This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2018.2823906, IEEE Embedded Systems Letters

Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers

Thiago Alves
Electrical and Computer Engineering
The University of Alabama in Huntsville
Huntsville, U.S.A.
thiago.alves@uah.edu

Rishabh Das
Electrical and Computer Engineering
The University of Alabama in Huntsville
Huntsville, U.S.A.
rd0029@uah.edu

Thomas Morris
Electrical and Computer Engineering
The University of Alabama in Huntsville
Huntsville, U.S.A.
tommy.morris@uah.edu

***Abstract*— During its nascent stages, Programmable Logic Controllers (PLC) were made robust to sustain tough industrial environments, but little care was taken to raise defenses against potential cyberthreats. The recent interconnectivity of legacy PLCs and SCADA systems with corporate networks and the internet has significantly increased the threats to critical infrastructure. To counter these threats, researchers have put their efforts in finding defense mechanisms that can protect the SCADA network and the PLCs. Encryption and Intrusion Prevention Systems (IPS) have been used by many organizations to protect data and the network against cyber-attacks. However, since PLC vendors do not make available information about their hardware or software, it becomes challenging for researchers to embed security mechanisms into their devices. This paper describes an alternative design using an open source PLC that was modified to encrypt all data it sends over the network, independently of the protocol used. Additionally, a machine learning-based IPS was added to the PLC network stack providing a secure mechanism against network flood attacks like Denial of Service (DoS). Experimental results indicated that the encryption layer and the IPS increased the security of the link between the PLC and the supervisory software, preventing interception, injection and DoS attacks.**

Keywords—SCADA, PLC, OpenPLC, Encryption, AES, Machine Learning, IPS

I. Introduction

Modern society is dependent on critical infrastructure including water treatment, electricity generation and distribution, petroleum refining, chemical processing, and automated manufacturing. Failure of these cyber physical systems can lead to significant financial and property losses and physical harm to the citizenry. Industrial Control Systems (ICS), also known as Supervisory Control and Data Acquisition (SCADA) systems, manage complex and potentially dangerous processes using peripheral devices such as Programmable Logic Controllers (PLC) that directly interface with the process plant or machinery.

ICS operational networks can be penetrated by cyber-attackers. Although firewalls, demilitarized zones and data diodes can often isolate ICS operational networks, it cannot be assumed to stop all penetrations. Even air gapped operational networks can be penetrated. Hackers can use infected thumb drives, infected software updates, insiders, and spear phishing attacks to penetrate heavily isolated and air-gapped operational networks. The Stuxnet malware [1] is a famous example of a worm which penetrated an air gapped network by exploiting a USB thumb drive autorun vulnerability.

The work presented on this paper makes the assumption that ICS operational networks will be penetrated, and therefore presents an embedded resiliency system that can defeat cyber-attacks after the penetration. Cyber-attacks from three categories are considered: interception, injection and Denial of Service (DoS).

Given that vendors do not make available any information about their hardware or software, this work augmented an open source PLC [2] with AES-256 encryption and decryption capabilities, and a machine learning-based IPS, to create the embedded resiliency system. The OpenPLC platform includes a program development environment, supports popular SCADA protocols such as Modbus/TCP and DNP3, and includes an open source Human Machine Interface (HMI) editor called ScadaBR. The OpenPLC project was created in accordance with the IEC 61131-3 standard [3], which defines the basic software architecture and programming languages for PLCs. This means that OpenPLC can be compatible with other PLCs that are compliant to the same standard.

II. Related Works

There are many examples in the literature where researchers applied cryptography to secure SCADA communication. The work put forth by Fovino [4] describes a secure version of the Modbus SCADA protocol that incorporates integrity, authentication, non-repudiation and anti-replay mechanisms. The integrity, authentication, and non-repudiation of the secure Modbus packet are guaranteed using SHA2 to compute a secure

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2018.2823906, IEEE Embedded Systems Letters

digest of the packet, combined with a RSA-based signature scheme. Since the original Modbus protocol was modified, a secure gateway must be provided to enable legacy devices to communicate.

Majdalawieh [5] proposed a security framework for the Distributed Network Protocol Version 3 (DNP3). Since DNP3 was never designed with security mechanisms in mind, the protocol lacks any form of authentication or encryption. The described DNPSec architecture adds confidentiality, integrity, and authenticity to the original DNP3 and uses encryption to protect against eavesdropping and to hide the frame source. This solution requires modifications to the DNP3 Data Link Layer and therefore cannot be applied to certified DNP3 devices without a proper gateway to translate the messages.

By analyzing the works put forth on this field, it seems obvious that it is not possible to secure legacy protocols without modifying their original structure. However, a modified protocol cannot be compliant with the current industrial standards and therefore will not be compatible with the actual industrial devices, such as PLCs. Adding gateways to translate the modified protocol can enable PLCs to communicate, but it is an expensive alternative, not to mention that it adds another hop on the network which can even prevent the real-time nature required by SCADA systems. A better alternative is to embed the secure layer in the PLC itself, which can only be done if the source code for the PLC is available.

Although encryption can protect data confidentiality and also offer passive integrity protection (given that the smallest change in the ciphertext will render the plaintext unusable), encryption cannot protect against network intrusions. Therefore, various algorithms have been proposed to identify network intrusions, where most of them rely on machine learning to identify the attack signatures or the normal pattern of the network.

Some applications of machine learning based Intrusion Detection Systems (IDS) for SCADA systems in the literature include Yang et al. [6] which proposed an IDS to detect man in the middle (MiTM) and denial of service (DoS) attacks. Zhang et al. [7] used a data mining approach to identify malicious data and possible cyber-attacks in communication traffic from different levels of networks. Researchers in [8] proposed anomaly detection techniques which extract behaviors from various communication protocols to create a full description of the communication pattern in an industrial control system. These approaches work with data strictly from the network and are applying either signature-based or anomaly-based IDS.

While a lot of work has been done to develop the accuracy and sophistication of intrusion prevention systems, the sheer size of data in SCADA systems is the main bottleneck for the performance of machine learning algorithms used in IPS. As SCADA is a time critical application, IPS on industrial control systems must be able to process large amounts of data quickly. Secondly, the network IPS, which is the last line of defense in the industrial network, can also be deactivated from a compromised node in case of network penetration. Hence the proposed work targets these issues and comes up with a novel architecture of an embedded IPS for OpenPLC. The embedded architecture improves performance, as each PLC analyzes only its own data, and also minimizes the exposure of the IPS to the network.

III. AES-256 Encryption Module

The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information, based on the Rijndael encryption algorithm created by Joan Daemen and Vincent Rijmen. The successful use of AES by the U.S. government led to widespread use of the algorithm in the private sector, leading AES to become the most popular algorithm used in symmetric key cryptography. Currently, AES is widely implemented in software and hardware to encrypt sensitive data.

A. AES-256 Implementation in OpenPLC

The AES Encryption Layer stands between OpenPLC's Network Layer and the Intrusion Prevention System. The key scheme used is Pre-Shared Key (PSK). Therefore, the user must provide a keyphrase for the AES Encryption Layer every time OpenPLC is starting. OpenPLC expects all data received to be encrypted. Therefore, every packet received is decrypted by the AES Encryption Layer using the PSK, and then sent to the Network Layer for further processing. If a response is needed from the Network Layer, the reverse process occurs, as the AES Encryption Layer will encrypt the message using the PSK and then forward the ciphertext to the external network. The encryption process is protocol independent, and therefore works with both Modbus and DNP3 implementations in the original OpenPLC.

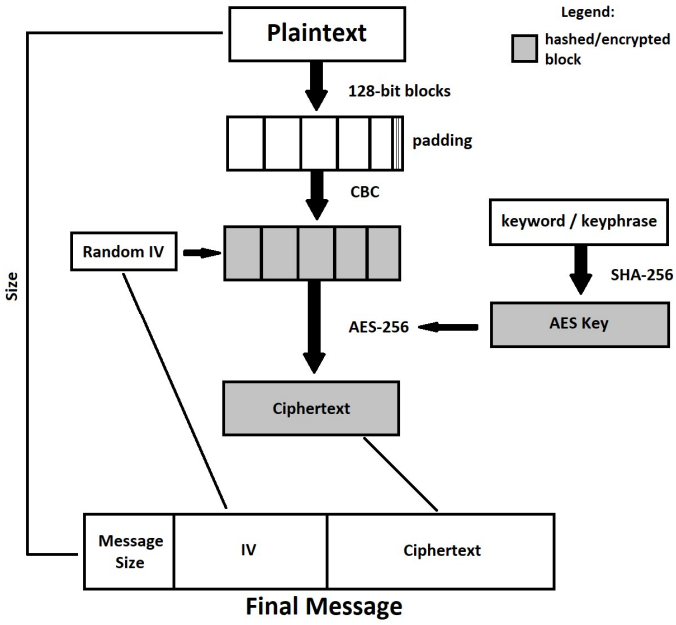

Fig. 1. OpenPLC Encryption Process

The encryption process requires five steps, as illustrated on Fig. 1. Each of the five steps are described above in more detail

- *Plaintext division*. Initially, the plaintext message is divided in chunks of 128 bits each. If the original

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2018.2823906, IEEE Embedded Systems Letters

message size is not multiple of 128, the last block is padded with random bits to form a full 128-bit block.

- *CBC Block Cipher Mode Operation*. The divided message is processed using CBC. A random Initialization Vector (IV) is provided as the starting pseudo-block.
- *Keyphrase Hash*. A keyphrase must be provided by the user for the encryption. Since AES-256 requires a fixed-sized key of 256 bits, the provided keyphrase is hashed using SHA-256 to generate a 256-bit hash that is used as a key for the AES-256 algorithm.
- *AES-256 Encryption*. The result of the CBC block cipher is processed by the AES-256 algorithm using the key generated in the last step.
- *Message Assembly*. The final message is assembled using the size of the original message, the random Initialization Vector, and the final ciphertext.

The decryption process is analogous to the encryption process illustrated before but executed in reverse order.

B. OpenPLC Secure Gateway

Although the AES-256 encryption process can secure messages from OpenPLC, the supervisory software cannot benefit from it if they cannot decrypt the message sent by OpenPLC. For this reason, a localhost gateway application was developed to provide a decryption mechanism inside the machine running the supervisory software. The OpenPLC Secure Gateway uses the same AES Encryption Layer implemented in OpenPLC. Once started, it prompts the user for the destination PLC IP Address, the keyphrase to be used for the encryption, and then starts a server listening for Modbus or DNP3 clients. Legacy SCADA software that works with the original OpenPLC should work with the secure OpenPLC, given that the messages go through the gateway.

IV. Embedded Intrusion Prevention System

The intrusion prevention system interfaces all packets received from the external network. It boasts off an implementation of an embedded unsupervised clustering algorithm to classify the incoming streaming data real-time to detect network anomaly and DoS attacks. If an attack is detected the IPS creates its own custom rules to block the attacker's IP from the network. The unsupervised clustering algorithm (K-means) makes the IPS generic and makes it adjustable to any changing network condition that a normal network might experience.

The trusted nodes connect to the IPS using the secure gateway application. The secure gateway application encrypts the messages received using AES 256 and only the TCP header, packet latency and packet processing information are available to the IPS. After collecting the required data while training or after inspecting the packet while monitoring, the IPS forwards the packet to the PLC. Fig 2. shows the network layout of the IPS and the trusted nodes connected through the secure gateway.

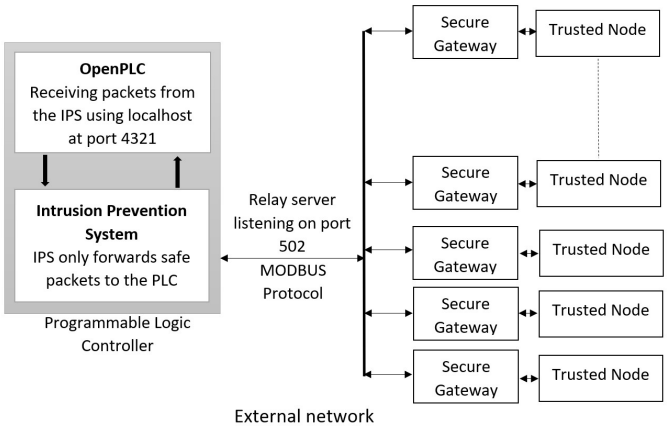

Fig. 2. External Network layout of the IPS

A. Data collection and preprocessing

The IPS incorporates a TCP proxy server which acts as an intermediary client while interacting with the PLC and acts as a server while receiving command from the trusted nodes. Hence, this module gives IPS the capability to receive and forward the incoming packets to the PLC after filtering out the anomalous traffic. This module allows the IPS to collect any data associated with the external network.

Two features were used for the training of the machine learning algorithm: packet interarrival time and packet processing time. The packet interarrival time is calculated by collecting the associated timestamp of any network packet being received by the relay server while a record is maintained about the client sending the packet. Once a second packet is received from same client, the time stamp of the second packet is subtracted from the stored time stamp to find the network latency associated with the concerned client. The packet interarrival time attribute is calculated by the data collection module right away once any network packet is received by the relay server module.

The processing time of a packet is the time duration spent by the PLC to process a network packet, perform the required operations associated with the network packet and to respond to the relay server with a valid response. To calculate this attribute, once a network packet is received a counter is started by the relay server module while the packet is forwarded to the PLC. Once the response for the packet is received by the PLC, the value of the counter is stored as the value of the attribute. Hence the packet interarrival time and the packet processing time are matched for a distinct network packet and are stored as a tuple of data in the data set being produced. The data set is sent to the data preprocessor every 200 tuples.

Deterministic clustering algorithms like bisecting K-Means are very sensitive to noise as the value of the cluster centers are distorted in the presence of outliers. Therefore, before training the algorithm, the outliers are removed from the training dataset.

Local outlier factor (LOF) algorithm computes a score reflecting the degree of abnormality of the observations. The local density of the data point with respect to its neighborhood is computed for each point. The points having lower density are considered outliers in the given data set and the tuples are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2018.2823906, IEEE Embedded Systems Letters

removed. This preprocessed data is forwarded to the machine learning module to train the IPS.

B. K-Means Clustering of Network Latency

K-means clustering is an unsupervised algorithm to partition n number of samples into K number of clusters in which each observation belongs to cluster with the nearest mean. If the set of clusters are represented as $\{S_1,S_2,S_3,\ldots.S_k\}$. the objective function can be written as

$$\arg\min_{s} \sum_{i=1}^{k} \sum_{x \in S_i} \left\| x - \mu_i \right\|^2 = \arg\min_{s} \sum_{i=1}^{k} |S_i| Var\, S_i$$

Where μ_i is the mean of the points in S_i

Once the PLC is commissioned into a new SCADA system the network latency and the processing time of data from different clients are collected real-time by the data collector module of the IPS. After the completion of the data collection, the collected data is clustered into n+1 different partitions using the K-Means algorithm, where n is the number of clients.

The attribute selection process creates distinct clusters for each client and hence the data precisely captures the network latency and query times of each of the clients connected to the PLC. The LOF algorithm ensures lower variance in each cluster thereby reducing the intra-cluster spread. This helps the unsupervised algorithm to eliminate the false positive scenarios and enables it to make accurate decisions. Secondly, to handle the outliers caused due to network congestion or network packet collisions, a thread monitors the number of anomalous packet and keeps a count of the packet for a 5 sec window. An attack scenario is affirmed once ten anomalous packets are detected in a period of 5 secs. If the number of anomalous packet is less than ten then the count of the anomalous packet is reset to zero. This method prevents outliers from triggering a false positive attack detection.

During a DoS attack, the huge volume of packets projected to the target computer creates new clusters and hence the DoS attack can be detected. Once the attack is detected the IPS uses iptables rules to block the attacker out of the PLC. Once blocked, the attacker cannot reach the PLC anymore from that IP. The information about the attack is written in the PLC attack log and the socket is restarted.

V. Experimental Results

The UAH SCADA Lab includes 3 laboratory scale industrial control systems: a gas pipeline, a water treatment facility, and a water storage tank. Each system is a working physical model with commercial actuators and sensors. The actuators and sensors for each system are connected to a Raspberry Pi model 3 attached to a UniPi industrial board running OpenPLC. Each PLC is programmed with ladder logic for distributed control. All 3 PLCs are networked through a set of Siemens switches to a HMI. The HMI is implemented using the ScadaBR software, which is provided with OpenPLC. The attacks for this work were performed on the water storage tank.

A. Interception Attack

When packets travel across a network, they are susceptible to being read or altered. During an interception attack, the attacker monitors data streams to or from a target, in order to gather sensitive information. With a simple packet sniffer, the attacker can read all plaintext traffic that is travelling across the network.

On the water storage tank SCADA system, the information about current mode, defined setpoints and manual triggers for the pump and valve are stored in PLC registers accessed by Modbus or DNP3. Due to the unsecured nature of both protocols, this information can be accessed and modified by anyone in the network.

After starting the modified Secure OpenPLC on the same UniPi device, the interception attack was performed once again. Although it was possible to intercept the packets on the network, it was not possible to decypher its contents due to the AES-256 encryption.

B. Injection Attack

Injection is an entire class of attacks that rely on injecting data into an application and possibly causing harm by altering the course of execution for the application. Given the open nature of the messages from the Modbus and DNP3 protocols, it was possible to create an injection attack for the original OpenPLC on the water storage tank SCADA system. The injected messages were crafted to lock the system in manual mode, with the pump turned on and the escape valve turned off. This is the worst-case scenario, since the tank will be constantly receiving water where no water can flow back to the reservoir. Although the HMI was still able to monitor the system, it was not possible to revert the system back to auto mode, or turn off the pump, as messages were being injected faster than the authentic messages from the HMI. It was also observed that by increasing the rate of the injected messages, it was possible to cause a Denial of Service (DoS) on the HMI, which eventually stopped receiving updated information from the PLC.

The same attack was performed with the modified Secure OpenPLC in place. Due to the fast pace of the injected messages, the embedded IPS detected an anomalous traffic and was able to quickly block the attacker node without causing any harm to the running system. Therefore, it was not possible to DoS the system using injected messages. After slowing down the frequency for the injected messages, it was possible to go undetected through the IPS. Although the injected messages were able to reach the PLC, it was still not possible to inject the commands due to the AES-256 encryption that was in place. Therefore, the messages were discarded, and no harm was caused to the system.

C. Denial of Service (DoS) Attack

A DoS attack is a cyber-attack where the perpetrator seeks to make a machine or network resource unavailable to its intended users. DoS is typically accomplished by flooding the target machine with superfluous requests in an attempt to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LES.2018.2823906, IEEE Embedded Systems Letters

overload the system and prevent legitimate requests from being fulfilled.

The DoS attack performed for this work uses the Low Orbit Ion Cannon (LOIC) software to flood the PLC with TCP packets on port 502, which is the same port used for Modbus communications. The attack was able to almost immediately cause a denial of service to the HMI, since the OpenPLC running on the UniPi got overwhelmed and could not respond to the HMI requests anymore. Although the HMI got unresponsive, the system continued functioning, running the ladder logic program. However, after about 3 minutes of continuous attack, OpenPLC's network stack crashed and the OS killed the application, stopping the ladder logic program execution.

The same attack using LOIC was performed against the Secure OpenPLC. Due to the fast nature of the flood packets, the embedded IPS running alongside OpenPLC quickly detected abnormal traffic and banned the attacker node. The attacker node was only able to attack the device for a few milliseconds before being banned. Although it was a short period of time, around 5,000 packets were sent to OpenPLC, which could cause slowdowns while processing the next legitimate packages. In order to recover from this, right after banning the attacker, the IPS flushed the network interface and restarted the socket to listen for new connections. The HMI was able to reestablish communication in a few seconds and no harm was caused to the OpenPLC application running the ladder logic program.

VI. Benchmark

As demonstrated in the experimental findings from previous section, the techniques implemented in the open source OpenPLC project indicate that they can thwart man-in-the-middle attacks, packet injection attacks, and Denial-of-Service attacks. However, given the strict real-time requirements of many industrial processes, if the encryption and machine learning IPS layers add significant delays, it might not be possible to deploy these techniques in the real world. Therefore, this section evaluates the performance penalty for applying the encryption and machine learning IPS through a benchmark test.

The test was performed on the original OpenPLC, OpenPLC Secure (which is OpenPLC enhanced with the encryption and machine learning IPS layers), and two other popular commercial PLCs: Siemens S7-1214C, and Allen-Bradley MicroLogix 1400. Both OpenPLC and OpenPLC secure ran on a Raspberry Pi model 3 hardware. All four PLCs were loaded with the water storage tank ladder logic program.

The real-time behavior of a PLC is usually associated with its scan time, which is the time the PLC takes to read all inputs, execute the logic program and write all outputs back. Therefore, for the proposed benchmark test, the water storage tank ladder diagram was augmented with an extra ladder logic line with one contact and one coil, as shown in Fig. 3.

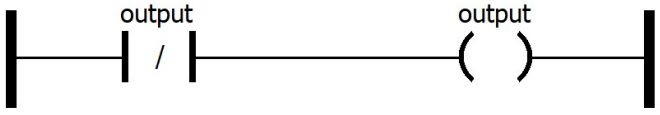

Fig. 3. Ladder Logic for the Benchmark Test

Since both contact and coil are associated with the same tag, this ladder logic line toggles the output tag on every scan cycle. Therefore, by connecting an oscilloscope to the port associated with the output tag and measuring its frequency, it is possible to accurately determine the PLC's scan cycle.

The benchmark test consisted in having the water tank program running on each PLC with the extra ladder logic line from Fig. 3. Each PLC was connected to a water tank HMI provided by ScadaBR, that constantly queries the PLCs for data every 100ms over Modbus/TCP. A microcontroller was used to measure the output toggle time, collecting 4000 samples from each PLC. In order to avoid errors introduced by delays on MOSFETs, transistors and relays, the microcontroller was wired directly to the processor (or controller) pin of each PLC. Fig. 4 shows the histogram curves for each PLC during the test. The standard deviation and average metrics were calculated for the 4000 samples and are displayed on Table I.

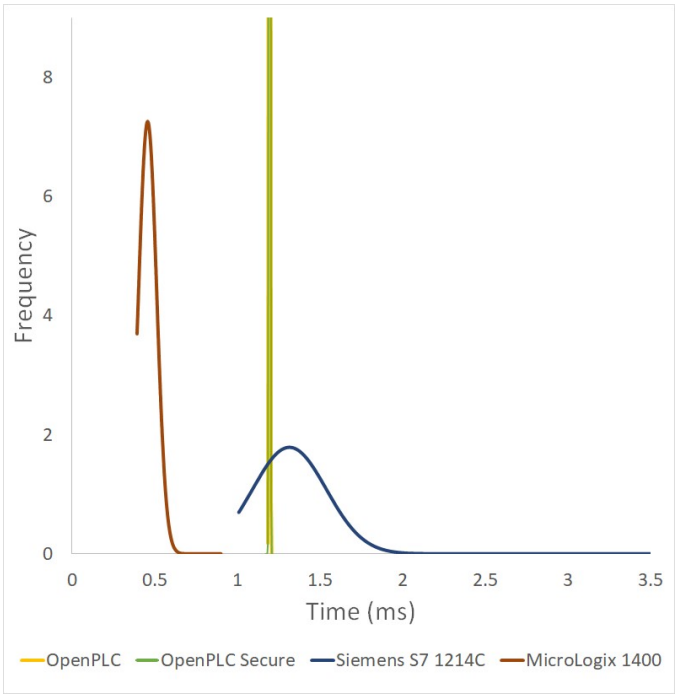

Fig. 4. Histogram for the Benchmark Test

The results indicate that both OpenPLC versions (original and secure) had the same scan time, which was also similar to the scan time of the other two PLCs analyzed. Additionally, the standard deviation for both OpenPLC versions was much smaller than the standard deviation for the other two commercial PLCs, which means that OpenPLC had much less jitter on the

TABLE I
Scan Time Average and Std. Deviation

PLC	Average	Standard Deviation
OpenPLC	1.1960 ms	0.0029 ms
OpenPLC Secure	1.1960 ms	0.0029 ms
Siemens S7-1214C	1.3159 ms	0.2230 ms
A-B MicroLogix 1400	0.4579 ms	0.0549 ms

scan time and a much better real-time response, even when the encryption and machine learning IPS layers were added.

The explanation for those results lies in OpenPLC's real-time library, that provides a better real-time response for the OpenPLC process and can really benefit from a multi-core architecture such as the one found in the Raspberry Pi model 3. Real-time in Linux is greatly affected by Linux's scheduling policy. When a running process is scheduled out, it is hard to know precisely when it will be scheduled back. This lack of determinism affects processes that require a real-time response. OpenPLC's real-time library improves Linux response on a multicore architecture by isolating OpenPLC process on a separated core, and by changing the scheduling policy for the OpenPLC main thread to SCHED_FIFO and setting its priority to the maximum value. Processes scheduled under the SCHED_FIFO policy with maximum priority will always immediately preempt any currently running process in the core. Additionally, processes under this policy do not have time slicing, which means that a SCHED_FIFO thread runs indefinitely until it yields the processor or is blocked by an I/O request. Since the OpenPLC process is the only process constantly running in the isolated core, it can provide an improved deterministic response, even when the system is under heavy load. By analyzing the load while OpenPLC was running, it could be seen that the encryption tasks and the IPS were scheduled on different cores than the one assigned to run the main OpenPLC task.

VII. Conclusion

The combination of the embedded encryption module and the intrusion prevention system makes the secure version of OpenPLC immune to a wide variety of attacks. Attacks like interception, injection and denial of service, which are the major threats in industrial control systems, were rendered ineffective by the implemented embedded security measure.

Additionally, as demonstrated in the experimental findings from previous sections, the techniques implemented in the open source OpenPLC project indicate that they not only can thwart man-in-the-middle attacks, packet injection attacks, and Denial-of-Service attacks, but also that the protection mechanisms did not interfere with the real-time characteristics of OpenPLC.

VIII. Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1623657. The travel to the Embedded Security Challenge finals was sponsored by the Office of Naval Research, under grant no. N00014-17-1-2515.

References

[1] Falliere, N., Murch, L., and Chien, E. W32/stixmet dossier. White paper, Symantec Corp., Security Response 5

[2] Alves, T. The OpenPLC Project. http://www.openplcproject.com. Accessed: 2017-11-06

[3] International Electrotechnical Commission. "IEC 61131-3: Programmable Controllers – Part 3 Programming languages". International Electrotechnical Commission, Geneva, Switzerland, 1993

[4] Fovino, I., Carcano, A., Masera, M. and Trombetta, A. 2009. Design and Implementation of a Secure Modbus Protocol. IFIP Advances in Information and Communication Technology. (2009), 83-96.

[5] Majdalawieh, M., Parisi-Presicce, F. and Wijesekera, D. DNPSec: Distributed Network Protocol Version 3 (DNP3) Security Framework. Advances in Computer, Information, and Systems Sciences, and Engineering. 227-234.

[6] Y. Yang, K. McLaughlin, S. Sezer, T. Littler, B. Pranggono, P. Brogan, and H. Wang, "Intrusion detection system for network security in synchrophasor systems," in Information and Communications Technologies (IETICT 2013), IET International Conference on, April 2013, pp. 246–252.

[7] Y. Zhang, L. Wang, W. Sun, R. Green, and M. Alam, "Distributed intrusion detection system in a multi-layer network architecture of smart grids," Smart Grid, IEEE Transactions on, vol. 2, no. 4, pp. 796–808, Dec 2011.

[8] H. Hadeli, R. Schierholz, M. Braendle, and C. Tuduce, "Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration," in Emerging Technologies Factory Automation, 2009. ETFA 2009. IEEE Conference on, Sept 2009, pp. 1–8.