Feature Extraction Optimization for Network Intrusion Detection in Control System Networks

Travis Atkison[1], Stanislav Ponomarev[2], Randy Smith[1] and Bernard Chen[3]
(Corresponding author: Travis Atkison)
Computer Science Department, University of Alabama[1]
PO Box 870290, Tuscaloosa, AL 35487, USA
(Email: atkison@cs.ua.edu)
BBN Technologies, Cambridge, MA, USA[2]
Computer Science Department, University of Central Arkansas[3]
201 Donaghey Ave, Conway, AR 72035, USA
(Received May 24, 2017; revised and accepted Aug. 27, 2017)

Abstract

Security measures for Industrial Control Systems (ICSs), until recently, have come mainly in the form of a physical disconnect by implementing an "air-gap". This disconnect isolated the nodes of an ICS network from other networks, including the Internet. While connecting an ICS network to the Internet is beneficial to both the engineers and companies that operate them, it places these ICSs in a situation where they are vulnerable to attacks as the protocols that are used by several of the ICSs have very little, if any, security mechanisms. This paper focuses on optimization of the feature extraction algorithms used in a continuing effort to develop a Network Telemetry based Intrusion Detection System (IDS). After development and testing of the optimizations described in this paper, the developed IDS was able to achieve 99.99% accuracy when differentiating between machines of an attacker and engineer on the same network.

Keyword: Control System Security; Network Intrusion Detection; Network Telemetry

1 Introduction

The use of Industrial Control Systems (ICS) has become common place in many businesses. While typically seen in the utilities industries (water, gas, oil, electricity), they are not limited to such. For instance, some factories use ICSs to control their assembly lines. Theme parks can use them to control their rides. Even the International Space Station uses control systems for many different operations. ICSs are designed to take complex data, process it, and complete designated tasks.

Since ICSs are used to control many sensitive operations, security is of extreme importance. Early ICSs used a security scheme known as "air-gap." Every node in an air-gapped ICS network was physically disconnected from other networks [9] and the Internet, as it was not widely used then. This physical disconnection meant that an attacker would need physical access to the system in order to perform an attack. Therefore, cyber security for an ICS was not considered a priority. Instead, the designers focused on the safety of the physical system operations and the availability of the ICS network.

"Air-gap" security measures worked well enough as long as the physical disconnect was maintained. However, as the Internet became widely used and networking equipment became more affordable, companies began to realize that they could save time and money by attaching their ICSs to the Internet. Engineers would no longer need physical access to the individual ICS networks. They would be able to monitor the ICS and address critical issues from anywhere.

Although there were benefits, attaching ICSs to the Internet exposed the ICS networks to new security threats. The main issue was the accessibility of the ICS network to would-be attackers. No longer did an attacker have to gain physical access to the network to launch an attack. Instead, they could use the Internet to access and launch a wide array of attacks against the ICS, such as non-patched system attacks, DDoS attacks, zero-day attacks, *etc.* ICS protocols were originally developed assuming "air-gapped" accessibility; therefore, cyber security measures were not designed into the protocols. By attaching these devices to the Internet, they became much more vulnerable to attacks.

Another issue inherent with an Internet connected ICS network is the difficulty of verifying the authenticity of the host sending the data and the data itself. Information, such as the sender's location, connection properties, configuration and architecture, is kept hidden from the control applications. This makes it much more difficult to

authenticate the host transmitting the data [16]. It also makes it easier for attackers to change the data within the data packet but make the packet itself appear authentic [26].

As vulnerabilities were exposed in Internet connected ICSs, it became apparent to the engineers that industry wide standards were lacking, often not used, and some companies were even creating their own. Although a few standards were available, companies were not required to use them. Having a wide array of ISC protocols made it hard to determine what the vulnerabilities were for all ICSs [12]. One protocol might have vulnerabilities that another protocol does not, and vice versa. Therefore, engineers were forced to use more time, energy and resources to find and fix the vulnerabilities.

One approach to addressing cyber-security in an ICS network is a network telemetry-based Intrusion Detection System (IDS). This approach looks at data that is sent through the network that is not used by the transmission protocol. This data is known as network telemetry. It includes information such as packet size, number of dropped packets, when the packet arrives, session sizes and times. The IDS, then, seeks to measure the telemetry data and verify it. In order to verify the data, the IDS must look at all the packets that pass through the network. From these packets, the IDS can calculate the telemetry data's running average and look for anomalies. When anomalies are found, the system triggers an alert for the engineers to investigate.

2 Background

Many cyber-attacks have been launched against ICSs [14]. Stuxnet is the most well-known example [10]. It was used to infiltrate the Iranian nuclear facilities' ICSs and change the spinning frequencies of the centrifuge motors. This caused damage to hundreds of Iranian centrifuges. Stuxnet was a well thought out, well written piece of malicious code that was able to duplicate and spread itself across a network. While it was able to spread itself over the Internet, it did not necessarily need an Internet connection. It was able to infect any USB drive that was inserted into an infected computer and use that USB drive to infect other computers [10, 22, 29]. Stuxnet was extremely difficult to find and made the centrifuge breakdowns look like unfortunate accidents.

Stuxnet opened the eyes of many ICS engineers to the need for cyber security measures to be added to ICS protocols. The consequences of a successful ICS cyber attack were too great for engineers to sit back and do nothing. Therefore, significant research is now being done on how to secure cyber-physical systems. The research has covered a wide array of approaches. Some focus on preventing a specific type of attack, i.e. denial of service or man-in-the-middle. Others focus on topics such as state estimation, traffic analysis, and hardware analysis.

One proposed method is to use a hardware fingerprint and duration [24–26]. By using patterns found in the communications of ICS network nodes, the researchers are able to create a digital fingerprint for each node that can be used to authenticate the data. In [33], Wallace *et al.* propose an IDS that can distinguish between the state of an ICS under attack and the state of an ICS operating normally. They suggest that the objects being controlled by the ICS have distinguishable underlying physical properties that can be used for state estimation.

Another approach has been to focus on network traffic. Carcano *et al.* [5] suggest an IDS that looks for state anomalies using ICS network traffic. It creates an image of the ICS from the network traffic that flows through it and uses that image to find state anomalies within the system. Temporal packet data, as seen in [28], can also be used to find anomalies in network traffic. The IDS developed in [28], which focused on the BACnet protocol, used probability functions to examine packet signatures and determine if the packet was authentic. The drawback to their method was the number of false positives resulting from anomalies found in the ICS's physical domain for which the engineers attempted to fix by reprogramming the programmable logic controllers (PLCs).

Cheung *et al.* [7] propose a model-based intrusion detection system for SCADA networks. In their system, an analysis of the communication model using Modbus is done to look for signatures of packet fields. When specific signatures are detected, an alarm is triggered.

Long *et al.* [17] chose to focus on denial of service attacks. They proposed a packet filtering system to help mitigate these types of attacks. Oh *et al.* [23] focused on man-in-the-middle attacks and suggested some modifications be made to the ICS protocols. The National Institute of Standards and Technology recommends using firewalls, MAC address locking and encryption, among other things, to help prevent man-in-the-middle attacks [31].

The latest research by Fovino *et al.* [11] developed a critical state based intrusion detection system that is able to prevent damage to physical plants. For an IDS to follow the state changes of an ICS, protocols used to transmit the data must be parsed, and state changes recorded. Fovino's latest IDS can parse Modbus and DNP3 protocols. Several filtering and monitoring techniques were developed that can describe unwanted states of the ICS. However, there is still no single solution that can guarantee the security of an ICS. While a state-based IDS will monitor the state of the system being secured, some measures to prevent the attackers from modifying the code run on PLCs have to be implemented and some of the control packets have to be blocked. To achieve this, an IDS provides a packet language that can describe signatures of unwanted Modbus and DNP3 packets.

2.1 Data Mining/Machine Learning in Network Security

By utilizing principal component analysis in [34], high accuracy state classification of a power grid was achieved.

The reduced feature set was compared to new power grid states using Hotelling's T^2 value. If the value was too high, then the new feature set could not exist and was determined to be malicious.

Mantere *et al.* [20] states that machine learning IDS can be very useful in a deterministic network such as an ICS network. Unlike typical business networks, an ICS network has a specific periodic packet flow that contains little to no noise during its normal operation. Mantere *et al.* proposes the use of throughput, IP address, average packet size, timing, flow direction, as well as payload data as features used for machine learning.

In their further work in [18], Mantere *et al.* analyzed many features listed in their original research, including network timing features - data that is critical to the research that will be presented in this paper. Mantere *et al.* was not able to detect useful behavior in timing features to detect anomalies. However, the presented research did not target any attacks on the network. The research concluded that ICS networks overall have many anomalies present in the traffic due to misconfiguration of the hardware.

The most recent research by Mantere *et al.* [19] focuses on creating a complementary network security monitoring using Self-Organizing Maps as a means of machine learning. Their approach targets restricted IP networks and does not use any network timing features, but uses packet data instead. The research concludes that deterministic properties of ICS networks make machine learning a viable tool for network anomaly detection.

Gao *et al.* [13] used a Neural Network to classify the ICS network traffic as normal and abnormal based on the operation of the MSU SCADA Security Laboratory water tank control system. The experiment resulted in a 100% accuracy classification of negative false data injection, 95% for positive false data injection, and 84.9% for a random data response injection. Unfortunately, the Neural Network developed achieved only 12.1% accuracy for a replay attack.

Network Telemetry based IDSs, such as the one presented in this paper, are useful in preventing unauthorized access to ICS PLCs. They are even able to differentiate between benign and malicious single-packet attack patterns. In order to demonstrate this ability, a specific type of denial-of-service attack was performed. It is known as a CPU shutdown attack. For this type of attack, a single packet is transmitted to the PLC, which commands its CPU to shutdown. For the system to function properly again, a physical reset of the PLC must be performed. While most systems would ban CPU shutdown commands, the telemetry based IDS did not need to do this. It was able to determine when the single packet shutdown command originated from an attacker and when it originated from the SCADA system.

2.2 Network Telemetry

The Internet Protocol (IP) forwards packets on a network between multiple nodes using Ethernet frames until the packet's final destination is reached. For this to happen, the Ethernet frame headers must include the media access control (MAC) address for the destination of the packet as well as the source. MAC addresses are different from normal IP addresses. IP addresses are assigned by self-configuration protocols or network administrators to different machine interfaces; however, MAC addresses are set when the controller circuit for the network interface is manufactured [21].

Because software can be used to spoof both the IP and the MAC addresses, it is often difficult for the server to determine where the packet is actually coming from [30, 35, 36]. To protect data integrity and prevent illegal access through spoofing, encryption is often used for security-critical algorithms [3]. While encryption is an effective method in the deterrence of most attackers, it can be broken. Encryption can, at the very least, be attacked through brute force, no matter the level of security of the cryptographic function [2]. Passwords that are weak can be easily cracked, phishing sites can trick users into giving out their passwords [4], and, like in Stuxnet, reverse engineering can be used on hashes [22]. Network telemetry data is very useful when trying to detect network intrusions. It can help to detect anomalies caused by an attacker using software that is not normally used or even using different machines.

Chandola *et al.* [6] provide a survey of multiple studies focusing on anomaly detection in which machine learning and data mining methods and techniques are used to detect anomalies. One particular data mining application that concentrates on traffic data interactions and correlations is NetMine [1]. Though the methodologies implemented are similar to those that are used in the research effort presented here, their studies are more interested in improving network stability and traffic quality than network security.

By analyzing derived transport layer statistics, Erman *et al.* [8] successfully clustered together packets that were similar. Without extracting packet data, they were able to identify protocols successfully by using DBSCAN and K-Means algorithms [8]. Using the received signal strength (RSS) of packets that are transmitted over wireless networks and analyzing the statistical fluctuations, Shen *et al.* were able to show the ability to detect host spoofing [30].

Wireshark is a software-based network analyzer that captures and displays network traffic in real time. One of Wireshark's many features is that it gives a system administrator the ability to keep all network packets that come in to the network so that they can be analyzed later to find any useful information. For this research effort, Wireshark was used to extract network packet arrival times. This data was placed in comma separated values format to interpret and generate graphs of the data [27].

Although the detection approach that is presented in this effort may not endure the dynamics of a traditional enterprise-wide Local Area Network (LAN), it is beneficial in control system LANs for the detection of spoofed hosts. Control system LANs are unique in their construction in that the hosts will commonly communicate in set intervals that are defined by the particular polling protocol that is used [15]. The detection method that was developed for this research effort has the ability to determine when communication is occurring outside of these set intervals and will trigger a potential intrusion alert when detected. In addition, with the continued use of open source tools that can automate the attack process targeting control systems [32], this effort proves to be successful in differentiating between benign and malicious network packets.

3 Telemetry Based Intrusion Detection System

One of the critical components of the developed Intrusion Detection System algorithm is the session capture method. When an attacker is spoofing MAC and IP addresses, there are two methods for them to inject data into the control system network - either by using TCP session spoofing, or instantiating new TCP connections. This section covers the basics of TCP sessions, and discusses experiments performed to evaluate different session instantiation techniques.

The possibility of spoofed addressing, as well as TCP session injections, makes it impossible to differentiate between communication sessions as defined by the TCP flow model. Therefore, to logically group incoming traffic into sessions, a new session definition must be created. Several session aggregation techniques were selected and tested for this research.

3.1 TCP Session Flows

An example TCP session flow is shown in Figure 1. TCP is a connection oriented protocol that utilizes internal state variables in order to maintain the connection. TCP session begins with a TCP handshake. The TCP handshake includes a 3-way packet exchange that is commonly explained as SYN, SYN/ACK, ACK. SYN is a synchronization packet sent from the client to the server. This packet starts a TCP session and asks the server to establish a connection. The server then responds with a SYN/ACK packet, acknowledging the connection request. Afterwards, the client acknowledges the server's acknowledgment with an ACK packet. This completes the TCP handshake and allows the data exchange to be started. The TCP session is closed with a FIN, ACK, FIN, ACK sequence. When the client decides to terminate the connection, it sends a FIN packet to the server. The server then sends an ACK packet, acknowledging the request to terminate the connection. After sending the ACK packet, the server has time to release all of the resources used by the connection, and sends a FIN packet of its own to notify the client that all of the resources have been cleared. The client then sends its final packet - ACK - to acknowledge the connection termination. TCP sessions are sets of all of the packets between two nodes that start with the TCP handshake and end with the FIN, ACK, FIN, ACK packet sequence.

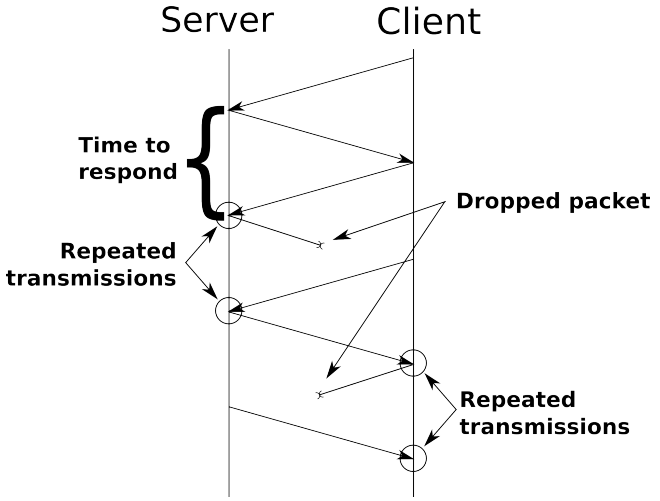

Figure 1: Client-server session graph

Whenever an attacker wants to spoof and inject information into the TCP session, he has to inject packets in the middle of an already established TCP connection; therefore, proper network telemetry can only be obtained if the data is extracted from a session flow model, rather than a by-packet basis because there is no performance information present in the analysis of a single packet. A single packet will only have the time stamp of its arrival to the server. In order to capture relevant performance information of the TCP communication, two methods have been created and tested for this research.

3.2 Session Duration Based Instantiation

Session duration based instantiation seeks to identify periods of communication silence when silence is larger than a time threshold. New sessions are defined as a set of packets between the detected silence intervals when the next packet is sent towards the server (PLCs). This method was selected to match the periodical silence of the polling nature of Modbus/TCP SCADA architecture. Silence time threshold was determined experimentally by using 0.1 second interval increments to maximize the C4.5 based classifier accuracy. Table 1 and Figure 2 show session counts for a given time interval, while Table 2 and Figure 3 show accuracies achieved by the C4.5 algorithm, as well as Model Build Times (MBT) in seconds.

A malicious session defines a session that has at least one malicious packet. As the length of the inter-session silence increased, the count of benign sessions decreased,

Table 1: Generated session counts for varying silence intervals

Interval (s)	Malicious Sessions	Benign Sessions	Total Sessions
0.1	51156	89943	141099
0.2	51021	42125	93146
0.3	50893	29972	80865
0.4	50893	24458	75351
0.5	50891	19430	70321
0.6	50891	14454	65345
0.7	50888	9648	60536
0.8	50886	4869	55755
0.9	50508	403	50911

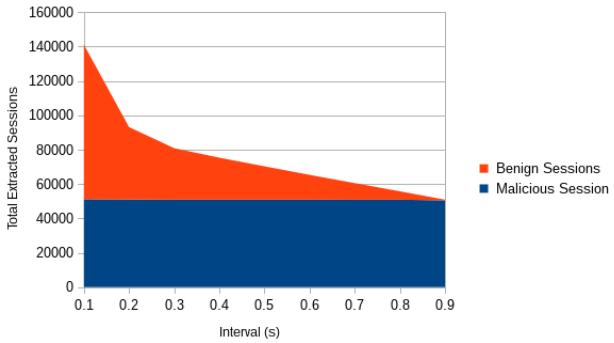

Figure 2: Silence interval detection session counts

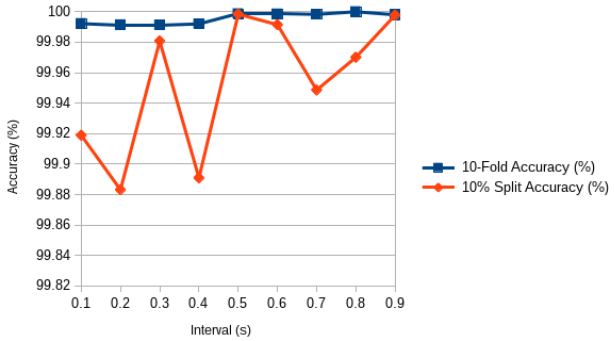

Figure 3: Silence interval detection accuracy

while the count of malicious sessions increased. This trade off happened due to the definition of the malicious session. This is shown in Table 1 where the number of benign sessions drop from 89943 for a silence interval of 0.1 to only 403 as the interval increases to 0.9. A similar drop is also seen in the total number of sessions. Figure 2 pictorially shows this dramatic drop in the number of sessions. A benign session could not have any malicious packets at all, while a malicious session could have any number of benign packets, as long as it had at least one malicious packet.

Table 2 provides accuracies for two different classification experiments: 10-Fold validation, and 10% data split, as well as the time in seconds the C4.5 classifier took to build its model (MBT). The 10-Fold validation experiment takes the dataset and splits it into 10 chunks. The classifier uses the first 9 chunks for training, and the last chunk for accuracy verification. The chunks are then rotated and the experiment is repeated ten times. The accuracy value in Table 2 is the average accuracy of classification for all ten experiments. While this accuracy is not a good estimate of the overall classifier accuracy, it can be used to verify the information distribution of the dataset. If some parts of the dataset contained lower information about the subject, the 10-Fold accuracy would be a lower number. Unlike 10-Fold validation, the 10% Split experiment uses the first 10% of the dataset for training, and the other 90% for validation. The 10% Split accuracy recorded in Table 2 refers to the classifier accuracy when classifying 90% of the dataset. 10% of the dataset translates to approximately 2.5 hours of captured traffic.

Table 2: Classification results of varying silence intervals

Interval (s)	10-Fold 1Accuracy (%)	10% Split Accuracy(%)	MBT (s)
0.1	99.9922	99.9189	3.33
0.2	99.9914	99.8831	0.87
0.3	99.9913	99.9808	0.42
0.4	99.9920	99.8909	0.33
0.5	99.9986	99.9984	0.29
0.6	99.9985	99.9915	0.25
0.7	99.9983	99.9486	0.17
0.8	100.000	99.9701	0.49
0.9	99.9980	99.9978	0.11

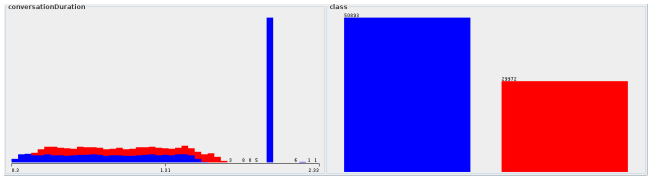

Figure 4: Silence interval session duration and class distributions for 0.3s silence interval

The interval of 0.9 was the last usable interval. Intervals above 0.9 are not listed because only malicious sessions were created while using such a large interval. While Classification accuracy seems to increase as the interval increases (Figure 3), the amount of malicious sessions in the dataset starts to outweigh the amount of benign sessions drastically. For example, if we take the amount of sessions for the 0.9 second interval, just choosing a malicious class over all of the data will result in 99.208% accuracy, as there are 50508 malicious features and only 403 benign features. However, selecting a malicious class for the 0.3 second interval would result in an accuracy of 63.936%, while the C4.5 classifier was able to achieve 99.9808% accuracy for a 10% Split experiment.

Examining the results presented in Table 2 and Figure 3 across all intervals shows an accuracy achievement of between 99.8909% and 99.9984% for the 10% Split ex-

Table 3: Generated session counts for varying conversation lengths

Length (packets)	Malicious Sessions	Benign Sessions	Total Sessions
1	343202	255209	598411
2	283280	157384	440664
3	183664	99144	282808
4	167121	53786	220907
5	132017	49889	181906
6	133396	21106	154502
7	120452	16223	136675
8	112428	3897	116325

periment and an accuracy between 99.9913% and 100% for the 10-Fold experiment. As shown in Table 2, the MBT are low and are relative to the total session size as noted in Table 1.

For the use of the silence interval as a session separation method, the best value is determined to be 0.3 seconds, as it provides high classification accuracy (99.9913% and 99.9808% for the 10-Fold and 10% Split experiments respectively) as well as approximately equal amounts of malicious and benign features. Figure 4 shows the distribution of features based on session duration and features class for the value of the 0.3 second interval. In addition, Figure 4 shows a duration based distribution of sessions based on the silence interval of 0.3 seconds.

3.3 Session Length Based Instantiation

Session length can be used as an alternative method of extracting sessions from a non-interruptible TCP traffic flow. In these experiments, sessions were separated by counting the amount of packets already in the session. If that number is larger than a given length and the next packet was sent to the server, a new session was instantiated. Table 3, as well as Figure 5, present the data from varying session length separation values.

The same trend found in Table 1 can be noticed in Table 3 for the silence based extraction - the larger the extracted sessions are, the less benign features are extracted due to the benign session being defined as having no malicious packets. However, the amount of malicious sessions extracted is not as stationary as with the silence based extraction. As the session length increases, the amount of extracted malicious sessions decreases from 343,202 malicious sessions for a packet length of 1 down to 112,428 malicious sessions for a packet length of 8, as seen in Figure 5.

At no point of using session length as a session detection parameter was there a comparable amount of malicious and benign sessions extracted as seen in the silence intervals. Moreover, when the amount of extracted sessions was compared to each other, the classification accuracy was significantly lower than that of a silence based extraction.

Table 4: Classification results of varying conversation lengths

Length (packets)	10-Fold (%)	10% Split (%)	MBT (s)
1	97.0796	97.1679	41.6
2	97.3172	97.2468	20.9
3	99.9346	99.7929	13.5
4	99.9570	99.8521	8.31
5	99.9082	99.7306	6.47
6	99.9663	99.8698	4.28
7	99.9188	99.5699	4.34
8	99.9003	99.8147	1.62

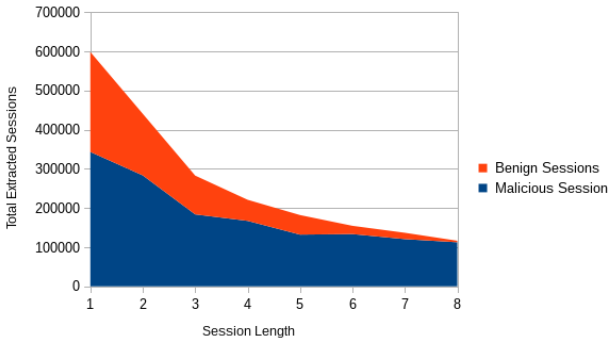

Figure 5: Session length based extraction counts

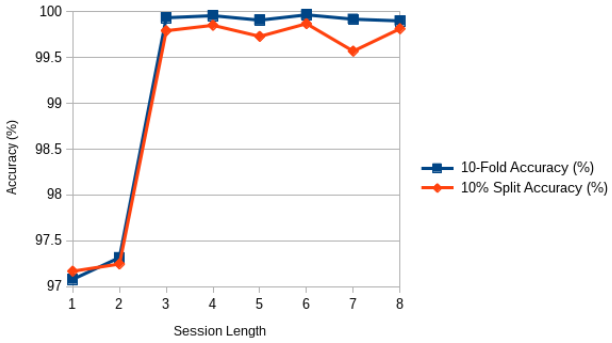

Figure 6: Session length extraction accuracy

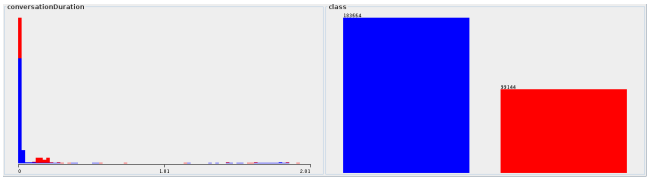

Figure 7: Session length of 3 session duration and class distributions

Similar to the results presented above for silence intervals, Table 4 and Figure 6 present classification and accuracy results for varying conversation lengths respectively. Examining the results presented across all intervals show an accuracy achievement of between 97.1679% and 99.8698% for the 10% Split experiment and an accuracy between 97.0796% and 99.9663% for the 10-Fold experiment. As shown in Table 4, the MBT are low and are relative to the total session size as noted in Table 3.

Comparable accuracy to that of the silence interval is achieved when the session has at least 3 packets. 10% Split Accuracy of 99.7929 and 10-Fold Accuracy of 99.9346 are achieved, however, it takes 13.5 seconds to build the model. When the distribution of extracted sessions is graphed (Figure 7), the majority of the sessions span very little time in contrast to silence based session extraction.

4 Discussion and Future Work

Through the use of different methods there may be improvement in the accuracy of the session classifier. One way to increase the classifier accuracy is by being able to make a distinction between delays that are introduced by a client machine's software and/or hardware and the delays that are introduced by networking hardware, such as routers. Because of idiosyncrasies of a given network's routing algorithm, the number of hops calculated and the delay measurement may not always be directly proportional. As an example, a router may select a connection path with a higher number of hops because of latency due to line congestion. As the separation decreases between the client and server, this problem diminishes, but the problem still needs addressing at the increased separation distances. The detection accuracy for the IDS that is developed and presented in this paper achieved high results. Furthermore, by using a specific set of features for which the attacker has little control over, the IDS provides robustness in network intrusion detections.

There are several threads of future work for this effort. These include looking at how changes in software will affect the detection accuracy, assessing how different hop counts from the target effect the accuracy curve, and expanding the variety of attacks to test against the IDS. Due to the fact that different variations of software and hardware can create unique delays in the propagation of the traffic, a possibility exists to create delay fingerprints that would identify nodes that communicate with the IDS. Work will be done to see if this fingerprinting technique can be accomplished.

5 Conclusions

When an attacker is spoofing their machine's addresses and injecting traffic in the middle of an already-established TCP session, it becomes impossible to isolate the attacker's TCP sessions from TCP sessions from a benign machine. In order to collect the information that can be used for an intrusion detection, two session aggregation methods were created and tested. Overall, silence based feature extraction presents better accuracy and efficiency of detecting network intrusions. Both classification accuracy and model build time were better for this method in comparison to session length based extractions. Results presented provide accuracies at 99% when differentiating between the machines of an attacker and an engineer on the same network.

References

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

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

[3] D. Basin, C. Cremers, K. Miyazaki, S. Radomirovic, and D. Watanabe, "Improving the security of cryptographic protocol standards," *IEEE Security & Privacy*, vol. 13, no. 3, pp. 24–31, 2015.

[4] J. Bonneau, C. Herley, P. C. V. Oorschot, and F. Stajano, "Passwords and the evolution of imperfect authentication," *Communications of the ACM*, vol. 58, no. 7, pp. 78–87, 2015.

[5] 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 Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 179–186, 2011.

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

[7] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA Security Scientific Symposium*, pp. 1–12, 2007.

[8] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, pp. 281–286, 2006.

[9] M. Fabro, T. Nelson, "Control systems cyber security: Defense-in-depth strategies," in *2007 ISA (International Society of Automation) Expo*, Oct. 2007.

[10] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *Security Response*, 2011.

[11] I. N. Fovino, A. Coletta, A. Carcano, and M. Masera, "Critical state-based filtering system for securing scada network protocols," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 10, pp. 3943–3950, 2012.

[12] J. Gao, J. Liu, B. Rajan, R. Nori, B. Fu, Y. Xiao, W. Liang, and C. P. Chen, "Scada communication and security issues," *Security and Communication Networks*, vol. 7, no. 1, pp. 175–194, 2014.

[13] W. Gao, T. Morris, B. Reaves, and D. Richey, "On scada control system command and response injection and intrusion detection," in *eCrime Researchers Summit (eCrime'10)*, pp. 1–9, 2010.

[14] B. Genge, I. Kiss, and P. Haller, "A system dynamics approach for assessing the impact of cyber attacks on critical infrastructures," *International Journal of Critical Infrastructure Protection*, vol. 10, pp. 3–17, 2015.

[15] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in scada systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[16] L. T. Heberlein and M. Bishop, "Attack class: Address spoofing," in *Proceedings of the 19th National Information Systems Security Conference*, pp. 371–377, 1996.

[17] M. Long, C. H. J. Wu, and J. Y. Hung, "Denial of service attacks on network-based control systems: impact and mitigation," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 85–96, 2005.

[18] M. Mantere, M. Sailio, and S. Noponen, "Network traffic features for anomaly detection in specific industrial control system network," *Future Internet*, vol. 5, no. 4, pp. 460–473, 2013.

[19] M. Mantere, M. Sailio, and S. Noponen, "A module for anomaly detection in ics networks," in *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, pp. 49–56, 2014.

[20] M. Mantere, I. Uusitalo, M. Sailio, and S. Noponen, "Challenges of machine learning based monitoring for industrial control system networks," in *26th International Conference on Advanced Information Networking and Applications Workshops (WAINA'12)*, pp. 968–972, 2012.

[21] J. Martin, E. Rye, and R. Beverly, "Decomposition of mac address structure for granular device inference," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pp. 78–88, 2016.

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

[23] S. Oh, H. Chung, S. Lee, and K. Lee, "Advanced protocol to prevent man-in-the-middle attack in scada system," *International Journal of Security and its Applications*, vol. 8, no. 2, pp. 1–8, 2014.

[24] S. Ponomarev and T. Atkison, "Industrial control system network intrusion detection by telemetry analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 252–260, 2016.

[25] S. Ponomarev and T. Atkison, "Session duration based feature extraction for network intrusion detection in control system networks," in *International Conference on Computational Science and Computational Intelligence (CSCI'16)*, pp. 892–896, 2016.

[26] S. Ponomarev, N. Wallace, and T. Atkison, "Detection of ssh host spoofing in control systems through network telemetry analysis," in *Proceedings of the CIRSC Conference*, pp. 1–12, 2014.

[27] C. Sanders, *Practical Packet Analysis: Using Wireshark to Solve Real-World Network Problems*, No Starch Press, 2017.

[28] N. Sayegh, I. H. Elhajj, A. Kayssi, and A. Chehab, "Scada intrusion detection system based on temporal behavior of frequent patterns," in *17th IEEE Mediterranean Electrotechnical Conference (MELECON'14)*, pp. 432–438, 2014.

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

[30] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell, "Detecting 802.11 mac layer spoofing using received signal strength," in *The IEEE 27th Conference on Computer Communications (INFOCOM'08)*, pp. 1768–1776, 2008.

[31] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," *NIST Special Publication*, pp. 800–82, 2015.

[32] N. Wallace and T. Atkison, "Observing industrial control system attacks launched via metasploit framework," in *Proceedings of the 51st ACM Southeast Conference (ACMSE'13)*,, pp. 22:1–22:4, 2013.

[33] N. Wallace, S. Ponomarev, and T. Atkison, "A dimensional transformation scheme for power grid cyber event detection," in *Proceedings of the CIRSC Conference*, pp. 1–12, 2014.

[34] N. Wallace, S. Semple, and T. Atkison, "Identification of state parameters for stealthy cyber-events in the power grid using pca," in *IEEE PES General Meeting Conference & Exposition*, pp. 1–5, 2014.

[35] G. Yao, J. Bi, and A. V. Vasilakos, "Passive ip traceback: Disclosing the locations of ip spoofers from path backscatter," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 471–484, 2015.

[36] J. Yu, E. Kim, H. Kim, and J. Huh, "A framework for detecting mac and ip spoofing attacks with network characteristics," in *International Conference on Software Security and Assurance (ICSSA'16)*, pp. 49–53, 2016.

Biography

Travis Atkison is an Assistant Professor of Computer Science at the University of Alabama. He is the director of the Digital Forensics and Control System Security Lab (DCSL). His current research efforts focus on the topic of cyber security. These efforts include malicious software detection, threat avoidance, digital forensics, and security in control system environments including transportation.

Stanislav Ponomarev is currently a Cyber Security Scientist at BBN Technologies. Dr. Ponomarev received his Master and PhD degrees from Louisiana Tech University in 2015.

Randy Smith is an Associate Professor in the Department of Computer Science at The University of Alabama.

Dr. Smith is an active researcher in transportation safety with over 15 years experience in crash data analysis, linear referencing systems and safety data integration. His research has been supported by various federal agencies including NASA, NSF, and the USDOT. In addition, Dr. Smith has worked with multiple states on transportation safety matters.

Bernard Chen received the Master and PhD degrees in Computer Science from Georgia State University in 2008. Currently, he is an Associate Professor in the Department of Computer Science at The University of Central Arkansas. His research interests include Data Science, Data Mining, Bioinformatics, and Wineinformatics.