuscation within industrial control system environments. We have shown that obfuscation is a valid approach to reduce attackers' appetite to interact with certain systems, as they believe the systems are honeypots and that obfuscation can successfully be deployed within an ICS environment. Our evaluation happened within a physical environment using a Pica8 Openflow SDN switch, a Ryu-based OpenFlow controller and a Siemens S7 ET 200S PLC. We further strengthened this from a theoretical approach via a Hidden Markov model. Even if the obfuscation ultimately would fail, due to the attacker identifying the system is real, they would have interacted

with the honeypot services. which would result in the SOC being alerted of activity on the honeypot and actions could be taken. If this device was Internet-connected, Shodan would have identified the Conpot default signatures

During the practical evaluation, we ran several penetration testing tools and networking tools. Our tests have shown that our obfuscator successfully presents the deployed honeypot and the Siemens PLC as the same system in a real environment. The tools ran on the system show no differences between them, and a Nmap scan on the PLC covers both the PLC and the honeypot. When we investigated the network traffic on Wireshark, we saw no connection originating from the honeypot going to the attacker; all packets were sent from the PLC with the correct MAC address. Both experts also found no evidence of the second system that we obfuscated. Furthermore, we have also shown that the obfuscator system does not interfere with operations on the PLC, as all S7comm communications still report back as expected. Therefore, the operations of the PLC itself have not been affected. The physical environment we evaluated the system in also performs better than the PLC itself. Any bottleneck would originate from the PLC rather than the SDN switch, controller or honeypot. This can change depending on the devices deployed within the obfuscator system.

We believe that the obfuscator presented is feasible as an additional security mechanism within a network to both discourage attackers from interacting with a real PLC and increase the chances of capturing attackers on the network. Even if an attacker decides to interact with the system because it is a honeypot, this should feed into the other security mechanisms that are deployed. The interaction should warn the SOC, which then can apply the proper mitigation. Consequently, the system can also be connected with other security mechanisms, as the data captured by the honeypot can be used in systems such as an IDS.

Although our evaluation focused on a Conpot honeypot combined with a Ryu OpenFlow controller, the system can be deployed in many configurations. Whereas we have leveraged SDN as a tool to enable traffic redirection, other systems are equally valid. This also links back to the SDN technology used, as this system can be deployed within P4, other SDN environments or intent-based networking as well [39, 47]. Further research within these fields might show new ways of enabling obfuscation to improve the obfuscator system.

ACKNOWLEDGMENTS

This research was supported by the Next Generation Converged Digital Infrastructure project (EP/R004935/1) funded by the Engineering and Physical Sciences Research Council and British Telecommunications. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) license to any Accepted Manuscript version arising.

REFERENCES

[1] Irfan Ahmed et al. “Programmable Logic Controller Forensics”. In: *IEEE Secur Priv* 15.6 (2017).

[2] Otis Alexander, Misha Belisle, and Jacob Steele. “MITRE ATT&CK for Industrial Control Systems: Design and Philosophy”. In: *MITRE Corporation* (2020).

[3] Adel Alshamrani et al. “A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities”. In: *IEEE Commun. Surv. Tutor.* 21.2 (2019).

[4] Daniele Antonioli, Anand Agrawal, and Nils Ole Tippenhauer. “Towards high-interaction virtual ICS honeypots-in-a-box”. In: *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. 2016, pp. 13–22.

[5] Giuseppe Bernieri, Mauro Conti, and Federica Pascucci. “Mimepot: a model-based honeypot for industrial control networks”. In: *2019 ieee international conference on systems, man and cybernetics (smc)*. IEEE. 2019, pp. 433–438.

[6] Ping Chen, Lieven Desmet, and Christophe Huygens. “A study on advanced persistent threats”. In: *IFIP CMS*. Springer. 2014.

[7] Mauro Conti, Denis Donadel, and Federico Turrin. “A survey on industrial control system testbeds and datasets for security research”. In: *IEEE Communications Surveys & Tutorials* 23.4 (2021), pp. 2248–2294.

[8] Paul Didier et al. “Converged Plantwide Ethernet (CPwE) Design and Implementation Guide”. In: (2011).

[9] Michael Dodson, Alastair R Beresford, and Daniel R Thomas. “When will my PLC support Mirai? The security economics of large-scale attacks against Internet-connected ICS devices”. In: *IEEE eCrime*. 2020.

[10] Miao Du and Kun Wang. “An SDN-enabled pseudo-honeypot strategy for distributed denial of service attacks in industrial Internet of Things”. In: *IEEE Transactions on Industrial Informatics* 16.1 (2019), pp. 648–657.

[11] Joseph Gardiner et al. “Controller-in-the-Middle: Attacks on Software Defined Networks in Industrial Control Systems”. In: *CPSIoTSec*. 2021.

[12] Hamid Reza Ghaeini and Nils Ole Tippenhauer. “Hamids: Hierarchical monitoring intrusion detection system for industrial control systems”. In: *CPS-SPC*. 2016.

[13] Benjamin Green, Marina Krotofil, and Ali Abbasi. “On the significance of process comprehension for conducting targeted ICS attacks”. In: *CPS-SPC* (2017).

[14] Benjamin Green et al. “{ICS} testbed tetris: Practical building blocks towards a cyber security resource”. In: *USENIX CSET*. 2020.

[15] Benjamin Green et al. “PCaaD: Towards automated determination and exploitation of industrial systems”. In: *Comput Secur* 110 (2021).

[16] Haoran Gu et al. “DEIDS: a novel intrusion detection system for industrial control systems”. In: *Neural. Comput. Appl.* (2022).

[17] Thorsten Holz and Frederic Raynal. "Detecting honeypots and other suspicious environments". In: *IEEE SMC*. 2005.

[18] Douglas W. Hubbard and Richard Seiersen. *How To Measure Anything in Cybersecurity Risk*. John Wiley & Sons, 2016.

[19] Forough Ja'fari et al. "An intelligent botnet blocking approach in software defined networks using honeypots". In: *J Ambient Intell Humaniz Comput.* 12.2 (2021).

[20] Peng Jie and Liu Li. "Industrial control system security". In: *IHMSC* 2 (2011).

[21] Mohammad Taha Khan et al. "An empirical analysis of the commercial vpn ecosystem". In: *IMC*. 2018.

[22] Neal Krawetz. "Anti-honeypot technology". In: *IEEE Secur Priv* 2.1 (2004).

[23] Sukwha Kyung et al. "HoneyProxy: Design and implementation of next-generation honeynet via SDN". In: *IEEE CNS*. 2017.

[24] Roberto di Lallo et al. "Leveraging SDN to monitor critical infrastructure networks in a smarter way". In: *IFIP/IEEE IM*. IEEE. 2017.

[25] Robert Larkin, Juan Lopez, and Jonathan Butts. "Evaluation of traditional security solutions in the SCADA environment." In: *ICIW* (2012).

[26] Heiner Lasi et al. "Industry 4.0". In: *Bus. Inf. Syst. Eng.* 6.4 (2014).

[27] Efrén López-Morales et al. "Honeyplc: A next-generation honeypot for industrial control systems". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 279–291.

[28] Peter Loshin. *Practical anonymity: Hiding in plain sight online*. Newnes, 2013.

[29] Sam Maesschalck, Vasileios Giotsas, and Nicholas Race. "World Wide ICS Honeypots: A Study into the Deployment of Conpot Honeypots". In: *ICSS Workshop*. 2021.

[30] Sam Maesschalck et al. "Don't get stung, cover your ICS in honey: How do honeypots fit within industrial control system security". In: *Comput Secur* 114 (2022).

[31] Stephen McLaughlin et al. "The Cybersecurity Landscape in Industrial Control Systems". In: *Proceedings of the IEEE* 104.5 (2016).

[32] Thomas Miller et al. "Looking back to look forward: Lessons learnt from cyber-attacks on Industrial Control Systems". In: *Int. J. Crit. Infrastruct. Prot* 35 (2021).

[33] MITRE. *ATT&CK for Industrial Control Systems*. 2020. URL: https://bit.ly/3MvQbXo.

[34] Nitin Naik et al. "Honeypots that bite back: A fuzzy technique for identifying and inhibiting fingerprinting attacks on low interaction honeypots". In: *FUZZ-IEEE*. 2018.

[35] Philip O'Kane, Sakir Sezer, and Kieran McLaughlin. "Obfuscation: The hidden malware". In: *IEEE Secur Priv* 9.5 (2011).

[36] PLC-Obfuscator. *Obfuscator*. 2022. URL: https://gitfront.io/r/user-2147465/EsugqX3pMaWD/obfsucator/.

[37] Steve Ranger. *Security surprise: Four zero-days spotted in attacks on researchers' fake networks*. 2020. URL: https://zd.net/3CYBXLC.

[38] Sandeep Gogineni Ravindrababu and Jim Alves-Foss. "Automated Detection of Configured SDN Security Policies for ICS Networks". In: *ICSS Workshop*. 2020.

[39] Mohammad Riftadi and Fernando Kuipers. "P4i/o: Intent-based networking with p4". In: *IEEE NetSoft*. IEEE. 2019.

[40] Neil C Rowe, Binh T Duong, and E John Custy. "Fake honeypots: A defensive tactic for cyberspace". In: *IEEE IWIA*. 2006.

[41] Siemens. *IPv6 in automation technology*. 2019. URL: https://sie.ag/3seDO97.

[42] Joseph Slowik. "Evolution of ICS attacks and the prospects for future disruptive events". In: *Threat Intelligence Centre Dragos Inc* (2019).

[43] George Stergiopoulos, Dimitris A Gritzalis, and Evangelos Limnaios. "Cyber-attacks on the Oil & Gas sector: A survey on incident assessment and attack patterns". In: *IEEE Access* 8 (2020).

[44] The Honeynet Project. *Know Your Enemy*. Addison-Wesley, 2001.

[45] FUJITA Tomonori. "Introduction to ryu sdn framework". In: *Open Networking Summit* (2013).

[46] Akihiro Tsuchiya et al. "Software defined networking firewall for industry 4.0 manufacturing systems". In: *J. Ind. Eng. Manag.* 11.2 (2018).

[47] Yoshiharu Tsuzaki and Yasuo Okabe. "Reactive configuration updating for intent-based networking". In: *IEEE ICOIN*. 2017.

[48] Joni Uitto et al. "A survey on anti-honeypot and anti-introspection methods". In: *WorldCIST*. Springer. 2017.

[49] Nikolaos Vakakis et al. "Cybersecurity in SMEs: The smart-home/office use case". In: *IEEE CAMAD*. 2019.

[50] He Wang and Bin Wu. "SDN-based hybrid honeypot for attack capture". In: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE. 2019, pp. 1602–1606.

[51] Hui Xu et al. "Layered obfuscation: a taxonomy of software obfuscation techniques for layered security". In: *Cybersecurity* 3.1 (2020).

[52] Tarun Yadav and Arvind Mallari Rao. "Technical aspects of cyber kill chain". In: *SSCC*. Springer. 2015.

[53] Jianzhou You et al. "Honeyvp: A cost-effective hybrid honeypot architecture for industrial control systems". In: *ICC 2021-IEEE International Conference on Communications*. IEEE. 2021, pp. 1–6.

Sam Maesschalck is currently with RHEA Group where he is the lead security trainer for the European Space Agency's new cybersecurity centre of excellence. Aside from this role, he is also a researcher at the Lancaster University School of Computing and Communications. He has obtained a PhD in Computer Science from Lancaster University. His research interests include honeypots, critical infrastructure protection and education.

Will Fantom is currently undertaking a PhD in Computer Science at Lancaster University. His research focuses on the deployment and management life-cycles of software-driven network infrastructures. This focus on network DevOps has resulted in research interests around network emulation and unikernel-driven telemetry systems.

Vasileios Giotsas is a lecturer at Lancaster University. He received his PhD from University College London (UCL), and he worked as a postdoctoral researcher at UCSD Center for Advanced Internet Data Analysis (CAIDA) and TU Berlin. His research focuses on network measurements, the analysis of the routing system, and the development of novel risk assessment and mitigation techniques.

Nicholas Race is Professor of Networked Systems at Lancaster University. His research focuses on developing future networking services built upon Software Defined Networks and Network Functions Virtualisation. This includes new techniques to enhance the detection and remediation of network anomalies. He leads the EPSRC Prosperity Partnership "Next-Generation Converged Digital Infrastructure" (NG-CDI) with BT, developing a future network that is "autonomic", with the capability to react and reconfigure infrastructure accordingly with minimal human intervention. He is also the lead at Lancaster of the EPSRC Prosperity Partnership "Future Personalised Object-Based Media Experiences Delivered at Scale Anywhere" with the BBC, which is building an intelligent network compute platform enabling the efficient utilisation of network compute and delivery resources at scale.