Dealing with Advanced Persistent Threats in Smart Grid ICT Networks

Florian Skopik, Ivo Friedberg, Roman Fiedler
Safety and Security Department
AIT Austrian Institute of Technology
firstname.lastname@ait.ac.at

***Abstract*—With the increasing use of novel smart grid technologies, a comprehensive ICT network will be established in parallel to the electricity grid, which due to its large size, number of participants and access points will be exposed to similar threats as those seen on the current Internet. However, modern security systems that are applied in today's highly dynamic ICT networks, including malware scanners and intrusion detection systems, apply a kind of black-list approach, where they consider only actions and behavior that match to well-known attack patterns and signatures of malware traces. We argue that for the smart grid a more restrictive approach, that cannot be circumvented by customized malware, will increase the security level tremendously. Therefore, in this paper we present a smart white-list approach. Our anomaly detection technique keeps track of system events, their dependencies and occurrences, and thus learns the normal system behavior over time and reports all actions that differ from the created system model. The application of such a system is promising in a smart grid environment which mostly implements well-specified processes, resulting in rather predictable and static behavior. We demonstrate the application of the system in a small-scale pilot case of a real utility provider.**

***Keywords*-anomaly detection, event correlation, ict security**

I. Introduction

A complex ICT network is currently being established in parallel to the existing power grid in order to make it smarter. This way, sensors can report operational data within fractions of seconds, and enable control loops to react much more dynamically to changing load conditions, and finally, enable more efficient energy distribution. Unfortunately, this added complexity makes the power grid also more vulnerable to non-conventional attacks. While in the past utility providers and other grid stakeholders had to deal with safety concerns and physical security only, a large ICT network with access points in every household opens up entirely new attack surfaces.

Advanced persistent threats (APTs) [1] refer to attacks carried out typically by a group with both the capability and the intent to persistently target a specific entity undercover. One prominent example that raised awareness of APTs in the SCADA (supervisory control and data acquisition) domain is the Stuxnet malware [2] which specifically targeted the widespread Siemens WinCC/PCS 7 systems. SCADA systems are an essential cornerstone of smart grid implementations. This type of computer systems control industrial processes on a large scale that can include multiple sites over large distances. In the smart grid, SCADA systems are used to remote control the operational behavior in power plants, distribution stations and substations. Since SCADA systems are increasingly coupled to the Internet to enable cost-efficient monitoring and remote maintenance, their security is a major concern of today's utility providers.

APT Scenario. In a common scenario, an attacker tries to first intrude into the corporate LAN, e.g., by exploiting a weakness in the Web server which enables SQL injection. This kind of attack gives the intruder elevated permissions and the chance to take over the Web server for his own purposes. Assuming that the intruder uses a zero day exploit, i.e, an unknown vulnerability, it is highly unlikely that this attack will be discovered easily. In a second step, once the intruder has established a beachhead, he uses the Web server to drive further attacks within the network. For example, he will try to expand his access to critical systems, such as a SCADA application server. Here he could try to change the configuration of control programs. If he is cautious, this second stage might be undiscovered, because all commands that are issued to the SCADA application server come from a trusted entity within the corporate LAN. As a third step, the intruder could use the application server to send fabricated commands to remote intelligent electronic devices (IEDs), e.g., in substations or relay stations, to disturb critical processes and produce considerable harm.

Contribution. The contributions in the paper are as follows:

- *APTs in Smart Grid ICT Networks:* We raise awareness for advanced persistent threats in smart grid ICT networks and outline the challenge of detecting them.
- *Anomaly Detection Mechanism:* We introduce a novel concept of dealing with APTs that works especially well for SCADA backends due to their rather restrictive and predictable behavior.
- *Pilot Case and Discussions:* We demonstrate the applicability of the proposed anomaly detection approach in a real smart grid ICT network.

The remainder of the paper is structured as follows. Section II provides an overview about related work. Section III introduces the rough concept of our proposed anomaly detection approach, while Sect. IV explains the model in detail. Then, Sect. V discusses its application in a areal context and Sect. VI concludes the paper.

II. Related work

The electric grid is perhaps the most critical infrastructure today, and thus, safety, reliability and availability are of top priority. In course of the roll out of smart grid components, security and privacy issues have become the focus of many discussions [3], [4]. Especially, security in the SCADA domain has been extensively discussed in recent years [5], and especially after the Stuxnet [2] incidents. Many of today's systems still lack security and authentication in the design, deployment and operation. Therefore, anomaly detection approaches have been massively studied [6] for decades – especially in context of network intrusion detection systems for computer networks [7]. However, since systems become increasingly complex and interconnected, novel approaches that capture the big operational picture of a large-scale system, such as the smart grid, and that can cope with today's large amount of data, aggregated from distributed systems, need to be developed. Our proposed approach utilizes some already existing techniques, but applies them in different context or on a larger scale. For instance, approaches in bioinformatics use fingerprinting techniques for data reduction, and alignment mechanisms for fast and fuzzy similarity search. These techniques have been used for single systems in the past, such as for detecting illegitimate accesses in database systems [8], and are now being applied on a much larger scale.

III. APT Discovery through Anomaly Detection

Discovering APTs requires a detection method which does not rely on predefined signatures, as anti-virus scanners or most intrusion detection systems do. The reason for this is that on the one side, APTs may use widely unknown zero day exploits to intrude into a system, on the other side the application of social engineering [9] allows the infiltration of systems without the exploitation of technical weaknesses. Thus, we introduce an approach that heavily relies on statistical analysis of system behavior reflected in system log files to detect anomalies that are potentially effects of APTs.

Approach. Almost every modern ICT component and service produces logging data to report events, internal state changes, and committed actions. This data is a valuable source to establish situational awareness about the current status of ICT networks and is utilized by our approach. From bottom to top, we utilize software to collect distributed log files and maintain the temporal order of log messages (in particular Graylog2). Once all messages have been aggregated and are available through a common interface, the system creates and extracts search patterns from single log lines (cf. Sect. IV). These search patterns are initially random text fragments and are periodically refined based on their occurrence and usefulness in later stages of the analysis. Using information about occurred search patterns enables the system to determine the event type(s) that caused the corresponding log lines (e.g., a connection has been opened, a switching command issued etc.). Again, this classification process is periodically refined using a self-learning system approach. So, on top of the third layer the system has a clear view about recent events in the infrastructure. Finally on the top layer, the focus of our algorithm lies on the timely correlation of different events, including their relative position to each other. For that purpose it creates hypotheses about causes and effects and evaluates them for a certain time span. If a hypothesis can be confirmed as valid, it becomes a part of the system model. An ongoing statistical analysis of correlation results allows the system to infer the degree of deviation of these "if-then" hypotheses and therefore, a degree of anomalous behavior.

IV. Model Definition

The proposed anomaly detection method is a self-learning approach that continuously creates hypotheses about correlated events and tests them at run-time. We distinguish between (i) a *learning phase* (to capture a stable system model) and (ii) an *operational phase* (to trigger alerts in case of deviations from the system model). However, both phases are executed in parallel[1],which means the system continuously learns also in its productive operation, and thus, has the ability to adapt to changing situations.

A. Basic Definitions

A basic unit of logging information, e.g., one line for line-based logging, one binary log data record or one XML-element is called a log-atom L_a which consists of a series of single symbols s (Eq. 1).

$$L_a = s_1 \dots s_n \tag{1}$$

A log event L_e (Eq. 2) is the association of a log-atom L_a with a timestamp t which describes when L_a has been created.

$$L_e = \langle L_a, t \rangle \tag{2}$$

A search pattern $\hat{P}_s$ (Eq. 3) is a substring (an *n-gram*) of a log-atom L_a. Search patterns are randomly created on the fly (see later).

$$\hat{P}_s = s_{1+i} \dots s_{m+i},\ where\ 0 \leq i\ and\ m+i \leq n \tag{3}$$

The process of vectorization transforms a log-atom L_a into an n-dimensional pattern vector - the so-called atom-vector $\vec{P}$ (Eq. 4). This step massively reduces the amount of data to be processed in the next steps and speeds up the overall anomaly detection tremendously. During the process, all known search patterns $\{\hat{P}_s\}$ are looked up in the log-atoms and component values $p_i \in \{0,1\}$ are set depending on a match or mismatch of the search process.

$$\vec{P} = p_1 \dots p_n,\ where\ p_i \in \{0,1\} \tag{4}$$

Smoothing and folding is the process of enriching the pattern vector with additional components that contain further attributes, e.g., coming from systems not under scrutiny. This enrichment vector $\vec{P'}$ can also carry contextual information, e.g., if a maintenance operation was going on when a log

[1]not considering the startup phase here

line has been produced. Notice that $\vec{P'}$ does not influence the further classification.

Log event classification is the process of determining to what event class C a log-atom L_a belongs to. One L_a can belong to a multitude of classes, e.g., a log-atom might be an 'incoming connection event', an 'ssh service event' and an 'IP-zone X service event' at the same time. On the other side, a log-atom might also belong to no class at all. Notice that L_a is categorized, not L_e, because the categorization is timestamp-independent.

An event E (Eq. 5) is finally a log-event L_e that belongs to a known (meaningful) class C. Only these events are further investigated by the anomaly detection system.

$$E = \left\langle L_e, \vec{P'}, C \right\rangle \tag{5}$$

A hypothesis H (Eq. 6) is a non-validated correlation rule of two events E_1 and E_2. The relation $\rightarrow$ is the logic consequence operator (i.e. $E_1 \rightarrow E_2$), while the time window t_w describes the time span (relative to t from L_e that triggered E_1) in which the implication has to hold. The system automatically creates such correlation rules and subsequently tests them in order to learn about event dependencies (see later). Notice, currently independent rules with a fixed t_w of either 10ms, 100ms or 1000ms are created. In the future we plan to let the system dynamically adapt t_w, i.e., starting with a high value and then step wise reduce it until rules just match.

$$H = \langle E_1, E_2, \rightarrow, t_w \rangle \tag{6}$$

The evaluation process determines over a longer time window t_e the number of rule matches and mismatches[2]. If the predictability, i.e., one of these numbers remains stable (or at least predictable), the initial hypothesis becomes a confirmed hypothesis. Otherwise, this means the evaluation results do not become stable after t_e, a hypothesis is discarded without further action. The collection of all confirmed hypotheses forms the system model M (Eq. 7), which describes the normal system behavior.

$$M = \{H \,|\, H \; is \; confirmed\} \tag{7}$$

In the operational phase of our anomaly detection approach, the confirmed hypotheses in M are continuously evaluated. This means the stability of the number of matches and mismatches for each single *confirmed* correlation rule is evaluated and if a significant statistic deviation is measured an alert is raised. An advanced evaluation mode accounts for multiple alerts that are triggered at the same time, and therefore amplifies the raised alert level. Notice, infrastructure changes (hardware and software; i.e., everything that alters the usual log output) will render a part of the learned hypotheses invalid and will need a cleanup.

[2]Notice, t_e is not constant but reduced over time depending on the stability of the rule evaluation results. This means in the beginning t_e will be rather long to easily find stable values, while later it will be reduced to allow for faster reactions on rule violations

B. Search Pattern Creation and Log Line Vectorization

Search patterns $\hat{P}_s$ are not created manually but by the system based on currently processed log atoms L_a. The basic idea is that the system uses patterns to cover/index the occurring log atoms best. Thus, it creates patterns for (i) a non-indexed L_a (e.g., when new log sources are being connected), but also (however less frequent) for (ii) a well-covered L_a in order to refine the system model M. For that purpose we apply a simple but effective token bucket algorithm. Here, every processed L_a increases the number of tokens in a bucket. If there are enough tokens in the bucket to 'buy' a new pattern, the system does so. An important configuration parameter is the price of a pattern. This is an easy way to overcome the *pattern balancing problem*. This problem deals with the fact that rarely occurring log atoms are not properly indexed by patterns in the current pattern vector, while frequently occurring log lines are indexed (i.e., covered with patterns) very well. However, especially the rarely occurring log atoms are the most interesting ones since they represent exceptional events. For that purpose we configure a cheaper price for rare log atoms, i.e., less tokens are withdrawn from the bucket when a pattern for a rare line is created, and thus, allows the creation of several patterns for one rare L_a. On the other side, buying a pattern for a well covered type of log atom is expensive, but still affordable from time to time.

The process of log line vectorization parses incoming L_a byte-wise and creates $\vec{P}$ according to matching search patterns. This processes uses simple but effective hashtable lookups to achieve linear complexity.

C. Event Classification

After the vectorization of an L_a follows the classification. An event class C (Eq. 8) is defined as the combination of a mask $\vec{C_m}$ and a target value $\vec{C_t}$.

$$C = \left\langle \vec{C_m}, \vec{C_t} \right\rangle \tag{8}$$

Each generated atom-vector $\vec{P}$ is assigned to one or more classes wrt. the result of a bit-wise comparison (Eq. 9).

$$\vec{C_t} \equiv \vec{P} \wedge \vec{C_m} \tag{9}$$

Event classes C are automatically defined using the following scheme. Each log-atom L_a is characterized by its corresponding $\vec{P}$. If a $\vec{P}$ does **not** match to any defined class, a new class is created, using the same token bucket algorithm as in the search pattern creation step.

D. Hypothesis Evaluation and System Model Updates

Hypotheses are continuously created after correlating two random event classes by setting them in an implication relation $E_i \rightarrow E_j|_{i \neq j}$. They are then evaluated and can become part of the system model. This evaluation process foresees one event queue Q_i for every existing hypothesis H (modeled as rule). All queues Q_i listen to events relevant to the respective rule, i.e. they add $E|_{E=E_1 \vee E=E_2}$. A periodic evaluation process acts on the entries in the respective Q_i as given in Table I.

occurence	evaluation result
$E_1 \wedge \neg E_2$	Given t_w has passed the rule evaluates to **false** and a mismatch counter is increased. E_1 will be deleted.
$E_1 \wedge E_2$	Given both events occurred within t_w the rule evaluates to **true** and a match counter is increased. E_1 and E_2 will be deleted.
$\neg E_1$	All occurrences of E_2 that lack an E_1 are deleted without an evaluation result being returned.

TABLE I
POSSIBLE EVALUATION RESULTS OF A HYPOTHESIS

Both, a match and mismatch counter are evaluated by a significance function sc (Eq. 10) to decide if an hypothesis should be part of the system model M, and – in case $H \in M$ – if alerts should be raised. Since the continuous evaluation of a hypothesis produces a binary stream, the whole process can be interpreted as a Bernoulli process, i.e., discrete-time stochastic process that takes only two values. The function sc currently implements a simple binomial test to evaluate the probability of the current bit-stream of a hypothesis. In case the evaluated rule is part of the system model, the value of θ reflects the probability of an anomaly.

$$\theta = 1 - sc(count_{matched},\ count_{missed}) \tag{10}$$

The subsequent application of Eq. 10 considers the last 10, 100, and 1000 evaluation results θ and tracks their development [10]. Looking at a larger set of evaluation results (e.g. the last 100) can trigger anomalies with higher certainty than those triggered by only 10 results. However, the smaller focus is required to detect short but sudden/dense anomalies.

V. PILOT USE CASE AND DISCUSSION

We evaluated the feasibility of our proposed approach in a real setting and rate its efficiency on data (approx. 200.000 log lines over 3 days) obtained from an Austrian utility provider.

A. Evaluation Preparation

Use Case Description. The testing environment is basically a branch of a real network infrastructure from the utility provider's SCADA system that is coupled to the corporate LAN. The infrastructure consists of (1) a firewall that records incoming connections, (2) a switch that forwards connections to a SCADA system, and (3) the SCADA system itself which issues switching commands and provides measurement readings on request. We collect logs from all these systems, merge them, and apply the proposed anomaly detection algorithm.

Data Set Description. Listing 1 provides a short excerpt of the firewall logs. Notice the first line which describes the semantics of the single fields. Furthermore notice that some information has been removed or altered due to security reasons. Basically, the firewall records accepted and dropped connection attempts from the corporate LAN and the outside world respectively.

```
"Number" "Date" "Time" "Interface" "Origin" "Type" "Action" "Service"
      "Source Port" "Source" "Destination" "Protocol" "Rule" "Rule
      Name" "Current Rule Number" "User" "Information" "Product" "
      Source Machine Name" "Source User Name"
"5487639" "6May2013" "10:59:58" "eth3.842" "mntfwp33" "Log" "Accept" "
      ntp-udp" "ntp-udp" "[removed-url].at" "[removed-url].at" "udp" "
      834" "--> RemoteNet to Sub" "834-MNT_ON_PX_RN_Global" "" "
      service_id: ntp-udp" "VPN-1 Power/UTM" "" ""
"5515746" "6May2013" "11:02:22" "eth4.1151" "mntfwp33" "Log" "Accept"
      "cust-tcp-3505-3506-iec104" "52094" "[removed-url].at" "[removed
      -url].at" "tcp" "839" "--> remnet IEC 104 to VX" "839-
      MNT_ON_PX_RN_Global" "" "service_id: cust-tcp-3505-3506-iec104"
      "VPN-1 Power/UTM" "" ""
```

Listing 1. Firewall log (excerpt from 221 lines).

Listing 2 shows an abstract from the SCADA logs. Here measurement values from the system under supervision are transfered to the requester through a previously established connection via the aforementioned firewall. Logs from the switch are not shown due to space limitations.

```
Tele000592/06.05.2013 11:01:20,12/In /Source=4123/Len=21 Measured
      float/36 Cause=3() Number=1 Common=27/16 floating point Info/
      Obje=12/17/66 Val=5.97 QDS=0x00 Date/Time=06.05.2013/11:01:20
      ,042 - IV=0 DST=1
Tele000593/06.05.2013 11:01:21,17/In /Source=4123/Len=21 Measured
      float/36 Cause=3() Number=1 Common=27/16 floating point Info/
      Obje=12/15/66 Val=99.65 QDS=0x00 Date/Time=06.05.2013/11:01:20
      ,780 - IV=0 DST=1
```

Listing 2. SCADA log (excerpt from 2854 lines).

B. Evaluation Run

During the evaluation run the algorithm created 26 search patterns (Listing 3, separated through double slashes) which are used to classify the log events; determined 14 distinct event classes (that characterize re-occuring events) and 25 stable (i.e., significant and not just random) rules that describe correlations between events on the given data set.

```
"service // .at // 5.2013/11: // 21 Measu //  130 // r=1   Comm //
5/66 // flo // 16  // =3()  // RN_Glob // 11/26/5 // ing poin //
Power/UTM" // easur // 3.842" " // 10/40 //  "-- // 1:24 // teNet to//
 .05.2013/ // =11/26 // cust-tcp // z-ns // mmon=27/1 // fo/Ob //
```

Listing 3. Generated search patterns for event classification.

Listing 4 shows a dump from one of the 25 stable rules, including detailed information of the two correlated events E_1 and E_2 (and their classes respectively) and the rule configuration (t_w (cf. Eq. 6), and significance value (cf. Eq. 10)). In detail, Line 3 and 8 hold $\vec{C_t}$, while Line 4 and 9 describe $\vec{C_m}$ of the corresponding event classes C (cf. Eq. 8) of the two identified events. The applied search patterns are given in Line 6 and Line 11 respectively.

```
 1 ImplicationCorrelationRule [47]
 2  ConditionType: EventClass [18]
 3    value=000000000000000000101001110001000
 4    mask= 000000000000000000111011110001000
 5    lastLine: /In /Source=4123/Len=21 Measured float/36 Cause=3()
        Number=1 Common=27/16 floating point Info/Obje=12/14/66 Val
        =96.83 QDS=0x00 Date/Time=06.05.2013/17:41:41,589 - IV=0 DST=1
 6    triggered by: easur, ing poin, =3(), 16 , flo, 21 Measu
 7  ImpliedType: EventClass [42]
 8    value=000000000000101010100100000000011
 9    mask= 000000000000110101011011010100011
10    lastLine: "1 "eth3.842" "mntfwp33" "Log" "Accept" "ntp-udp" "ntp-
        udp" "[removed-url].at" "[removed-url].at" "udp" "836" "-->
        RemoteNet to Sub" "834-MNT_ON_PX_RN_Global" "" "service_id: ntp-
        udp" "VPN-1 Power/UTM" "" ""
11    triggered by: teNet to ,  "--, 3.842" ", Power/UTM", RN_Glob, .at,
         "service
12  Tw: [-1000ms, 0ms]
13  Aging Score: [704.0]
14  Significance: [0.93]
```

Listing 4. Machine generated system rule consisting of a SCADA event (conditional event) and an implied (firewall) event (at most 1 second before the SCADA event) and instance data of the last match (compare with log).

Anomaly Injection. After the system has started and learned a set of rules, we inject an anomaly to test the detection capabilities. For that purpose, we bypass the firewall and request some measurement values from the SCADA system. Many common attacks would cause a similar behavior deviation, including the APT described in the introduction.

We expect our algorithm to detect this change in the system utilization behavior by discovering that SCADA events occur without a corresponding firewall entry. Subsequently, the rule in Listing 4 should fail and trigger an alert.

C. Evaluation Results

Anomaly Detection Performance. The algorithm analyzes the degree of anomaly for the (series of) incoming events. The subsequent application of Eq. 10 accounts for the last 10 and 100 evaluation results (cf. (t_eval)) and tracks their trend. Figure 1 depicts the outcome for the rule given in Listing 4. As one can see, the injected anomaly at tick 200 causes a value drop immediately for t_eval=10 (blue curve), and shortly delayed for t_eval=100 (red curve). Notice, Figure 1 shows the full experiment run, including the startup phase. Therefore, in the beginning the system needs to collect enough evaluation results (10 or 100) in order to calculate reasonable results. While a short evaluation interval (reflected by the blue curve) allows a fast detection of the anomaly, a longer interval allows a more reliable detection, however will react less dynamically since an adequate number of evaluation results must be calculated prior to detection.

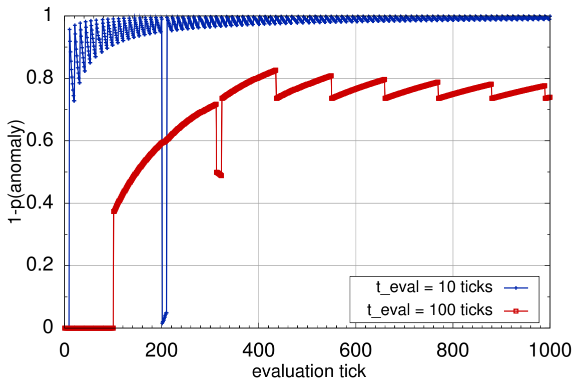

Fig. 1. Evaluation results over a short time interval.

Accuracy and Coverage. The generated system model M describes the overall system behaviour rather than just security related events. Hence, we could define *false-positives* in our approach as behaviour which is considered anomalous with respect to our model but not being security-relevant (e.g., deviating measurement values etc.). However, given the nature of the approach we argue that these types of alerts should not be considered typical false-positives since they are indeed anomalies with respect to the overall system behaviour model. We further argue that this is not a design weakness of our algorithm but an essential strength to detect attacks whose anatomy is not known in advance (APTs). We further define *false-negatives* as deviations of normal system behaviour that are not discovered by the algorithm. This happens in case log lines occur too rarely and thus never become part of the system model. This is not the case in our evaluation dataset.

In the investigated system, after about 7.000 line we can assume that we cover every log line by at least one stable rule. After 50.000 lines we can assume an almost stable rule set (Note, the rule set is continuously adapted and thus never becomes fully stable) covering most possible implications. Figure 2 shows how patterns, event classes and rules reach a level of saturated coverage over time. In case of the SCADA system, the stable values are 11-13 patterns per log line, 7-9 event classes per line and > 5 rules per line. In this setting we reach a throughput of 1.200 lines/sec on a standard PC.

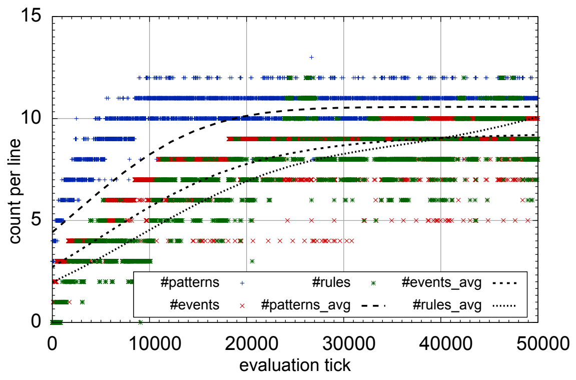

Fig. 2. Saturation of M over a long time interval.

VI. Conclusion and Future Work

In this paper we demonstrated a novel system to detect unforeseen security incidents, possibly caused by advanced persistent threats. In contrast to existing security solutions, including IDSs and SIEMs, our approach is able to automatically adapt to arbitrary log file formats, which make them especially well applicable for (proprietary) industrial solutions. Furthermore, patterns and signatures are not predefined, but are selected by the system based on their statistical relevance. This approach is promising for SCADA systems and communication networks with rather static usage behavior, and thus specifically valuable for the smart grid.

Future work deals with the introduction of hypothesis weights to prioritize system model rules, the correlation of rules in order to improve the derived operational picture in case of attacks, and aging models to remove outdated rules.

References

[1] C. Tankard, “Advanced persistent threats and how to monitor and deter them,” *Network Security*, vol. 2011, no. 8, pp. 16–19, 2011.
[2] R. Langner, “Stuxnet: Dissecting a cyberwarfare weapon,” *IEEE Secu. & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.
[3] P. D. McDaniel and S. E. McLaughlin, “Security and privacy challenges in the smart grid,” *IEEE Secu. & Privacy*, vol. 7, no. 3, pp. 75–77, 2009.
[4] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, “Smart-grid security issues,” *IEEE Security & Privacy*, vol. 8, no. 1, pp. 81–85, 2010.
[5] R. Chandia, J. Gonzalez, T. Kilpatrick, M. Papa, and S. Shenoi, “Security strategies for scada networks,” in *Critical Infrastructure Protection*, 2007, pp. 117–131.
[6] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, pp. 1–58, July 2009.
[7] P. Garcia-Teodoro *et al.*, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.
[8] S. Y. Lee, W. L. Low, and P. Y. Wong, “Learning fingerprints for a database intrusion detection system,” in *ESORICS*, 2002, pp. 264–280.
[9] M. Workman, “Gaining access with social engineering: An empirical study of the threat,” *Inf. Syst. Sec.*, vol. 16, no. 6, pp. 315–331, 2007.
[10] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994.