Submitted for publication. Author Copy - do not redistribute.

Monitoring Advanced Metering Infrastructures with Amilyzer

Robin Berthier and William H. Sanders

Information Trust Institute and Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{rgb,whs}@illinois.edu

Abstract. Advanced metering infrastructures (AMIs) enable two-way communications between smart meters and utility control centers. They are a key component of the smart grid initiative as they encompass a large number of devices and technologies and provide important applications that improve energy consumption and grid reliability. They could also attract diverse cyber-physical threats, and thus they require efficient monitoring solutions to ensure the detection of known and unknown abnormal activities. This paper presents Amilyzer, a specification-based intrusion detection system (IDS) that leverages tight control over authorized AMI protocol operations to provide a practical network situational awareness solution for AMI operators.

1 Introduction

The progressive replacement of traditional power meters with smart meters over the past few years was made possible through the deployment of important communication infrastructures by energy utilities. Those infrastructures, called *advanced metering infrastructures* (AMIs), consist of sets of wide-area networks (WANs) and neighborhood-area networks (NANs) that enable utilities to exchange information with meters in near real-time. That capability saves utilities from having to send human meter readers every month and allows new applications to improve grid reliability, energy savings, and consumer awareness. As an example of an AMI-enabled application, demand/response programs give consumers the ability to participate in load reduction during times of peak load, thanks to price signals sent by utilities. Those signals, received by meters, are used to automatically control home appliances that can accept delayed starts or slower running cycles, such as heating, ventilation, and air conditioning.

The communication capabilities of AMI also mean that utilities and parts of the distribution grid could be exposed to the risk of cyber intrusions. Motivations for malicious actions include access to large networks and an important number of low-computational devices, as well as access to consumption data and consumer information. Moreover, disruptions of grid operations through an AMI might have high visibility and high impact.

As a result, important efforts have been made by vendors, utilities, and regulators to ensure the resiliency of AMIs. In particular, strong encryption and authentication mechanisms are part of communication standards such as the ANSI

C12.22. While those protective measures are crucial to preventing attacks, they do not replace the need for efficient monitoring solutions. Indeed, the rapid evolution of attack techniques means that some protections might become obsolete, and successful intrusions should be promptly mitigated through an adequate combination of detection and response systems. The application of traditional information technology (IT) security solutions offers mixed results, and research and development efforts are still required to meet the needs of utilities [3].

This paper focuses on the detection aspect of the resiliency equation. Over the past two decades, various techniques for efficiently detecting intrusions in IT environments have been developed and evaluated. Through the Trustworthy Cyber Infrastructure for the Power Grid (TCIPG) research center[1], we studied how the unique characteristics of AMI compared to traditional IT networks could be leveraged in order to design highly efficient and practical intrusion detection systems (IDSes) [4,11]. In this paper, we present a sensor called *Amilyzer* that implements a specification-based approach to white-list the behavior of thousands of identical meters by observing and validating their network communications against protocol standards and a security policy.

The paper is organized as follows. Section 2 describes the scope of Amilyzer and the categories of threats that it monitors. Section 3 presents the internal architecture of Amilyzer and discusses deployment options. Section 4 describes our approach to define a sound security policy through the study of failure scenarios. Section 5 reports on lessons learned during the evaluation of Amilyzer in a testbed environment and in an AMI of more than 10,000 meters. Finally, Section 7 concludes the paper and discusses future work.

2 Threat Model and Failure Scenarios

Unlike a traditional IT network, AMI networks carry traffic for a limited set of applications using a single communication protocol, and smart meters are all designed to behave similarly. This tight control over devices and communications means that precise specification of expected traffic characteristics becomes a tractable problem; the control also opens up the possibility of implementing efficient white-listing technologies.

The scope of Amilyzer is to monitor network traffic and to focus on malicious behaviors that are visible in network activities. Thus, intrusions that reside entirely system-side, such as firmware compromise or penetrations into the collection engine application, are excluded from our scope. They could be detected by Amilyzer if they impact the network or if an attack vector is transferred through the network. Additionally, Amilyzer assumes complete visibility over payloads, which means that encrypted packets or packet captures that are incomplete because of sampling are currently not supported. We reviewed techniques to support encryption in [2], and improvement of Amilyzer so that it can be integrated into an IDS-friendly encrypted network environment is part of our future work.

[1] http://www.tcipg.org

To understand the types of malicious activities that Amilyzer should detect, we partnered with the Electric Power Research Institute (EPRI) and leveraged the work of the National Electric Sector Cybersecurity Organization Resource (NESCOR). NESCOR produced a report on possible failure scenarios for the different domains of the smart grid [18]. This report includes a threat model that consists of 4 categories of threat agents: criminals (economic, malicious, or recreational), activist groups, terrorists, and hazards. Insider threats such as disgruntled or careless employees are also included. This model helped us understand possible intents, threat characteristics, and risk prioritization. After studying failure scenarios for AMIs, we extracted 12 categories of failures that affect network activity. Those categories are presented in Table 1.

1	Adversary Issues Invalid Mass Remote Disconnect
2	Adversary Manipulates Meter Data Management System to Over/Under Charge
3	Mass Meter Rekeying Required when Common Key Compromised
4	One Compromised Meter in a Mesh Wireless Network Blocks Others
5	False Meter Alarms Overwhelm AMI and Mask Real Alarms
6	Unauthorized Pricing Information Impacts Utility Revenue
7	Spoofed Meter "Last Gasp" Messages Cause Fake Outage
8	Breach of Cellular Provider's Network Exposes AMI Access
9	Out of Sync Time-stamping Causes Discard of Legitimate Commands
10	Stolen Field Service Tools Expose AMI Infrastructure
11	Threat Agent Performs Unauthorized Firmware Alteration
12	Rogue Firmware Enables Unauthorized Mass Remote Disconnect

Table 1. Categories of network-related AMI failures extracted from the NESCOR report on failure scenarios

Each category in Table 1 groups one or several failure scenarios that we translated into security policy rules (detailed in Section 4). We then proceeded to extract the information needed for the successful detection of rule violations. First, Amilyzer has to access source and destination information, as well as identifiers for calling and called devices, from the network and application layers. That makes it possible to uniquely identify devices and to monitor security policy rules, such as "remote disconnects can only be issued from the collection engine." Second, Amilyzer has to understand the requests and responses issued by AMI devices so that security policy rules, such as "price updates should respect a given price window," can be supported. That means that headers and payloads from the application layer have to be captured and parsed. Finally, Amilyzer has

to be able to track the behavior of devices over time in order to monitor rules, such as "remote disconnects cannot be sent to more than N meters in less than M minutes." That means that the detection operations should be stateful.

3 Architecture

To achieve the objective of monitoring network activity and applying constraints from the specifications of valid behaviors, Amilyzer uses a modular architecture that is shown in Figure 1. The application was developed in Python and consists of about 2,300 lines of code. We describe the different modules in the next subsections.

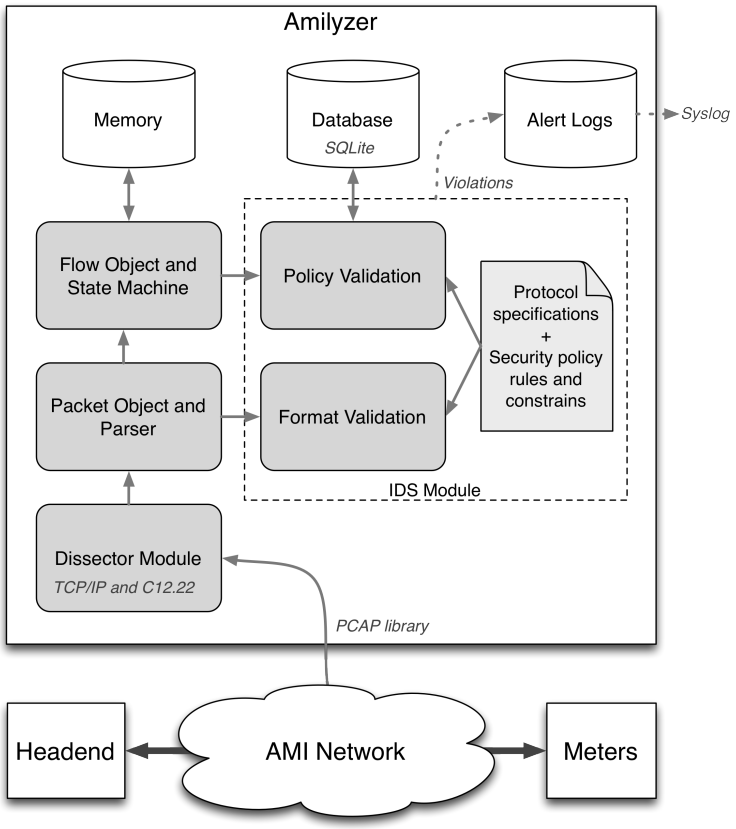


Fig. 1. Internal components and architecture of Amilyzer

3.1 Dissector

The application captures AMI traffic using the PCAP library and sends network packets to the dissector. The dissector includes parsers for TCP/IP, ANSI C12.22, and ANSI C12.19. A packet structure is instantianted for each new packet received, and information from the network, transport, and application layers are stored. At the application layer, the C12.22 header consists of an application security context element (ASCE) from which calling and called identifiers are extracted, an extended protocol specification for electric metering (EPSEM) header, and optional C12.19 data elements. The most important information extracted by the dissector is the information on EPSEM services that define the operations being requested. For instance, issuing of a remote disconnect involves sending of a *write* operation to update a specific register on the meter. An example of a dissected C12.22 packet representing a *full read* request and captured in the TCIPG testbed is provided below. The first line shows the timestamp, source IP address, and destination IP address. The next two lines show the raw bytes captured at the application layer. The first byte "60" indicates the beginning of a C12.22 frame. This frame has an ASCE header and an ESPEM section for which the different fields are detailed.

```
10:10.45.899059 10.9.0.1 > 10.9.2.6
app-layer:          60 15 a4 03 02 01 00 a8 03 02 01 00
        be 09 28 07 81 05 80 03 30 00 00
  --Decoding C12.22------- 21 bytes
  ACSE section:
       Called AP-invocation-identifier = 2.1.0
       Calling AP-invocation-identifier = 2.1.0
       User Information Element
       User Information External (length: 7)
       User Information Octet String (length: 5)
  EPSEM section: User Information Element (length: 5)
      EPSEM control: 128 10000000
       security: cleartext
       response_control: always respond
       --EPSEM request-response 1--
        Service length: 3
        Service: Full read
        Data: 00 00
```

3.2 State Machine

Once a packet structure has been created, Amilyzer tries to match it with an existing flow. Flows are defined as C12.22 sessions in which calling and called identifiers match and the flow is still active. Flows are expired after a couple of minutes of inactivity. If no flow is found, a new flow is created. Otherwise, the existing flow structure is updated. Each flow maintains a state machine for the caller and the called device. The state machine is built from the working state

of the device, which can be *in use*, *failed*, or *offline*, and the C12.22 protocol state, which can be *idle*, *sessionless processing*, *session idle*, or *session processing*. That state machine has been described and illustrated in [4]. A violation is reported to the intrusion detection module when an invalid transition is attempted. In addition, the sequence of EPSEM services requested and responded to is stored to allow the intrusion detection module to validate the flow against the specifications of known AMI operations.

3.3 Intrusion Detection System

Once a flow structure has been created or updated, the intrusion detection module analyzes the source and destination of the flow as well as the ordered sequence of requests and responses. Three types of checking operations are conducted. First, if a protocol violation has been recorded by the parser or by the state machine, an alert is logged. Second, the tuple {source, destination, sequence} is matched against a database of known valid flows. If the tuple has never been seen before, an alert is also, logged and an operator will have to approve the newly discovered flow or escalate the alert to create an incident. An approach to initial population of the database consists of running Amilyzer for a few days in *learning mode* before switching the system to *checking mode*. Third, the tuple is checked against a set of constraints. Those constraints are defined according to the security policy rules that are described in the next section. Five fields can be defined for each constraint: 1) a sequence pattern, which can be defined using a regular expression over the sequence of EPSEM services sent and received; 2) a source address or calling identifier; 3) a destination address or called identifier; 4) a condition, which can be expressed through functions such as *maximum rate per hour* or *scheduled time of validity*; and 5) an action, which might trigger a low-, medium-, or high-level alert. An example of a constraint is provided below for the security policy rule "*Allow only scheduled remote disconnect commands during authorized time windows.*"

```
– Pattern:      remote_disconnect
– Source:       any
– Destination:  any
– Condition:    NOT(9am–5pm)
– Action:       alert_high
```

An important feature of the intrusion detection module that improves the practicality of Amilyzer is the lookup mechanism. It enables operators to label caller ID, called ID, or C12.22 sequences of requests and responses with human-readable names. As shown in the example constraint above, the keyword *remote_disconnect* in the pattern field actually represents the sequence {*Logon; Full read; Partial write table 7; Logoff*}.

4 Security Policy

As mentioned in Section 2, we used the report on failure scenarios produced by NESCOR [18] to define a security policy for AMIs. A first step in writing the policy was to produce a human-readable set of rules that were extracted from the different scenarios that relate to network activity. The scope of those rules is to define what activities are allowed and prohibited in an AMI, and under which conditions. Conditions include timing, user roles, and system state requirements. A second step has been to translate the security policy rules into constraints that can be understood by Amilyzer.

A total of 23 rules were identified. For each rule, 6 fields were defined: 1) action to perform when a violation occurs; 2) best practice to prevent violations of the rule; 3) follow-up response actions after a violation occurred and an alert has been issued; 4) optimal sensor location to monitor the information required by the rule; 5) optimal location to issue response actions; and 6) information required for a successful detection. The 23 rules and 4 of the 6 fields are detailed in Table 2.

The optimal location to monitor the security policy rules depends on the information required. For instance, threats local to a neigborhood area network would not be visible at the headend. The Amilyzer sensor has been designed to be lightweight and to support flexible deployment schemes. Sensors can be deployed either centrally at the headend, or distributed in the field within access points, meters, or dedicated monitoring field devices. The central approach offers the advantages of low deployment cost, reliable alerting, and simple configuration. However, the central location lacks visibility over the edge of the network and can suffer from scalability issues in the case of a large AMI with hundreds of thousands of meters. In [6], we introduced a framework to assist utilities in selecting the right deployment strategy based on the characteristics of their AMIs.

5 Evaluation

The evaluation of Amilyzer was first conducted in a testbed environment. The TCIPG research center has a metering lab [21] that we used to capture normal meter operations and to test detection capabilities by injecting malicious traffic. The testbed consists of 20 meters deployed on 3 floors of a building. Each floor represents a NAN and has a wireless access point. The 3 access points are communicating using C12.22 over TCP/IP to a collection engine. An Amilyzer sensor was deployed on a switched port analyzer (SPAN) of a switch that connects the collection engine to the access points. The collection engine was configured to periodically send meter reading queries to meters. We then proceeded to inject the following attacks: 1) a remote disconnect from the collection engine towards a single meter at an unauthorized time; 2) a set of remote disconnects from the collection engine towards 10 meters at once; 3) a remote disconnect from a rogue IP address towards a single meter; 4) a burst of read requests from a rogue IP

Policy Rule	Prevention	Location	Information Required	Alert
Allow remote disconnect only from authorized headend	authentication; access control	headend	network access (app layer); Authorized headend	high
Allow remote disconnect to be issued only by employees with sensitive function credentials	least privileges	headend	network traffic (app layer); Employee ID	high
Remote disconnect commands affecting more than N1 meters require 2-person authentication	prevent command from reaching meter	headend	network traffic (app layer); N1 threshold	high
No remote disconnect command affecting more than N2 meters in less than M1 minutes should be authorized	prevent command from reaching meter	headend	network traffic (app layer); N2, M1 thresholds	high
Remote disconnect commands affecting more than N3 meters in less than M2 minutes require 2-person authentication	prevent command from reaching meter	headend	network traffic (app layer); N3, M2 thresholds	high
Allow scheduled remote disconnect commands only during authorized time windows	defining tight time windows	headend	network traffic (app layer); authorized time period	high
Pricing information must be flowing from the head end to meters	traffic control	headend; field	network traffic (app layer)	moderate
Only authorized system components can access the metering servers	device authentication; network access control lists	headend; field	network traffic (app layer) authorized devices ID	moderate
Cryptographic keys must never be transmitted in clear on the network	use cryptographic module	headend; field	network traffic; cryptographic key format	moderate
Allow software/firmware upgrades only from authorized headend systems or authorized field devices	authorization and privileges	headend; field	headend and device ID	moderate
Meters should respond to communication requests in less than S seconds	network health	headend; field	meter response time	low
Allow authorized operations to be sent to and received only from meters	white-listed network	headend; field	network traffic (app layer); list of authorized operations	moderate
Allow meter alarms to be issued only by registered meters	authentication, message authenticity, non-repudiation	headend; field	network traffic (app layer); list of authorized operations	moderate
Time-of-use price updates should fit between minimum and maximum acceptable values	authentication; data integrity	headend; field	network traffic (net + app layer); minimum and max pricing values	moderate
Time-of-use price updates should be executed by 2 persons	authentication	headend	authentication logs network traffic (app layer)	moderate
Meters in last-gasp state should have limited functionalities	up to date database of registered meter statuses	headend; field	network traffic (app layer); limited functionality definition	low
Traffic should be encrypted on leased networks	communication confidentiality	leased networks	network traffic (app layer); perimeter definition for leased networks	moderate
Traffic on leased networks must be isolated per utility	communication confidentiality; network configuration	leased networks	network traffic (app layer); perimeter definition for leased networks	moderate
Allow time synchronization updates to be issued only by authorized time servers	authentication of time servers	headend; field	network traffic (app layer); list of authorized time servers	low
Allow only correctly time synchronized communication messages	time synchronization process; communication integrity	headend; field	network traffic (app layer); correctly time synchronized: maximum of X milliseconds	low
Allow access to AMI only from authorized and authenticated field service equipment	authentication and authorization	headend; field	network traffic (net + app layer); list of authorized devices	low
Allow authorized operations only from field service equipment	authentication; authorization; privileges	headend; field	network traffic (net + app layer); list of authorized devices	moderate
Allow firmware upgrade only from valid firmware signature	authorization; privileges	headend; field	network traffic (net + app layer); firmware signature	high

Table 2. Security policy rules defined based on failure scenarios

address towards a single meter. Amilyzer was configured to accept remote disconnects only if they occur during an authorized time window, towards less than 2 meters per hour, and if they were issued from the collection engine. We verified that alerts were correctly triggered for each of the attack injected.

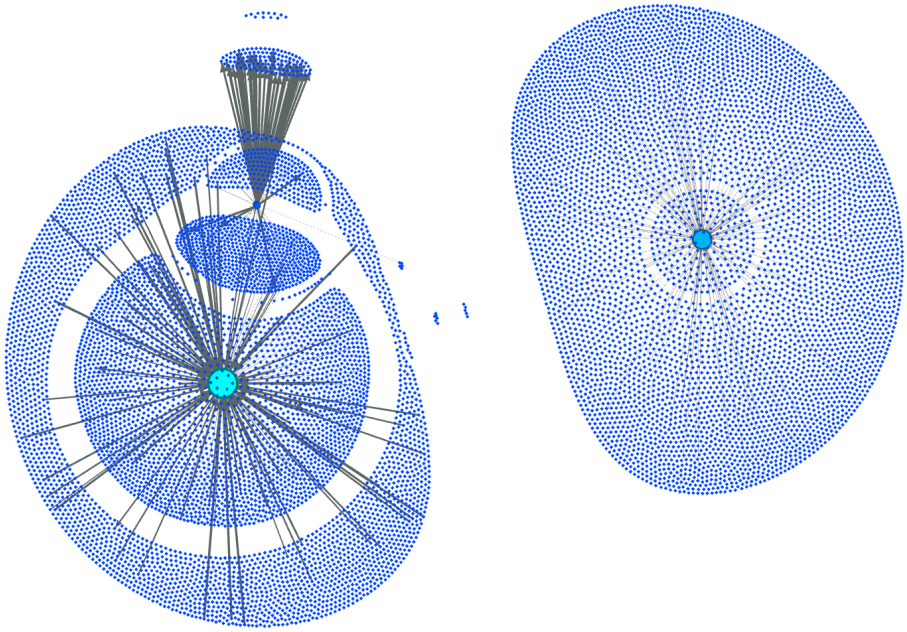
Once Amilyzer successfully passed accuracy and performance tests in the lab, we partnered with a large utility in the United States to deploy a sensor at the headend of a real AMI. The number of meters in this AMI grew from 6,000 at the beginning of the deployment in December 2012, up to 32,000 meters in October 2013. This large-scale deployment enabled us to identify and fix performance bottlenecks, as well as to tune the security policy and better understand the situational awareness needs of AMI operators. A total of 10 versions of the application were deployed over the past 10 months to take into account crash reports and data logged. Three types of instabilities were reported: 1) frequent crashes of the C12.22 dissector module; 2) large number of packets that could not be parsed; and 3) intrusion detection module too slow to keep up with the rate at which packets were captured. We solved the first issue by improving exception handling in the dissector. The second issue was due to many packets being segmented at the application layer. We completed the implementation of the dissector to handle reassembly of C12.22 segments. Finally, we replaced the file-based input/output system used by the intrusion detection module with an SQLite database. The database significantly improved the speed at which search and save queries could be handled. As an example, for one week of traffic collected in October 2013, the average traffic rate measured has been 593 packets per minute (ppm), with a minimum of 17 ppm and a maximum of 2942 ppm. At the end of those 7 days, Amilyzer was maintaining a database of 29 MB that consisted of half a million flow processed, 144,175 unique flows identified, and 234 unique sequences of C12.22 operations. During that week, C12.22 operations recorded included 9,084 meter registration requests, 988,607 identification requests, and 2,791 write requests.

Figure 2 is a screenshot of a visualization application that represents the network topology from the logs produced by Amilyzer. It also allows us to filter the traffic monitored by type of requests and responses. Each dot in Figure 2 represents a smart meter and arrows represent directed traffic captured between the meters and one of the 3 collection engines (larger circles at the center of the clusters). In this screenshot, only response errors are shown, indicating areas of the AMI in which errors occur more frequently. (The thickness of arrows indicates traffic volume.) That representation is used by operators to identify misconfigurations and security issues.

6 Related Work

IDSes for critical infrastructures have been the focus of an increasing number of studies over the past few years. A first category of work has looked at supervisory control and data acquisition (SCADA) protocols. [7] leverages the regular traffic patterns of SCADA networks to develop a model-based IDS. The approach mon-

Fig. 2. Visualization of response errors sent by meters to a collection engine for an AMI of 12,000 meters. Meters are clustered around 3 collection engines and edges represent network flows for a specific C12.22 operation.



itors Modbus protocol fields to update models of the protocol. Rules to detect violations of the critical characteristics of those models have been implemented in Snort. MHINDS [12] extended that approach by specifying additional temporal and spatial model characteristics using a Petri net implementation. The authors of [10] also introduced a state-based network intrusion detection system. They focused on the detection of complex multi-step attacks targeting systems that use the DNP3 or the Modbus protocols. At the core of this approach is a rule language that allows operators to describe critical device states for which alerts should be triggered. [14] implemented a complete parser for DNP3 in the open source Bro IDS, along with a set of rules to detect violations of the protocol semantic. Those rules enable the detection of both errors within a given DNP3 packet, and of state-based issues across a sequence of DNP3 operations. [20] proposed a model-based sensor working on top of the WirelessHART protocol to monitor and protect wireless process control systems. The architecture consists of a central component that collects information periodically from a set of sensors distributed in the field. On the side of host-based IDSes, [16] presents a probability model based on stochastic Petri nets to specify the behavior of a cyber physical system (CPS). A contribution of that work is a framework that can be dynamically adjusted to tune detection parameters of IDS sensors in order to better monitor and respond to attackers. [8] looked at the specific issue of supply-chain threats that target critical-infrastructure embedded systems. The

hardware-based IDS monitors the analog signal response of a resistor-capacitor circuit to identify the presence of hardware Trojans.

A second category of work has investigated the specific domain of AMIs. Detection of and protection against firmware compromises has been studied in [13], where an architecture called the *cumulative attestation kernel* enables utilities to remotely audit firmware updates in smart meters. Another host-based IDS is described in [9], which proposed to instrument meters, data concentrators, and the headend with sensors that can apply data-mining techniques on the continuous stream of data exchanged. Finally, [19] studied how to instrument a trusted meter device with an anomaly-based IDS. Closer to our approach, a few network-based IDSes for AMI have been recently introduced. [15] presented BRIDS, a behavior-rule based IDS that can translate rules about the expected behavior of AMI components into state machines. States are defined according to the combined values of predicates expressed in normal form. [17,1] described an anomaly-based IDS for NAN that uses a combination of central and distributed sensors to support monitoring of encrypted traffic.

Our present work builds upon [5], which explained the motivation for using specification-based IDSes in the context of an AMI, and [4], which leveraged a modeling technique and a theorem-proving framework to formally verify that a detection operation correctly implements the security policy. Our contributions in this paper are the following. First, the complete design of Amilyzer is presented and grounded in a review of realistic AMI failure scenarios. Second, a comprehensive security policy for AMI is introduced. Third, lessons learned after evaluating Amilyzer in a large AMI for 9 months are presented.

7 Conclusions and Next Steps

Amilyzer is a specification-based intrusion detection sensor designed to monitor AMI communications. This paper presented the threat model covered by Amilyzer, as well as the modular architecture of the sensor. A unique aspect of this project is that it leverages an industry-established set of realistic failure scenarios to define a comprehensive security policy. The policy, which consists of 23 rules, was described and translated into machine-checkable constraints. Several months of deployment of Amilyzer at the headend of a large AMI enabled us to improve the scalability and reliability of the IDS. Amilyzer has reached a development phase at which it is starting to address the pressing need for an efficient and practical monitoring solution for AMIs.

Amilyzer is still a prototype, and there are still important research challenges to investigate as part of future work. First, Amilyzer does not support encrypted traffic. We are looking into IDS-friendly key-sharing mechanisms to increase the visibility of the sensor over meter communications in the field. Second, Amilyzer currently works as an individual sensor, but we believe that the efficiency of the intrusion detection approach could gain from being distributed and coordinated across multiple sensors. Finally, we are looking at how to improve the resiliency of the monitoring infrastructure against attacks that target IDS sensors.

Acknowledgments

This material is based upon work supported in part by the Department of Energy under Award Number DE-OE0000097 and by the Electric Power Research Institute. The opinions expressed are those of the authors alone. The authors would like to thank Annabelle Lee for her input and insightful feedback on the security policy and Jenny Applequist for her editorial assistance.

References

1. Nasim Beigi Mohammadi, Jelena Mišić, Vojislav B Mišić, and Hamzeh Khazaei. A framework for intrusion detection system in advanced metering infrastructure. *Security and Communication Networks*, 2012.
2. Robin Berthier, Jorjeta G Jetcheva, Daisuke Mashima, Jun Ho Huh, David Grochocki, Rakesh B Bobba, Alvaro A Cárdenas, and William H Sanders. Reconciling security protection and monitoring requirements in advanced metering infrastructures. In *Smart Grid Communications (SmartGridComm), Conference on*. IEEE, 2013. To appear.
3. Robin Berthier and Galen Rasche. Intrusion detection system for advanced metering infrastructure. Technical report, EPRI, 2012. http://www.energycollection.us/Energy-Metering/Intrustion-Detection-System.pdf.
4. Robin Berthier and William H Sanders. Specification-based intrusion detection for advanced metering infrastructures. In *Dependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on*, pages 184–193. IEEE, 2011.
5. Robin Berthier, William H Sanders, and Himanshu Khurana. Intrusion detection for advanced metering infrastructures: Requirements and architectural directions. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 350–355. IEEE, 2010.
6. Alvaro A Cárdenas, Robin Berthier, Rakesh B Bobba, Jun Ho Huh, Jorjeta G Jetcheva, David Grochocki, and William H Sanders. A framework for evaluating intrusion detection architectures in advanced metering infrastructures. *Smart Grid, IEEE Transactions on*, 2013. To appear.
7. Steven Cheung, Bruno Dutertre, Martin Fong, Ulf Lindqvist, Keith Skinner, and Alfonso Valdes. Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium*, pages 1–12, 2007.
8. Nathan J. Edwards. Hardware intrusion detection for supply-chain threats to critical infrastructure embedded systems. Master's thesis, University of Illinois at Urbana-Champaign, 2012.
9. Mustafa Amir Faisal, Zeyar Aung, John R Williams, and Abel Sanchez. Securing advanced metering infrastructure using intrusion detection system with data stream mining. In *Intelligence and Security Informatics*, pages 96–111. Springer, 2012.
10. Igor Nai Fovino, Andrea Carcano, T De Lacheze Murel, Alberto Trombetta, and Marcelo Masera. Modbus/DNP3 state-based intrusion detection system. In *Advanced Information Networking and Applications (AINA), International Conference on*, pages 729–736. IEEE, 2010.
11. David Grochocki, Jun Ho Huh, Robin Berthier, Rakesh Bobba, William H Sanders, Alvaro A Cárdenas, and Jorjeta G Jetcheva. AMI threats, intrusion detection

requirements and deployment recommendations. In *Smart Grid Communications (SmartGridComm), International Conference on*, pages 395–400. IEEE, 2012.
12. Adam Hahn. *Cyber security of the smart grid: Attack exposure analysis, detection algorithms, and testbed evaluation.* PhD thesis, 2013.
13. Michael LeMay and Carl A Gunter. Cumulative attestation kernels for embedded systems. *Smart Grid, IEEE Transactions on*, 3(2):744–760, 2012.
14. Hui Lin, Adam Slagell, Catello Di Martino, Zbigniew Kalbarczyk, and Ravishankar K Iyer. Adapting Bro into SCADA: building a specification-based intrusion detection system for the DNP3 protocol. In *Annual Cyber Security and Information Intelligence Research Workshop*, page 5. ACM, 2013.
15. Robert Mitchell and Ing-Ray Chen. Behavior-rule based intrusion detection systems for safety critical smart grid applications. 4(3):1254–1263, September 2013.
16. Robert R Mitchell III. *Design and Analysis of Intrusion Detection Protocols in Cyber Physical Systems.* PhD thesis, Virginia Polytechnic Institute and State University, 2013.
17. Nasim Beigi Mohammadi. An intrusion detection system for smart grid neighborhood area network. *Journal on Information Technology and Applications*, 2(1):7–13, 2012.
18. National Electric Sector Cybersecurity Organization Resource (NESCOR). Electric sector failure scenarios and impact analyses. Technical report, EPRI, 2012. http://www.smartgrid.epri.com/doc/NESCOR_10_25_12.pdf.
19. Massimiliano Raciti and Simin Nadjm-Tehrani. Embedded cyber-physical anomaly detection in smart meters. In *Critical Information Infrastructure Security (CRITIS12), International Conference on*, 2012. To appear.
20. Tanya Roosta, Dennis K Nilsson, Ulf Lindqvist, and Alfonso Valdes. An intrusion detection system for wireless process control systems. In *Mobile Ad Hoc and Sensor Systems, International Conference on*, pages 866–872. IEEE, 2008.
21. Tim Yardley, Robin Berthier, David Nicol, and William H Sanders. Smart grid protocol testing through cyber-physical testbeds. In *Innovative Smart Grid Technologies (ISGT)*, pages 1–6. IEEE, 2013.