Intrusion Detection System Test Framework for SCADA Systems

Henrik Waagsnes and Nils Ulltveit-Moe
Department of ICT, University of Agder, Jon Lilletunsvei 9, 4879 Grimstad, Norway

Keywords: IDS, Test Framework, SCADA, IEC 60870-5-104, SIEM.

Abstract: This paper presents a SCADA intrusion detection system test framework that simulates SCADA traffic and detects malicious network activity. The framework combines several existing components such as Kali Linux, Conpot, QTester104 and OpenMUC in a virtual machine based framework to provide realistic SCADA traffic. It is agnostic to Intrusion Detection System (IDS) type, and is demonstrated in a case study comparing two popular signature-based IDS engines: Suricata and Snort. The IDS engines include rule-sets for the IEC 60870-5-104 and other SCADA protocols. Detected events from IDS sensors are sent to a distributed Elastic cluster which visualises them using Kibana dashboards. The experiments show that there is some difference in behaviour between Suricata and Snort's ability to detect malicious traffic using the same SCADA ruleset, but these issues are relatively easy to mitigate. The IDS test framework also measures the latency from detection and until the IDS alerts are presented in the incident management system, which shows that Suricata has slightly better performance than Snort.

1 INTRODUCTION

This paper describes the design and implementation of a testbed for assessing IDS performance in SCADA networks. It combines off-the-shelf components such as the Elastic stack, Kali Linux, and published IDS rule sets to generate simulated traffic and analyse results. The paper's experimental section evaluates two popular open–source IDS engines against SCADA-related traffic produced by several public tools. Overall, this provides a useful end-to-end solution for testing IDS tools against SCADA–oriented attacks, which are an increasingly prevalent security issue nowadays. The long-term objective is to extend it as a control system platform for performing realistic exercises in a simulated operations centre as shown in Figure 1. The aim is to improve the situational awareness for security analysts and other stakeholders (personnel managing power lines etc) during emergency situations. This would allow performing realistic exercises on cyber-attack events that naturally will occur infrequently, thereby increasing the readiness towards such attacks. It will especially allow performing exercises that simulate advanced persistent threat scenarios caused by government agencies or other organisations with significant funding that aim at performing targeted attacks on critical infrastructure as demonstrated for example by the Ukrainian BlackEnergy attack[1].

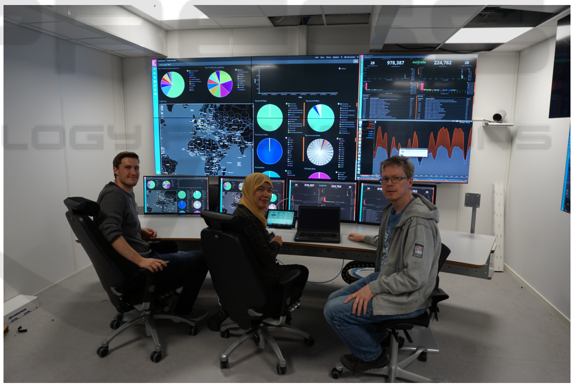


Figure 1: CIEM Test lab for evaluating situational awareness.

SCADA systems used in the energy sector encompasses the following: collecting of information via Remote Terminal Units (RTUs), transferring it back to the central site, carrying out necessary analysis and control, and then displaying that information in a Human Machine Interface (HMI). A SCADA communication protocol is a standard for data representation and data transfer over a communication channel on a master/slave basis. IEC60870-5-104 and DNP3 are

[1]Cyber-Attack Against Ukrainian Critical Infrastructure https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01

two of the most frequently used SCADA communication protocols in the electrical energy industry. IEC 60870-5 is widely used in Europe, while DNP3 is widely used in North America.

The main contributions of this paper are threefold: first, it presents a novel intrusion detection system test framework for SCADA networks based on a set of virtual machines that can be used to simulate different attack scenarios. This framework builds upon and integrates several other SCADA tools in a novel way, and also allows for simulating the HMI of a power plant. Secondly, this framework is demonstrated in a comparative study between the signature-based IDS Snort[2] and Suricata[3] to verify whether an IEC60870-5-104 SCADA ruleset for Snort indeed is compatible with Suricata or not. Third, it performs a comparative study of the efficiency of Snort and Suricata when integrated in the Elastic stack[4] (formerly known as the Elasticsearch, Logstash and Kibana (ELK) stack).

This paper is organised as follows: The next section gives an introduction to SCADA protocols. Section 3 discusses related works. Section 4 describes the proposed framework architecture and developed dashboards. Section 5 discusses experiments and results in context of validation and testing of the implemented framework. Section 6 contains a discussion of the proposed framework, evaluation of results, and other thoughts about the conducted research. Section 7 contains a conclusion of the presented framework and achieved results, and finally section 8 contains a discussion around possible future work and suggestions for improvements.

2 INTRODUCTION TO SCADA PROTOCOLS

IEC 60870-5-104 is a standard SCADA communication protocol for telecontrol in electrical energy systems. The protocol describes data representation and data transfer over communication channels on a master/slave basis. The SCADA Master polls data from the connected SCADA slaves (RTUs). A RTU is used in the electrical power sector to control circuit breakers, transformer stations etc. The SCADA Master can display the information in a Human-Mahine Interface (HMI). Figure 2 shows a simplified SCADA archiecture.

[2]Snort: https://www.snort.org
[3]Suricata https://suricata-ids.org
[4]Elastic: https://www.elastic.co

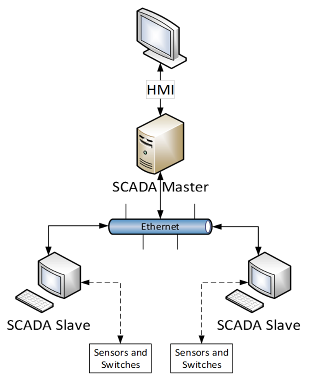


Figure 2: Example SCADA system.

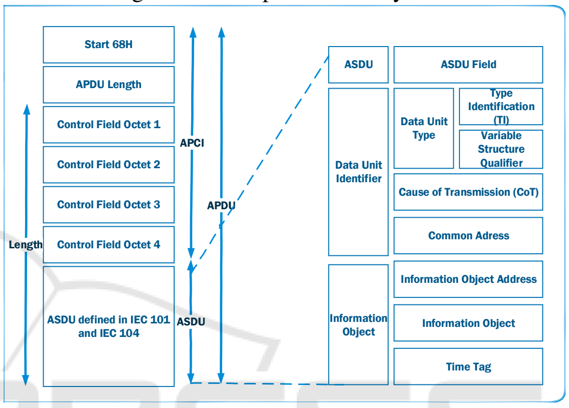


Figure 3: IEC 60870-5-104 frame format.

2.1 IEC 60870-5-104

IEC 60870 is a collection of open standards created by the International Electrotechnical Commission (IEC) for the transmission of SCADA telemetry control and data (International Electrotechnical Commission, 2006). IEC 60870 normally refers to IEC 60870-5-101 when discussed in context of SCADA in the electrical energy sector. This is a standard for power system monitoring, control and associated communications. When it was launched in 1995, the protocol was designed for serial communication. IEC 60870-5-104 is an extension of IEC 60870-5-101 that was released in 2004, which allows the serial frames to be transmitted over TCP/IP. The Application Protocol Data Unit (APDU) shown in figure 3 consists of two parts: the Application Protocol Control Information (APCI) and Application Service Data Unit (ASDU).

The APCI is comprised of the first 6 octets in the APDU, and contains a start character, 68H, a length field (containing the length of the APDU) and a control field. The APCI control field can be of the following types: Information, Supervisory or Unnumbered. The last two bits indicate the type; 10 for supervisory, 11 for unnumbered and 00 for information.

The ASDU contains the data unit identifier and the data payload of one or more information objects. The Type Identification (TI) field defines the data types by referring to the 8-bit code types. The TI groups that are currently defined by IEC. TI number 9 for example refers to the reference code M_ME_NA_1 indicating "Measured value, Normalised value".

2.2 Security Issues

The security in SCADA systems and the communication protocols used to exchange data is important, to prevent cyber-attacks on critical infrastructures that they control. The root cause of cyber security vulnerabilities in industrial control systems have been found to be (Graham et al., 2016):

- Poorly secured legacy systems;
- Lack of trained cyber security specialists;
- Delayed patching of software vulnerabilities;
- Lack of cyber-security situational awareness;
- Lack of communication security;
- Remote access for system engineers and vendors.

Many SCADA systems lack security incident event detection and monitoring (SIEM) functionality to increase the cyber-security awareness and detect cyber-attacks on these systems. Without network monitoring, it is impossible to detect suspicious activity and identify potential threats. Another problem is that some systems are rarely or never updated. This means the SCADA systems might contain vulnerabilities in firmware or software that can be exploited by attackers. Some vendors even allow SCADA devices to communicate remotely over unencrypted communication links. Authentication solutions are often configured with poor passwords or even with default passwords, and SCADA protocols typically lack authentication and cryptographic message integrity checks. The reason for this being that the Remote Terminal Unit (RTU) devices originally did not have the processing capability to do such operations. Many SCADA protocols are therefore designed to be open, robust (from a control perspective), reliable and easy to operate, and not necessarily designed to provide secure communication.

The IEC 60870-5-104 protocol which is used to control RTUs in the electrical energy sector has the following security properties:

- ***Lack of Confidentiality:*** All IEC 60870-5-104 messages are transmitted in clear-text across the network.
- ***Lack of Integrity:*** There are no cryptographic integrity checks built into the protocol.
- ***Lack of Authentication:*** There is no authentication of communicating entities in the protocol.

3 RELATED WORKS

This paper proposes an IDS test framework based on virtual machines for SCADA systems. It uses a combination of the Elastic stack as a big-data based SIEM solution, the SCADA HoneyNet ConPot and Kali Linux as the attacker. The solution is experimentally verified in a comparative analysis of the performance of Snort and Suricata for SCADA rules focusing on the IEC 60870-5-104 telecontrol protocol for the electrical energy sector. The solution and comparative analysis is, to the best of our knowledge, not formely described in the scientific literature.

There are some examples of previous SCADA test beds, for example the US Idaho National Lab SCADA test bed (Wei et al., 2010). University of Arizona set up a test bed called TASCA for analysing the security of SCADA control systems (Davis et al., 2006). The latter uses PowerWorld as a grid simulator to simulate the effect of cyber-attacks on the power grid. Our framework currently supports grid simulation based on the Open Substation HMI (OSHMI)[5], and plans more elaborate grid simulation support in the future.

Queen's University Belfast has published a paper based on research in the PRECYSE FP7 project that presents a SCADA-specific cyber-security testbed used for investigating man-in-the-middle attacks (Yang et al., 2012). They built a framework with three Windows-based hosts to simulate real-time SCADA master/slave communication that displays the output on a HMI. The framework also includes a Linux host to simulate an attacker. Due to confidentiality and security concerns, the specific software used in this framework is withheld by the publishers (Yang et al., 2012). This solution does not use a virtualised infrastructure, which makes it less flexible for dynamically setting up and changing test scenarios. It furthermore only covers the IDS host, which runs a proprietary IDS (ITACA), and does not contain a SIEM solution that can be used for the comparative analysis as well as for doing research on situational awareness. Our solution is more flexible by running as a virtualised infrastructure which uses different virtual machines for different test components.

The same researchers published a paper, presenting a set of Snort IDS rules for IEC 60870-5-104 SCADA networks (Y. Yang et al., 2013). They used a Deep Packet Inspection (DPI) method, which in-

[5]OSHMI https://github.com/focusenergy/oshmi

cludes signature-based and model-based approaches. Our experiments use a set of Snort IDS signatures for SCADA systems from various sources, amongst others the model-based IDS ruleset for the IEC 60870-5-104 protocol from (Y. Yang et al., 2013), as well as the Modbus and DNP3 IDS rules that come with the Snort VRT ruleset. These signatures can not only detect several known malicious attacks and suspicious threats, but also identify the sources of the attacks which can aid security analysts in potentially preventing future intrusions from these sources. The Modbus and DNP3 rules are not described in this paper, since the results do not add significant value compared to the findings in the IEC 60870-5-104 rule set.

We reuse this IDS rule set to perform a comparative analysis of how well Suricata and Snort perform on these rules, something that is not done in the original paper. This also acts as a proof of concept experiment for verifying the whole SCADA IDS test framework.

An ENIP (Ethernet/IP) SCADA rule set for Suricata was developed in (Wong et al., 2017). This research confirms that Suricata is a good choice for SCADA IDS monitoring on resource constrained hardware, but does not perform a comparative analysis of other SCADA protocols, including IEC 60870-5-104.

A quantitative analysis of Snort and Suricata is performed in (White et al., 2013). They used large amounts of pcap data and custom scripts to compare the two IDS engines on flexible and scalable hardware. To conduct a quantitative analysis they captured a variety of metrics, including packets per second, memory usage and CPU utilization. The IDS rulesets Emerging Threats (ET) Open/ET-Free, ET-Pro and Snort VRT were used in their experiments. They conclude that Suricata outperforms Snort, even on a single core. Suricata also has an average lower memory and CPU utilization than Snort. This paper does not do a comparative analysis of SCADA specific rulesets to detect compatibility issues as well as a latency comparison between the IDSs in the Elastic stack.

There are also some other examples of comparative analyses of Snort and Suricata, for example (White et al., 2013; Pihelgas, 2012; Brumen and Legvart, 2016), but none of these consider SCADA protocols.

Our framework simulates IEC60870-5-104 communication, and can be configured to simulate Modbus TCP and DNP3 as well. In contrast to other quantitative analyses of IDS engine performance, we have compared the detection capabilities of Snort and Suricata in context of SCADA rule-sets.

4 APPROACH

The SCADA IDS Test Framework provides a controlled environment for penetration testing, real-time simulation, intrusion detection and visualisation of attacks targeting SCADA networks. It can simulate real-time IEC 60870-5-104 traffic between SCADA devices and attacks can be performed by utilizing the attacker machine in the framework. The framework also includes a simulated Siemens S7-200 PLC to perform attacks against the Modbus protocol. Two separate intrusion detection systems (IDSs) are individually looking for malicious SCADA traffic in the controlled environment. An alert is triggered each time one of the IDS implementations detect malicious traffic. The alert is then forwarded to a centralised security information and event management (SIEM) solution, responsible for collecting, analysing, indexing and visualising IDS events.

4.1 Framework Architecture

The framework's architecture is categorised into four elements, as illustrated in figure 4:

- Security information and event management (SIEM)
- Intrusion detection systems (IDS)
- SCADA target side
- Attacker side

The attacker side is a Kali Linux machine[6] with over 600 penetration testing tools. The SCADA target side consist of three machines running the client and server side of the IEC 60870-5-104 communication, and one machine simulating a Siemens S7-200 PLC. The Attacker and SCADA machines are connected to the same local network. The network traffic traversing the lab network is mirrored and analysed by two separate IDS implementations: Suricata and Snort. The SCADA rules for IEC 60870-5-104 and the Digital Bond Quickdraw SCADA rules for Modbus and DNP3 are used in both IDS solutions.

Suricata is configured to log alerts in a structured Extensible Event (EVE) JSON format. This output format is not available in Snort. For the basis of comparison between alerts triggered by Suricata and Snort, it is advantageous if the output log format is the same. To achieve this, Snort has been configured to log alerts in Unified2 format. The Python library py-idstools is then used to convert the Unified2 format to EVE JSON (Ish, 2017).

[6]Kali Linux: http://www.kali.org

The Elastic stack[7] (formerly known as ELK stack) is used as a SIEM solution in this framework (Lahmadi and Beck, 2015). It includes the following services:

Logstash is a service that collects, parses and transforms logs. It is configured to listen for log data on TCP port 5044 in the management network. Filebeat is log file transfer service that is installed on both IDS implementations and configured to forward new entries (IDS alerts) in the log file to Logstash on port 5044. Logstash then analyses the incoming data and adds/removes specific fields and tags (e.g. GeoIP information) as will be discussed later. The refined data is then sent to the search engine (Elasticsearch based on Apache Lucene) and stored in a cluster of nodes. Kibana is used as a data analysis framework to explore the Elasticsearch data. It provides the possibility to investigate historical and real-time events by utilizing a search engine and use custom dashboards to visualise the data.

X-pack[8] is a commercial SIEM solution implemented on top of the open source Elastic stack. This framework is available with a basic license which only includes a monitoring feature. It provides real-time monitoring of the performance of the Elasticsearch cluster and the Kibana work load. Figure 4 gives an overview of the framework architecture.

To provide automatic start up at reboot, scheduled log rotation and synchronous time settings, the system was automated using system daemons, scripts for automation as well as network segmentation with a separate management and simulation network to reduce the risk that attackers are aware of the security monitoring.

4.2 Dashboards

Kibana uses a dashboard based approach for data visualisation and analysis. An advantage with this framework, is that it allows security analysts to easily reconfigure the user interface according to their operative needs. For our test lab, this means that different scenarios and visualisations can be tested out to evaluate the usability, accessibility and situational awareness of the operators.

In total six dashboards were developed for different purposes. The first dashboard is a visualisation of the traffic flow data generated by Suricata. Two separate dashboards were created to analyse the alert data from Suricata and Snort, respectively. A dashboard was also developed to compare alert data generated from the two IDS implementations. An experimental dashboard was created to visualise relationships between data. The last dashboard was created to analyse the additional latency added by Logstash processing. The time frame for each dashboard can easily be changed by adjusting a parameter at the top right. The time frame can be anything from the past 15 minutes to the past 5 years.

[7]Elastic stack: https://www.elastic.co

[8]X-pack: https://www.elastic.co/x-pack

Network Traffic Monitoring

The network traffic dashboard shown in figure 5 analyses the network traffic in real-time. The left side uses the geographical data added by Logstash. The dashboard displays pie chart visualisations of the top 10 countries and cities generation traffic, and plots the geographical location into a map. The right side analyses the actual traffic and visualises the destination IP addresses, source IP addresses, source port, destination port, underlying protocols and TCP states.

IDS Alert Monitoring

The IDS alert dashboard provides a histogram that counts the unique events at any given time. The dashboard also provides a measurement count of alerts with various severity levels. If the count reaches a custom set limit, the measurement changes color from green to yellow or red. The geographical location of any source address triggering an alert, is plotted in a map. Pie charts are used to visualise relevant data, and a table is used to count the unique signatures triggered.

IDS Comparison

The IDS comparison dashboard in figure 6 is used to compare Suricata's and Snort's ability to detect attacks. Snort is monitored on the right side and Suricata on the left side. The dashboard contains measurement visualisations to count unique signatures and the total number of alerts triggered by Suricata and Snort, respectively. The size of the IDS name text in the middle is automatically adjusted by number of alerts triggered. At the end of the dashboard is a table that lists every triggered alert and counts every occurrence.

Data Relation Dashboard

The experimental dashboard implemented in this framework, is a network graph that can be used to visualise the relations between the data.

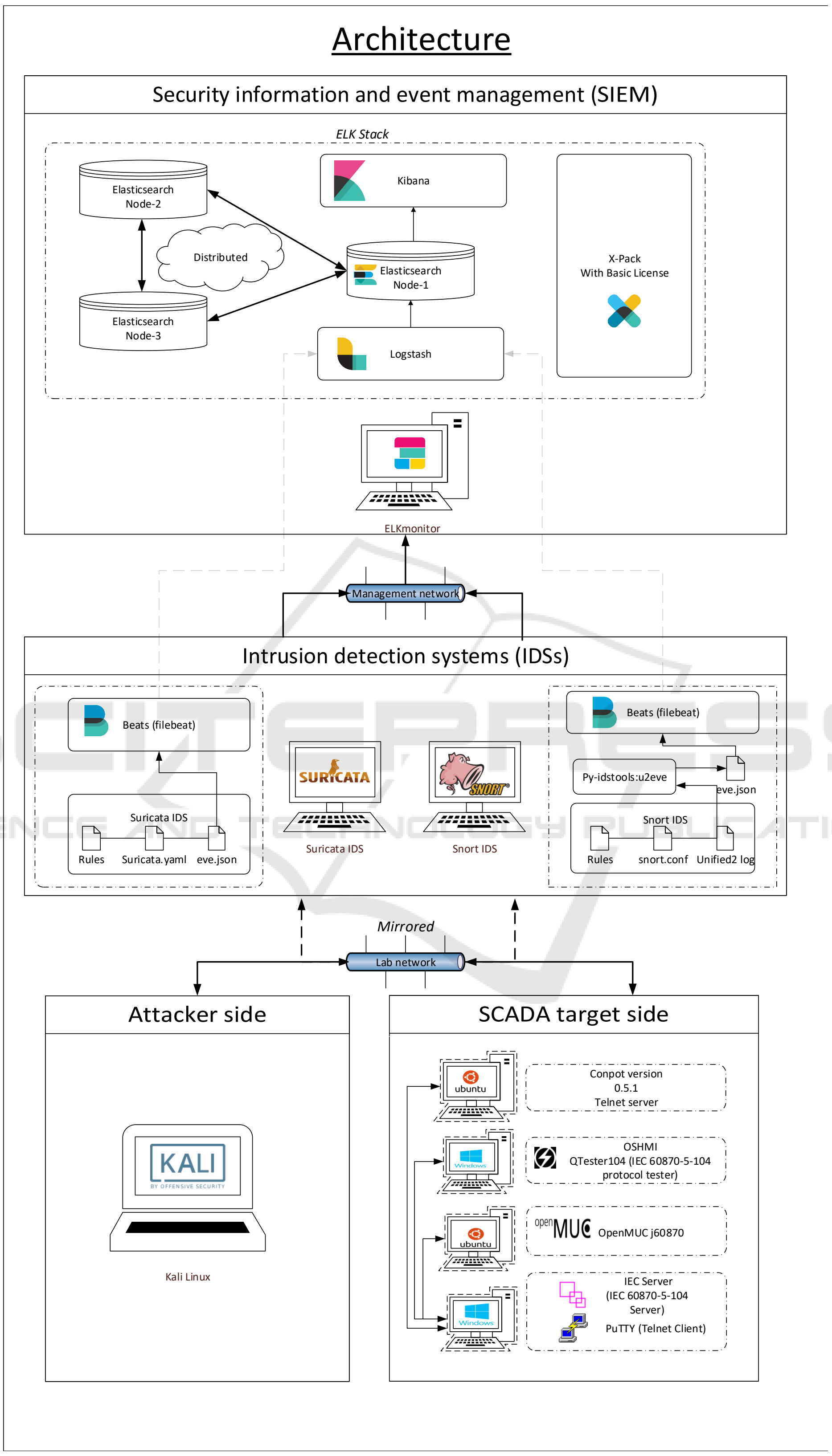


Figure 4: System Architecture.

Processing Latency Dashboard

The last dashboard implemented in this framework, is used to visualise the additional latency added by Logstash processing. The IDS adds a timestamp field to the eve.json file when it detects malicious traffic. Logstash also adds a timestamp field when the data is processed and shipped to Elasticsearch. These timestamp fields can be used to find the latency by calculating the deviation between them. This is done by using a Kibana functionality called scripted fields which performs calculations on the fly by analysing the incoming Elasticsearch data. The implemented dashboard compares latency for Suricata and Snort alerts. It plots the average latency in a line chart every second and calculates max, min, average, median and standard deviation (upper/lower) on the fly. The dashboard also visualises the latency in a pie chart.

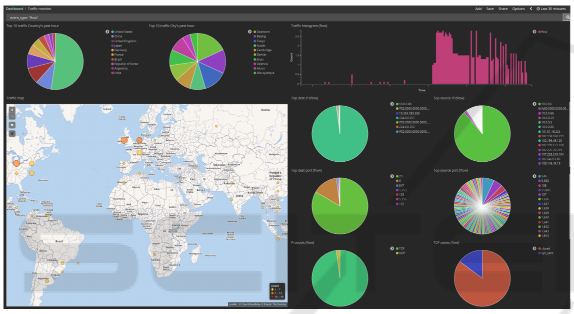


Figure 5: Network traffic monitoring dashboard.

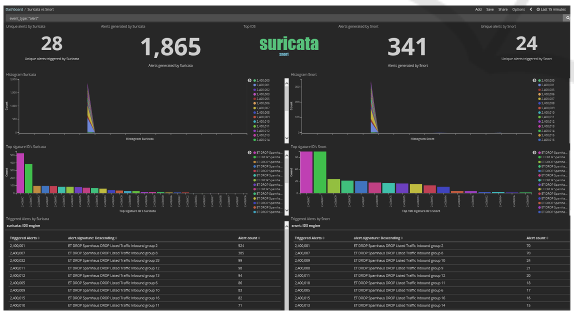


Figure 6: IDS comparison dashboard.

5 EXPERIMENTS AND RESULTS

This section discusses experiments and results in the context of validation and testing of the implemented framework. Five experiments were performed to measure the latency of the framework and verify that:

- Normal traffic does not trigger any false positives;
- Signature-based rules do not trigger on traffic from an unauthorised client;
- Protocol-violation rules trigger on IEC 60870-5-104 protocol violations;
- Traffic pattern based rules do not trigger on illegal traffic patterns.

5.1 IEC 60870-5-104 client/server Communication

The main goal of carrying out these experiments, is to trigger alerts specified in the implemented IEC 60870-5-104 rule-set (Y. Yang et al., 2013). This is done by generating realistic IEC 60870-5-104 background traffic in the lab network and in addition perform attacks and Illegal actions. The alerts triggered by Suricata and Snort are compared by log analysis and real-time visualisation. The signatures included in the rule set can be categorised into three categories; signature-based rules, protocol based rules and traffic-pattern-based rules.

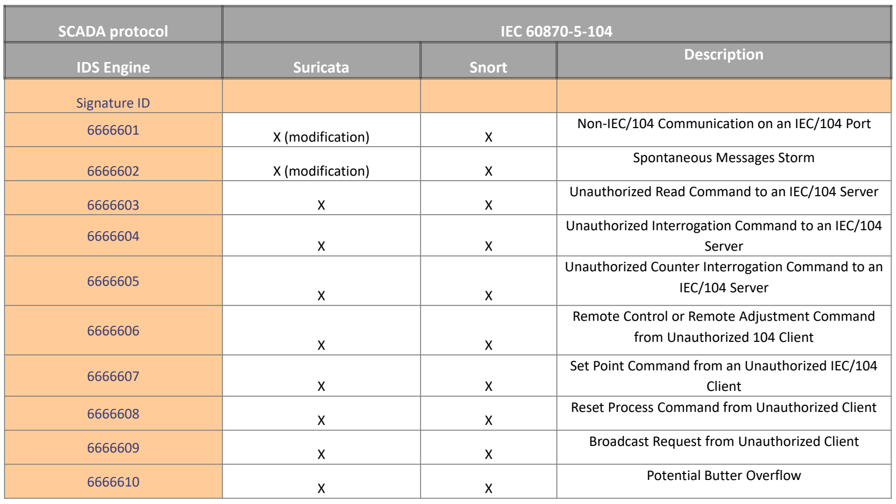
5.1.1 Normal Communication:

To simulate normal traffic patterns between authorised IEC 60870-5-104 clients and servers, experiments were conducted using the IEC Server software on a virtual machine with IP address 10.0.0.75, listening on port number 2404 (Jaentsch, 2013). QTester104 is used as an IEC client on a virtual machine with IP address 10.0.0.66 (Olsen, 2017). Wireshark[9] is used to inspect and validate the IEC 60870-5-104 traffic traversing the lab network. Since Suricata is configured to log all data traffic, Kibana can be used to analyse normal traffic as well. No alerts (false positives) were triggered either by Suricata or Snort during this experiment, as expected.

5.1.2 Signature-based Rules:

Several experiments were conducted to simulate illegal SCADA operations to trigger the signature-based rules, by using tools like QTester104, IEC Server, OpenMUC j60870 and packet sender. Initially, Suricata did not detect and trigger alerts for two of the signatures as shown in figure 7. The compatibility issue was that the regular expression matching in Suricata with the /R (relative) option requires a preceeding match in the same buffer, but there is no buffer keeping track of previous matches. This was the case for two of the signatures in this rule set, sid: 6666601 and sid: 6666602. This issue was mitigated by removing the "R" from the regular expression.

[9]Wireshark: https://www.wireshark.org



SCADA protocol	IEC 60870-5-104		
IDS Engine	Suricata	Snort	Description
Signature ID			
6666601	X (modification)	X	Non-IEC/104 Communication on an IEC/104 Port
6666602	X (modification)	X	Spontaneous Messages Storm
6666603	X	X	Unauthorized Read Command to an IEC/104 Server
6666604	X	X	Unauthorized Interrogation Command to an IEC/104 Server
6666605	X	X	Unauthorized Counter Interrogation Command to an IEC/104 Server
6666606	X	X	Remote Control or Remote Adjustment Command from Unauthorized 104 Client
6666607	X	X	Set Point Command from an Unauthorized IEC/104 Client
6666608	X	X	Reset Process Command from Unauthorized Client
6666609	X	X	Broadcast Request from Unauthorized Client
6666610	X	X	Potential Butter Overflow

Figure 7: Signature-based rules.

5.1.3 Protocol Violation and Traffic-pattern based Rules:

To trigger the protocol violation rules, an experiment was conducted using using the Ettercap plugin developed by PMaynard for performing a man-in-the-middle attack (Maynard, 2017). The attack monitors the IEC 60870-5-104 communication between a client and a server, and injects illegal protocol packets into the network.

Both Suricata and Snort detected all malicious traffic injected in the experiment, and triggered alerts based on the protocol-based rules (SIDs 6666611 - 6666621). All the traffic-pattern based rules (SIDs 6666622 - 6666624) were also triggered for both IDSs using generated traffic.

5.2 Latency

The two different IDS implementations in the test framework detect malicious traffic, and use Filebeat to transmit the log entries to Logstash. Logstash analyses, modifies and transfers the data to a Elasticsearch cluster. The latency considered here is the time it takes from malicious traffic is detected by an IDS until the event is indexed and stored in the cluster. This experiment uses a custom latency dashboard implemented in the framework to analyse the additional latency added by Logstash and conversion-tools. Normal amounts of IEC 60870-5-104 traffic are sent between client and server, using the QTester104 and IEC Server tools (Olsen, 2017)(Jaentsch, 2013).

The first experiment lasted ten minutes and triggered approximately 200 alerts for both Suricata and Snort during normal conditions. The boxplot in figure 8 is an ensemble of ten latency measurements to get a representative picture of the variance of the test data. The figure shows a comparison of Suricata and Snort under normal conditions. The thin line describes the highest and lowest latency measured for both IDS solutions. Outlier values are excluded from the figure. The box represents the upper and lower quartile of the latency and the yellow line in the middle represent the median. From the figure it appears that Snort generally has higher latency a Suricata. The median value was approximately 2.5 seconds for Suricata and 5.1 seconds for Snort. The upper and lower quartile of Snort latency is higher with a larger variance than Suricata. Both solutions have a bit skewed distribution compared to a normal distribution.

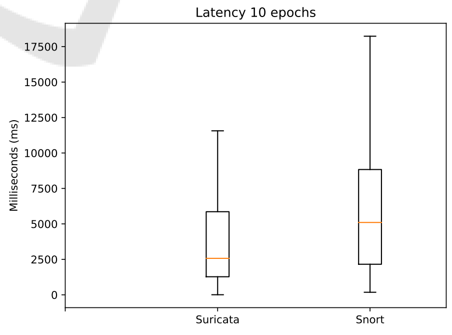


Figure 8: Boxplot comparison of latency in Suricata and Snort under normal traffic.

6 DISCUSSION

The main advantage of using the Open Source Elastic stack rather than a fully commercial alternative such as Splunk for the IDS test framework, is that Elastic has an affordable licencing policy, and does not severely restrict the number of dashboards such as Splunk does in its evaluation mode. These restrictions essentially made Elastic a better alternative for our project. Comparing Splunk and Elastic as SIEM solutions under similar conditions would be an interesting extension in a larger follow-up study.

Another possibility would be to use Graylog2[10], which has similar features as Elastic, is Open Source, and also is based on Elasticsearch for log data mining. Graylog2 excels at postmortem debugging, security and activity analysis. Our main reason for choosing Elastic over Graylog2, is that it has more flexible graphing and dashboard handling. Such functionality would be useful for extending the framework as a more general situational awareness platform supporting both security monitoring and crisis management in the future. Kibana is user friendly by allowing data scientists to fairly easily add more functionality to the system in the form of new dashboards. It is also possible to add support for more advanced time series analytics by integrating Grafana[11]. A limitation with Kibana is that it lacks advanced user management. It is however possible to set up simple access control using HTTP basic authentication. More advanced solutions will require purchasing X-Pack which provides encryption, authentication and authorisation.

The solution proposed here is based on virtual machines. This plug-in architecture for IDS engines makes the framework easily extensible for other IDS technologies. Other similar architectures have been suggested based on Docker containers, which are more lightweight than virtual machines and allows for enabling independence between cloud applications and infrastructure[12]. Our solution will needs to use at least some virtual machines to be able to test real operating system instances. Docker could be considered as an option in the future.

Elasticsearch can be used for building complex search requests, but the main challenge in future research will be not only conveying the IDS alerts but also performing more complex data analysis, alert correlation and data mining to identify the relevant information for security operators and other stakeholders and reduce the amount of false alarms. Clustering, behaviour analysis and machine learning techniques used in anomaly detection would be natural extensions of this research to improve the overall attack detection capabilities of the system in the future.

Possible missions for the target framework includes testing the performance in terms of accuracy and speed of detection, validating new IDS solutions or validating different rule sets on IDSs. Another possibility is to extend the framework as a comprehensive hybrid SIEM/IDS solution that uses several different tools such as log analysis, network and host-based intrusion detection systems.

Future challenges that can be investigated using this framework includes research on how to model new protocols and how to simulate really big data scenarios where a large cluster of sensors as well as Elastic shards need to collaborate on the data mining. Autoconfiguration of the framework is another challenge which can use techniques such as Network Function Virtualisation and Software Defined Networking. Alert normalisation is to some extent handled by Logstash. Future research and standardisation is however required to define a common ontology that ensures semantic interoperability between different types of alerts (Krauß and Thomalla, 2016). The platform will also act as a research vehicle for visualising and analysing results as to prove specified simulation scenarios as well as for improving situational awareness during such scenarios.

The framework is extensible and scalable. The IDS side can be extended with additional IDS solutions by adding new virtual machines to the framework that support a given IDS technology. The Logstash configuration can also be adapted to categorise different kinds of IDS solutions. Logstash is horizontally scalable meaning that the performance can be scaled up by adding more hardware nodes. It can furthermore form groups of nodes running the same information pipeline. Adaptive buffering capabilities in the Elastic stack provides smooth data streaming even with variable throughput loads. If Logstash becomes a bottleneck, then more nodes (cloud instances/virtual machines) can be added.

Elasticsearch is also horizontally scalable by allowing the performance to be scaled up using more hardware nodes. The nodes in a cluster form a full mesh topology, which means that each node maintains a connection to each of the other nodes. The cluster has a single master node which is chosen automatically by the cluster and which can be replaced if the current master node fails. An index is a logical namespace which points to primary and replica shards, which are instances managed automatically by Elasticsearch[13]. Each document is stored in a single primary shard. When a document is being indexed, it is indexed first on the primary shard, then on all replicas of the primary shard. A replica is a copy of the primary shard, used to increase failover and performance. The number of primary and replica

[10]Graylog2: https://github.com/Graylog2

[11]Grafana https://grafana.com/

[12]Docker: https://elk-docker.readthedocs.io/

[13]https://www.elastic.co

shards can be manually configured to optimise implementation.

The case study used the implemented framework to carry out several experiments. The experiments demonstrate which attacks and types of communication that trigger different signatures in the rules for IEC 60870-5-104. The experiments were performed manually, which is an expensive and more error-prone approach than automatic testing. Automating the test setup using OpenMUC with automated testsuites for example based on JUnit is therefore a natural next step in improving the test framework.

The experiments aim at demonstrating IEC 60870-5-104 traffic that triggers alerts using IDS rulesets for this protocol. Normal traffic did not trigger any of the signatures specified in the IEC 60870-5-104 rule-set. Several methods and tools were used to trigger the signature-based rules. The protocol-based rules and trafficpattern-based rules were demonstrated using a man-in-the-middle packet injection and demonstration unauthorised traffic. This case study detected some compatibility issues between the two IDSs that relatively easily could be mitigated.

The experiments also demonstrated a SYN flood denial-of-service attack and analysis of the additional latency applied by the Logstash processing and conversion-tools. During some of the experiments, Suricata triggered a larger amount of alerts than Snort. This is because the default alert threshold settings are different for Suricata and Snort, respectively. The purpose of these is to avoid that the security operations centre becomes swamped by too many alerts if the same rule triggers too many times. These values can be manually changed by editing the threshold configuration file for both solutions.

The platform should in the future be extended to use the OpenMUC j60870 library to simulate all the IEC 60870- 5-104 communication. This library allows simulating all IEC message types on both the client and server side. Both QTester104 and IEC server have their limitations because they not are complete implementations of the IEC 60870-5-104 protocol.

7 CONCLUSION

This paper proposes a SCADA Intrusion Detection System Test Framework that can be used to conduct research on security in SCADA communication and validate existing IDS signatures before implementing them in a production network. The framework consists of four parts; attacker side, SCADA target side, IDS side and SIEM side. It simulates real-time IEC 60870-5-104 communication between a client and a server. Both Suricata and Snort IDS are included in this framework to analyse the network traffic and detect malicious activity. The solutions are compared in a comparison dashboard that was created for the framework.

There is generally little difference between Suricata and Snort's ability to detect malicious traffic. Suricata is mostly compatible with signatures written in Snort's lightweight rules description language. We discovered some compatibility issues that can be mitigated by modifying the ruleset slightly. All IEC 60870-5-104 signatures were triggered for both IDSs after these modifications.

The perceived latency is generally higher for Snort events than for Suricata events. The reason for this is probably the additional processing time applied by the unified2 to eve.json conversion.

8 FUTURE WORK

As future work, we aim at expanding the SCADA Intrusion Detection System Test Framework. The natural next step would be to implement other rule-sets, create custom signatures, use anomaly based intrusion detection features and experience with other IDS packages, like Bro or commercial IDS solutions.

The HMI software that is implemented in the framework is currently only used for simulation purposes. A possible extension of this implementation, could be to interconnect the HMI software with the IEC 60870-5-104 client. It would then be possible to see and simulate the effect of an attack in the HMI. Another possible extension is to implement host based IDS solutions (HIDS) in the framework to analyse machine level activity, detect abnormal login patterns and file changes. New entries in the HIDS and SCADA logs could be shipped to Logstash and be stored in the cluster. This extension would provide the network administrators with more information that can be used to detect malicious activity in the SCADA system.

Another idea is to integrate Apache Hadoop and connect it to the cluster using the ES-Hadoop connector. This would connect the massive data storage and deep processing power of Hadoop with the real-time search and analytics of Elasticsearch. It would also be interesting to use the machine learning feature included in X-pack to predict attacks based on past events. Another possibility is to implement the IEC 62351 security enhancement of IEC 60870-5-104 protocol in the framework. Finally, future research includes to perform usability testing and experiments to determine the level of situational awareness for operators using the framework.

ACKNOWLEDGEMENTS

This research has been supported by NC Spectrum AS and CIEM, the Centre for Integrated Emergency Management at University of Agder, Norway. Thanks to the anonymous reviewers for good suggestions on how to improve the paper.

REFERENCES

Brumen, B. and Legvart, J. (2016). Performance analysis of two open source intrusion detection systems. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1387–1392.

Davis, C. M., Tate, J. E., Okhravi, H., Grier, C., Overbye, T. J., and Nicol, D. (2006). SCADA Cyber Security Testbed Development. In *2006 38th North American Power Symposium*, pages 483–488.

Graham, J., Hieb, J., and Naber, J. (2016). Improving cybersecurity for Industrial Control Systems. In *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, pages 618–623.

International Electrotechnical Commission (2006). International Standard IEC 60870-5-104: Telecontrol Equipment and Systems — Part 5-104: Transmission Protocols — Network Access for IEC 60870-5-101 using Standard Transport Profiles. IEC, Geneva, Switzerland. Second Edition (Reference number IEC 60870-5-104:2006(E)).

Ish, J. (accessed 2017). py-idstools: Snort and Suricata Rule and Event Utilities in Python. https://github.com/jasonish/py-idstools.

Jaentsch, K. (2013). IEC Server Manual. http://area-x1.lima-city.de/.

Krauß, D. and Thomalla, C. (2016). Ontology-based detection of cyber-attacks to scada-systems in critical infrastructures. In *2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pages 70–73.

Lahmadi, A. and Beck, F. (2015). Powering Monitoring Analytics with ELK stack. In *9th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2015)*.

Maynard, P. (accessed 2017). Ettercap-104-mitm: Plugin for IEC 60870-5-104.

Olsen, R. L. (accessed 2017). QTester104 iec 60870-5-104 protocol tester. https://sourceforge.net/projects/qtester104/.

Pihelgas, M. (2012). *A Comparative Analysis of Open-Source Intrusion Detection Systems*. PhD thesis, Tallinn University of Technology & University of Tartu.

Wei, D., Lu, Y., Jafari, M., Skare, P., and Rohde, K. (2010). An integrated security system of protecting smart grid against cyber attacks. In *Innovative Smart Grid Technologies (ISGT), 2010*, pages 1–7. IEEE.

White, J. S., Fitzsimmons, T., and Matthews, J. N. (2013). Quantitative analysis of intrusion detection systems: Snort and suricata. *SPIE Defense, Security, and Sensing*, pages 875704–875704.

Wong, K., Dillabaugh, C., Seddigh, N., and Nandy, B. (2017). Enhancing Suricata intrusion detection system for cyber security in SCADA networks. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–5.

Y. Yang, K. McLaughlin, B. Pranggono, T. Littler, S. Sezer, and H. F. Wang (2013). Intrusion Detection System for IEC 60870-5-104 Based SCADA Networks. Technical report, Queen's University Belfast and Brunel University.

Yang, Y., McLaughlin, K., Littler, T., Sezer, S., Im, E. G., Yao, Z. Q., Pranggono, B., and Wang, H. F. (2012). Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid scada systems. In *International Conference on Sustainable Power Generation and Supply (SUPERGEN 2012)*, pages 1–8.