Exploiting Bro for Intrusion Detection in a SCADA System

Robert Udd
Sectra AB
Linköping
Sweden

Mikael Asplund
Dept. Comp. and Inf. Sci.
Linköping University
Sweden
mikael.asplund@liu.se

Simin Nadjm-Tehrani
Dept. Comp. and Inf. Sci.
Linköping University
Sweden
simin.nadjm-tehrani@liu.se

Mehrdad Kazemtabrizi
Ind. Inform. and Cont. Syst.
The Royal Institute of Technology
Stockholm Sweden
mkaz@kth.se

Mathias Ekstedt
Ind. Inform. and Cont. Syst.
The Royal Institute of Technology
Stockholm Sweden
mathias.ekstedt@ics.kth.se

ABSTRACT

Supervisory control and data acquisition (SCADA) systems that run our critical infrastructure are increasingly run with Internet-based protocols and devices for remote monitoring. The embedded nature of the components involved, and the legacy aspects makes adding new security mechanisms in an efficient manner far from trivial. In this paper we study an anomaly detection based approach that enables detecting zero-day malicious threats and benign malconfigurations and mishaps. The approach builds on an existing platform (Bro) that lends itself to modular addition of new protocol parsers and event handling mechanisms. As an example we have shown an application of the technique to the IEC-60870-5-104 protocol and tested the anomaly detector with mixed results. The detection accuracy and false positive rate, as well as real-time response was adequate for 3 of our 4 created attacks. We also discovered some additional work that needs to be done to an existing protocol parser to extend its reach.

Keywords

Anomaly detection, IDS, Bro, SCADA, IEC 60870-5-104

1. INTRODUCTION

The electrical grid is becoming increasingly dependent on information infrastructures and services. These allow greater flexibility to accommodate alternative energy sources, lower cost of operation and potentially improved resilience. For reasons of compatibility and cost modern Supervisory Control and Data Acquisition (SCADA) systems now use Internet protocols for communication within as well as between substations. Moreover, SCADA systems are increasingly being connected to the Internet to allow remote operation, software updates and system monitoring.

In order to protect these critical cyber-physical systems (CPS) from unauthorised access, they are often separated from normal Internet traffic through firewalls, VPN systems, and in some highly-critical cases using physically separated networks. However, most security experts agree that security should be constructed using a *defence in depth* approach. Among others, the standards developed for management of power systems propose specific measures for monitoring network protocols and end system health status[1].

One reason for this is that SCADA operated networks cannot be considered as fully isolated from social engineering induced malware proliferation and insider threats [17]. A recent study by Volmetrics shows that only 11% of the 800 businesses interviewed considered their systems resistant against insider threats (by privileged users, contractors and service providers, and business partners)[2]. Another reason is that the added complexity posed by a mix of modern and legacy systems leads to benign software-dependent misbehaviour and the traditional security mechanisms are often targeted towards intentional breaches.

The combination of new protocols and sensors to traditionally closed SCADA systems requires a new way of approaching the defence in depth idea in this context. As new standards appear and interoperable network technologies emerge, a need to augment the existing monitoring capabilities in an efficient manner arises. Platforms that facilitate rapid deployment of modular incident detection techniques are valuable, and here we explore one such platform for building our anomaly detection mechanism.

In this paper we propose an intrusion detection mechanism for SCADA systems. It is intended to operate on the internal network of an electrical substation and is based on anomaly detection. Anomaly detection (as opposed to misuse detection) has the benefit of being able to detect previously unknown attacks. The main drawback of the anomaly detection approach is the potentially large number of false

CPSS'16, May 30-June 03 2016, Xi'an, China

ISBN 978-1-4503-4288-9/16/05…$15.00

DOI: http://dx.doi.org/10.1145/2899015.2899028

[1]IEC/TS 62351-7 Ed.1: Power systems management and associated information exchange – Data and communication security, Part 7: Network and system management (NSM) data object models

[2]2015 Voltometric Insider Threat Report: http://www.vormetric.com/campaigns/insiderthreat/2015/

alarm coming from benign traffic that just does not match the trained normality model of the detector. However, in closed SCADA systems, the traffic pattern does not change as frequently as in enterprise networks.

Our proposed intrusion detection component combines two anomaly detection mechanisms, automatic whitelists and a statistical model of packet timing characteristics. The system has been implemented as components in the state-of-the art intrusion detection framework Bro [14]. This modular framework has support for a wide range of existing network protocols. Moreover, we have extended Bro with a parser for the IEC 60870-5-104 SCADA protocol in order to test the system on a small-scale test network with real Remote Terminal Units (RTUs).

We evaluate the performance of the detector by constructing four different attacks (two sniffing attacks, one man-in-the-middle attack and one spoofing attack). The results are show some potential with three out of the four attacks perfectly detected with an acceptable amount of false positives.

In summary, the contributions of the paper are threefold:

- An automatic whitelisting anomaly detector for SCADA networks
- A Bro parser for the IEC 60870-5-104 protocol
- Implementation of four SCADA network attacks, three generic attacks and one specific for the IEC 60870-5-104 protocol

The rest of the paper is organised as follows. Section 2 discusses related work. Section 3 describes the intrusion detection components that we add to the Bro framework. Section 4 contains a description of the evaluation we performed on a small SCADA testbed using real hardware components. Finally, Section 5 concludes the paper.

2. RELATED WORK

Several recent papers show that opening up the SCADA networks to external devices enables adversaries to perform attacks on the networks (that obviously relate to critical services). Among the protocols that appear in the context of SCADA networks we find Modbus, DNP3, and IEC-60870-5-104. Several recent works address attacks and countermeasures for networks operating these protocols.

To mention a few, a man-in-the middle attack on the IEC-60870-5-104 protocol was described by Maynard et al. [12]. Further work by Yang et al. [16] lists eight known attacks on the IEC-60870-4-105 communication and builds a signature-based defence approach on these. Other work describes attack step sequences that include crafting legitimate but malicious DNP3 packets so that 4 circuit breakers can be opened simultaneously in a 30 bus network [10]. Hoyos et al. [8] describe a message authentication attack on a network operating with the IEC-61850 standard and running the GOOSE protocol. Several examples of attacks feasible on Programmable Logic Controllers (PLCs) that run a Modbus protocol are described in the literature [7]. Moving on to smart grid infrastructures we see descriptions of potential attacks on smart grids and how the dynamics of such networks differ in terms of timing characteristics compared to traditional ones [15]. The study of intrusion detection approaches is typically performed using data that is not available for comparative purposes [13]. Neither is the code for any other anomaly detector for the IEC-61870-5-104 protocol available to the best of our knowledge. Thus, it is difficult to compare one approach against another. We are only aware of one other work in collaboration with industrial partners [1] that has datasets based on the same protocol as ours, but to the best of our knowledge it does not make the dataset available for testing. Common data sets and available security building blocks would promote a more systematic approach to anomaly detection in SCADA systems.

Altogether, these analyses indicate that a serious look at risks is needed before an unprotected device is embedded in potentially sensitive contexts. However, the residual risks may not be easy to mitigate as we indicate below.

A basic problem that has to be resolved with respect to securing networks extended with new CPS devices and protocols is the question of resource efficiency for security building blocks.

Although studying the cost of hardware designs for security is subject to meticulous studies (see e.g. Good and Benaissa [6]), the resource costs of adding security implemented in software is much less studied. A relatively early study of the resource costs was in the context of adding security mechanisms to tactical networks [11]. Lake et al. [9] state that the "the biggest challenge from the device side is that a lot of M2M/IoT devices do not have enough capability to do the encryption on the device."

One of the most well-known and well-used intrusion detection mechanisms is Snort with rules dynamically updated to recognise tens of thousands of adverse conditions. Chang et al. [2] compare the RAM usage benchmarking of Snort that at peak rate shows a 1.2 GB memory usage with the 512MB of RAM in a Raspberry Pi computer. They then describe the steps taken to enable lightweight intrusion detection by implementing memory-efficient representations of Snort rules and CPU-efficient algorithms to work in real-time in tactical devices. Any new mechanism that addresses adding new devices to a network needs to enhance the arsenal of defence mechanisms that work in the resource-constrained settings.

A timing attack on the IEC-61850-8-1 authentication mechanism [8] shows that since the computation capacity of embedded processors for running an authentication algorithm currently exceeds the needed 4ms response time, a successful attack would be able to create an automation breakdown, including damaging circuit breakers and power transformers.

A recent survey of IoT technologies [5] includes some exposure to how IoT technologies (device management, wireless connectivity, protocols) address security issues. In general, no IoT-specific security mechanisms with well-understood resource footprints are currently available.

Our work shows that a) the Bro framework is runnable on a Raspberry Pi and can be inserted in adequate numbers to monitor segments of a SCADA network. The Hilti parser that we use is not yet ported to the small footprint platforms, but our experiments on the standard computer show that the detection performance is fast enough for the network sizes specified. Despite its current limitation our parser can be made available for further improvements by the research community.

3. DETECTING ANOMALIES

Our approach is built using two mechanisms: The automatic whitelisting (AW) anomaly detector and timing analy-

sis. The system consists of three main parts that are all integrated in the Bro framework. First the parser ensures that traffic using the IEC 60870-5-104 protocol can be treated in Bro, second, a learning component that automatically creates a number of whitelists of normal traffic as well as collecting timing statistics during some training period of the system, and finally a detection component that uses whitelists and timing statistics to flag for anomalous traffic. We now proceed to describe the rationale and basic three components.

3.1 IEC 60870-5-104 Parser

The IEC 60870-5-104 protocol parser has been implemented using the Spicy parser generator and a Bro plugin. This framework allows an analyser to be constructed in Bro by specifying a protocol grammar and a number of event hooks that triggers a Bro event handler. We now proceed to describe the basic building blocks of the IEC 60870-5-104 protocol, that correspond to the parser structure. Figure 1 shows the basic frame format of the Application Protocol Data Unit (APDU) frame as it is defined in the IEC 60870-5-104 protocol. The Application Protocol Control Information (APCI) is always present in in the frames, but the Application Service Data Unit (ASDU) is optional [3]. An APDU frame can be in U, S or I format. The unnumbered control format (U) is used as a start and stop mechanism for communication flows. The supervisory format (S) controls the transport of APDUs. Finally, the information instruction format (I) is used for APDUs that contain an ASDU and carry the actual SCADA instructions and information items.

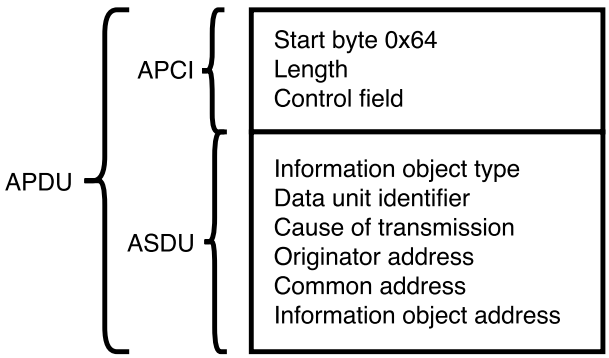

Figure 1: Frame format for IEC 60870-5-104 packets

The parser is built in a hierarchical design with each frame format represented with a dedicated module. The parser has full support for the entire IEC 60870-5-104 protocol specification.

3.2 Learning component

Whitelists are commonly used to ensure that only known and approved traffic is allowed in the network, something that can be implemented using manually created firewall rules. However, rule creation can be a tedious and error prone task. The learning component therefore creates whitelists automatically based on a data from a training period. In addition to these whitelists we consider the use of traditional statistical anomaly detection techniques applied to timing characteristics in the network to further increase the detection ability of the system.

3.2.1 Whitelist creation

Figure 2 shows the basic functionality of the learning component when processing an incoming packet. The detector maintains three separate whitelists (1) ARP whitelist, (2) pair communication whitelist and (3) a TCP control whitelist. The ARP (Address Resolution Protocol) whitelist is built up from ARP packets where MAC addresses and IP addresses are paired together. This ensures that only known hosts are participating in the network. The pair communication whitelist contains information of sender-receiver node pairs. This list can be populated both by ARP packets and by IEC60870-5-104 packets. In the latter case, the type of APDU-packet that is being sent (S,U, or I), and in the case of an I-type, what type of instruction is also recorded. Finally, the TCP control whitelist records the IP-number and port number for all TCP packets that are not IEC 60870-5-104 packets (e.g., acknowledgement packets). This ensures that only the intended connections patterns are allowed. That is, if computer node A normally connects to service S on node B, then if A would suddenly try to access service S' on B or to access node C, an alarm would be raised.

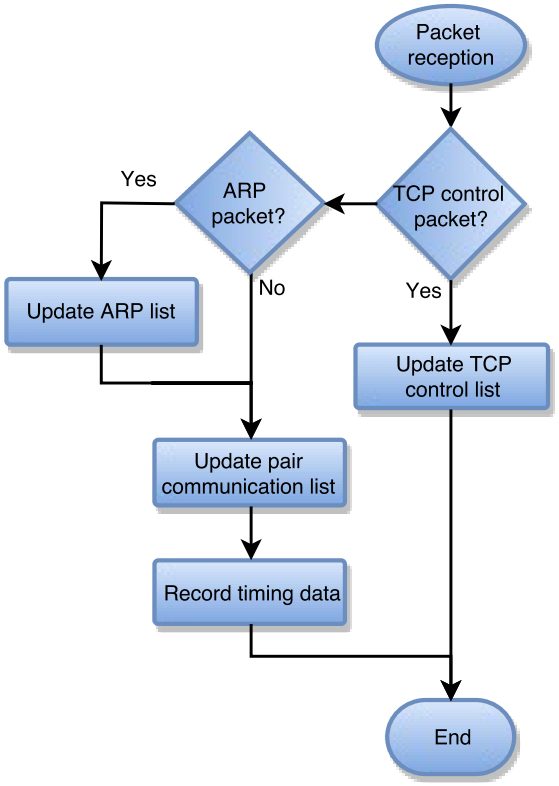

Figure 2: Flowchart of the learning component.

As can be seen in the flowchart, the whitelists are constructed by adding a new item every time a packet appears that does not match any existing item in the whitelists. Moreover, for the ARP and IEC 60870-5-104 packets the timing characteristics are stored in a separate record for

making statistical analysis, as we will now describe in the following.

3.2.2 *Timing statistics*

The purpose of keeping timing statistics is to be able to detect deviations of packet inter-arrival times. Figure 3 illustrates the basic idea of how such timing detection is supposed to function. The first three packets arrive in a normal fashion, whereas the packet arriving at time T_4 arrives more than a threshold D later than what is considered normal by the model

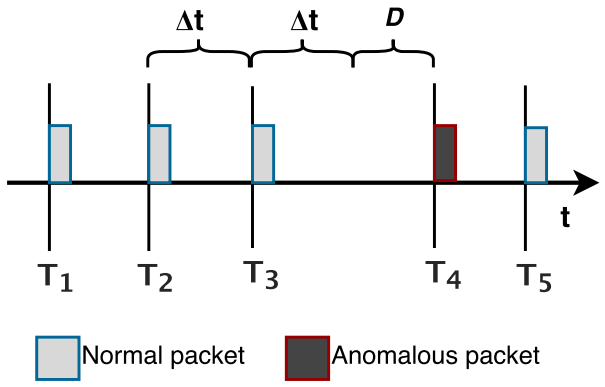

Figure 3: Timing anomaly detection.

In order to create such a timing model of packet arrivals, we record the following information features:

- The minimum difference in arrival time, Δt_{min}
- The maximum difference in arrival time, Δt_{max}
- The mean value of the difference in arrival time, μ
- The sample variance S_n
- The variance of the difference in arrival time σ^2
- The time of the previous packet t_{n-1}
- The number of analysed packets n

We then use the technique proposed by Finch [4] to iteratively calculate the variance as each packet arrives. This means that we only need to store 7 numerical values for each connection.

3.3 Detection component

The detection component processes packets during normal operation of the system, and should raise an alert if the traffic deviates from the normality model. Figure 4 shows the operation of the detector for each processed packet. Basically, the detection is performed in two steps. First, the packet is compared with one of the three appropriate whitelists (ARP, pair communication, TCP control). If the packet does not match any of the whitelists, an alert is raised. For example, if a packet has a previously unknown sender, receiver or port number then there will be no matching item in any of the whitelists.

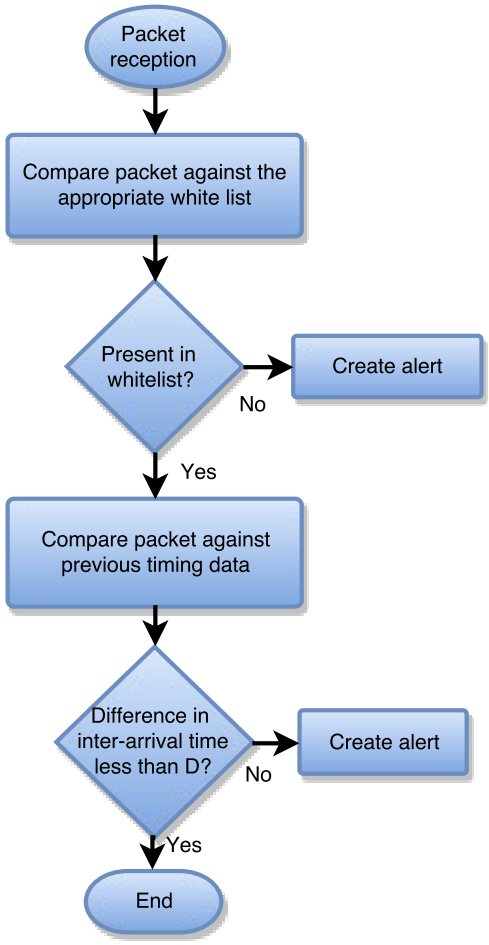

Figure 4: Flowchart of the detection component.

The detector will also perform a timing check for IEC 60870-5-104 packets and ARP packets. The timing check uses the statistical information recorded in the learning phase to determine if the time elapsed since the last packet differs more than D from the average inter-arrival time. The setting of the D parameter affects both the detection rate and false positive rate. In our tests we achieved the best results with $D = 6\sigma$.

4. EVALUATION

To test the effectiveness of the automatic whitelisting in a SCADA network a prototype implementation was developed and tested with some key attack types using data from a small-scale testbed. This section contains a description of how the normality data was collected, the attacks and their generation, and the resulting performance of the prototype IDS using this data.

4.1 Normal data collection

The Department of Industrial Information and Control Systems at the KTH Royal Institute of Technology in Stockholm maintains a SCADA laboratory which was used to record normal traffic data (see Figure 5). The setup contained four Remote Terminal Units (RTU) of the model NETCON RTU 28-IP, a HP ProCurve 1800-8G switch, and

a user terminal machine running Zenon (COPA-DATA 2013). The traffic data was collected using a Raspberry Pi model B+ through a mirroring port on the switch. A total of 6 days and 16 hours of data was collected. Traffic between RTUs 1 and 4 was further parsed and used as input to the training and testing of the intrusion detection system.

The traffic data was split in two parts, training and test data. The training period lasted for 24 hours in the tests described here, and the remaining days were used for testing. The attacks were added to the data traces by simulating the effect of a real attack to the system.

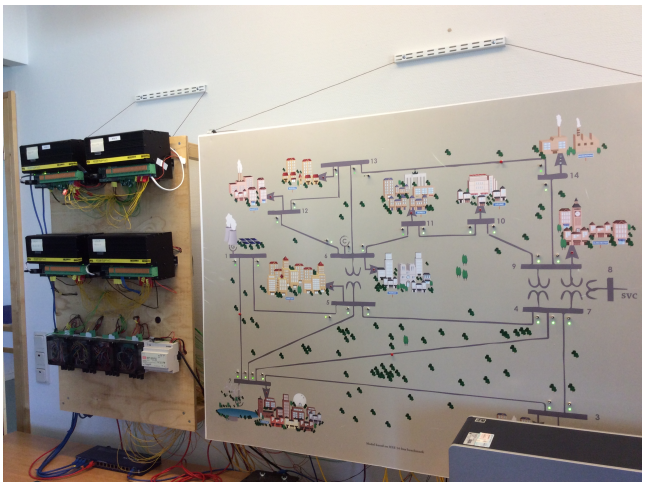

Figure 5: SCADA testbed at KTH Royal Institute of Technology used for collecting network traces.

4.2 Attacks

The intrusion detection system was tested with four different attacks: two port scan attacks, one man-in-the-middle (MITM) attack, and one spoofing attack, as described below.

Attack 1, port scan from an unknown host.

In this attack a host that was not previously part of the network sends a number of different connection requests with varying port numbers to see if the target responds to any of these ports. This information can then later be used to attack the host using exploits that target these specific services. To make the attack stealthier, we employ a technique where the connection is never fully established. This will reduce the risk that the attacked system creates logs that might trace back to the attacker. This is done by just sending the first request packet and if the host responds immediately shutting down the connection.

Attack 2, port scan from a known host.

This attack is identical to attack 1 except that the attack originates from a host that is already part of the system. Such an attack can occur if a node is infected by malware.

Attack 3, man-in-the-middle.

This attack uses the ARP to trick other hosts in the network to send their packets through the attacking host, thereby allowing the attacker to freely modify the packets in transit. The attack is initiated by the attacker by creating forged ARP packets. The packets are created to look like the only way for the hosts to communicate with each other is by sending their packets to the attackers interface.

Attack 4, spoofing.

The purpose of this attack is also to modify packet contents, but the effect is to send a packet that arrives just before the real packet. IEC60870-5-104 runs on top of TCP that uses a combination of sequence (SEQ) and acknowledge (ACK) numbers to keep track of packets and to sort them back into order upon arrival and to prevent replay attacks where a sniffed packet is sent again from the attacker to create an unwanted effect. The initial SEQ number should be chosen randomly to prevent prediction attacks. However, when a connection has been established the SEQ/ACK numbers are increased in a deterministic manner based on the length of the data transferred in the packet. In IEC 60870-5-104 connections can last for weeks or more. Another important aspect is that the length of the IEC 60870-5-104 packets is determined based on the type of instruction. This makes it possible for an attacker to send a forged packet shortly before the real packet is supposed to be sent. Since both packets will arrive, the destination host will discard the late packet as if it was a re-transmitted packet caused by packet losses.

4.3 Attack generation tools

A variety of tools were used to implement the studied attacks. All tools listed are freely available online.

Wireshark[3] is the world's leading protocol analyser. Wireshark has been an important tool used to analyse and understand the IEC 60870-5-104 protocol. Wireshark was also used to confirm that the attacks were manipulated and inserted into the recordings in the correct way. Tshark is a command-line version of Wireshark included in the Wireshark distribution. Tshark uses the same packet dissectors as Wireshark and were used at times when Wireshark was too slow due to the big recording file sizes.

Tcprewrite[4] was used to remove VLAN tags that the switch introduced. Tcprewrite was also used to change port numbers in the attack files to match the port numbers used in the real system.

Mergecap is a command line tool included in the Wireshark distribution used to merge different recordings together. One example of a use case is when the manipulated attack packets were inserted into the recording of the normal traffic. Mergecap was also used to convert between the .pcapng and the .pcap format of the recorded files.

Bittwiste[5] is a command line tool that were used to change the IP and MAC addresses in the attack files to match the actual recording from the SCADA environment. Nmap[6] is a tool that is used for security auditing and network discovery. Nmap was used in the port scan attacks. The first port scan emulates an attacker that is new to the network and tries to find open ports on the RTU. The other port scan emulates an attacker using a previously known host to find open ports. The attacker could have compromised the host were the port scan originates from earlier.

[3]https://www.wireshark.org/
[4]http://tcpreplay.appneta.com/wiki/tcprewrite.html
[5]http://bittwist.sourceforge.net/index.html
[6]https://nmap.org/

Arpspoof is a tool included in the dsniff package[7]. Arpspoof was used to create the MITM attack. Arpspoof sends ARP replies to both communicating hosts in order for them to stop communicating directly with each other. The hosts will think that in order to reach each other they will have to send their packets through the attacker. The attacker can not only read the packets passing but also alter them freely.

The most important tool used is Scapy[8] which is a packet manipulation program running on Python. Scapy allows manipulation of all fields in a wide variety of protocols. Even though the IEC 60870-5-104 protocol is not supported it is still possible to change the data transmitted. Scapy was used heavily to produce the data necessary to emulate a spoofing attack. Packets from the real recording were copied and moved a little earlier to emulate the attacker sending spoofed packets. The packets were also changed to carry modified data as if the attacker was injecting false information.

4.4 Merging attack data with normal data

The process of adding an attack to the recorded data starts with recording attack packets. This was done by setting up an emulated version of the data collection network (see Section 4.1) in which attacks could be performed. Depending on the attack type, the attacker mounts attacks on one or both of the RTU and HMI. The recorder is connected to a mirroring port in the switch. When the attack has been recorded in this setting, all attack packets are extracted from the recording. Since the MAC and IP addresses do not match the actual system, they have to be modified before insertion into the normal data stream for test purposes. The port numbers of the attack packets were also changed. The prediction attack needed manipulation of the packet payload. To accomplish this Scapy was used as described above.

When the packets have been modified to fit the actual system the packets are inserted into the normal recording. The detection was run on the resulting recorded file with attacks interleaved with normal traffic.

4.5 Detection performance

We proceed to give a sample of some of the results from this study. The data is based on traces from two RTUs since the data for the other RTUs led to the Hilti parser crashing. Table 1 summarises the total number of packets used in the experiment.

The overall best results (considering all attacks) were obtained when setting the threshold for normality at a very high level (6 sigma). Therefore we proceed to provide the results with this setting.

Figure 6 shows that the whitelisting mechanism successfully detects both port scan attacks and the man-in-the-middle attack as expected. The timing analysis confirms the detection of the whitelist for the man-in-the-middle attack. However, the spoofing attack which manipulated only timing was not expected to be detected by the whitelists. To our surprise the timing detector did not detect it either. Upon further investigation we found that this was due to filtering performed by Bro which suppresses duplicate packets. Since our timing manipulated packets arrive before the real packets, the real ones will be dropped by Bro. We have good reason to believe that by disabling this filter reasonable results can also be achieved for the spoofing attack, but this needs further research. The original developers of the Bro systems have been notified of this issue.

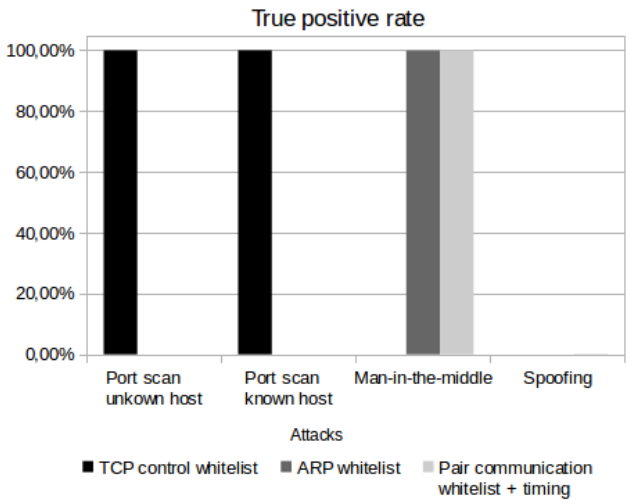

Figure 6: True positive ratio

Figure 7 depicts the false positive rates for the timing element of the detector. The whitelisting mechanism does not cause any false positives. The 6σ bar (green colour) shows that the false positives are around 0.01% when using the timing analysis. This corresponds to around 385 false alarms during the week of experiment, or on average 2.7 alarms per hour to be managed by an operator. This should be compared to the timing characteristics of the true alarms which was at far higher frequency (e.g., for the man-in-the-middle attacks 4195 alarms was generated for one RTU during one hour). Obviously for a real system post processing can be applied to reduce the load on the operator due to the same attack.

4.6 Timing performance

The 9,089,861 packets collected in the experiment were analysed using our system in the order of hours on a standard computer. Considering that the protocol packets themselves were not more than 10 packets per second per RTU, and that we used one day's data for learning, we conclude that testing of the packets was around 25 times faster than the protocol data rate. That means that in principle this approach is adequate for testing around 25 RTUs in real-time.

5. CONCLUSIONS AND FUTURE WORK

Detection of adverse events in critical cyber-physical systems is a complex task that needs knowledge about the physical system as well as the accompanying IT infrastructure. In certain instances the IT infrastructure alone can create adverse events and this can be detrimental to the physical process. This paper has studied a number of attack scenarios that build on deficiencies associated with one current protocol running in SCADA systems, namely the lack of authentication in a protocol devised over a decade ago, and for

[7]http://linux.die.net/man/8/dsniff
[8]http://www.secdev.org/projects/scapy/

Table 1: Experiment parameters

Total number of IEC 60870-5-104 packets	4,987,112
Total number of ARP packets	44,208
Total number of TCP control packets	4,058,541
Length of experiment	585,027s (≈1 week)
Packets used for learning	727,445
Number of attack packets inserted for port scan from an unknown host	4,002
Number of attack packets inserted for port scan from a known host	4,002
Number of attack packets inserted for the man-in-the-middle attack	18,594
Number of attack packets inserted for the spoofing attack	4,737

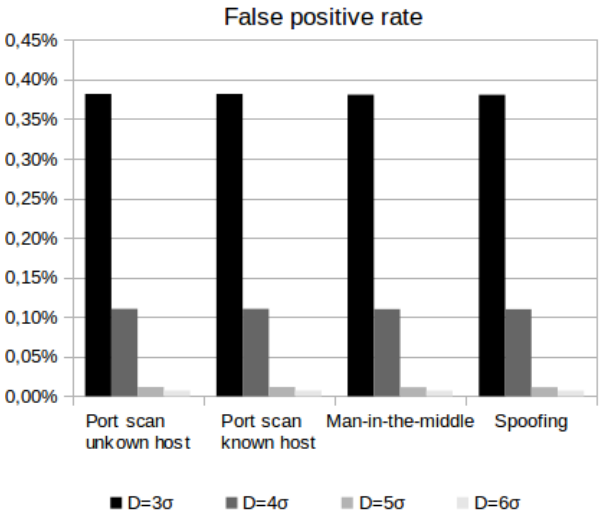

Figure 7: False positive ratio for the timing analysis

which the lack of memory/processing capacity in end units may exclude addition of such security mechanisms also in the future. In any case, the legacy issue in the power domain leads to possible exploitation of this vulnerability, as well as timing aspects explicit in the running of the protocol.

In order to defend SCADA systems with a monitoring element that only builds on the normal packet traces expected in such networks, the community needs efficient means of adapting the monitoring elements to various protocols and end system characteristics. We believe that building a protocol parser for various protocols is a viable approach and in this work have made progress in this direction.

While the results in this case study are mostly positive, there were also some challenges along the way. In order to get the network data into a form so that the proposed additions to the Bro tool could be applied, a parser had to be written. Unfortunately, it seems that the Spicy framework needs further development since we experienced some hard to debug crash cases while analysing the traces for two of our 4 RTUs, which seemed to be due to bad memory handling. Hence, the task of implementing the vision was shown to be demanding both with respect to expertise and effort. The thresholds for the anomaly detector were experimentally determined in the available network. In a live system one would have to monitor the performance of the anomaly detector and potentially adjust the thresholds during an observation and adaptation period, which requires expert knowledge and careful analysis of the running scenarios.

The preliminary results showed, however, that the idea of whitelisting for anomaly detection in the IEC-61870-105-4 protocol was fruitful and the false positive rates for applying the timing analysis based approach were satisfactory. Moreover, the volume of data generated in our small test bed was satisfactorily handled in the time intervals considered. This provides an indication that scaling up this approach for networks with many RTUs and distributed monitoring using small and dedicated embedded devices is worth pursuing in more detailed future studies. In addition, more sophisticated attacks whereby the payloads of intercepted packets are manipulated and modified have not been dealt with here and are interesting topics for further work.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable and constructive feedback. The work was initiated in a project supported by Vinnova, Formas and the Swedish Energy Agency under the IoT strategy program and completed within RICS: the research centre on Resilient Information and Control Systems (www.rics.se) financed by Swedish Civil Contingencies Agency (MSB). The second author was also supported by CENIIT project 14.04. Finally, we wish to thank Robin von Post and Anders Hansson at Sectra Communications.

6. REFERENCES

[1] R. R. R. Barbosa. *Anomaly detection in SCADA systems: a network based approach*. PhD thesis, Enschede, 2014.

[2] R. J. Chang, R. E. Harang, and G. S. Payer. Extremely lightweight intrusion detection (elide). Technical report, Adelphi, Army Research Laboratory, 2013.

[3] G. R. Clarke, D. Reynders, and E. Wright. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems*. Newnes, 2004.

[4] T. Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*, 4, 2009.

[5] V. Gazis, M. Gortz, M. Huber, A. Leonardi, K. Mathioudakis, A. Wiesmaier, F. Zeiger, and E. Vasilomanolakis. A survey of technologies for the internet of things. In *Wireless Communications and Mobile Computing Conference (IWCMC)*, 2015. , doi: 10.1109/IWCMC.2015.7289234.

[6] T. Good and M. Benaissa. Asic hardware performance. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 267–293. Springer Berlin Heidelberg, 2008. , doi: 10.1007/978-3-540-68351-3_19.

[7] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. Through the eye of the plc: Semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, ACSAC '14. ACM, 2014. , doi: 10.1145/2664243.2664277.

[8] J. Hoyos, M. Dehus, and T. Brown. Exploiting the goose protocol: A practical attack on cyber-infrastructure. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, 2012. , doi: 10.1109/GLOCOMW.2012.6477809.

[9] D. Lake, R. Milito, M. Morrow, and R. Vargheese. Internet of things: Architectural framework for ehealth security. *Journal of ICT*, 3&4, 2014.

[10] H. Lin, A. Slagell, Z. Kalbarczyk, and R. K. Iyer. Semantic security analysis of scada networks to detect malicious control commands in power grids (poster). In *Proceedings of the 7th International Conference on Security of Information and Networks*, SIN '14. ACM, 2014. , doi: 10.1145/2659651.2659746.

[11] B. Matt. The cost of protection measures in tactical networks. In *Proceedings for the Army Science Conference (24th), Orlando, Florida.*

[12] P. Maynard, K. McLaughlin, and B. Haberler. Towards understanding man-in-the-middle attacks on iec 60870-5-104 scada networks. In *Proceedings of the 2Nd International Symposium on ICS & SCADA Cyber Security Research 2014*, ICS-CSR 2014. BCS, 2014. , doi: 10.14236/ewic/ics-csr2014.5.

[13] R. Mitchell and I.-R. Chen. A survey of intrusion detection techniques for cyber-physical systems. *ACM Comput. Surv.*, 46(4), 2014. , doi: 10.1145/2542049.

[14] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24), 1999. , doi: 10.1016/S1389-1286(99)00112-7.

[15] T. Shawly, J. Liu, N. Burow, S. Bagchi, R. Berthier, and R. Bobba. A risk assessment tool for advanced metering infrastructures. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, 2014. , doi: 10.1109/SmartGridComm.2014.7007777.

[16] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. Wang. Intrusion detection system for iec 60870-5-104 based scada networks. In *Power and Energy Society General Meeting (PES), 2013 IEEE*, 2013. , doi: 10.1109/PESMG.2013.6672100.

[17] K. Yim, A. Castiglione, J. H. Yi, M. Migliardi, and I. You. Cyber threats to industrial control systems. In *Proceedings of the 7th ACM CCS International Workshop on Managing Insider Security Threats*, MIST '15. ACM, 2015. , doi: 10.1145/2808783.2808795.