http://dx.doi.org/10.14236/ewic/ICS2016.4

A Practical flow white list approach for SCADA systems

Antoine Lemay
École Polytechnique de Montréal
antoine.lemay@polymtl.ca

Jonathan Rochon
École Polytechnique de Montréal
jonathan.rochon@polymtl.ca

José M. Fernandez
École Polytechnique de Montréal
jose.fernandez@polymtl.ca

Abstract: The blatant vulnerability of industrial control systems, including those controlling critical infrastructure, is now well known. There is a need for immediately applicable security solutions that do not interfere with normal operations. Intrusion detection through flow white listing is an approach that can detect multiple components of modern attacks such as pivoting and command and control channels. However, the white list approach is not compatible with current black list-based IDS technology. This paper presents a practical approach for implementing flow white listing in SCADA system. The approach extracts a flow white list from a known good packet capture and inverts the decision logic to programmatically generate a rule set that can be consumed by a black list-based IDS. A performance evaluation shows that the approach is viable for SCADA systems, where the number of communication pairs is limited and traffic is mostly deterministic.

Keywords: white listing, critical infrastructure protection, SCADA networks.

1. INTRODUCTION

Since the discovery of Stuxnet (Falliere-2011), the spotlight has been directed to the security of Industrial Control Systems (ICS) and Supervisory Control And Data Acquisition (SCADA) systems. That light has revealed that the overall security posture of those system is lacking, with rampant vulnerabilities, lack of any form of authentication or integrity validation and minimal security literacy on the part of network operators. Considering that these systems control multiple components of critical infrastructure (CI) such as electric grids, water treatment and aqueduct, manufacturing, oil and gas and so on, the general lack of security is worrying. In that sense, there is a need for simple high impact solutions that can be implemented by SCADA network operators.

However, due to operational constraints, there are only a limited number of levers which can be used to enhance security. For example, the requirement for 24/7/365 operation makes the regular application of software patches more complex. Alternatively, the SCADA system integrator, on which many owners rely for maintenance and support, may not allow a change in the network configuration. This means that security solutions which have little to no interaction with deployed systems should be prioritized. One such solution is network security monitoring (NSM). In particular, automated detection of network intrusions would allow SCADA system operators to quickly respond to incidents and minimize the potential impact on the physical systems, which are often CI.

To implement a NSM program, defenders must have a good idea of what is legitimate and what is illegitimate. So, providing defenders of SCADA networks the ability to create a flow white list, i.e. a list of explicitly allowed network flows, for NSM could significantly increase their defensive capabilities immediately.

This paper presents a practical approach for implementing flow white listing in SCADA systems. It starts by reviewing the background of white listing approaches and presenting relevant related work. Then, it offers the details of the suggested approach and a performance evaluation of that approach. Finally, the paper summarizes the main findings and provides avenues for further research in a brief conclusion.

2. BACKGROUND

Of the threats facing SCADA networks, the threat of Advanced Persistent Threat (APT) actors is one of the most dangerous. While conventional malware and script kiddies can still affect some SCADA systems due to their high exposure, most SCADA systems sit behind corporate firewalls and remote access gateways. While these protection measures can deter and prevent these unfocused threats, they cannot prevent intrusions highly motivated and resourceful attackers, like APT

actors. As such, it is important to detect and respond to APT attacks at the earliest opportunity, before the network is fully compromised.

For example, it is typical for an attacker to acquire an initial point of presence in the network from spear phishing, water holing or providing infected media. To reach their target from the initial point of compromise, the attackers must expand their presence within the network. This task is called pivoting. To perform this task, the attacker must perform reconnaissance (e.g. port scanning) and send exploits between peers within the same network. If the network artefacts from these actions could be detected, the attack would reveal his presence and be kicked out of the network.

Similarly, once an attacker has achieved a point of presence or expanded his presence, it is necessary for him to establish communication paths between himself and the machines he has compromised to be able to realize his goals. If these communication paths, called command and control channels, could be detected, it would be another opportunity to find and remove the hacker's presence from the network.

In their top 4 mitigation strategies to protect information and communication technologies against targeted attacks by APT actors (Australian Signals Directorate-2012), the Australian Signals Directorate identifies application white listing as the most secure approach. The idea behind this recommendation is that it severely limits the attacker's ability to execute arbitrary code on the machine. The same reasoning applies to network security monitoring: if it were possible to create a white list of approved network communication paths, it would limit an attacker's capability to perform critical tasks like pivoting or establishing command and control.

In traditional I.T. systems, it is not possible to completely enumerate the list of allowable communication paths. A large portion of the traffic occurring in I.T. networks is the result of human actions. Employees need a broad range of applications and access to numerous different servers to perform their tasks. These tasks are performed whenever is convenient to the employees. Furthermore, the contents of an I.T. network are constantly changing due to dynamic addressing, constant patching and routine evolution of the network to fit the business needs. This makes the creation and maintenance of a white list of traffic flows impractical at best. So, I.T. networks must rely on black lists, which require constant updating to keep with the latest attacks and which perform badly against previously unknown patterns.

On the other hand, SCADA networks are very stable. Unless there is a significant change in the wiring of the network, SCADA traffic follows deterministic patterns. As evidenced by the related work, SCADA network use fixed IP addressing schemes, usually follow a master-slave architecture and feature mostly machine-to-machine communications. When human communication is involved, it is done through very specific paths, such as the use of a human machine interface (HMI) from an operator workstation to a master terminal unit (MTU) or from an engineering workstation to a controller.

The consequence of this these deterministic patterns is that, while the network physically could allow fully meshed communications as is the case in traditional I.T. networks, communications occur only through well defined communication pairs. For example, the operator workstations will communicate with the MTU through the port used by the HMI application, the MTU will communicate with the controllers on the SCADA protocol's port and the engineering workstation will communicate on the controllers with the port required to update the controllers' configuration (e.g. tfpt). Some other communication, such as default windows communication to support Workgroup or domain networking, may occur, but only in a machine to machine context. This creates a limited set of traffic patterns that can be used as a white list as shown by (Barbosa-2013). This white list would require little maintenance because it is based on expected traffic, which seldom changes, instead of attack patterns, which always change.

The ideal solution to perform flow white listing for intrusion detection would be to enter a list of permitted flows in an IDS designed to reason on white lists. All the flows entered would be considered legitimate, and all packets not parts of one of these flows would be considered anomalous. However, there are currently no product that implements this approach. Only black listing approaches, where anomalous is explicitly defined and all traffic not explicitly anomalous is normal, are implemented. As such, there is a need for an adaptation of the white list approach to a black list paradigm.

3. RELATED WORK

A number of researchers have worked on intrusion detection in SCADA systems. This section presents work done in that field that is relevant to the white listing approach.

In their work, Hadeli et al. (Hadeli-2009) have studied leveraging determinism for what they call anomaly detection and reliable security communication. In particular, they observe that the IP addresses and polling times have to be hard coded in the configuration files for the SCADA systems to work. This means that IP addressing is required to be static and that communication

should occur at predictable intervals. This so called determinism is leveraged to create a custom detection mechanism for missing packets as well as custom built firewall rules. However, their reliance on SCADA configuration files alone does not capture the entirety of the traffic on SCADA networks. Other traffic, such as the human machine interface (HMI) flow between operator workstations and the master terminal unit (MTU) also needs to be included in a white listing approach.

In his work on implementing a network behavior-based IDS, Langill (Langill-2011) goes a little further in the analysis by analyzing the communication patterns based on each node's role in the network. In that sense, he incorporates traffic not covered by Hadeli et a.. However, once the complete list of flows is extracted from the network and the visualization is produced, the IDS rule set is generated by hand. Furthermore, the rule set is generated to explicitly detect known communication paths, such as servers initiating connections to clients, rather than detect everything that is not part of a known path. The limited automation and the expert knowledge-based black listing approach limits the impact of this research for SCADA network operators, who may not be sufficiently proficient in security.

Another possible approach to white listing SCADA communications is the specification-based approach. The work of Cheung et al. (Cheung-2007) on model-based intrusion detection for SCADA networks is an example of such an approach. By creating a model of normal communications by protocol inspection and then validating that the traffic does not violate that model, enables them to detect any communication not specifically allowed by the protocol. However, as with Hadeli et al., this approach does not cover the entire range of protocols present in SCADA networks. Furthermore, this approach requires in-depth knowledge of the protocol's specification. This knowledge might not be available for many proprietary protocols.

Furthermore, the work of Lemay (Lemay-2013) shows that full protocol inspection may not be required to detect many attacks. He notes that many modern attacks require lateral propagation inside networks and the maintaining of a command and control channel. These requirements generate periodic traffic flows that do not follow the heavily constrained characteristics of traditional SCADA traffic. However, he limits himself to SCADA protocols and does not cover all the other traffic flows in the network. Furthermore, while the suitability of a white list approach is alluded to, it is left for future work.

The work of Barbosa et al. (Barbosa-2013) offer the only true flow white listing approach. They take the traffic captures from water treatment plant analyzed in (Barbosa-2012), and extract the various flows within to create a white list intrusion detection system. To do so, they learn existing flows from a capture of the network traffic. Once the learning phase is complete, their program detects any flow that is not part of the list and marks it as anomalous. Their results show the validity of the white list approach, both in terms of viability and in terms of scalability for SCADA networks. Their results, based off packet captures of production networks, show a list of around 40 host pairs in the water control network, 20 in the water field network and 542 in the electric-gas network. Compared with the 56 billion communication pairs on the university network. However, their solution relies on a custom-built classifier to implement the white listing approach. To apply their results, current defenders of SCADA networks would not be able to leverage their existing intrusion detection setups.

In contrast, the works of Rusu et al. (Rusu-2013) and Genge et al. (Genge-2014) on SPEAR also implement a form of white listing, but using a black list engine to perform the detection. This is achieved by building a model of the ICS network, including a specification of all the flows. Such an approach provides a significant advantage over true white listing approach as it does not require the development of a new IDS engine. However, while the proposed approach automates the generation of rules for intermediary nodes, it does not enable the automated extraction of flows. This limits the ability of the system to be scaled to production environments.

4. A PRACTICAL FLOW WHITE LISTING APPROACH

If the flow white listing approach could leverage existing black list-based IDS and automate the rules generation, defenders of SCADA network could reap the benefits right away. This section presents a practical flow white listing approach that extracts existing flows, representing a flow white list, and generates automatically a black list covering all flows not part of the white list.

Let us consider a simple SCADA consisting of one MTU (A:192.168.0.100) and two controllers (B:192.168.0.101, C:192.168.0.102). The MTU can establish a connection to the controllers on the Modbus port (502) and all machines can establish connections to one another for Workgroup networking (port 139). These flows are considered legitimate while all other flows are not. The range of allowable communications is summarized in Table 1.

***Table 1:** Allowable communications*

Source	Destination	Port	Legitimate?
192.168.0.100 (A)	*192.168.0.101 (B)*	*502*	*Yes*
192.168.0.100 (A)	*192.168.0.102 (C)*	*502*	*Yes*
192.168.0.100 (A)	*192.168.0.101 (B)*	*139*	*Yes*
192.168.0.101 (B)	*192.168.0.100 (A)*	*139*	*Yes*
192.168.0.100 (A)	*192.168.0.102 (C)*	*139*	*Yes*
192.168.0.102 (C)	*192.168.0.100 (A)*	*139*	*Yes*
192.168.0.101 (B)	*192.168.0.102 (C)*	*139*	*Yes*
192.168.0.102 (C)	*192.168.0.101 (B)*	*139*	*Yes*
Other flows		*No*	

Let us say that each legitimate flow is represented by a rule R in a white list. The decision of the intrusion detection engine could be summarized as follow:

$$R1 \vee R2 \vee ... \vee Rn \rightarrow Pass \quad (1)$$

However, because we have to rely on a black list IDS engine, we need a reasoning that would instead look like the following:

$$R'1 \vee R'2 \vee ... \vee R'n \rightarrow Alert \quad (2)$$

In other words, if a flow matches the definition of any flow not in the allowed flows, then generate an alert. In all other cases, the engine lets it pass. To reach this new reasoning state, we can invert the logic presented in equation 1.

$$\neg(R1 \vee R2 \vee ... \vee Rn) \rightarrow \neg Pass \quad (3)$$

Which can also be presented as follows:

$$\neg R1 \wedge \neg R2 \wedge ... \wedge \neg Rn \rightarrow Alert \quad (4)$$

We cannot directly use this representation as a rule set because of the conjunction operator between the rules. IDS engines match the entire content of each rule and match each rule independently. As such, we are looking for conjunction operators between the conditions of the rules and disjunction operators between the rules. To reach this point, we can further distribute the negation operators. A rule based on a flow white list would look like the following rule:

$$IP\_src = 192.168.1.100 \wedge IP\_dst = 192.168.1.101 \wedge Port\_dst = 502 \quad (5)$$

Replacing the source IP clause by the statement A, the destination IP clause by the statement B and the port clause by the statement C and distributing the negation operator, we obtain the following rule.

$$\neg A \vee \neg B \vee \neg C \quad (6)$$

We can then express equation 4 as follows:

$$(\neg A1 \vee \neg B1 \vee \neg C1) \wedge (\neg A2 \vee \neg B2 \vee \neg C2) \wedge ... (\neg An \vee \neg Bn \vee \neg Cn) \rightarrow Alert \quad (7)$$

We can perform the conjunction and to rearrange the equation as a series of disjunctions:

$$(\neg A1 \wedge \neg B2 \wedge \neg A3 \wedge ... \wedge \neg An) \vee (\neg A1 \wedge \neg A2 \wedge \neg B3 \wedge ... \wedge \neg An) \vee ... \vee (\neg C1 \wedge \neg C2 \wedge ... \wedge \neg Cn) \rightarrow Alert \quad (8)$$

This equation, where a series of conjunctions are separated by disjunction operators could be used as-is as a rule set for a black list IDS. However, we can make a number of simplifications. For example, if multiple flows have the same source IP, for example if A1 equals A2, it is possible to regroup some of the propositions. Similarly, if all but a single destination IP are disallowed, it is possible to express that rule as a positive statement rather than a negative statement.

There are multiple ways to reduce the number of rules. We settle on the following: any flow that comes from an unknown source or goes to an unknown destination is considered anomalous. Any flow that comes from a known source and goes to an unusual destination for that source or that goes to a known destination, but does not come from one of the usual sources should also raise an alarm. Finally, any flows that comes from a known IP and goes to a usual destination, but uses an abnormal port would also be considered abnormal. Naturally, as the number of flows grow, it quickly becomes impractical to perform this transformation by hand. However, this task is easy to automate.

In order to automate the process, we would normally require a description of known good network flows. However, there is usually no easily importable documentation of those flows. So, the observed flows in the network will be considered legitimate flows. To obtain those flows, a packet capture or flow summary record, such as netflow records, would be used as a representation of legitimate traffic. This representation would go through a flow extractor that would extract the full list of flows present in the network. This list would then be reduced to a unique list of 3-tuples containing the source IP, the destination IP and the destination port number. The list is then stored in an intermediary representation that can be validated to make sure no non-legitimate flows are present. Once the list is validated, this represents the white list of flows in the network. The white list is then passed through a script that inverts the detection logic as shown previously and generates

a SNORT rule set that raises an alarm if the flow is not part of the white list.

5. PERFORMANCE EVALUATION

This section presents the performance evaluation of our practical flow white list approach. It starts by describing the implementation used in the performance evaluation. Then the experiment design is presented. Finally, the results and the corresponding analysis is offered.

5.1 Implementation

To test the performance of our approach, an implementation of the rule set generator was created based on the design presented in section 3. The tshark command-line utility (Wireshark. Tshark-2015) was used as the flow extractor. For each TCP packet with a SYN flag present but no ACK flag present (the first packet of a 3-way handshake), tshark stored the source IP, destination IP and destination port in a CSV file. Then a python script parsed the CSV file to create a unique list of source IP, destination IP and destination port 3-tuple to create the white list. Then, the python script uses the unique list of 3-tuples to generate the SNORT rule.

The SNORT rules generation is done in five steps:

(i) The rule covering packets with an source IP that is not present as a source IP in the white list is generated;
(ii) The rule covering packets with a destination IP that is not present as a destination IP in the white list is generated;
(iii) The rules covering packets with a destination IP that is present as a destination IP in the white list, but where the source IP does not match that particular destination IP in any 3-tuple is generated;
(iv) The rules covering packets with a source IP that is present as a source IP in the white list, but where the destination IP does not match that particular source IP in any 3-tuple is generated;
(v) The rules covering packets matching a specific source and destination IP pair, but where the destination port does not match.

This creates a mutually-exclusive SNORT rule set that covers all packets that are not part of the white list.

In this particular implementation, the number of rules is a function of possible communication pairs. Step (i) and step (ii) create a single rule each. Step (iii) and step (iv) create a number of rules based on the number of destination IP and source IP respectively. So, in a network where 20 machines can act as sources and 4 machines can act as destinations, the total number of rules created by steps (iii) and (iv) would be 24. Step (v) creates a number of rules equal to number of IP communication pairs. So, going back to our previous example with 20 sources and 4 destinations, we might have 70 unique source IP, destination IP 2-tuples. This would generate 70 rules.

Using this approach, we can calculate an upper bound on the number of rules produced. If we have a network comprised of n machines that communicates with every other machine, step 1 and 2 would create one rule each, step 3 and 4 would create n rules each and step 5 would create n(n-1) rules. So, the rule set generation script would generate a total of n2 + n + 2 rules. If we want to compare the number of rules generated by the script with the number of rules of a more traditional rule set, we can take the Emerging Threats rule set (Emerging Threats Pro, LLC-2015) as an example. The ET rule set has over 21 000 rules in its rule set. This would create an upper bound of around 145 machines, given a fully meshed network and much higher if there were limited communication pairs as is the case in SCADA networks. Furthermore, a large number of rules in the ET rule set must search for content within the payload of the packet. This consumes more resources than rules based on information in the IP header like the rules produced by our tool. As such, while this approach clearly does not scale in traditional I.T. environment, it seems like a perfectly reasonable approach in a SCADA environment.

5.2 Experiment Design

To test the performance of our approach, we ran our implementation on a number of packet captures with varying sizes and varying number of endpoints. For each run, we recorded the time required to complete the extraction, the generation of the unique list and the generation of the rules and we also recorded the number of rules generated by each run. The time metric will allow us to estimate the feasibility of applying our approach to a SCADA network at scale and allow us to identify any performance bottlenecks in our approach. The second metric will allow us to gauge the distance between our estimated higher bound on the number of rules and the reality of SCADA network. This will serve as a proxy for the ability of our system to scale up to production level networks.

To generate the data sets, a number of SCADA configurations were implemented in a SCADA sandbox (Lemay-2013) and the traffic was recorded. Among the configurations, a number of Modbus configurations built using ScadaBR (Sourceforge-2014) and Modbus-tk (Jean-2014)

were used. Additionally, multiple packet captures from a single configuration of DNP3 using GE GenE software (General Electric-2011) was used. In the Modbus configuration, the length of the capture was fixed at one hour, but the number of controllers and the polling time were modified to alter the number of communication pairs and the size of the packet capture.

To supplement the SCADA traffic data sets with more challenging packet captures, we also used our rule set generator on more traditional I.T. datasets. While the white listing approach does not seem appropriate in such a context, these datasets are much larger and include more communication pairs than our SCADA datasets. This allows us to test the performance at a larger scale. In particular, we used the DARPA 1999 inside and outside datasets (Lincoln Laboratory-2014) and the DFIR Monterey 2015 Forensics challenge dataset (SANS DFIR-2014).

The tests were run on a commodity off-the-shelf laptop. Faster results could be achieved with more computing resources.

5.3 Results

The results were compiled and are summarized in Table 2.

Table 2: *Performance results*

Data Set	Data Size (KB)	Δt (s)				Rules Generated
		Extraction	*Reduction*	*Gen.*	*Total*	
Modbus						
polling only	*4272*	*4.41*	*0.12*	*0.11*	*4.64*	*15*
2M, 3R, 2s	*24635*	*22.44*	*0.12*	*0.12*	*22.68*	*18*
2M, 3R, 10s	*6835*	*5.88*	*0.12*	*0.11*	*6.11*	*18*
2M, 3R, 15s	*5720*	*4.69*	*0.10*	*0.10*	*4.89*	*18*
2M, 6R, 10s	*1677*	*10.34*	*0.11*	*0.11*	*10.56*	*36*
2M, 8R, 10s	*14107*	*12.59*	*0.12*	*0.11*	*12.82*	*33*
2M,10R,10s	*1736*	*15.12*	*0.12*	*0.11*	*15.35*	*39*
2M,12R,10s	*19536*	*17.22*	*0.12*	*0.11*	*17.45*	*45*
DNP3						
Trace 1	*1118*	*1.26*	*0.11*	*0.11*	*1.48*	*15*
Trace 2	*1167*	*1.28*	*0.11*	*0.12*	*1.51*	*18*
Trace 3	*1197*	*1.29*	*0.11*	*0.11*	*1.51*	*15*
Trace 4	*1153*	*1.27*	*0.11*	*0.12*	*0.5*	*19*
DIMVA 1999						
inside	*333035*	*108.12*	*0.13*	*1.41*	*109.66*	*2995*
outside	*316242*	*104.27*	*0.14*	*2.17*	*106.58*	*2919*
2015 DFIR Network Forensics Challenge						
ftp-example	*35268*	*3.00*	*0.10*	*0.10*	*3.2*	*5*
nitroba	*55465*	*9.87*	*0.12*	*0.18*	*10.17*	*764*

Studying the results, we can observe that the largest portion of the time required to process the results is caused by the flow extraction method. Even in the case of traditional I.T networks, where a large number of rules are generated, the extraction method accounts for more than 98% of the total execution time. A more in-depth study of the completion time shows that the extraction time grows more or less linearly with the size of the packet capture and completely overshadows both the reduction time and the generation time. Figure 1 illustrates this relationship.

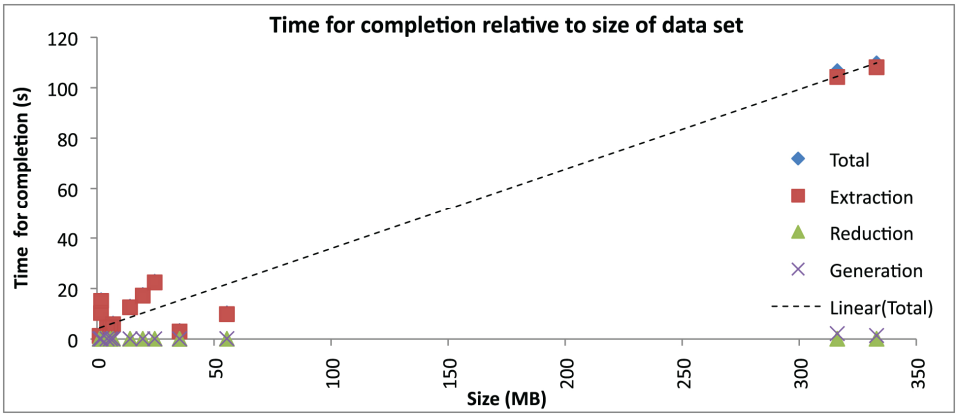

Figure 1: *Time for completion in relation to size of the data set*

Even considering the massive contribution of the flow extraction phase, which could be optimized or replace with a method based on flow summary records, the time for completion obtained represent an insignificant amount of time invested to produce the rule set. At worst, for the I.T. systems where there is a large volume of data to process and a large number of recorded flows, our solution completes in under two minutes. For SCADA systems, the solution completes in under 30 seconds. Taking into account that the rule generation script only needs to be run only when there is a major modification in the SCADA network, which seldom occurs unless the physical process is in constant flux, these completion time values seem reasonable.

In terms of scalability, we can see that the number of rules generated in the SCADA system is far fewer than the number of rules we could expect from our higher bound. In the 2 MTU,12 RTU and 10 seconds polling interval case (2M,12R,10s row), we have 14 nodes. Our higher bound on the number of rules for a 14 node full mesh network would be 212 and we get 45 rules. This would tend to indicate that our higher bound significantly exaggerates the number of communications in a SCADA network. In that sense, we could reasonably expect to scale up in the multiple hundreds of nodes, which is an order of magnitude more than the water treatment data sets and in the same order of magnitude as the electric-gas data set presented in (Barbosa-2013).

5.3 Validation

In order to validate the applicability of this approach in an industrial context, we obtained data from a real SCADA deployment. The data contained traffic from the production network of a large natural gas provider in Eastern Canada and was captured over the course of a few days. On average, the captures contains around 2.1 gigabytes of data per hour, for a total of around 300 gigabytes. This creates a sizeable sample to test the validity of our approach.

One of the most salient metric for usability of intrusion detection approaches is the number of false positives produced. If the number of false positives is too high, the alarms will eventually get ignored. As such, we will record the number of false positives generated by a rule set extracted from the production network.

Another important metric is the scalability of the rule set. As shown in Section 5, SCADA networks should have a smaller number of flows when compared to production networks. This should translate in a smaller number of rules. Therefore, we will also record the number of rules generated by our tool.

Because we are using data collected from a production network, it is not possible to fully know the ground truth. However, it is unlikely that this production system contains attack data. So, we will consider any alert generated as a false positive. We have also performed manual verification of some of the alarms to confirm that they are indeed false positives.

To perform the experiment, the data set was divided in data samples of various time increments (15 minutes, 30 minutes, 1 hour and 2 hours). This would simulate the time an analyst would take to set up a white listing solution in a production network. For each of these time increments, a data sample is chosen randomly to act as the "known-good" capture and a rule set is generated and the number of rules is recorded. We then run the intrusion detection process on all the other data samples to detect if there are any new TCP connections that were not identified in the rule set and record the number of alarms. We repeat this process 20 times to achieve statistically significant results (except for the 2h division where only 15 runs were successfully completed due to an unscheduled reboot). The results of the experiments are presented in Figure 2 and 3.

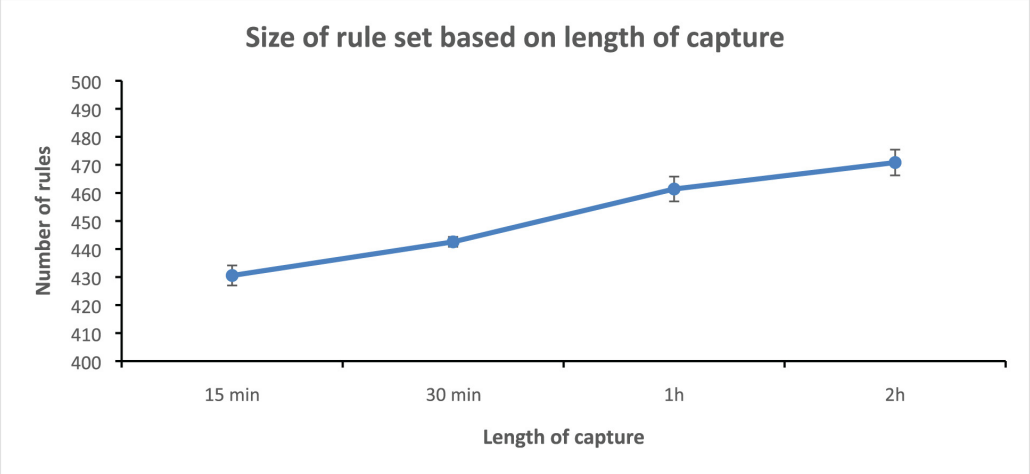

Figure 2: *Size of rule set based on length of capture*

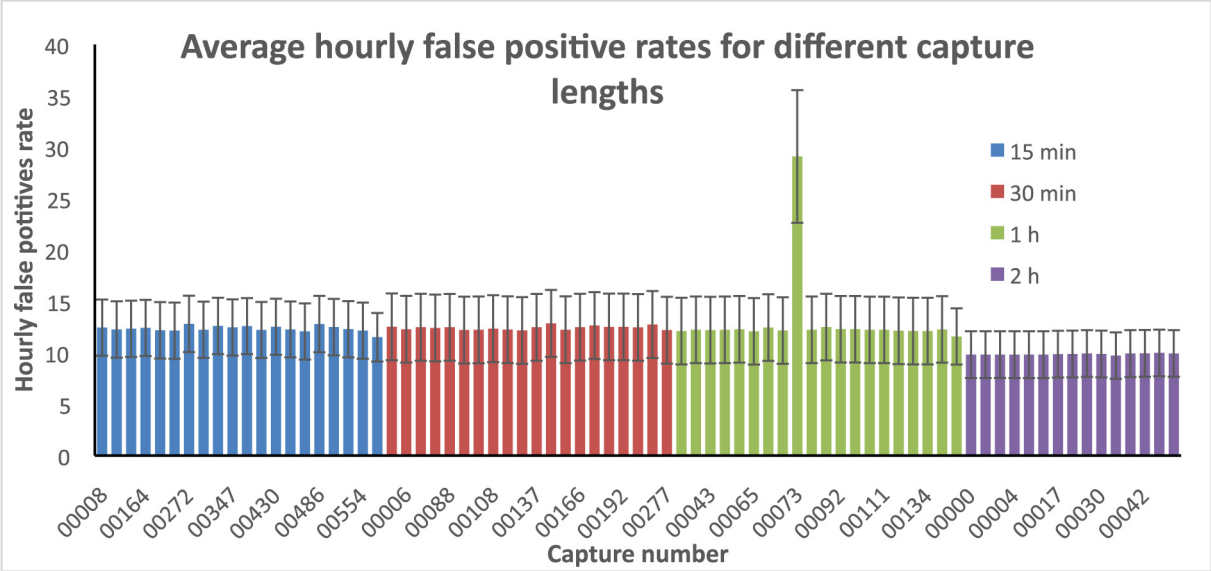

Figure 3: *Number of false positives based on length of capture*

In terms of the number of false positives, the numbers are still arguably high, with a rate of around 12 false positives per hour of traffic. However, the standard error on that value is relatively high. This represents the fact that a sizeable proportion of packet captures contain no alarms, but some generate a high number of alarms. This is mainly due to occasional activities (maintenance for example) that generate a lot of connections, but are not captured in the sample used for white list. A further study of the alarms generated also identify some of the low frequency SCADA communication as an important source of alarms, as a number of connections are made when these communication occur. The fact that the longer captures (2h) seem to incur less alarms seem to coincide with the capture of a greater range of behaviours.

This analysis suggests that, while the current number of false positives is relatively high, it could be rapidly brought down when alarms are resolved and new behaviour is captured.

6. CONCLUSION

In this paper, we presented a practical approach to implement a flow white list-based intrusion detection system for SCADA network. The proposed solution leverages existing black list-based IDS, such as SNORT, instead of duplicating the network monitoring infrastructure. This is achieved by creating a white list and inverting the decision logic to generate a black list covering any flow that was not included in the white list. The transformation can then be used to programmatically generate a rule set for a black list based IDS, such as SNORT.

This approach was successfully tested with a number of SCADA data sets. Results show that the approach is reasonably quick and presents a profitable one-time investment of effort. Further analysis has identified that the main bottleneck of the approach is the extraction of flows from packet captures. This is especially true for larger packet captures as the extraction time is proportional to the size of the pcap file. The results also show that the rule set produced in SCADA network contains a small number of rules, many orders of magnitudes smaller than rule sets that could be expected from I.T. systems. This shows promise for applicability in real deployments and was confirmed by a validation experiment using data from an actual SCADA deployment.

Additional research could further enhance the proposed approach. In particular, addressing the flow extraction bottleneck problem could significantly reduce the time required for completion, and enhance scalability to larger systems or increase fidelity of the white list by being able to record legitimate traffic for longer periods of time. Currently, the extraction method only covers TCP flows. Adding UDP flow extraction, more difficult because of the lack of a handshake, would address this limitation. Finally, a study of how to further exploit intermediary representations produced by this practical approach, for example by leveraging them for automated compliance, also seems like a viable path forward.

7. ACKNOWLEDGMENTS

This research was funded in part by the Canadian Center for Security Science (CSS). In addition, we would like to thank the National Energy

Infrastructure Test Center (NEITC) for additional testing and support and our industrial partner for providing access to the data.

REFERENCES

Falliere, N., Murchu, L. O., and Chien, E. (2011) W32.Stuxnet Dossier Version 1.4. Symantec Security Response.

Australian Signals Directorate. (2012, Nov.) Top 4 Mitigation Strategies to Protect Your ICT System. Available from http://www.asd.gov.au/publications/protect/top_4_mitigations.htm. [Accessed 6 May 2015].

Barbosa, R. R. R. et al. (2013) Flow whitelisting in SCADA networks. In: *Seventh Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*, Washington.

Hadeli, H. et al. (2009) Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration. In: *IEEE Conference on Emerging Technologies & Factory Automation, 2009. ETFA 2009*, Mallorca.

Langill, J. (2011) Implementing a Network-based Intrusion Detection System for Control Systems. In *ICSJWG 2011 Fall Conference*.

Cheung, S. et al. (2007) Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium*, Miami Beach.

Lemay, A. (2013) Defending the SCADA network controlling the electric grid from advanced persistent threats. École Polytechnique de Montréal.

Barbosa, R. R. R. et al. (2012) A First Look into SCADA Network Traffic. In: *Network Operations and Management Symposium (NOMS)*, Maui.

Rusu, D. A., et al. (2013). SPEAR: A Systematic Approach for Connection Pattern-based Anomaly Detection in SCADA Systems. In: *The 7th International Conference Interdisciplinarity in Engineering, INTER-ENG 2013*, Romania.

Genge, B. et al. (2014) A Connection Pattern-based Approach to Detect Network Traffic Anomalies in Critical Infrastructures. In: *EuroSec'14*, Amsterdam.

Wireshark. tshark - The Wireshark Network Analyzer 1.12.2. Available from https://www.wireshark.org/docs/man-pages/tshark.html. [6 May 2015].

Emerging Threats Pro, LLC. (May, 5 2015). Emerging Threats.net Open rulesets. Available from http://rules.emergingthreats.net/. [6 May 2015].

Lemay, A. (2013). An isolated virtual cluster for SCADA network security research. In: *1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013)*, Leicester.

Sourceforge. (2014, 16 Dec.) Sourceforge project - SCADA BR. Available from http://sourceforge.net/projects/scadabr/. [Accessed 26 January 2015].

Jean, L. (2014, 3 Nov.) modbus_tk 0.4.3. Python Software Foundation. Available from https://pypi.python.org/pypi/modbus_tk. [Accessed 6 May 2015].

General Electric. (2011, Jan.) GENe Software Suite. Available from http://www.gedigitalenergy.com/products/brochures/uos/GENeSoftwareSuite.pdf. [Accessed 8 April 2013].

Lincoln Laboratory, M.I.T. (2014). 1999 DARPA Intrusion Detection Evaluation Data Set. Available from http://www.ll.mit.edu/ideval/data/1999data.html. [Accessed 6 May 2015].

SANS DFIR. (2014, 5 Dec.) SANS Digital Forensics and Incident Response Blog. DFIR Monterey 2015 Network Forensics Challenge. Available from https://www.dropbox.com/s/8mtz0m1gd32nk0m/2014-11%20DFIR%20Network%20Forensics%20Challenge.zip?dl=0. [Accessed 6 May 2015].