Encryption in ICS Network: Is it enough?

Pritam Dash
University of British Columbia

Elaine Yao
University of British Columbia

Abstract

Industrial Control Systems (ICS) have evolved from closed and isolated systems to connectivity over the internet. This has led to efficient system operations and flexibility in managing the distributed system that is usually spanned over a large area. However, this has opened doors for new attacks e.g., malicious command injection, replay attacks etc., which were previously not possible in the closed system design. Most of these attacks were possible due to lack of authorization and encryption in intra-ICS communication. To protect ICS from such attacks secure communication protocols have been proposed (e.g., secure Modbus). These protocols are becoming increasingly popular in the ICS community. In this paper, we analyze if secure Modbus protocol is vulnerable to side channel information leaks that can be leveraged to perform attacks against the ICS despite of the encryption and authentication. We use secure water treatment plant (SWaT) as our case study for this work. We find that the attacker can learn SWaT activities by monitoring packet lengths and timing variance in the encrypted network traffic and profile the ICS activities. Using this information the attacker can drop or delay critical commands which result in malfunctions such as water tank overflow and delay in various processes of the plant.

1 Introduction

Industrial control systems (ICS) are important infrastructures. They are widely used in water treatment plants, power plants, energy and petroleum industry, etc. Infiltration in these critical infrastructure can raise alarming consequences such as large scale black-outs, shutting down of oil pipelines, etc.

Traditionally ICS were closed and isolated systems, which were cumbersome to manage, challenging to upgrade, and add new features. Over the last decade, ICS have moved from closed systems design to connectivity over the internet. Modern ICS use TCP protocol to connect various components of the distributed system (servers, workstations, PLCs, sensor and actuators), support sending commands over wireless network, support remote login over internet, etc. [15]. This evolution of ICS has led to efficient system operations such as the ability to monitor multiple sensors and control actuators spanning over a large area, ability to issue commands remotely. However, the inter-connected nature of the ICS presented new security risks. Attacks targeting cyber components (e.g., control system, network, servers etc.) of the ICS have been demonstrated [9, 11, 16, 17, 24, 26]. Recent events such as the Colonial pipeline attack [1] and the Florida water treatment plant attack [3] have raised an important question i.e., how secure are the critical ICS systems?

Most of these attacks were possible because the network protocols used in ICS were designed without security considerations (e.g., no encryption, no authentication). Therefore, it was possible for an attacker with relatively less privilege to launch attacks by modifying commands in network packets, replaying past commands, changing control logic [26]. The attacker does not need to compromise any component of the ICS, the above attacks could be launched simply by monitoring the intra-ICS network traffic. To overcome this problem secure communication protocols for ICS have been proposed. One such example is secure Modbus [6, 25]. The secure Modbus protocol is built on existing TCP protocol and utilizes TLS for encrypted channel and authentication [6]. Modern ICS are increasingly deploying encrypted communication protocols. The use of encryption in ICS communication protects the ICS network traffic from command modification attacks, session hijacking attacks, to privacy attacks. In this project, we analyze if the state-of-the-art security protocols used for securing modern ICS communications are adequate for preventing network attacks in the ICS. Specifically, we show that an attacker can leverage side channel information leaks to launch attacks despite of encrypted communication channel.

Our motivation comes from website fingerprinting attacks [10, 14, 18, 21, 23, 27] that target web applications and learn users' internet activities (e.g., which websites the user is visiting, which videos the user is watching) in spite of encrypted network channel. The main idea behind website fingerprinting attacks is to monitor and analyze the encrypted

network packets and profile details such as packet lengths and timing relationships between packets to disclose users' internet activities.

In the context of ICS, secure Modbus protocol prevents the attacker from modifying commands or monitoring realtime network in plaintext. However, some of the crucial information about the intra-ICS communication might still get exposed from the traffic metadata (e.g., source and destination IP, packet timing, packet sizes, etc.) that are not obfuscated by encryption. We find that an attacker can snoop on the encrypted network communication, collect traffic metadata information and profile the information to learn ICS activities (e.g., timing variance of various commands, when a critical command is sent). This can be achieved by monitoring the timing behaviour of the communication between devices, e.g., Programmable Logic Controller (PLC) in the ICS, and monitoring the packet lengths of the encrypted network traffic (referred to as side channel information leaks henceforth).

Using the side channel information, the attacker can drop a few packets containing a critical command (e.g., close the valve and stop water supply) or delay critical commands (this is a temporary delay instead of the traditional DoS attack). As most actuator commands in ICS are time-critical in nature, dropping or delaying such commands will cause the ICS to malfunction. Therefore, despite of the encrypted communication channel and authentication, an attacker with relatively few privileges can still attack the ICS.

In this work, using the SWaT testbed [20] as a case study. We believe that our work is not tied to a specific ICS and it extends to other ICS which are distributed in nature and use secure Modbus (or similar) protocols. Our contributions are as follows: :

1. We find that network traffic metadata reveals significant details about the ICS activities such as cyclic nature of operations (i.e., certain tasks are performed at fixed time and in fixed intervals). In addition, the network patterns e.g., source and destination of traffic, reveal the current state of the ICS.

2. We demonstrate that crucial information about the ICS can be obtained through side channel leaks. The attacker can learn the stage transitions in ICS operations simply by profiling the network packet lengths and burst frequency.

3. We demonstrate that attackers can leverage information about the ICS learned from the side channel leaks to perform active attacks that result in ICS malfunctioning and disruptions in ICS operations.

4. Our results on MiniCPS simulator [4] shows that 4 hours of network traffic profiling is enough to launch attacks that results in water tank overflowing and delaying the process stages which will lead to water supply shortage.

2 Background

In this section, we first present the ICS we use for our case study, followed by the network protocol we are studying and finally our threat model.

2.1 Water Treatment Plant (ICS) Architecture

We use a simulated version of the secure water treatment plant (SWaT) in our experiments, which is a test bed from SUTD [20] in Singapore. This test bed is widely used to experiment hardware and software developments for real-world ICS. SWaT represents a small scale representation of a modern water treatment plant.

Figure 1a shows the SWaT testbed which is a distributed control system that has 5-6 process stages. For each process stage, a local PLC is used for computation and communicating with sensors (e.g., water level sensors) and actuators (e.g., valve). Each process stage is controlled by the local PLC, and the process stage transitions are sequential. The PLCs communicate over wireless network, and share their current state information.

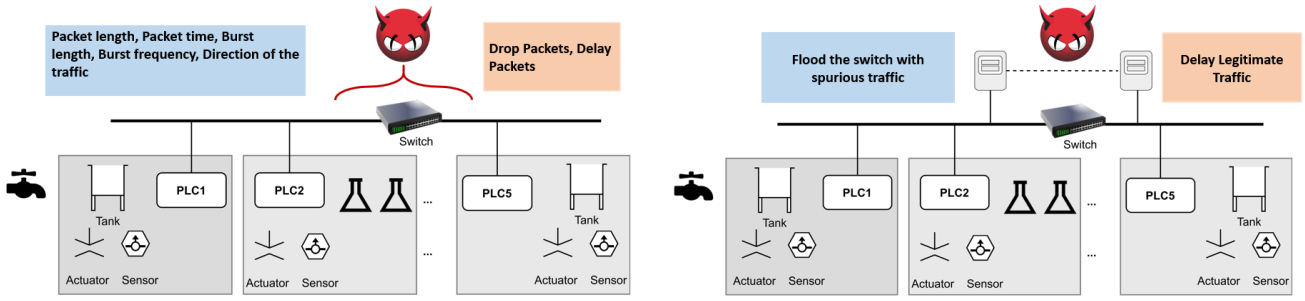
For example, in stage 1 (the first square from right in Figure 1a), the PLC obtains the sensor readings and controls the actuators, e.g., turn ON the pump or opening a valve to let water flow in to the tank and constantly monitor the water level. Based on the sensor readings that show the water level, the PLC decides when to close the value and turn water pump OFF. SWaT uses several other sensors in subsequent stages to check the physical and chemical properties of water. The current states of one $PLC1$ affect the states of other PLCs. The plant operator can also manually control the actuators through the HMI, and the SCADA system.

2.2 SWaT Network

The SWaT testbed uses WiFi to communicate between the PLCs, and for communication between the process stages and the HMI and SCADA. SWaT can be configured to use Modbus/TCP or EtherNet/IP protocols. We limit our focus to Modbus/TCP protocol.

2.2.1 Network Protocol

Modbus is protocol proposed by Modicon in 1979 for communication among programmable logic controllers(PLCs). Various versions of Modbus protocol have been proposed including Modbus/RTU(series transmission) and Modbus/TCP(Ethernet transmission) [5]. However, Modbus/TCP is lack of authentication to establish the sessions and encryption of the communication [2], which exposes many vulnerabilities. Attackers may spoof the master/slaves devices or interfere with the TCP/IP connection [7]. Parian *et al.* [22]



(a) Attacker compromises network access point

(b) Attacker floods the switch

Figure 1: Water treatment plant architecture and Attack Model

proposed infecting the master with malware and man-in-the-middle attacks on Modbus, making the master accept incorrect information about the slave.

To limit the above attacks encrypted Modbus/TCP was proposed by adding the TLS layer, which is called Modbus/TCP Security [6] or secure Modbus. TLS provides authentication capability so that the clients and servers need keys and certificates to identify each other before establishing connections. Several papers analyze the system performance of secure Modbus, such as the connection time [13]. However, no prior work has studied what type of information the protocol reveals despite encryption that can be leveraged to learn the ICS operations.

2.3 Threat Model

The goal of the attacker is to learn the ICS operations and obstruct the normal operation of the ICS. In case of SWaT, the attacker aims to cause a water tank over-flow which could damage the plant equipment, delay the process stages. This would delay the water purification timeline and eventually lead to water supply shortage. In our work, we consider two types of attacker: Attacker A who can compromise a network access point such as a switch (shown in Figure 1a) and Attacker B can connect to the WiFi network of the plant (shown in Figure 1b).

Attacker A can compromise a network access point by scanning for vulnerable ports and installing malware in a network access point [19]. Attacker A can be an IXP (or ISP) employee who can record the network traffic of the distributed ICS connected to the internet. Attacker A has the following capabilities: The attacker knows the ICS design such as number of process stages, PLCs, types of sensors and actuators, number of tanks etc. In addition, the attacker can obtain a specification of the ICS and knows which tasks are performed by which PLCs (e.g., PLC1 controls water level in Tank1, PLC2 performs chemical processes). The attacker can compromise a network access point (e.g., switch) [12] to snoop on the encrypted network traffic and profile ICS activities based on the network packet metadata and side channel information leaks (e.g., packet length, timing, burst, direction of packet etc). The attacker can delay and drop packets that pass through the compromised network device.

Attacker B has all the above capabilities of Attacker A, except that Attacker B cannot compromise a network access point. Instead, Attacker B is located close to the plant and connects to the plant wireless network. This can be achieved by exploiting bad security practices such as not changing the default wireless password or through social engineering attacks. With this, Attacker B can transmit spurious packets between its fake nodes and delay the legitimate ICS network traffic.

Attackers can also compromise a network access point by exploiting known vulnerabilities such as CVE-2021-1374 and CVE-2021-1611. However, the attacker cannot decrypt or modify the network packet contents. We also assume that the attacker cannot compromise any of the ICS components (gain physical access, or network access), and cannot inject spurious commands (false commands) in the network packets. We assume that end to end encryption is used in the ICS network.

3 Methodology

In this section we describe the various challenges to perform the proposed attacks and the methodology addressing those challenges.

3.1 Challenges

The following are the challenges that an attacker has to overcome:

C1 Identifying the various process stages and critical commands, e.g., in the SWaT testbed (i) identifying when commands are sent to PLC1 or PLC2, (ii) when PLC1 closes the valve.

C2 Identifying which packets to drop or delay and launch attacks with minimal packet drops or delays.

C3 Because the intra-ICS communication between various components does not occur at same frequency (i.e., some components send signals at second level granularity, other components operate at microsecond level), it is hard to derive correlations in SWaT operations by manual analysis.

Unlike website fingerprinting attacks where the attacker can collect network traffic data and record web flows for multiple web services and compare that with the victim's network traffic, in the case of ICS the attacker has limited profiling capabilities as the attacker won't have the ability to compare the ICS network traffic with any ground truth. One of the main challenges to perform this attack is to identify the critical commands (e.g., turn off water supply) or sequence of plant operations that is related to issuing of a critical command. Then, identify the corresponding network traffic signature associated with critical commands. Dropping such critical commands will result in malfunctions (e.g., the water tank overflows). In addition, identifying dependencies of a particular critical commands can disrupt (or delay) SWaT operations. For example, PLC_*a* and PLC_*b* perform some computations and that is crucial for PLC_*b* to send a command. Delaying the communication between PLC_*a* and PLC_*b* will delay the issuing of the command.

As we assume end to end network encryption, we learn the correlation between various ICS operations and transitions in process stages, and make an educated guess of when a critical command is sent and from which component.

3.2 Attack Preparation

To address the above challenges we propose an attack in two phases: (1) Offline profiling phase where we (the attacker) collect network traffic of the victim ICS (SWaT) and we record features such as source and destination IPs, replicate the target ICS design (in simulation) and run the various components to record the network traffic. (2) Active attack phase, where we will drop or delay packets in the ICS network communication.

We assume that the attacker can compromise a network access point close to the ICS hardware and collect the encrypted network traffic. We also assume that the attacker can obtain a specification of the ICS and knows the various process stages involving the PLCs and can figure out which PLCs interact with water tanks and which PLCs perform chemical processes. In the offline profiling phase, our goal is to learn the various system dependencies, timing behaviour of various intra-ICS communications and identify the process stage transitions. The following are the steps in the attack preparation phase:

1. We use Wireshark to collect and dissect SWaT network traffic. We record the communication between each pair of PLCs to get a fine grained understanding of the SWaT activities. In the profiling phase, we also collect the water level in the tanks to draw correlations between network traffic and the SWaT process stages.
2. We use Wireshark to remove all the unwanted non-ICS protocols (e.g., SSL). Packets tagged as TCP or UDP are ICS communication related, and hence interesting to us.
3. We collect network traffic data for various duration (e.g., 1h, 2h, 4h, 8h) of SWaT operations.
4. By observing the source and destinations, and the acknowledgment sequence in wireshark we can identify the sender and the receiver PLCs for a particular packet.

After we collect network traces for profiling, we plot the packet length and the water level in the tank to find the correlation between water level and communication between the sender and receiver PLCs. We find that change in the packet length has a cyclic nature e.g., the packets of certain fixed sizes are sent between PLC_a and PLC_b periodically. This periodic change in packet length shows the state transitions in the SWaT system. For example, by observing the packets patterns of PLC1, we can guess when stage 1 completes and stage 2 begins(Addressing C1).

We also noticed invariably large packets being sent in the network. These packets are 2X larger than the other packets. We identified that these packets represent service requests, and dropping these large packets will result in service unavailability (Addressing C2).

In order to address C3 related to finding correlations between encrypted network packets, we derive a cyclic pattern detection algorithm (CDA). The CDA leverages the stage transition in SWaT and identifies when to drop the packets in order to delay the following stage.

3.3 Cyclic Pattern Detection Algorithm

In order to find the cyclic pattern in the modbus packet lengths, we devise our own cyclic pattern detection algorithm(Algorithm 1). We choose the TCP segment length as the feature as it has close relationship with the Modbus packet length. The packets with length 0 are ignored as they are mainly ACK packets which don't carry the information of commands. And we define the pattern as the combinations appearing at least twice.

Algorithm 1 shows how we get the cyclic pattern. Line 1 to Line 7 are used for initialization. We first define that the length of the potential pattern p is two. A dictionary *p_Dict* is used to store the patterns that are already been detected. It is initialized to empty at the beginning. Line 8 to Line 25 is the while loop where it will search for all the possible patterns in the series. Line 9 first search that if the potential pattern p has already appeared in the dictionary *p_Dict*, if so, Line 10 adds this to the final pattern array *p_Array*. If not, Line 11 will compare p with the next 2 elements in the time series. If they matches, then Line 12 and Line 13 declare that a new pattern

p is found and add this to p_Dict. If they still fail to match, p will be expanded by 1 in Line 16. In other words, if the length of previous p is two, then it's three. And in Line 17 to Line 19, it continues to compare the p with the next 3 elements in the time series. If they still doesn't match, in Line 21, the length of p and the elements to be compared will increase. At last, this algorithm will stop when the whole array has been explored.

Algorithm 1 Cyclic pattern detection algorithm

```
1:  arrayofPac: array of the TCP segment length
2:  p: potential pattern
3:  p_Array: final pattern array
4:  p_Dict: dictionary of detected pattern
5:  idx: the number of elements to match
6:  idx ← 2
7:  p ← arrayofPac[0 : 1]
8:  while arrayofPac is not fully checked do
9:      if p is in p_Dict then
10:         add p into p_Array
11:     else if p matches next two elements in arrayofPac
    then
12:         add p into p_Array
13:         add p into p_Dict
14:     else
15:         while idx elements are unchecked do
16:             idx++
17:             if p matches next idx elements in arrayofPac
    then
18:                 add p into p_Array
19:                 add p into p_Dict
20:             else
21:                 idx++
22:             end if
23:         end while
24:     end if
25: end while
```

3.4 Packet Bursts Detection

For this we use timing as a feature. Network communication in SWaT testbed occurs at variable frequency i.e., communications between some PLCs happen at microsecond level granularity and others happen at second level granularity. To measure the bursts, we record the packet transmission at second level granularity for each pair of PLCs in the network. We use wireshark to observe the network traffic and record communication between the target pair of PLCs. We notice that a lot of packets carrying acknowledgments, hence they do not carry any payload. Thus, to improve our observation of burst patterns we record only the TCP segment length (payload) instead of entire packet size. We also record the water level in the tanks to correlate how the burst patterns associate with the water level and other SWaT activities (e.g., close value).

3.5 Active Attack

In this phase, we launch attacks based on the information collected about the SWaT activities in the attack preparation phase. Recall that in attack model we assume that the attacker can compromise a network access point and the attacker cannot see the packet contents as the SWaT system uses end to end encryption. Considering the attacker's capabilities we launch the following two attacks: 1) Drop network packets - As most modern ICS use software defined networking (SDN), an attacker can configure a switch with a rule to drop certain packets, e.g., packets of certain signature and time when to drop packets. 2) Delay network packets - Similarly, the attacker can define rules to delay certain packet, and sequence of packets intermittently.

Ideally, the attacker will observe the cyclic pattern and the burst patterns and start packet drop or delaying the packets when such network traffic patterns are observed. The cyclic pattern or the burst patterns suggest a change in the SWaT's process stages, and droping or delaying the network packets that are crucial to a particular transition the SWaT process stage will result in malfunctions. Moreover, the attacker will aim to drop and delay packets in a stealthy manner in order to avoid detection. For example, the attacker will drop the packets for a certain duration and then allow the network to recover.

4 Experiments

In this section, first, we present our experimental test bed, followed by the evaluation methodology for our approach.

4.1 Experiment setup

To evaluate our attack experimentally, we implemented a realistic setting. The implementation details of the main components are described as follows.

Water Treatment Plant We choose the secure water treatment plant(SWaT) as our target ICS as it is one the most widely adopted ICS in prior academic work. We use a simulation of the SWaT testbed called MiniCPS [8]. The simulation is realistic and has the exact number of PLCs and other components as the real testbed.

Network Simulation MiniCPS is build upon Mininet [4], and it is extended with tool to simulate CPS components such as the PLCs and the industrial protocols(Modbus/TCP). It also provides the network topology for SWaT testbed. All the PLCs are connected to the supervisory SCADA system through a common switch in an Ethernet star topology.

Network Protocol We apply the built-in unencrypted Modbus/TCP protocol in the network simulation. We could not

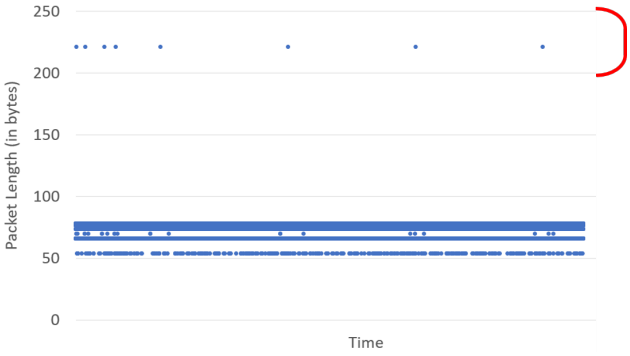


Figure 2: Plot of Modbus packet lengths

configure secure Modbus protocol which is an end to end encryption protocol. However, to make our analysis realistic we do not look into the contents of the Modbus packets (emulating observing encrypted network packets). The attacker is only able to observe the size and timing information of the network packages and not the contents.

Launch Attacks To simulate an attacker who can compromise the network access point, we apply `ovs-ofctl` to drop packets as other options such as `iptables` and `tc qdisk` work better on Linux switches. In the SDN-based controller, `ovs-ofctl` could be leveraged to drop packages according to its ip address and packet length. As for packets delaying, `tc qdisk` is used based on the specified delay time and ip address.

4.2 Attack 1: Drop Largest Packets

Figure 2 shows a plot of the packet lengths of modbus packets for 2 hours of SWaT operations. We find that most of the packets are of length 66-72 bytes. As can be seen in the figure (marked by the red bracket), a small number of packets are invariably larger than the rest of the packets. We looked in to the SWaT code to investigate why these small number of packets are larger than the rest, and we found that PLCs make a request periodically to discovery the running services. This service request and service discovery transaction are carried via largest packets we see in Figure 2.

In our first attack, we drop the large packets. To perform this attack we write a simple `ovs-ofctl` script that monitors the modbus packet lengths and drop packets which are larger than 200 bytes. Note that all the service request and service discovery related packets are larger than 200 bytes.

The attack is launched soon after the simulation starts. We notice that the attack crashes PLC_1 program (SWaT setup is shown in Figure 1b). As rest of the PLCs operate independently, they perform their respective tasks without any updates from PLC_1. After almost 10 minutes, we observe the water level of the tank exceeds the threshold i.e. 800 cm.

This experiment shows that an attacker that can observe the modbus packet lengths can launch an attack to drop the largest packets to cause a tank overflow.

4.3 Attack 2: Cyclic Pattern Attack

To launch this attack, first we collect network traffic data and perform offline profiling. Particularly, we observe the TCP segment length and use Algorithm 1 to detect the cyclic pattern in SWaT network traffic. In our experiment, we observe two types of patterns namely Pattern 1 that has TCP segment length $[12, 11]$ and Pattern 2 $[12, 12]$. In each pattern, the value of the elements represents the TCP segment length in each network packet. For instance, Pattern 1 $[12, 11]$ suggests that every time a packet with the TCP segment length 12 shows up, it will be followed by a packet with the TCP segment length 11.

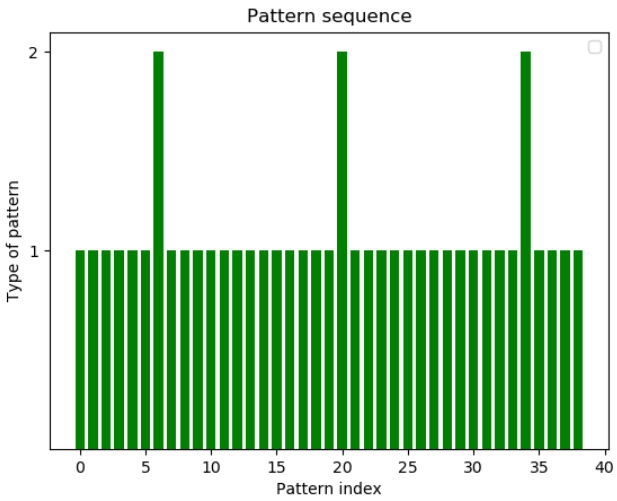
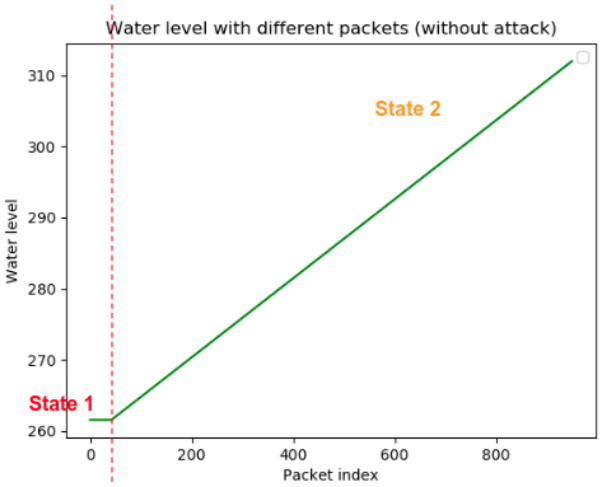


Figure 3: Detected cyclic pattern with Algorithm 1

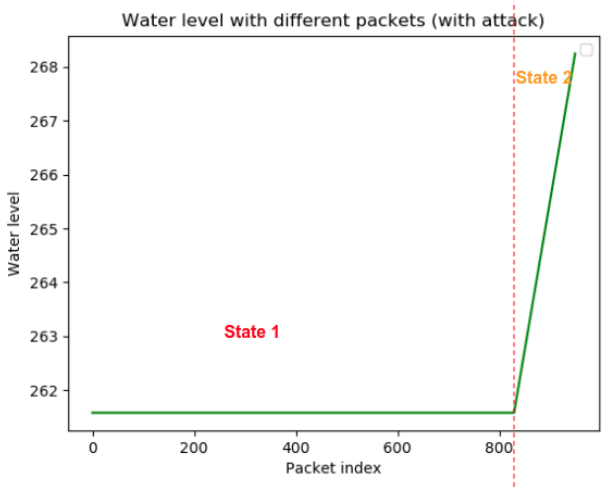
Figure 3 shows a part of the patterns and the corresponding TCP segment length during the offline profiling process. Value = 1 in y-axis represents that it's Pattern 1 and Value = 2 in y-axis represents that it's Pattern 2. The x-axis shows the index of the patterns. We can observe from Figure 3 that Pattern 1 is the dominant pattern, the shift in Pattern 1 and Pattern 2 is cyclic in nature, and Pattern 2 follows 5 instances of Pattern 1.

Our goal is to disrupt the cyclic nature of Pattern 1 and Pattern 2. Therefore, the attack strategy is to drop packets for 3 minutes every time we observe the Pattern 2. We choose to drop packets for 3 minutes because TCP will resend the packets that are not received at the destination. Longer attack time assures that the dropped packets won't be received by other PLCs. In the attack phase, we wait till the PLCs start to communicate, and we monitor the length of the TCP segments. We wait till Pattern 2 is observed in the network traffic, and then we start dropping packets for 3 minutes.

The consequences of packet dropping are as follows: PLC_1 requests the value of the status of the valve from



(a) Water level with different packets(without attack)



(b) Water level with different packets(with attack)

Figure 4: Water level with different packets under different scenarios

other PLCs. The status of the valve is represented as *HMI_MV201_STATUS* which can be set to 1 or 0 i.e., on or off. . Note that we looked in to the code to identify this, an attacker cannot see this information. Due to the attack, PLC_1 fails to receive the valve status and therefore it can't update the current valve status of other PLCs. As a result, the process stages of the water treatment is delayed. Figure 4a shows the process stage transition under no attack scenario. As can be seen the water level begins to increase after 40 packets. Figure 4b shows the process stage transition after dropping packets. When there is no attack, it takes 10 seconds for the water level to change, which is about after 20 packets are sent. However, after the cyclic pattern attack is launched, it takes 254 seconds for the water level to enter State 2, which is about 820 packets are sent. In other words, the attacker successfully delays the time needed for stage transition of water level by 25 times.

Our results shows that an attacker can delay the regular operations of SWaT by observing the cyclic pattern in packet length and dropping packets for 3 minutes.

5 Discussions

In this section we discuss the limitations of our experiments, followed by possibility of more observation from timing side channels, and finally, we discuss how the attacker will know whether the attack was successful.

5.1 Limitations of our Experiments

We have performed all our experiments on Modbus/TCP which transmits network packets in plaintext, though we emulate a realistic attack setting by not looking into the network packets for launching the attacks. One of the concerns with our experimental setup is the possibility of losing packet length pattern observation when encryption is applied. Typically, TLS communication uses AES 128 (or larger size key) for encrypting the payload. AES encryption will add padding to the plaintext payload when encrypting. We performed analysis to identify what happens when the same network traffic analysis is performed on secure Modbus protocol that may use AES 128.

Our findings are as follows:

1. Modbus packets that are greater than 200 bytes in size will become 871 bytes and 917 bytes when encrypted with AES-CBC and AES-GCM modes respectively. AES key length is 128 bits in both cases. Both CBC and GCM are popular modes of implementing AES encryption in high-level languages. These packets are crucial to performing Attack-1, and we find that even with encryption it is possible to observe the invariably large packets and launch Attack 1 using our methodology.

2. TCP segment lengths of size 11 and size 12 bytes will become 96 bytes after encryption with AES 128 if CBC mode is used. This will result in losing the cyclic pattern observations that is key to performing Attack 2 which delays SWaT process stage transitions. However, if AES 128 with GCM mode is used, TCP segments of 11 and 12 bytes will become 161 and 165 bytes respectively. Therefore, if GCM mode is used we can observe the cyclic patterns in TCP segments and perform Attack-2 on secure Modbus protocol. We suspect that the difference in the length of encrypted packets when using CBC and GCM is due to the way these two methods apply padding (our guess, we have not investigated this).

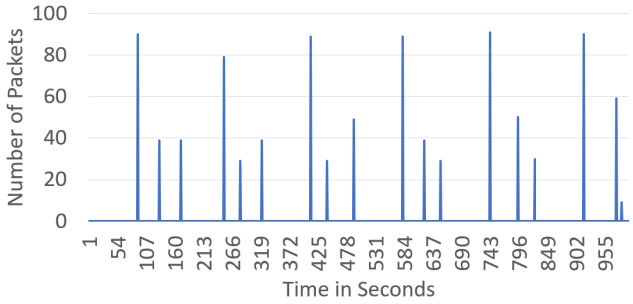


Figure 5: Plot of Packet burst observed at 1 second interval

We make similar observations for AES 256. When encrypted with CBC mode, TCP segments of 11 and 12 bytes are padded to same size after encryption. However, when using AES encryption with GCM mode, the packet sizes vary after encryption. Furthermore, this analysis shows that packet length alone is not a very interesting features. To draw more interesting correlations timing should be considered as the feature.

5.2 Observation using Timing Channels

We analyzed what can be learned by observing timing channel, particularly we focus on burst patterns. Figure 5 shows the packet burst pattern sampled at 1 second frequency. We plot the TCP segments alone, this is to eliminate TCP acknowledgment packets which do not carry any payload. We can see a consistent cyclic nature in packet burst patterns. We have not performed any attack leveraging this observation, but from our experience in performing Attack 2 we imagine that we can use the cyclic pattern detection algorithm and drop or delay packets that will disrupt the SWaT process stages.

More interesting observations can be made by correlating the burst patterns with the direction of the traffic and observing the water-level in the tank during the offline profiling phase. We leave this for future work.

5.3 Limitations of Attack 1

We mentioned in Section 4 that Attack 1 successfully causes the tank to overflow. However, as this attack crashes one of the PLC programs, it is not stealthy in nature. There will be significant latency from the time the attack is launched to the time the tank overflows. If a platform operator notices one of the PLCs stopped functioning and then restarts that PLC, Attack 1 will not have any implication.

However, it is hard for the plant operators to diagnose and locate the origin of the attack. Thus, an attacker can repeatedly or intermittently perform Attack 1 that may eventually result in the tank overflow.

5.4 Feedback to the Attacker

In our attack model we consider that the attacker can be an external player and can target the ICS. It is important to identify how the attacker will know whether the packet drops or delays satisfied their attack goals such as tank overflow or delaying ICS process stages. We find there are mainly two ways for the attacker to get feedback: 1) The attacker can collect a large amount of legitimate (no attack) ICS network traffic and profile the patterns of legitimate traffic to identify the traffic patterns of the responses (e.g., acknowledgments) corresponding to events such as process stage transitions, burst patterns, etc. When the attack is launched and if the attack is successful, the response traffic patterns will differ from the response traffic of legitimate regular events. The varying response traffic can be used as feedback. 2) The attacker may gain access to live camera feedbacks of the distributed ICS setup and observe the current actions of ICS after launching the attacks.

6 Related Work

Attacks have been demonstrated against web application despite encrypted network. Chen *et al.* [10] shows attacker can infer the user sensitive data based on the web traffic pattern. Their main idea is that modern web application make stateful state-transitions, the authors profile victim's internet activities based on this observation and disclose sensitive information about the victim. Heyes *et al.* [14] proposes k-fingerprinting, a more powerful website fingerprinting technique with random decision forests. Panchenko*et al.* [21] maps network traces to a class and proposes a classifier to get the feature of the traffic. Schuster*et al.* [23] shows the encrypted video stream can be identified be a remote attacker. The authors use packet bursts as a key feature to fingerprint encrypted streams. Zhuo *et al.* [27] focuses on user's hyperlink transitions between websites and applies Profile Hidden Markov Model(PHMM) to determine a series of pages.

We use side channel information such as packet length, patterns of packet size, burst patterns etc. which were used in some of the prior works as well. However, in all the prior work, the attacker has the ability to collect large amount of internet traffic data as the web services are public, therefore, the attacker can interact with the web services. Because the attacker can collect large amount of data, they can use classification techniques to to compare victim's data with ground truth. In our case, the attacker cannot follow the same methodology, as ICS system programs are not public, therefore, the attacker cannot obtain ground truth for network traffic patterns. Therefore, our approach is to correlate various features found in network traffic and use specific observation such as cyclic patterns and burst patterns to make the best guess about ICS activities.

All the prior work in website fingerprinting use side-

channel leaks to disclose user's internet activities. In contrast, this work goes one step further and leverages side channel information leaks to perform active attack on the ICS.

7 Conclusion and Future Work

In this project we investigated the possibility of using side channel leaks to learn about activities (e.g., process stage transitions, service requests, burst patterns) of Industrial Control Systems and leveraging the above information to perform active attacks against the ICS. We find that despite of end to end encryption and authentication in the ICS network, an adversary can learn crucial ICS activities by profiling the side channel leaks such as packet length, payload length, packet burst patterns, etc. We used secure water treatment plant (SWaT) as a case study for our project, and we collected network traffic and dissected the traffic to obtain the aforementioned side channel leaks. Our main findings are as follows- 1) By dropping a single packets it is possible to disrupt service requests made by PLCs (Attack 1) which results in water tank overflow. 2) By dropping packets for 3 minutes it is possible to delay the process stage transitions by 25 times. 3) More interesting observations can be made using burst patterns and combining the timing channels and packet length.

In future work, we plan to explore more side channels such as inter-packet timing, direction of the traffic and devise attacks based on the new observations. We could not get to exploring the the attacker presented in Figure 1b that can gain access to the ICS wifi and initiate communication between nodes to flood the switches that may delay legitimate traffic. We imagine the results of this attack will be similar to the Attack 2 presented in Section 4.

Acknowledgment

We thank Gargi Mitra (PhD student at IIT Madras, India) for her collaboration on this project, helping us with debugging various software components of SWaT simulation testbed, and for the numerous brainstorming sessions.

References

[1] Cyberattack forces a shutdown of a top u.s. pipeline. https://www.nytimes.com/2021/05/08/us/politics/cyberattack-colonial-pipeline.html. Accessed: 2021-10-13.

[2] Evolving towards secure modbus. https://www.incibe-cert.es/en/blog/evolving-towards-secure-modbus. Accessed: 2021-10-13.

[3] The florida water plant attack signals a new era of digital warfare. https://www.darktrace.com/en/blog/the-florida-water-plant-attack-signals-a-new-era-of-c Accessed: 2021-10-13.

[4] Mininet. http://mininet.org/. Accessed: 2021-10-13.

[5] Modbus messaging on tcp/ip implementation guide v1.0b. https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. Accessed: 2021-10-13.

[6] Modbus/tcp security. https://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf. Accessed: 2021-10-13.

[7] Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, 2008.

[8] Daniele Antonioli and Nils Ole Tippenhauer. Minicps: A toolkit for security research on cps networks. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, CPS-SPC '15, page 91–100, New York, NY, USA, 2015. Association for Computing Machinery.

[9] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Mark Felegyhazi. The cousins of stuxnet: Duqu, flame, and gauss. *Future Internet*, 4(4):971–1003, 2012.

[10] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *2010 IEEE Symposium on Security and Privacy*, pages 191–206, 2010.

[11] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

[12] Davide Fauri, Bart de Wijs, Jerry den Hartog, Elisa Costante, Emmanuele Zambon, and Sandro Etalle. Encryption in ics networks: A blessing or a curse? In *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 289–294. IEEE, 2017.

[13] Matheus K. Ferst, Hugo F. M. de Figueiredo, and Gustavo W. Denardin. Connection time in modbus/tls for secure communications on photovoltaic systems. In *2019 IEEE 15th Brazilian Power Electronics Conference and 5th IEEE Southern Power Electronics Conference (COBEP/SPEC)*, pages 1–6, 2019.

[14] Jamie Hayes and George Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1187–1203, Austin, TX, August 2016. USENIX Association.

[15] Ki-Seob Hong, Hyo-Bin Kim, Dong-Hyun Kim, and Jung-Taek Seo. Detection of replay attack traffic in ics network. In Roger Lee, editor, *Applied Computing and Information Technology*, pages 124–136, Cham, 2019. Springer International Publishing.

[16] Johannes Klick, Stephan Lau, Daniel Marzin, Jan-Ole Malchow, and Volker Roth. Internet-facing plcs-a new back orifice. *Blackhat USA*, pages 22–26, 2015.

[17] Cheng Lei, Li Donghong, and Ma Liang. The spear to break the security wall of s7commplus. *Blackhat USA*, 2017.

[18] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, page 255–263, New York, NY, USA, 2006. Association for Computing Machinery.

[19] John Matherly. Complete guide to shodan. *Shodan, LLC (2016-02-25)*, 1, 2015.

[20] Aditya P Mathur and Nils Ole Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016.

[21] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. Website fingerprinting at internet scale. *Proceedings 2016 Network and Distributed System Security Symposium*, 2016.

[22] Christopher Parian, Terry Guldimann, and Sajal Bhatia. Fooling the master: Exploiting weaknesses in the modbus protocol. *Procedia Computer Science*, 171:2453–2458, 2020. Third International Conference on Computing and Network Communications (CoCoNet'19).

[23] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Beauty and the burst: Remote identification of encrypted video streams. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1357–1374, Vancouver, BC, August 2017. USENIX Association.

[24] Ralf Spenneberg, Maik Brüggemann, and Hendrik Schwartke. Plc-blaster: A worm living solely in the plc. *Black Hat Asia*, 16:1–16, 2016.

[25] Luo Xuan and Li Yongzhong. Research and Implementation of Modbus TCP Security Enhancement Protocol. In *Journal of Physics Conference Series*, volume 1213 of *Journal of Physics Conference Series*, page 052058, June 2019.

[26] Hyunguk Yoo and Irfan Ahmed. Control logic injection attacks on industrial control systems. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 33–48. Springer, 2019.

[27] Zhongliu Zhuo, Yang Zhang, Zhi-li Zhang, Xiaosong Zhang, and Jingzhong Zhang. Website fingerprinting attack on anonymity networks based on profile hidden markov model. *IEEE Transactions on Information Forensics and Security*, 13(5):1081–1095, 2018.