Industrial Control System Network Intrusion Detection by Telemetry Analysis

Stanislav Ponomarev and Travis Atkison

Abstract—Until recently, industrial control systems (ICSs) used "air-gap" security measures, where every node of the ICS network was isolated from other networks, including the Internet, by a physical disconnect. Attaching ICS networks to the Internet benefits companies and engineers who use them. However, as these systems were designed for use in the air-gapped security environment, protocols used by ICSs contain little to no security features and are vulnerable to various attacks. This paper proposes an approach to detect the intrusions into network attached ICSs by measuring and verifying data that is transmitted through the network but is not inherently the data used by the transmission protocol—network telemetry. Using simulated PLC units, the developed IDS was able to achieve 94.3 percent accuracy when differentiating between machines of an attacker and engineer on the same network, and 99.5 percent accuracy when differentiating between attacker and engineer on the Internet.

Index Terms—Networked control systems, nonlinear network analysis, control systems, intrusion detection, telemetry

1 Introduction

Industrial control systems (ICSs) are designed, implemented, and deployed in most major spheres of production, business, and entertainment. From orchestrating complex maneuvers of the International Space Station to controlling the speed of a roller coaster, ICSs are able to process complex data and safely perform designed tasks.

Safety is a number one priority when dealing with physical devices. ICSs are commonly split into two subsystems—programmable logic controllers (PLCs) and supervisory control and data acquisition (SCADA) systems—to achieve high safety, allow engineers to observe states of an ICS, and perform various configuration updates. PLCs are small processing systems which are able to modify the behavior of the controlled devices and receive input from the system's sensors. SCADA systems allow engineers to monitor the ICS state and modify its parameters as needed.

Before wide adoption of the Internet, ICSs used "air-gap" security measures, where every node of an ICS network was isolated from other networks, including the Internet, by a physical disconnect [1]. To perform an attack on an isolated ICS, an attacker would have to gain physical access to the ICS' network, and either penetrate and use the hardware of the ICS to transmit malicious commands, or attach a new node to the ICS network. This level of security allowed ICS protocol designers to concentrate on availability and safety of operation of physical systems while decreasing the need for many cyber security implementations.

- *S. Ponomarev is with the College of Engineering and Science, Louisiana Tech University, Ruston, LA 71272. E-mail: spo013@latech.edu.*
- *T. Atkison is with the Cyber Engineering and Computer Science Department, Louisiana Tech University, Ruston, LA 71272. E-mail: atkison@latech.edu.*

Manuscript received 1 Oct. 2014; revised 3 Apr. 2015; accepted 8 May 2015. Date of publication 10 June 2015; date of current version 16 Mar. 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TDSC.2015.2443793

As the price of networking devices fell, and the Internet received global adoption, the benefits of attaching ICSs to wide and global area networks became evident. Engineers gained an ability to monitor and fix critical problems remotely, eliminating travel times to the ICS, which gave them more time to work on problem solving when the system malfunctioned. The price of implementing geographically dispersed ICSs, where PLCs and SCADA systems may be miles away from each other, decreased with the spread of the Internet. The critical infrastructure Smart Power Grid is one such ICS where each house has its own controller—a smart meter—that transmits usage information to the power company and may turn off the house's power for maintenance or lack of payment [1].

Attaching ICS networks to the Internet benefited companies and engineers who used them. However, as these systems were designed for use in the air-gapped security environment, protocols used by ICSs contain little to no security features and are vulnerable to various attacks. In addition, there exists a multitude of different ICS protocols. While some standard ICS protocols were created, different manufacturers choose to use different protocols, which make it difficult to assess possible vulnerabilities for all ICSs [2].

Network models that are currently implemented allow for easy access to various networked systems through a multitude of mediums, devices, and implementations. Cell phones can communicate with high-end servers or even supercomputers over WiFi, desktops can use cell phone's cellular networks, smart watches can communicate with the same servers by using Bluetooth and a cell phone's connection. Network models create abstraction layers and keep some information such as properties of the sender's connection, sender's architecture and configuration, as well as sender's location, away from applications. Lack of such information reduces the ability to establish authenticity of the host transmitting the data [3]. Additionally, an attacker can perform data manipulations on the network to achieve

penetration of target machines and spoof the authenticity of the transmitted packets [4].

One of the approaches to securing the communication of network attached ICSs is a network telemetry-based intrusion detection system (IDS). Such a system operates by measuring and verifying data that is transmitted through the network but is not inherently the data used by the transmission protocol. Network telemetry may include temporal data of packet arrival, packet sizes, session times and sizes, amount of dropped packets and more. To achieve this, telemetry-based IDSs need to monitor all packets traversing the ICS network, have a running average of the telemetry data, and be able to alert system engineers when anomalies in traffic are detected.

2 Background

ICSs have been the target of many cyber-attacks [2]. Unlike attacks on information technology, attacks on ICSs may not only destroy a company's rating and cause monetary loss, but may also damage equipment, the environment, or harm workers. When engineers design the logic flow of ICS controllers, their primary goal is to maintain system stability and safety. However, as ICS controllers are attached to wide and global area networks, they become vulnerable to a multitude of attacks.

Stuxnet was one of the largest and most complicated attacks deployed that targeted an industrial control system [5]. Stuxnet was a worm, a malicious application that is capable of replicating and spreading itself over a network, discovered in June 2010. However, unlike most worms, Stuxnet was developed to target industrial control networks. To hide the worm from the cyber-security community, the attackers prevented it from inflicting any damage in networks that did not contain specific properties of an ICS network. Once inside a vulnerable computer with no ICS software, the Stuxnet worm could propagate using USB drives or a network connection. Stuxnet could duplicate itself only three times when propagating over a USB drive, but there was no limit on network propagation [5], [6], [7]. However, when ICS properties were discovered, Stuxnet would present itself to PLCs as a SCADA system and perform man-in-the-middle attacks, reprogramming PLCs to perform different actions, thereby bringing the system to a critical state. Stuxnet penetrated the SCADA system of Iranian nuclear facilities and caused centrifuge motors to change their spinning frequency, which destroyed centrifuges. [5].

Stuxnet was a wakeup call to many ICS engineers. Securing industrial control systems from cyber attacks became a top priority. ICSs could not afford for any attacks to be successful due to physical consequences of such an attack. As a result, security of cyber-physical systems is now a hot topic world-wide, and various intrusion detection, attack mitigation, and attack obfuscation techniques are being researched.

Unique intrusion detection implementations have been suggested. In [4] a hardware fingerprinting method is proposed to establish data authenticity. Unlike business and residential networks, ICS networks maintain a steady packet flow, which allows the system to fingerprint different network nodes, based on the patterns of their communication. Wallace et al. [8] suggests designing an IDS custom-tailored to a specific ICS, where underlying physical properties of the objects being controlled manifest themselves as features of the ICS that can be used to distinguish between the states of ICS normal operation and attacks on the given ICS. Carcano et al. [9] propose a state-based intrusion detection system which listens for the ICS network traffic, maintains an ICS image based on that traffic, and monitors for a set of state anomalies. In [10], a model-based IDS is developed, where the communication model using Modbus protocol is analyzed, and an alarm is raised when signatures of packet fields are detected.

Long et al. [11] proposed methods of mitigating denial of service attacks that originate from either local or wide area networks by modelling stochastic process of packet delay jitter and loss. PLC shadowing and data duplication can be used to mitigate the attacks on industrial control systems [12]. Attack mitigation techniques include the use of firewalls, recommended by the National Institute of Standards and Technology [13], and some protocol modifications to prevent man-in-the-middle attacks [14].

Sayegh et al. [15] proposed an IDS which uses temporal packet data to identify traffic anomalies. Different packet signatures were generated for a given protocol, and probability functions were used to identify if a given packet was expected to arrive to the SCADA system. The paper targets BACnet protocol but mentions that new protocols can be supported without changing the core functions of the IDS. This approach, however, can produce many false positives when the anomaly happens in the physical domain of the ICS (for example: damaged plant, broken wire, low pressure, etc.), and the engineers try to reprogram PLCs to mitigate the problem.

In order to evaluate an intrusion detection system, a list of possible types of intrusions and attacks must be populated. While geographically concentrated ICSs are vulnerable to many attacks, geographically dispersed ICSs commonly utilize wide area networks or wireless communication to transmit valuable information, which increases the amount of attack vectors in comparison to concentrated ICSs. The following sections will overview attacks for both of these ICS types.

The attack vectors for attacking PLCs and SCADA systems alike include man-in-the-middle, DoS, spoofing, packet injection, and reconnaissance attacks. However, depending on the direction of packet flow, these attacks may affect PLCs, SCADA, or both.

2.1 SCADA-Side Attacks

SCADA attacks target the packets transmitted to the SCADA system. SCADA systems are commonly run on PCs [12], which are vulnerable to any attacks on their operating system. An attacker may acquire protocol-specific SCADA identifiers to mask her/himself as an engineer to inject PLC values into the network that do not reflect current network state, can stall the SCADA machine, and prevent the SCADA engineer from knowing the state of the ICS [16].

2.2 PLC-Side Attacks

PLC attacks target the packets being sent to the controller. Once communication with the controller is established

without a proper IDS, an attacker has the ability to shut down the CPU, disable memory protection and perform code injection. This allows an attacker to modify the code that the PLC is running, get the state of a system the PLC is monitoring, find what kinds of data the PLC sends to SCADA, overwrite values the PLC is reporting to SCADA, and overwrite some or all logic that is used to control the ICS. While some of these attacks may result in information leakage, others can damage the physical system being controlled or misrepresent the system state to the monitoring engineer [17].

Network Telemetry based IDSs can be used to protect PLCs from unauthorized access. For this particular research, a CPU shutdown attack was performed. This attack transmits a single packet to the PLC that results in its CPU shutdown (denial-of-service attack) until the PCL can be physically reset. However, instead of banning packets that result in CPU shutdown, the IDS was able to differentiate between CPU shutdown commands from the SCADA system and CPU shutdown commands from an attacking system. This attack was chosen to demonstrate the ability to differentiate even between single-packet attack patterns.

2.3 Network Telemetry

The Internet Protocol (IP) commonly utilizes Ethernet frames to forward packets between multiple nodes of the network until they reach their final destination. To achieve this, Ethernet frame headers contain the source and destination media access control (MAC) addresses. While IP addresses are being assigned to different machine interfaces by network administrators or self-configuration protocols, MAC addresses are uniquely assigned to each network interface during the manufacturing process of the interface's controller circuit [18].

Both IP and MAC addresses can be spoofed by software, preventing the server from ascertaining whether the packets are arriving from an authorized location [19], [20]. Spoofing is one of the reasons security-critical algorithms implement encryption to prevent illegal access and enforce data integrity [21]. Encryption, while effective in deterring most attackers, can be broken, and no matter how secure a cryptographic function can be, it can be attacked with at least a brute force attack [22]. Hashes can be reversed like in Stuxnet [7], users can be tricked into giving their passwords to a phishing site, and weak passwords can be guessed [23]. Network telemetry data can be used to detect a network intrusion when an attacker is using a different machine or even a different software of the authenticated machine.

There have been various studies in anomaly detection which use data mining and machine learning facilities to detect anomalies [24]. NetMine is a data mining application that specializes in understanding traffic data correlations and interactions [25]. While implementing methodologies similar to the ones used in this research effort, these studies focused on feature generation to improve traffic quality and network stability rather than network security.

In their works, Erman et al. [26] were able to successfully cluster similar packet types by analyzing transport layer statistics. By using K-Means and DBSCAN algorithms, they were able to successfully identify protocols being used without extracting the data from packets [26]. Sheng et al. showed that it is possible to detect host spoofing by analyzing statistical fluctuations in received signal strength of the packets transmitted over wireless networks [20].

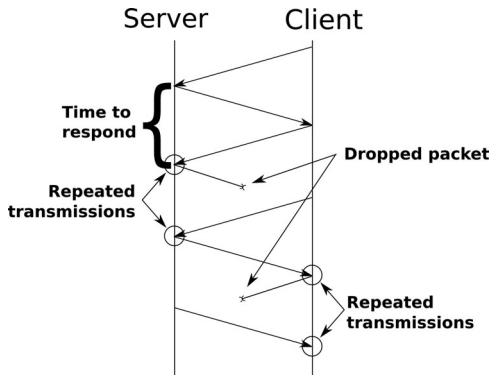

Fig. 1. Client-server session graph.

Wireshark is a software network analyzer which captures network traffic and displays it in real time. It also allows system administrators to save all of the received packets for future analysis and extract useful information about these packets. Wireshark was used in this research effort to extract packet arrival times into a comma separated values format used to generate graphs and interpret the data [27].

Though the detection approach presented here may not withstand the dynamics of a typical enterprise local area network (LAN), the approach will be beneficial in the detection of spoofed hosts in control system LANs. Control system LANs are unique in that hosts generally communicate in set intervals set by the polling protocol utilized [28]. The detection scheme developed for this research effort is able to determine when communication is initiated and maintained outside of these intervals and, upon detection, will alert on a possible intrusion. Furthermore, with the use of open source tools that automate the attack process against control systems [29], this work proves to be fruitful as it can distinguish between malicious and benign packets.

3 Telemetry Based Intrusion Detection System

The developed IDS utilizes aggregation and analysis of the traffic generated by Conpot—a contol system honeypot project aimed to simulate an industrial control system network to attract possible intruders [30]. The SCADA part of the ICS is implemented in python using Modbus protocol stack—pymodbus. Conpot was set to simulate a control system network that has two Siemens SIMATIC S7-200 PLCs.

A telemetry based IDS utilizes the understanding of session flow in a networked server-client model (Fig. 1). Experimental scenarios include a benign SCADA system and the malicious attacker's machine (insider threat) located 1 hop away from the PLCs, and another 8 hops away. The 8 hop network represents a real-world scenario of system engineer modifying the ICS from a remote location (for example, to troubleshoot time critical errors) and an outsider attacking the ICS. The origin of control system traffic may change the amount of hops between the nodes, and increased hop length can introduce packet delays and dropped packets. Possible attacks can originate from within the control system LAN, within the corporate LAN, or within the Internet.

The goal of Network Telemetry based IDSs is to protect PLCs and the underlying physical plant from malicious activity—unauthorized access and control of PLC hardware. Therefore, this IDS is implemented as a standalone device that monitors traffic between the PLCs and the rest of the network. Once the anomaly is detected, system engineers must be informed about an intrusion.

3.1 Classification Algorithms

In order to achieve the highest accuracy, many algorithms were tested. Some qualities of the data set such as noise and size were taken into account when choosing classifiers. Aside from individual classifiers, boosting was used as a supervisory algorithm to reduce the bias of other classifiers. Boosting is an ensemble learning algorithm that aims to create a strong classifier out of many weak ones. Bayesian classifiers' performance is linearly scaled with the data set and can be used in a time critical code. Bagging is also known as bootstrap aggregating and is an ensemble machine learning algorithm that can improve accuracy of its base algorithm. For this research, REPTree algorithm was used as a base machine learning algorithm. REPTree uses information variance to build a decision tree, and then prunes it using reduced-error pruning. A set of bagging-aided classifiers was chosen because they perform well with training sets containing large noise [31]. Several decision tree based classifiers were also used due to a flow-like dependence of the outcome of the classification on the features. The following paragraphs describe the algorithms used in this research with details on their characteristics and implementation.

Naïve Bayes classifier assumes that all the features are independent of each other, and it follows a Bayesian probabilistic model:

$$p(C|F_0, F_1, \ldots, F_n) = \frac{p(C) \cdot p(F_0, F_1, \ldots, F_n|C)}{p(F_0, F_1, \ldots, F_n)},$$

where C is the class of dataset, $F_0, \ldots, F_n$ is a set of features, and $p()$ is a probability function [32].

The multinomial model of the Bayesian classification, however, is dependent on the frequency of reappearing features in the data-set. This allows for a higher classification accuracy of data with some repeating patterns [33].

Simple Logistic classifier utilizes a binary logistic regression to describe an outcome in only two possible classes. It takes a set of features and applies regression analysis to create classification parameters [34].

Ripple-Down Rule learner generates a default rule and creates an exception list using weighted error rates. Then exceptions are rated and filtered to remove conditions that fit multiple exceptions [35].

A *Decision Stump* generates a single-feature condition that results in a binary classification. This results in a high speed classification but lacks accuracy in a high-dimensional orthogonal data [36].

J48 is an implementation of *C4.5* machine learning algorithm. This algorithm generates a decision tree based on the information gain ratio if the classification was split on a given tree node [37].

3.2 Telemetry Data

A list of features was generated by analyzing a total of 838,818 packets generated by the SCADA communication with the honeypot over a period of two days. The gathered data is the result of capturing the traffic from designated benign and malicious machines to the PLCs emulated by Conpot using libpcap—a library created for capturing live networking data. Both malicious and benign machines ran the same code to achieve similar patterns of packet transmissions, which represent situations when both an attacker and an engineer want to execute the same code but for a different purpose—for example, if an engineer wants to launch the normal PLC operation after a maintenance, and an attacker wanting to launch normal PLC operation during a maintenance. The transmission of similar functions is done to test the accuracy of the IDS for similar packet patterns. Different functions can have unique transmission patterns which are much easier to detect. The packet class was identified by packets' source IP address, however during the classification process, IP addresses were not selected as features. The following features were selected:

- time it takes the client to respond to server's message.
- amount of client-side dropped packets.
- amount of server-side dropped packets.
- time between the repeated packet transmissions when packet drops happen.

Fig. 1 shows the features listed above in a communication session graph. After the data was captured, the features described above were extracted from the recorded pcap file, and converted to an ARFF format that can be used with a machine learning environment.

A TCP latency model described in [38] was used to determine a set of telemetry features for the IDS. While Cardwell et al. published their model 15 years ago, it is still appropriate for modelling short-lived TCP connections [39]. The features were selected based on their variability from different machines, as well as the level of difficulty for the attackers to modify those features. The time to respond feature includes round trip time for the communication path as well as processing delays introduced by the client's machine. While an attacker can introduce delays to his code, speeding up packet delivery times requires relocation and/or an upgrade of the attacking machine. The amount of dropped packets on either side of the nodes relates to network congestion and the delays between repetition provide information about delays introduced by network schedulers. The attacker can change these values if she/he have control of all of the nodes between the malicious and benign machines. Otherwise, her/his option would be to relocate the malicious machine to a network path that matches a benign path in its congestion. Fig. 1 shows the relationship of these features.

The selected features are protocol independent and are based on the data that an attacker has little to no control over. For example, an attacker may be able to introduce delays into the packet flow of their machine, but the amount of dropped packets is largely dependent on the state of the network between client and server. Introduced delays would also have to be adjusted to match the current value of the IDS, which would require attacking the IDS first and acquiring

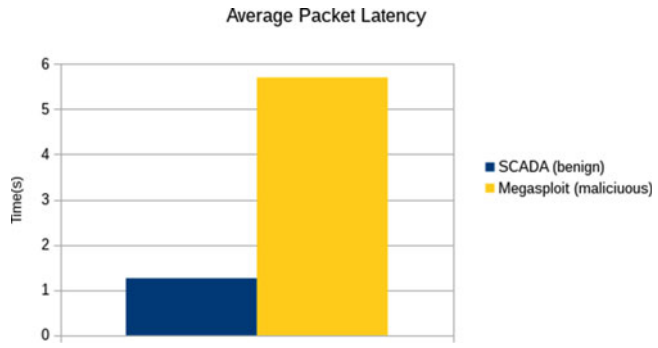

Fig. 2. Time delays introduced by spoofing.

the data. In addition, if the attacker's system is not fast enough to match the delays of the benign software, the only way to speed it up would be to upgrade her/his system.

Due to the nature of the telemetry data, an identical hardware and software combination will present itself as identical to this IDS, which will not generate any anomalies. However, different attack software used to spoof MAC and IP addresses can introduce different telemetry signatures. Fig. 2 shows an average delay between packet arrival on two similar machines; one uses benign SCADA software to query the PLCs, and the other uses a metasploit plugin to spoof mac and IP address polling the PLCs afterwards. If the attacker uses the SCADA machines for an attack, a Network Telemetry based IDS is one of few IDSs that can detect an intrusion, as long as some packet transmitting software was changed, for example, an added backdoor.

3.3 Data Acquisition

Classification accuracy of a Network Telemetry based IDS highly depends on processing times introduced by computer hardware as well as the network communication paths between the benign machines and PLCs and malicious machines and PLCs. For these experiments, both benign and malicious machines were identical at a given hop-distance, and the following machines were used:

1-Hop Classification

- CPU: Intel(R) Core(TM)2 Duo CPU E7500 @ 2.93 GHz
- RAM: 2 GB
- NIC: Intel Corporation 82567LM-3 Gigabit Network Connection
- OS: Gentoo Linux, kernel-3.12.13-gentoo
- Python version: 2.7.1

8-Hop Classification

- CPU: Intel(R) Core(TM) i7-4770K CPU @ 3.50 GHz
- RAM: 16 GB
- NIC: Broadcom Corporation NetLink BCM57781 Gigabit Ethernet PCIe
- OS: Gentoo Linux, kernel-3.12.13-gentoo
- Python version: 2.7.1

For all of the experiments, the PLCs were emulated using conpot on a machine with the following configuration:

- CPU: Intel(R) Core(TM)2 Duo CPU E7500 @ 2.93 GHz
- RAM: 2 GB
- NIC: Intel Corporation 82567LM-3 Gigabit Network Connection
- OS: Gentoo Linux, kernel-3.12.13-gentoo
- Python version: 2.7.1

The IDS is designed to notify system engineers of malicious traffic. For this research malicious traffic is defined as any traffic directed to the PLCs that are not originated from the designated SCADA system. To achieve classification, the IDS must receive all of the network packets sent to the PLCs. The IDS utilizes libpcap [40] to capture all the traffic transmitted to the PCLs over the Modbus protocol. For this research, Conpot emulated Modbus/TCP protocol on port 502. The system maintains a floating delay separation boundary (FDSB) which separates traffic fingerprinting methods to outsider and insider classification engines. This is done to improve accuracy of classifying traffic originating from local to the ICS machines, versus the traffic originating from machines on the internet. For example, after analyzing Netgear WNDR4500, D-Link DI-604, and Linskys WRT300N routers, the delay feature range is under 5 milliseconds while the communication utilizes local area network, but it can be several orders of magnitude larger if the communication path connects distant nodes on the internet. However, not all of the traffic originating from a distant node can be malicious (as in the example of engineer using remote access to troubleshoot an ICS at the remote site). Separation must be established in order to compare local paths with local paths and distant paths with distant paths. The floating delay separation boundary is defined as

$$FDSB = \frac{\sum_{i=0}^{n} \frac{max_i(I)}{n} + \sum_{i=0}^{n} \frac{min_i(O)}{n}}{2},$$

where I is the set of delay values from the packets originating from the inside of the ICS, O is the set of delay values from the packets originating from the outside of the ICS, max_i and min_i are the ith maximum and minimum values in the given set, and n is the amount of samples to take for the delay value which is the same for both sums. Having a value of n too high will use older network states which may not take place any more. However, having a value of n too low will result in a lower accuracy of the threshold calculation. For this experiment $n = 5$ is used, while sets of O and I have 50 items. These values were determined by iteratively running the classification over 5 minute intervals for all the captured dataset and determining the combination that results in the best classification. The initial $FDSB$ value was set to 0.025. These values were determined by analyzing 838,818 packets acquired over two days of operation of an ICS. Fig. 3 shows minimum and maximum response times between two different machines on the same LAN (malicious and benign 1 hop), and the delays introduced by moving the same machine further along the communication path (malicious 1, 2, and 8 hops).

Once the packet is identified as outsider/insider and sent to a proper classifier, its arrival time is recorded and compared against the previous packet that was sent to the same classifier. Inside-originating packets are only compared with the telemetry of other inside-originating packets, and outside-originating packets are compared with the telemetry of

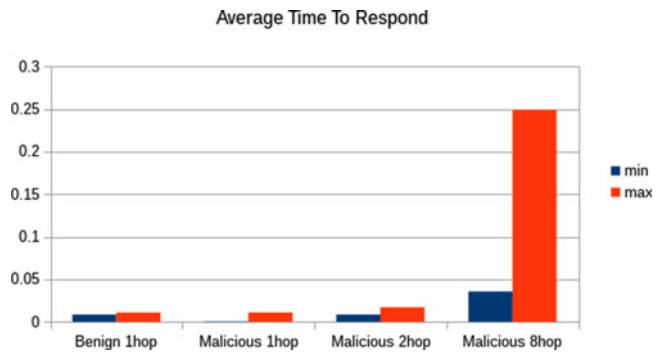

Fig. 3. Average response time in a session over ONE day.

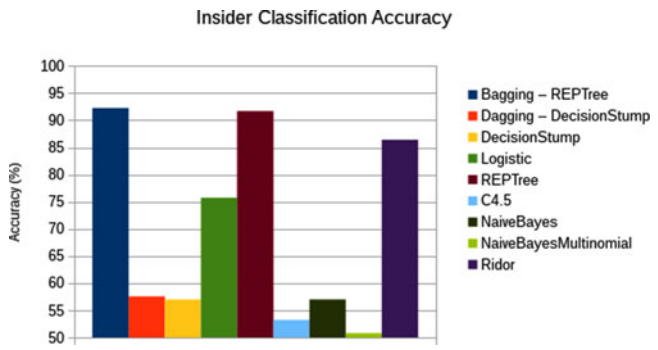

Fig. 4. Classifier accuracy for insiders.

other outside-originating packets which allows the IDS to mark some external traffic as benign as in the case of a system engineer working from home. Traffic to the PLCs is tagged as inside/outside traffic based on the subnet of the packet's IP address. Code running on the PLCs is assumed benign and IP address spoofing from the PLCs is not expected.

On every new packet arrival, telemetry data discussed in Section 3.2 is extracted and sent to the classifier. A new session is instantiated on every new packet. However, the calculations of features is done for all the packets received in the 2 second interval—the same interval that an experimental SCADA system uses for polling PLCs. For example, if there were nine packets received by the classifier in the past 2 seconds the first packet of session $n+8$ is the last packet of session n.

Several machine learning classifiers have been tested for this IDS as discussed earlier: Naïve Bayes, Multinomial Naïve Bayes, Logistics, REPTree, Bagged REPTree, DecisionStump, Degged DecisionStump, Ridor, and C4.5. Naïve Bayes classifiers were chosen to determine probability models for telemetry data. Bagging modelling of REPTrees was chosen because they perform well with training sets containing large amounts of noise [31].

3.4 Hop Relationship

Telemetry based measurements are highly dependent on the amount of hops between the client and server. As nodes are separated by a larger amount of hops, added network systems introduce different delays into packet propagation; this introduces noise. The IDS carries two classification profiles—for insider traffic and for outsider traffic. Fig. 3 shows that the delay increases with the amount of hops.

Differentiating between insider and outsider traffic is trivial as an outsider's delays are an order of magnitude larger than the insider's (Fig. 3). If an outside attack is further from the PLCs than the system engineer's outside point of entry, differentiating between an attack and normal control packets becomes trivial as well. However, when benign and malicious traffic originates from the same amount of hops away from the PLCs, the system's hardware is the one that introduces the most important delays, i.e, hardware and software delays become signatures that can be used for traffic classification. Most of the tested classifiers performed better when analysing outsider traffic than insider traffic due to additional network delays present in varying network paths. The results show that the network path adds a significant amount of data to the used feature set to differentiate between two machines with a high accuracy.

4 Results

4.1 Insider Classification

The bagging technique of REPTree classifier was able to reach the highest accuracy of 92.2 percent when classifying traffic between two computers of different hardware configuration. These computers were separated by 1 hop. Fig. 4 shows the classification accuracy of all used classifiers.

The Bagging method for the REPTree classifier achieves the maximum accuracy. However, considering the time required to create classification models, REPTree classifier achieves the best accuracy while maintaining lower processing requirements. Fig. 5 shows the relationship of classifiers' accuracy versus the time it took the classifier to build a classification model. The time axis is logarithmic to better show model development duration distribution. This telemetry based IDS includes network load information in its feature set. Therefore classification is a time critical process that requires fast training of the classifiers and accurate results.

Despite very fast processing speeds of modern computers, differences between computer hardware manifests itself in network telemetry in such a way that machine learning classifiers can still detect those differences for a successful classification.

4.2 Outsider Classification

Unlike the Insider classification results, most classifiers were able to achieve very high accuracy of classifying packets from different machines. The bagging method of REPTree classifier was able to achieve an accuracy of

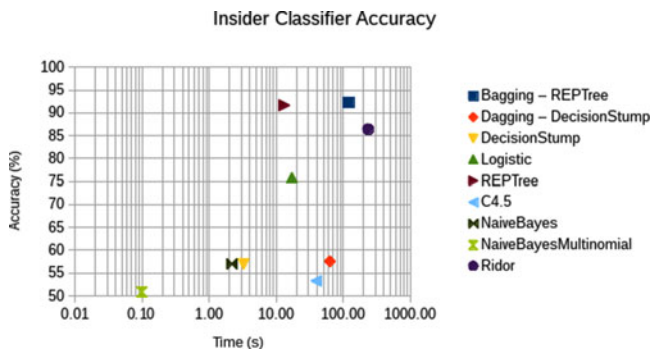

Fig. 5. Classifier accuracy versus model build time.

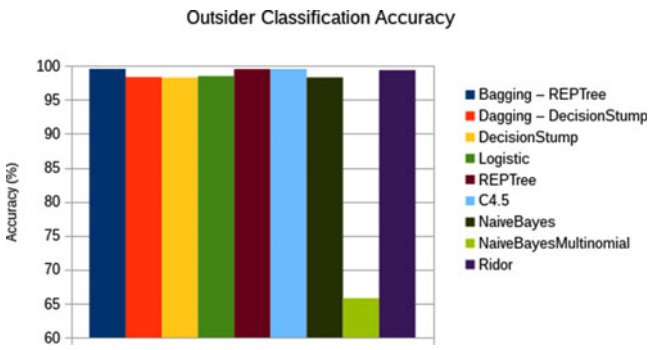

Fig. 6. Classifier accuracy for outsiders.

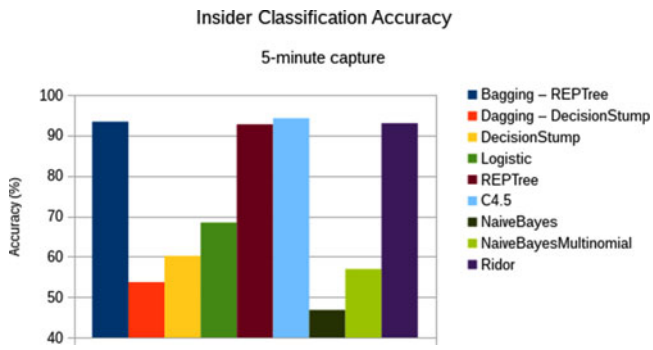

Fig. 8. Classifier accuracy for insiders for traffic captured over 5 minute window.

99.6 percent. The C4.5 classifier fell shortly behind having an accuracy of 99.5 percent (Fig. 6).

The Decision Stump and Naïve Bayes classifiers achieved a faster model build time—1.71 and 1.41 s respectively—while maintaining high classification accuracy of 98.3 percent (Fig. 7). Having traffic separated into insider and outsider groups also allows for a mix and match of different classifiers for different tasks.

Features extracted from the communication of machines separated by large network paths contain enough information to be accurately classified by machine learning classifiers. It may be further possible to generate signatures based on these features that would allow not only to detect an intrusion on an industrial control system's network, but to be able to fingerprint an attacker and determine whether the attacker has attempted intrusions before.

4.3 Decreasing Capture Intervals

The results above show classification accuracy for all of the Modbus/TCP traffic captured by the IDS over a day. However, fluctuations of the network's congestion and throughput may reduce the classifier's accuracy. To account for this, another traffic capture of the communication separated by 1 hop was performed for an interval of 5 minutes. Fig. 8 shows the classifier accuracy for this interval. The accuracy of Bagging for REPTree, REPTree, C4.5, and Ridor increased in comparison to the one day capture interval (Fig. 4). Ridor was able to achieve the highest accuracy of 94.3 percent.

The times required to build classification models have all been reduced significantly (Fig. 9) in comparison to a one day capture (Fig. 5). This is an important result as ICS IDS is processing time critical data. REPTree was the fastest classifier with model build time of 6.00 milliseconds, and an accuracy of 92.7 percent. The best performing classifier, C4.5 had a model build time of 110 milliseconds which may still be used for a dynamic classification of incoming packets.

By decreasing the capture window time to 5 minutes, accuracy of many classifiers was increased while the time to build classification models was reduced by several orders of magnitude. This goes along with the fact that network delays fluctuate with time due to different usage factors.

5 Discussion and Future Work

Accuracy of the session classifier may be improved by different methods. Being able to differentiate between the delays introduced by networking hardware, such as routers, and delays introduced by the client machine's hardware and software should increase the accuracy of the classifier. Due to a network's routing algorithms, the hop amount may not always be directly proportional to the delay measurements. For example, routers may choose a different connection path that has more hops yet lower latency due to line congestion. This problem diminishes as the separation of client and server decreases, but it needs to be addressed at higher separation distances. The developed IDS achieved high accuracy detection, and by utilizing a set of features with low control by an attacker, this IDS becomes more robust in network intrusion detections.

Future work includes acquiring an accuracy curve for attacks initiated from different hops from the target, determining the importance of software changes for detection accuracy, and testing IDSs with a variety of other attacks not covered in this paper. Since combinations of different hardware and software create unique delays in traffic propagation, it may be possible to create delay signatures to

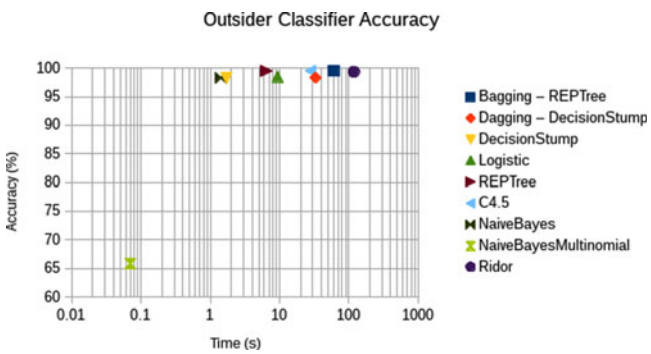

Fig. 7. Classifier accuracy versus time to build model for outsiders.

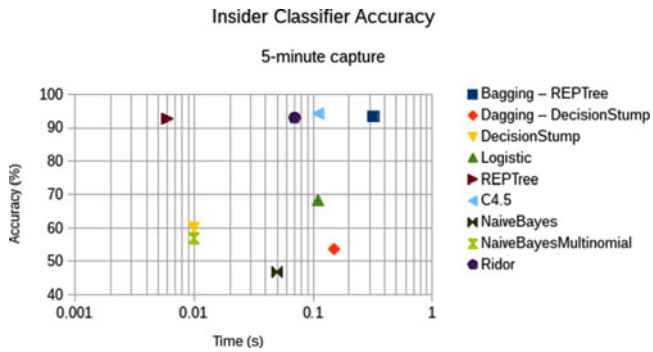

Fig. 9. Classifier accuracy versus time to build model for insiders for traffic captured over 5 minute window.

identify nodes on the network that communicate with the ICS. Future work will also be done to determine if this fingerprinting is possible.

6 Conclusions

The developed intrusion detection system can identify communication of different machines by analyzing telemetry data of the network. The classification is done at two different server-client separation stages which allows the IDS to differentiate between the inside and outside originating traffic. The IDS was able to achieve 94.3 percent accuracy with no false negatives and 5.70 percent false positives. While these values are, on average, near accuracies of other IDS, the use of network telemetry data as features results in a harder-to-obfuscate IDS. The IDS should notify system engineers of all detected intrusions and let them decide on further actions.

References

[1] CSSP and DHS, "Recommended practice: Improving industrial control systems cybersecurity with defense-in-depth strategies," *US-CERT Defense In Depth*, 2009.

[2] J. Gao, J. Liu, B. Rajan, R. Nori, B. Fu, Y. Xiao, W. Liang, and C. Philip Chen, "SCADA communication and security issues," *Security Commun. Netw.*, vol. 7, no. 1, pp. 175–194, 2014.

[3] L. T. Heberlein and M. Bishop, "Attack class: Address spoofing," in *Proc. 19th Nat. Inf. Syst. Security Conf.*, 1996, pp. 371–377.

[4] S. Ponomarev, N. Wallace, and T. Atkison, "Detection of SSH host spoofing in control systems through network telemetry analysis," in *Proc. Cyber Inf. Security Res. Conf.*, 2014, pp. 1–12.

[5] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White Paper, Symantec Corp., Security Response*, 2011.

[6] B. Schneier, "The story behind the stuxnet virus," *Forbes. com*, 2010.

[7] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the microscope," *ESET LLC*, Sep. 2010.

[8] N. Wallace, S. Ponomarev, and T. Atkison, "A dimensional transformation scheme for power grid cyber event detection," in *Proc. Cyber Inf. Security Res. Conf.*, 2014, pp. 1–12.

[9] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. N. Fovino, and A. Trombetta, "A multidimensional critical state analysis for detecting intrusions in SCADA systems," *IEEE Trans. Ind. Informat.*, vol. 7, no. 2, pp. 179–186, May 2011.

[10] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using Model-based intrusion detection for SCADA networks," in *Proc. SCADA Security Sci. Symp.*, 2007, pp. 1–12.

[11] M. Long, C.-H. J. Wu, and J. Y. Hung, "Denial of service attacks on network-based control systems: Impact and mitigation," *IEEE Trans. Ind. Informat.*, vol. 1, no. 2, pp. 85–96, May 2005.

[12] G. C. Goodwin, S. F. Graebe, and M. E. Salgado, *Control System Design*, vol. 240. Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[13] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," *NIST Special Publication*, pp. 800–82, 2011.

[14] S. Oh, H. Chung, S. Lee, and K. Lee, "Advanced protocol to prevent Man-in-the-middle attack in SCADA system," vol. 8, p. 1, 2014.

[15] N. Sayegh, I. H. Elhajj, A. Kayssi, and A. Chehab, "Scada intrusion detection system based on temporal behavior of frequent patterns," in *Proc. 17th IEEE Mediterranean Electrotechnical Conf.*, 2014, pp. 432–438.

[16] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on SCADA systems," in *Proc. Int. Conf. Internet Things 4th Int. Conf. Cyber, Phys. Social Comput.*, 2011, pp. 380–388.

[17] D. Beresford, "Exploiting siemens simatic s7 plcs," *Black Hat USA*, 2011.

[18] F. Guo and T.-c. Chiueh, "Sequence Number-based MAC address spoof detection," in *Proc. 8th Int. Conf. Recent Adv. Intrusion Detection*, 2006, pp. 309–329.

[19] A. Belenky and N. Ansari, "IP traceback with deterministic packet marking," *IEEE Commun. Lett.*, vol. 7, no. 4, pp. 162–164, Apr. 2003.

[20] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell, "Detecting 802.11 MAC layer spoofing using received signal strength," in *Proc. IEEE 27th Conf. Comput. Commun.*, 2008, pp. 1768–1776.

[21] M. Bellare, T. Kohno, and C. Namprempre, "Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol," in *Proc. 9th ACM Conf. Comput. Commun. Security*, 2002, pp. 1–11.

[22] K. Apostol, *Brute-Force Attack*. SaluPress, 2012.

[23] Y. Zhang, F. Monrose, and M. K. Reiter, "The security of modern password expiration: An algorithmic framework and empirical analysis," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 176–186.

[24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surveys*, vol. 41, no. 3, p. 15, 2009.

[25] D. Apiletti, E. Baralis, T. Cerquitelli, and V. DElia, "Characterizing network traffic by means of the NetMine framework," *Comput. Netw.*, vol. 53, no. 6, pp. 774–789, 2009.

[26] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proc. SIGCOMM Workshop Mining Netw. Data*, 2006, pp. 281–286.

[27] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Syngress, 2006.

[28] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *Int. J. Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[29] N. Wallace and T. Atkison, "Observing industrial control system attacks launched via metasploit framework," in *Proc. 51st ACM Southeast Conf.*, 2013, pp. 22:1–22:4.

[30] L. Rist, J. Vestergaard, D. Haslinger, and J. Smith. Conpot ICS/SCADA honeypot. Honeynet Project. [Online]. Available: conpot.org, 2015.

[31] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York, NY, USA: Springer, 2013.

[32] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proc. 10th Eur. Conf. Mach. Learning*, 1998, pp. 4–15.

[33] A. McCallum, K. Nigam et al.,"A comparison of event models for naive Bayes text classification," in *Proc. AAAI-98 Workshop Learning Text Categorization*, 1998, vol. 752, pp. 41–48.

[34] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Introduction to the Logistic Regression Model*. New York, NY, USA: Wiley Online Library, 2000.

[35] C. L. Devasena, T. Sumathi, V. Gomathi, and M. Hemalatha, "Effectiveness evaluation of rule based classifiers for the classification of iris data set," *Bonfring Int. J. Man Mach. Interface*, vol. 1, no. Inaugural Special Issue, pp. 05–09, 2011.

[36] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical machine learning tools and techniques with java implementations," 1999.

[37] J. R. Quinlan, *C4. 5: Programs for Machine Learning*, vol. 1. San Mateo, CA, USA: Morgan Kaufmann, 1993.

[38] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proc. 19th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2000, vol. 3, pp. 1742–1751.

[39] F. Hisch, "Performance evaluation of TCP flows," *Network*, vol. 79, pp. 79–88, 2014.

[40] L. M. Garcia, "Programming with libpcap±sniffing the network from our own application," *Hakin9-Comput. Security Mag.*, vol. 9, pp. 2–2008, 2008.

Stanislav Ponomarev is currently working toward the PhD degree in engineering at Louisiana Tech University with concentration in cyber security. His research topics of interest include image enhancement, hard drive forensics, malicious application detection, network intrusion detection, and windows executable memory attacks.

Travis Atkison received the BS degree in electrical engineering and the BS degree in computer science in 1995, the MS degree in computer science in 1997 from the University of Alabama, and the PhD degree in computer science in 2009 from Mississippi State University. He is currently an assistant professor of computer science and cyber engineering and the director in the Digital Forensics and Control System Security Lab, Louisiana Tech University.