Towards A Stateful Analysis Framework for Smart Grid Network Intrusion Detection

BooJoong Kang  Kieran McLaughlin  Sakir Sezer
Centre for Secure Information Technologies, Queen’s University Belfast
Belfast Northern Ireland, United Kingdom
{b.kang, kieran.mclaughlin, s.sezer}@qub.ac.uk

Cybersecurity is a primary issue in the development of smarter grid systems. Smart grid systems utilize a number of application protocols in order to implement their devices and services, and the information in the application protocols is useful for intrusion detection which is one of major security solutions. Stateful analysis based intrusion detection monitors network and system behaviours and keeps tracks of the behaviours in order to make detection decisions. In smart grid systems, monitoring these behaviours requires expert knowledge and tailoring for particular application protocols. In this paper, we present a framework for smart grid intrusion detection allowing stateful analysis methods to define its stateful rules that can be run on an open source network intrusion detection system, Suricata, in order to process their stateful analysis. A stateful rule defines a particular state of smart grid devices and will be examined with incoming network traffic in order to find any match. We also develop an application for IEC 61850 stateful analysis to show how the proposed framework can be implemented and work.

1. INTRODUCTION

Securing smart grid systems becomes one of major security research topics as cyber threats to smart grid systems have been growing. Since we discovered Stuxnet in 2010, physical infrastructures of smart grid systems in the real world have been under attack compromising and taking control over smart grid systems. It has been reported that cyberattacks caused real damages to the real world smart grid systems (Kushner 2013). In December 2015, an advanced persistent threat (APT) type cyberattack via BlackEnergy caused power outages for at least 80,000 customers in Ukraine (Assante 2016). Stuxnet and BlackEnergy have demonstrated that “security by obscurity” is no longer an adequate scheme for smart grid systems.

Cybersecurity protection against cyberattacks involves not only threats against data exchange between systems, but also security threats within systems as a result of the intrusion of illegitimate messages or malicious activities (IEC 2007). In response to these issues, a number of IT security techniques have emerged. For example, the IEC Technical Committee 57 Working Group 15 proposes appropriate countermeasures such as encryption, access control, firewall, authentication, intrusion detection system (IDS), and antivirus/antimalware (Cleveland 2005).

For internal system cybersecurity, one of the vital security mechanisms is intrusion detection via an intrusion detection system (IDS). Such an IDS can be deployed in smart grid systems to monitor the traffic on supporting IT networks and attempt to identify malicious or suspicious activities. In addition, smart grid systems frequently contain legacy systems that are difficult to update, patch, or protect by conventional IT security techniques. Therefore, new and robust approaches are required to detect malicious behaviours and actions in smart grid communications that take account of the special environments, operations and behaviours typically encountered in the electricity domain that are not present in a generic or commercial IT domain.

Compared with traditional IT networks, smart grid system networks are likely to have distinguishing features, such as a fixed number of communication devices, limited and known communication protocols, and consistent communication and behaviour patterns. Based on such assertions, it is hypothesised that an intrusion detection methodology can provide an effective approach that is able to identify events and communication behaviour representative of cybersecurity attacks or unexpected behaviours (Cheung 2007).

As efforts to develop intrusion detection methodologies, various machine learning algorithms
have been investigated with different types of attributes including network and application layer information. Many stateful analysis based IDSs have also been proposed, which require building a specification of devices or systems and monitoring network and system behaviour in order to identify any deviation from the normal behaviour or known malicious activities. In terms of applying these IDS technologies to smart grid systems, it is important to handle detection of information at the application layer, which can also involve inspection of the packet payload. This means that an IDS focusing at the depth of the application protocols requires expert knowledge and tailoring for particular application protocols.

In this paper, we present a framework for smart grid intrusion detection allowing stateful analysis methods to define its stateful rules that can be run on an open source network intrusion detection system (NIDS), Suricata, in order to process their stateful analysis. A signature of Suricata can specify detection plugins for additional packet inspection tasks rather than basic string comparison. A stateful rule defines a particular state of smart grid devices and will be examined with incoming network traffic in order to find any match. We also develop an application for IEC 61850 in order to show how the proposed framework can be implemented and work.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 describes our proposed framework, and Section 4 explains an IEC 61850 stateful analysis of the proposed framework. Section 5 discusses some limitations and other issues. Finally, Section 6 concludes the paper and outlines avenues for future work.

2. RELATED WORK

A traditional IDS approach is signature-based detection that maintains already known signatures or identities for each specific intrusion event. Currently, there are some contributions in signature generation for smart grid systems (Digital Bond) but it is difficult to establish a comprehensive set of signatures because there is lack of detailed information existing cyberattacks in smart grid systems. Anomaly detection establishes normal behaviour profiles of the system and identifies anomalous behaviours, as intrusions, which deviate from the normal behaviour profiles. Anomaly detection approaches can be categorised as statistical-based, heuristic, rule-based, and learning-based approaches, which are used to identify events considered outside normal operational bounds.

Genge et al. proposed a whitelist approach for network traffic anomaly detection in critical infrastructures (Genge 2014). Snort rules are used as detection rules, which include source/destination IP addresses, source/destination port numbers and a protocol. With these Snort rules, the IDS can whitelist allowed traffic and detect prohibited connections. However, an attacker can still exploit defined connections or legitimate machines. For example, a man in the middle attack is possible to manipulate the traffic of two legitimate machines as described in (Kang 2015).

Premaratne et al. investigated statistical properties of normal and attack traffic (Premaratne 2010). They launched simulated attacks and packet sniffing attacks on IEDs and showed that those attacks can be detected by the statistical information such as the number of sessions, ARP packet rates and etc. However, there is no explain how to derive Snort rules for this statistical detection.

Recently, researchers start investigating application layer information such as requested commands and transmitted measurements in order to identify anomalies in terms of device behaviours. Hong et al. derived rules to identify anomalies for GOOSE and SV protocols in IEC 61850 (Hong 2014). The rules define normal and critical states, and are processed in their own IDS over incoming traffic.

Pan et al. defined a state as a list of measurements and a path as a sequence of states (Pan 2015). Normal, fault attack scenarios are also defined and common paths for each given scenario are extracted in order to be compared with observed paths. If there is no common path, or more than one common path that matches with an observed path, they detect the observed path as unknown or uncertain, respectively.

Almalawi et al. proposed consistent and inconsistent states of SCADA systems for anomaly detection (Almalawi 2014). A state represents a combination of SCADA data and the consistency of the state is determined by a statistical likelihood of the state. A state that statistically has the higher likelihood of being generated by the same mechanism that generated the majority of states is a consistent state. Any inconsistency indicates a malicious action.

Various machine learning algorithms have been utilized as a method of anomaly detection as well. Caselli et al. generate and update discrete-time Markov chains for monitored industrial control system (ICS) device operations and continuously compute a weighted-distance among Markov chain states to detect anomalies (Caselli 2015).

Yoo et al. proposed a NIDS for anomaly detection using one-class support vector machine (Yoo 2014). They monitor network traffic and collect header information, GOOSE information, n-gram of MMS payload and traffic rates for anomaly
Towards A Stateful Analysis Framework for Smart Grid Network Intrusion Detection

Kang ● McLaughlin ● Sezer

Detection. Expectation maximization (EM) clustering is used to remove outliers from the training set. There is no evaluation with attack traffic and no explanation how to handle categorical features such as dataset.

Hink et al. investigated several machine learning algorithms for anomaly detection in smart grid systems (Hink 2014). They monitor 116 PMU measurements from 4 synchrophasors and control panel logs, Snort logs and relay logs for each synchrophasor.

Smart grid systems utilize application protocols in order to implement their devices and services, and the information in the application protocols is useful for monitoring the devices and services in order to identify behavioural anomalies. Stateful analysis based IDSs build and update state information of devices or services in order to monitor network and system behaviours. In terms of applying these IDS technologies to smart grid systems, it is important to handle detection of information at the application layer, which can also involve inspection of the packet payload. This means that IDS focusing at the depth of the application protocols require expert knowledge and tailoring for particular protocols. This paper, we present a framework for stateful analysis based IDSs, allowing stateful rules to specify particular states of smart grid devices that can be used to build a specification and to examine with incoming traffic.

3. STATEFUL ANALYSIS FRAMEWORK

In this section, the propose framework for stateful analysis in smart grid systems is described. Essential functions for stateful analysis and Suricata are also explained to help understandings of the proposed framework.

3.1 Essential functions for stateful analysis

We have identified essential functions for stateful analysis from the existing work as follows:

(i) stateful rules,
(ii) decoders,
(iii) rule match engines,
(iv) state manager.

A set of stateful rules is required to define critical or normal states of protected devices. If any application layer information is being used in stateful rules, then proper application protocol decoders are required to decode packets and collect the application layer information.

Rule match engines are responsible for the major task of stateful analysis that examines incoming packets with stateful rules and passes match results back to the main detection engine. Two common inspections can be processed for stateful analysis: contents inspection and state inspection. The state manager is responsible for maintaining and updating the states of protected devices that will be used in future examination.

3.2 Suricata

Suricata is an open source NIDS that has been developed by the Open Information Security Foundation (OISF) with financial support from the US Department of Homeland Security (Garcia-Teodoro 2009). Suricata is multi-threaded since its first release in 2010, while Snort will be multi-threaded from the version 3 (Caswell 2007). This allows Suricata to maximize its ability to receive and process packets (Park 2016).

Figure 1 shows the packet pipeline in Suricata that consists of 4 different modules: capture, decode, stream and detect modules. The capture module collects packets from a given pcap device, e.g. eth0, and then acts as a thin wrapper in order to make packets compatible with decoders. The decode module converts packets to Suricata support data structures. The stream module keeps track of TCP connections and exists of two engines: stream tracking and reassembly engines. The stream tracking engine monitors the state of a connection and the reassembly engine reconstructs the flow as it used to be. The detection module loads all signatures and initializes detection plugins, creates detection groups for packet routing, and finally runs packets through all applicable signatures.

Figure 2: Suricata signatures
As shown in Figure 2, a Suricata signature consists of action, header and rule options like Snort rules. The action determines what will happen when a signature matches. The header includes a protocol, two pairs of an IP address and a port number for source and destination respectively, and a direction between the two IP addresses. Rule options define additional settings for Suricata and inspection rules in different packet sections and network layers. Each rule option is given as “keyword: settings;” or “keyword” only.

(alert tcp any any -> any [80,8080](msg:“SURICATA HTTP but not tcp port 80, 8080”; flow:to_server; app-layer-protocol:http; sid:2271001;rev:1;)
alert tcp any any -> any 443(msg:“SURICATA Port 443 but not TLS”; flow:to_server; app-layer-protocol:tls; sid:2271003;rev:1;)
alert tcp any any -> any 502(msg:“SURICATA TCP port 502 but not MODBUS”; flow:to_server; app-layer-protocol:modbus; sid:2271018;rev:1;)

**Figure 3: Detection Plugins**

A Suricata signature can specify detection plugins as rule options that perform additional inspection processes and return process results, 0 or 1, which will affect the match result of the Suricata signature. In Figure 3, three signatures have a common rule option for the app-layer-protocol detection plugin that compares the application protocol of the current packet with a given protocol such as HTTP, TLS and so on. If they are the same protocol, the detection plugin will return 1. Otherwise, it will return 0. Note that ‘!’ represents the ‘not’ operation that will convert 1 to 0 and vice versa. More details can be found at (Suricata).

### 3.3 Stateful analysis framework

Figure 4 shows the overall process of the proposed stateful analysis framework for smart grid intrusion detection. The main component of the proposed framework is a stateful analysis plugin that performs the three requirements of stateful analysis: decoders, the rule match engine and the state manager.

<table>
<thead>
<tr>
<th><strong>Algorithm 1: Stateful Analysis</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: a packet ( P ), app-layer protocol ( AP ), content condition ( C_c ), state condition ( C_s ) and a signature id ( sid )</td>
</tr>
<tr>
<td><strong>Output</strong>: the stateful analysis result (0 or 1)</td>
</tr>
<tr>
<td>1. ( P_s \leftarrow \text{Decode}(P, AP) )</td>
</tr>
<tr>
<td>2. for ( \forall c \in C_c ) // Contents Inspection</td>
</tr>
<tr>
<td>3. if ConditionCheck(( P_s, c )) = 0</td>
</tr>
<tr>
<td>4. return 0</td>
</tr>
<tr>
<td>5. end for</td>
</tr>
<tr>
<td>7. for ( \forall s \in C_s ) // State Inspection</td>
</tr>
<tr>
<td>8. if ( s ) is not set in the global state flags</td>
</tr>
<tr>
<td>9. StateUnset(sid)</td>
</tr>
<tr>
<td>10. return 0</td>
</tr>
<tr>
<td>11. end for</td>
</tr>
<tr>
<td>12. StateSet(sid)</td>
</tr>
<tr>
<td>13. return 1</td>
</tr>
</tbody>
</table>

As shown in Algorithm 1, a packet will be decoded according to the application protocol before the packet inspection and examined through two inspection methods: the content inspection and the state inspection. Content and state conditions are given as a set of conditions that should match. In the state inspection, it simply checks that corresponding flags have been set in the global state flags that are shared among different invocations of the stateful analysis plugin and represent the current state of protected devices. If any of the corresponding flags is not set, then it will return 0.

The contents inspection checks all conditions and return 0 if any of the conditions is not met. However, content conditions are different from
application protocol to application protocol. Therefore, the condition check method, the \textit{ConditionCheck} function should be able to handle different application protocols or should be different methods for each application protocol.

\textit{StateSet/Unset} functions are responsible for the state update that set/unset a given state \textit{sid}. In the \textit{ConditionCheck} function, the \textit{StateUnset} function will be processed if some conditions are not met. For example, if there is a stateful rule detecting a read response containing a specific range of values for an attribute, then the state manager will set the flag of this stateful rule for every match. However, if there is a read response containing a value of the attribute, not in the range, then the flag should be reset.

4. IEC 61850 STATEFUL ANALYSIS

In this section, an implementation of IEC 61850 state analysis is described as an application of our framework. Experiments on a testbed are also presented to show how it works.

4.1 IEC 61850

IEC 61850 is an object oriented substation automation standard that defines how to describe devices and exchange the information (Mackiewicz 2006).

\textbf{Table 1: MMS objects and services in use}

<table>
<thead>
<tr>
<th>MMS Object</th>
<th>IEC 61850 Object</th>
<th>MMS Service in Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Process VMD</td>
<td>Server</td>
<td>Initiate, Conclude, Abort, Reject, Cancel, Identify</td>
</tr>
<tr>
<td>Named Variable Objects</td>
<td>Logical Nodes and Data</td>
<td>Read, Write, InformationReport, GetvariableAccessAttribute, GetNameList</td>
</tr>
<tr>
<td>Named Variable List Objects</td>
<td>Data Sets</td>
<td>GetNameVariableListAttributes, GetNameList, DefineNamedVariableList, DeleteNamedVariableList, Read, Write, InformationReport</td>
</tr>
<tr>
<td>Journal Objects</td>
<td>Logs</td>
<td>ReadJournal, InitializeJournal, GetNameList</td>
</tr>
</tbody>
</table>

The IEC 61850 information model is based on two main levels of modelling: the breakdown of a real device (physical device) into logical devices, and the breakdown of logical device into logical nodes, data objects and data attributes. IEC 61850 also standardizes the set of abstract communication services (Abstract Communication Service Interface–ACSI) allowing for compatible exchange of information among components of a power system. IEC 61850 offers three types of communication models: client/server type communication services model, a publisher-subscriber model and sample values model for multicast measurement values. For the client/server type communication services model, IEC 61850 defines mappings between the abstract services/objects to a specific protocol such as manufacturing messages specification (MMS). Table 1 shows the MMS objects and services in use within IEC 61850.

4.2 IEC 61850 stateful analysis

The AIT SmartEST lab offers an environment for testing, verification and R&D in the field of large scale distributed energy system integration and smart grids applications. The laboratory setup includes a commercial off-the-shelf 20 kW PV inverter connected to a PV simulator as power source and a laboratory current sink as model for the power grid connection. The inverter itself has no IEC 61850 capabilities. These are added by a gateway component on the basis of Raspberry Pi (R-Pi) hardware, which essentially serves as a programmable gateway between an IEC 61850 SCADA network and the inverters in-built Modbus interface (Bründlinger 2015). This testbed has been used our previous work that demonstrated attacks to IEC 61850 devices (Kang 2015).

\textbf{Figure 5: Testbed setup}

Figure 5 shows the testbed setup. Our NIDS is connected to the switch and getting mirrored traffic.
from the switch. Suricata is running on the NIDS with pre-defined stateful rules for the critical states of the PV inverter. The attack scenarios from our previous work (Kang 2015) are demonstrated to test our NIDS. The attack scenarios are related to the active power limitation that leads to reboot the PV inverter when it is set to a low value, less than 10 in our setup.

Figure 6 shows an example of stateful rules for IEC 61850 and the inspection process of IEC 61850 stateful analysis. The header of every incoming packet will be examined in the core engine of Suricata. If any match with stateful rules has found, our stateful analysis plugin will be invoked along with a relevant packet. Then, the packet will be decoded according to the given application protocol. Note that the first passing argument is the application protocol of the packet.

In the stateful analysis plugin, the packet will be decoded according to MMS and application information such as a MMS service, names of devices, commands and measurements will be collected. The collected application information will be questioned with the given conditions in the matched stateful rule in order. If any condition is not met, then the rest inspection processes will be skipped and ‘false’ will be returned to the core engine of Suricata to indicate “no match”. Otherwise, state conditions will be examined and the final match results will be passed to the core engine.

The contents inspection will be different by which MMS service is being questioned. In this example, the MMS write request is investigated as given in the example of stateful rules. By monitoring the write request, any attempt to overwrite settings and commands for protected devices can be monitored. The example specifies any write request to ‘FORTE_GWLDevice1’ that overwrites ‘DRCC1$SP$MaxWLimPct$setMag$f’ with any value less than 10.

Figure 7 shows a packet of writing the attribute to 10 injected by an attacker. Since any encryption algorithm is not used in MMS, all information can be collected as shown in Figure 7. In the payload of the write request, ‘domainId’ represents the logical device and ‘itemId’ represents the attribute that this write request wants to overwrite. At the end of the payload, the writing value is also given as well as the type of the value, (i.e. floating-point). This information will be decoded and collected in the decoder, and will be examined with the given conditions in the contents inspection.

In the stateful analysis plugin, the packet will be decoded according to MMS and application information such as a MMS service, names of devices, commands and measurements will be collected. The collected application information will be questioned with the given conditions in the matched stateful rule in order. If any condition is not met, then the rest inspection processes will be skipped and ‘false’ will be returned to the core engine of Suricata to indicate “no match”. Otherwise, state conditions will be examined and the final match results will be passed to the core engine.
A read request queries current values of attributes and a read response will send the current values of the requested attributes back. Figure 8 shows a read request for 'DRCC1$SF$MaxWLimPct$setMag$f', the active power limitation of the PV inverter. The read response will contain the current value. In this example, the value is 10.

Figure 9 shows another stateful rule for the low active power limitation in a read response. As shown in Figure 8, a read response does not contain the domain and attribute information that are required by the stateful rule to compare with the conditions. The missing information is in the corresponding read request so the information should be stored for the future inspection. When any read request is processed, requested domain and attribute information will be stored along with the 'invokeID', in order to pair the read request and response. Then, the stored information can be retrieved with the 'invokeID' when any stateful rule for the read response is processing.

```
alert tcp any 102 -> any any (msg:"Low active power limitation"; stateful-analysis:iec61850,read-response,FORTE_GWLDevice1,DRCC1$SP$MaxWLimPct$setMag$f,v<=10,3310011;sid:3190008;)
```

Figure 9: Low active power limitation with read response

5. DISCUSSION

We designed our framework to use Suricata rules as stateful rules rather than implementing independent pre-processors or detection engines. By this design, it is possible to implement flexible stateful analysis methods by defining stateful rules without major modification in the plugin. However, there is an issue in this design. Stateful rules will be examined with a packet in order. This order will affect the final decision since the state inspection can be different by this order. Therefore, it is required to consider the order of stateful rules when we design a stateful analysis and generate stateful rules.

In our implementation on Suricata, all functionalities are implemented in a single detection plugin of Suricata. For efficiency, decoders can be separated from the detection plugin and implemented as decoders of Suricata. In order to decode a packet only once, we check whether the packet has been decoded or not, before the decoding procedures. Without this check routine, every stateful rule will decode the same packet that degrades the efficiency.

The information of read requests should be stored when any relevant stateful rule with read response exists in the current stateful rules, in order to avoid wasting system resources. In addition, for every read response, the corresponding read request information should be removed after the read response is processed. Otherwise, it will cause memory issues.

6. CONCLUSION

In this paper, we presented a framework for smart grid NIDS, utilising stateful analysis methods to define stateful rules. A stateful rule goes beyond normal signature based detection on single packet features, and defines a particular state associated with smart grid devices. An open source NIDS, Suricata, has been adapted to enable states to be examined with incoming network traffic in order to find any stateful rule matches, as required by the system operator. The demonstrated framework and implemented toolset has the benefit of being capable of operating stateful IDS rules as well as the usually presented signature based approach offered by Snort IDS.

Furthermore an application for IEC 61850 stateful analysis was developed to show how the proposed framework can be implemented and operated within a realistic environment. In the experiment with the presented testbed, we showed that manipulated attacks with dangerous values can be detected by the proposed NIDS framework. With the proposed framework implemented as a functioning toolset, future work is proposed to include functionalities for other application protocols for smart grid systems such as Modbus and IEC 60870-5-104.

REFERENCES


Towards A Stateful Analysis Framework for Smart Grid Network Intrusion Detection

Kang ● McLaughlin ● Sezer


