Intrusion Detection in SCADA Systems via Network Analysis

Justin Jones
Department of Computer Science
Sam Houston State University
Huntsville, Texas 77341
Email: jxj037@shsu.edu

Yesenia Valles
Department of Computer Science
Sam Houston State University
Huntsville, Texas 77341
Email: yev001@shsu.edu

***Abstract*—Given the nature of SCADA systems, we postulate that it is possible to detect intrusions or otherwise malicious activity via network analysis. Utilizing machine learning, we develop methods to passively monitor systems and identify possible intrusions and complete a general study of SCADA networks.**

***Index Terms*—scada, network analysis, machine learning, whitelist.**

1. Introduction

SCADA (Supervisory Control and Data Acquisition) [1] systems are used to support and monitor the infrastructure that serves as pillars for many industrialized areas. An example SCADA system can be seen in Fig. 1. SCADA systems were historically created with an emphasis on availability of data as opposed to implementation of the X.800 security architecture. With the emerging prevalence of IoT security and edge-intrusion testing, the security of these systems has become an increasingly pertinent subject in the security community as networking and remote access becomes more and more desirable. Given that the purpose of SCADA systems is to enable users to manage and perform tasks on a large scale in industrial infrastructures as quickly and easily as possible, security vulnerabilities have the potential to cause serious damage if an attacker gains access to a system. One such example of the consequences can be seen from the Stuxnet worm, controlling and damaging Iranian nuclear reactors in 2010 [2]. The use of open standards for SCADA communication protocols are also increasing, largely due to low cost of implementation, so the risk of future security breaches will likely be applicable to multiple systems rather than limited to single product lines from a particular manufacturer.

2. Theory

It is now obvious that the security of SCADA systems is a pertinent subject for study. As there can be any number of zero-day attacks for these systems, it will likely be more productive to shift focus to detecting these intrusions rather than trying to prevent them. Given the nature of SCADA

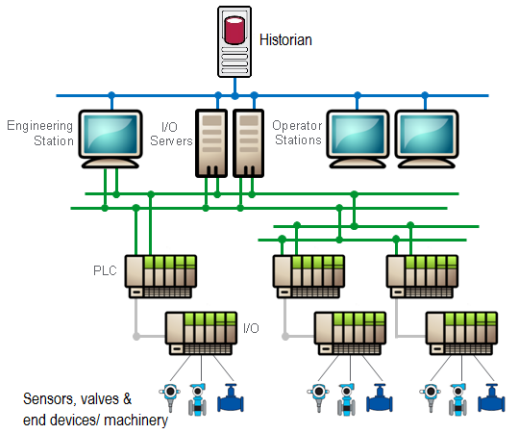

Figure 1. A basic SCADA system

systems, it can be presumed that their network traffic is reasonably predictable. This leads to the proposition that there are detectable network patterns occurring with some regularity for any given system or otherwise some normal level of activity. As stated previously, SCADA traffic patterns are generally stable in comparison to traditional IT networks. Once a certain normalization is established, one may then use that as a baseline to compare future traffic to in order to detect anomalies that may possibly indicate active attacks. Our proposal is that, instead of focusing on systems or network exploits themselves, we use a tool to establish and then passively detect any network anomalies.

2.1. Machine Learning & Network Captures

To implement this tool, we intend to utilize machine learning-based analysis trained on known *regular* traffic levels. We will then introduce network/control anomalies indicative of non-normal system behavior to demonstrate the detection capabilities and to test proper training levels of the algorithm. An obvious source of data to train the algorithm on are just source and destination IP addresses, however were we to go this route it would likely be easier to instead implement a hardware or software firewall with IP filtering at the network gateway level.

Utilizing *Python* along with *Numpy*, *Pandas*, and *Sci-kit Learn*, we focus on the timing and host/source interactions on the network to indicate anomalies. The creation of white-lists attributed to normal traffic coupled with a collection of timing statistics will aid in flagging anomalous traffic.

2.1.1. Timing Analysis. In analyzing typical pcap files acquired from various SCADA networks, it can be seen that the timing of host interactions can be used as an indicator for anomalous activity. Consider Fig. 2 below:

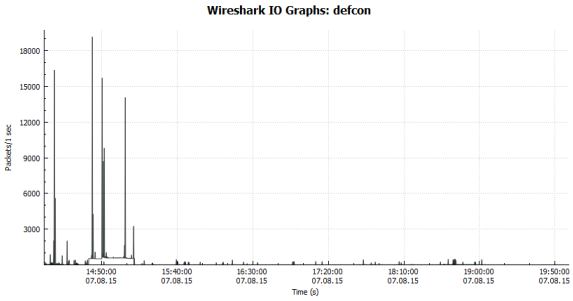

Figure 2. Anomalous Traffic in a SCADA System

This data has been retrieved from a publicly available pcap file from DEFCON SCADA 23 ICS Village via Netresec. [4] The data exhibits timing anomalies, displayed as packets per second, around the 14:90 mark. A more localized view of the graph follows in Fig. 3, focusing on the anomalies seen in the overview above.

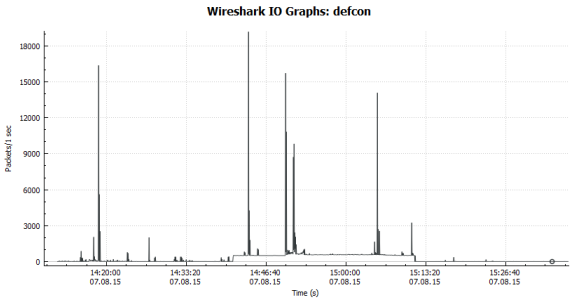

Figure 3. A more detailed overview of anomalous traffic

Given that each increment represents one second, we see spikes of more than 18,000 packets in just milliseconds of traffic. Our algorithm would flag the data based on rate of communication, and report the source and destination addresses. A clearer version of the traffic is shown in logarithmic view, which is similar to how a machine learning algorithm would visualize things. An example of this is provided in Fig. 4.

2.1.2. Whitelist Automation. Our usage of ML will initially undergo a *learning phase* over a certain period of time to create a type of whitelist. The rationale for this is that SCADA systems commonly have frequent and consistent interactions amongst the same hosts. They act in such a way

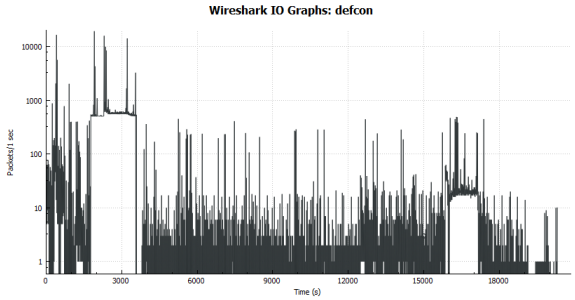

Figure 4. A logarithmic view

because all network traffic (incoming and outgoing) should interact and be handled by one "SCADA server" [5]. These common communications will serve in identifying several MAC and IP address pairs. It is safe for us to assume that the common pairs identified during the learning phase are authenticated devices.

To create the whitelist, a few items are taken into consideration. These list entries are *client IP address, server IP address, port, and IP protocol*. These packet items were chosen to be consistent with research previously performed, specifically involving flow whitelisting in SCADA Networks. [7] Systems themselves that have been identified as authenticated devices are whitelisted in conjunction to their MAC and IP addresses. IP protocol is important because traditionally, industry control systems work off of a *Modbus* communication protocol. Modbus is modeled in a master-slave dynamic. That is, a Modbus will connect a supervisory computer (master) with a subordinate *remote terminal unit* (RTU). [8] The core attributes of the Modbus protocol we are interested in is the function code and data components. Function codes have the capability of telling us what instructions are being carried out to the system. Using WireShark, analysis of our pcap file from DEFCON SCADA 23 ICS, it was noted that 71% were of the TCP/Modbus protocol. Since control systems act in a behavior that is fairly predictable, analysis during the learning phase of what commands constitute as normal will aid us in detecting anomalies. [6] A visual representation of our whitelisting components is shown in Fig. 5 below.

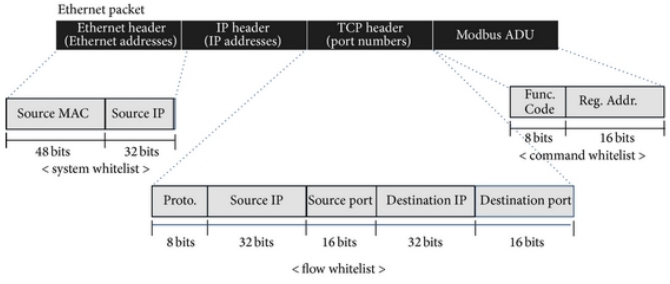

Figure 5. Representation of whitelist components

Once network traffic has been processed through the

learning phase, it is our expectation that any non white-listed connections and instructions will raise alarm. Therefore, we ensure that only known hosts are participating on the network as well as expected system interactions. By automating this process through ML, the need for human involvement is mitigated during pcap analysis.

For the act of whitelisting to be effective, we will have to assume that our list is a 1) reasonable, manageable size and 2) relatively stable in its nature. The ratio of "acceptable" entries versus the overall size of the capture should be smaller by comparison. Similarly, we would like to have this list structured and stable. A highly dynamic, changing list would likely correlate to an influx of false positives. As with all Intrusion Detection Systems, this is not a desirable trait.

Apart from common IT infrastructure, it would not be wise to automatically quarantine suspected malicious content in SCADA systems. In the event of a false positive, "legitimate" traffic communications not being able to be made have a possibility of causing large-scale implications such as black-outs or extreme water levels.

2.2. Common Attacks

Considering the preceding explanations on timing analysis and whitelist automation, we explore common attacks in SCADA systems that our software may have the capability of detecting.

The injection of requests to a network can indicate an attempt of discovering available hosts and services. An attacker would then use the revealed information in to maliciously impact the system. It is our theory that our software be able to flag these forms of information gathering attacks, as it violates system as well as flow whitelisting.

Modbus/TCP protocol holds a vulnerability making a system susceptible to *Denial of Service*(DOS) attacks. This is due to an implementation error when a system is processing *Read Discrete Inputs* during request/response messages. Remote, unauthenticated attackers can exploit this in a way that allows them to alter the request or response parameters. [6] Attackers can replace these otherwise harmless parameters with malicious values in the "data field" option, then sending it to a system Modbus/TCP implementation. This processing of messages has the possibility of triggering a Denial of Service attack to a vulnerable system.

3. Implementation

The implementation software is divided into two basic sections - the analysis module and the capture module. The capture module controls the retrieval and subsequent conversion of the *pcap* data into *csv* files and calls routine analysis on the data via the analysis module.

In order to run machine learning analysis on input data, the file must first be converted to numerical values which is done by hashing values together. To implement the machine learning and training, we are using an open-source program retrieved from [9] that is wrapped with an additional script to control capture. The general analysis scheme can be seen in Algorithm 2, and Algorithm 1 details the basic collection scheme.

Data: Target Network
Result: PCAP file
for *Specified time period* **do**
  | *output_file* ← *packet data*
end

Algorithm 1: Data Collection

⇓

Data: Interpreted Data
Result: Trained Model
split data
for *row* ∈ *input* **do**
  | trained model ← (SRCIP, DESTIP, TIME,
  | PROTOCOL)
end
for *row* ∈ *test* **do**
  | verify results
end

Algorithm 2: Model Training and Verification

4. Experiments

For our experiment, we used our own university's SCADA lab as a source of traffic that is not infiltrated as a control variable. This was done so that we could confirm our predictions with our own packet capture analysis of data controlled by us. Our findings were consistent with what we observed prior in that traffic was very redundant in nature and exhibited a normal structure expected for a stable, secure system. As an example, it maintained a fixed amount of communicating clients and the pairs involved in the communication were static, demonstrating the standard "Master-Slave" architecture of Industrial Control Systems. Similarly, the timing between the communications were predictable with few outliers, while also reflecting the common protocols explained prior. Our findings on the fixed environment at Sam Houston State University's SCADA lab is shown below in Fig. 6 for a total of 58,995 packets as read in by Snort over a duration of 14 minutes, captured by InduSoft v8.1 [1].

1. http://www.indusoft.com

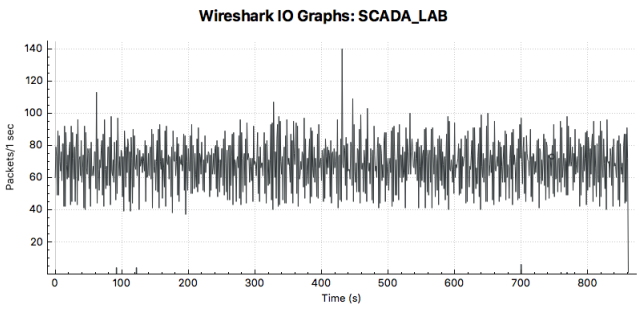

Figure 6. SCADA Lab Traffic

Snort analysis found no suspect packets, and a general view of the IO graph would corroborate this as well. Running this capture through the software we're using also results in no suspect results. We observed, however, that the traffic was unencrypted so it would be possible to view the payloads and control outputs once we had access to the network. This would lend itself to a 2-phase listen/replay attack for even a relatively unsophisticated malicious 3rd party, so network security should properly implemented.

Next, we began to perform an attack to this same SCADA system in the hopes of revealing any indication that our predictions for what would change in network traffic analysis held true. To begin this, we first performed a Denial of Service (DOS) attack on the SCADA software environment in the lab, which acts as the Human Machine Interface (HMI). Our targeted port was 1234, which plays a role in system communications. This was to see if it would provide unusual reporting to the database. In our testing, we observed that we were able to bring down the remote web viewer for the HMI. The IO graph can be seen below in Fig. 7 displaying the disrupted communications as the network repeatedly tries to recover.

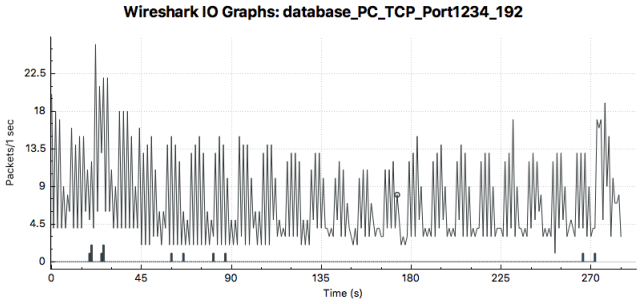

Figure 7. Port 1234 Attack Traffic

In addition, we performed this same attack to a PLC itself, a Direct Logic 06 Koyo. It is important to note this because different types of PLCs will result in different use of protocol, and this one was proprietary in nature as observed by the packet capture. In this attempt, we attacked port 28784 as it was the source port to the PLC. The output device in this scenario was a tower lamp. After performing a DOS attack on the PLC at its address, it was observed that the timing intervals between requests were elongated and the output device ceased to perform tasks as normal. The indicators on the HMI would show that the performance was still occurring as desired by the system, though the light itself was stagnate at one color instead of performing a cycle of red, yellow, and green. This is likely due to the reliance on UDP - the control software is not aware that the commands aren't being received by the device due to the 'connectionless' aspects of UDP. Our network observations can be seen below in Fig. 8 showing the cyclic overloads and recovery attempts.

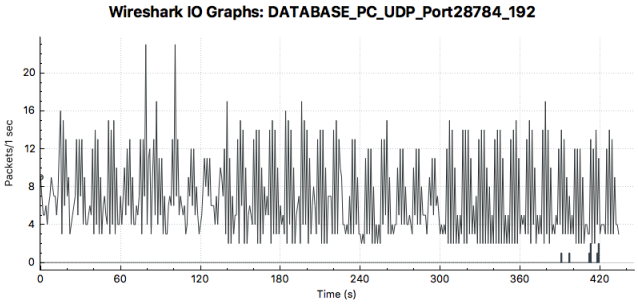

Figure 8. Port 28784 Attack Traffic

Finally, we tested an attack on a "middle-man" for communication between our HMI that uses Endusoft and a PLC. It seemed to bring down the system quicker attacking this vector as opposed to the PLC itself, but this could have been due to buffer size. This node serves as a converter from modbus to the Schneider PLC. The HMI knew and explicitly showed that it's connection to the PLC was severed, as opposed to our previous testing, as seen below in Fig. 9. This is a notable observation because the our prior testing does not allow the end-user or likely its packet captures to express a killed communication. In Fig. 10 we can see the results. By changing the timing of a TCP attack on Port 502 we were able to cause irregular behavior in the indicator bulb controlled by the PLC.

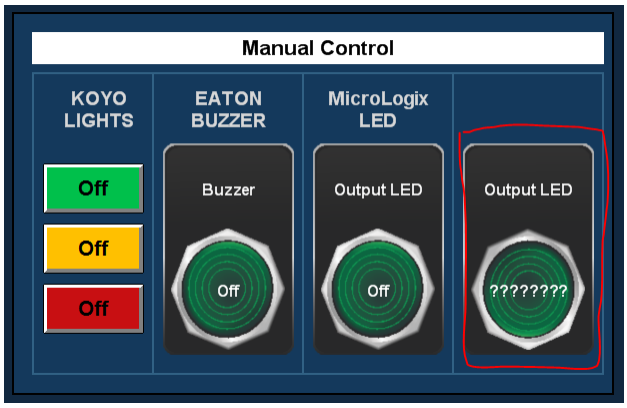

Figure 9. InduSoft Severed Connection

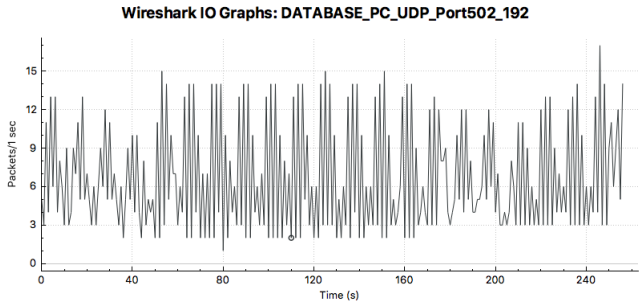

Figure 10. Port 502 Attack Traffic

We further observed that PLCs utilizing TCP communications were also affected when we attacked over UDP, however this could be due to switch overloading or the way the PLCs handle packet drops.

Once we gathered malicious captures, we found that we were able to not only check for the attacks but realized we would be able to classify attacks with modification to the training algorithm given the capture inputs.

5. Conclusion

After experimentation and the collection of results, it is clear that our previous expectations of a secure SCADA system holds true. Our results in comparing the controlled environment network statistics versus the attacked clearly shows differences between source and destination pairs and the frequency of communications. In the future, we would like to explore more into testing the injection of malicious parameters into the data field within network packets being sent to a system with Mobdus/TCP implementation. Due to time constraints, we were not able to investigate further into this commonly used attack. Despite this, we our confident in our predictions. We further encountered issues with the actual machine learning implementation regarding pcap files and input handling, which is why we ended up utilizing the software from [9]. The software successfully indicated the attacks given samples of normal traffic and anomalies, however we realized without preexisting normal captures and malicious captures it wasn't incredibly useful and further the output did not lend itself to our original goals of white-listing and pointing out specific malicious packets. In the future we would like to extend this software to include these capabilities.

References

[1] En.wikipedia.org. (2017). SCADA. [online] Available at: https://en.wikipedia.org/wiki/SCADA [Accessed 19 Sep. 2017].

[2] Kushner, D. (2013). The Real Story of Stuxnet. [online] IEEE Spectrum: Technology, Engineering, and Science News. Available at: https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet [Accessed 19 Sep. 2017].

[3] Gaurav, K. (2016) Open Source For You. [Online]. Available at: http://opensourceforu.com/2016/09/deep-learning-network-packet-forensics-using-tensorflow/. [Accessed: 29 Oct 2017].

[4] Netresec. (2017). Publicly available PCAP files. [online] Available at: http://www.netresec.com/?page=PcapFiles [Accessed 29 October 2017].

[5] Pleijsier, E. (2015). SCADA Networks. [online] Available at: http://referaat.cs.utwente.nl/conference/18/paper/7382/towards-anomaly-detection-in-scada-networks-using-connectionpatterns.pdf [Accessed 29 October 2017]

[6] DongHo, K. (2014). Whitelists Multiple Filtering. [online] Available at: https://www.hindawi.com/journals/jam/2014/597697/ [Accessed 28 October 2017]

[7] Ramos, R. (2015). Flow Whitelisting in SCADA Networks. [online] Available at: https://ris.utwente.nl/ws/portalfiles/portal/6147728 [Accessed 28 October 2017]

[8] En.wikipedia.org. (2017). Modbus. [online] Available at: https://en.wikipedia.org/wiki/Modbus [Accessed 19 Sep. 2017].

[9] Kacer, M. (2017) Anomaly Detection. [online] GitHub Repository, Available at: https://github.com/H21lab/Anomaly-Detection [Accessed 4 Dec. 2017].