SmartAnalyzer: A Noninvasive Security Threat Analyzer for AMI Smart Grid

Mohammad Ashiqur Rahman, Padmalochan Bera, Ehab Al-Shaer
Department of Software and Information Systems
University of North Carolina at Charlotte
Email: {mrahman4,bpadmalo,ealshaer}@uncc.com

***Abstract*—The Advanced Metering Infrastructure (AMI) is the core component in the smart grid that exhibits a highly complex network configuration comprising of heterogeneous cyber-physical components. These components are interconnected through different communication media, protocols, and secure tunnels, and they are operated using different data delivery modes and security policies. The inherent complexity and heterogeneity in AMI significantly increase the potential of security threats due to misconfiguration or absence of defense, which may cause devastating damages to AMI. Therefore, there is a need for creating a formal model that can represent the global behavior of AMI configuration in order to verify the potential threats.**

In this paper, we present *SmartAnalyzer*, a security analysis tool, which offers manifold contributions: (i) formal modeling of the AMI configuration including device configurations, topology, communication properties, interactions among the devices, data flows, and security properties; (ii) formal modeling of AMI invariants and user-driven constraints based on the interdependencies among AMI device configurations, security properties, and security control guidelines; (iii) verifying the AMI configuration's compliances with security constraints using Satisfiability Modulo Theory (SMT) solver; (iv) generating a comprehensive security threat report with a possible remediation plan based on the verification results. The accuracy, scalability, and usability of the tool are evaluated on an AMI testbed and various synthetic test networks.

I. Introduction

Smart grid provides innovative and efficient energy management services that offer operational reliability and value-added advantages to both customers and energy providers. The potential market for smart grids shows that it will be the most widely deployed critical infrastructure in the 21st century. AMI is the major component in a smart grid that, unlike the traditional networks, consists of heterogeneous devices, such as smart meters, intelligent data collectors, headend systems, hosts, routers, firewalls, etc. AMI devices communicate with one other through various communication protocols, physical media, and secure tunnels. These devices transfer energy usage data following different modes of data deliveries, which are controlled by alternative security policies. Security attacks on such networks due to misconfigurations or constraint violations have the potential to cause critical damages including power outages and destruction of equipments [5][11].

In this work, we develop *SmartAnalyzer*, an automated security analysis tool for AMI smart grid network, that takes AMI configuration and organizational security requirements as inputs; formally models the configurations and various invariants and security constraints; and verifies the compliances of the configurations with the constraints using satisfaction checking. The tool generates a comprehensive threat report that includes the traces and reasoning behind various constraint violations and potential reconfiguration plans. The performance of the tool is analyzed on various real and synthetic data.

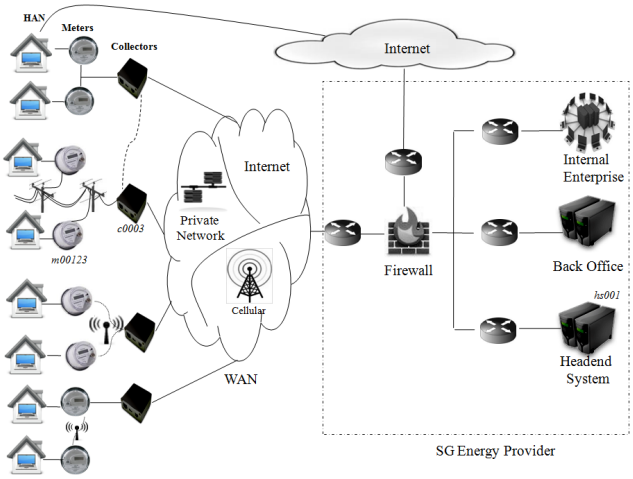

Fig. 1. A typical AMI smart grid network.

A. Background, Challenges, and Objective

Fig. 1 shows the general structure of an AMI that consists of smart meters (SM), intelligent collectors (IC), and headend system (HS) as the main components. Smart meters must be configured to perform security pairings with a specific IC, establish secure connections, monitor and report energy usage data periodically. ICs are configured to collect the data from a group of meters and to forward the data to a headend over secure connections. They forward control commands, patches, etc from the headend to the meters. There is one or more firewall(s) for restricting access between AMI and the energy provider's utility network. AMI devices transfer data using different modes of data delivery. Fig. 2 and Fig. 3 collectively present a partial template of AMI configuration. It shows the operational and security properties of AMI components including the network topology. These properties are described later in Section III.

AMI networks are more complex than traditional networks mainly due to the following reasons. Firstly, AMI is a hybrid

Meter Class	ID	Type	Patch Info	Sampling Info	Reporting Mode (to Collector)	Report Schedule	Auth Property	Encrypt Property	Ports in Service	Comm Protocol				
meter	m00003	ge	pm011, pm115	18,40	push	15,40	auth1	encrypt1	nil	lontalk				
meter	m00123	echelon	pm115	15,30	push	20,30	auth0	encrypt1	nil	lontalk				
meter	m00129	echelon	pm0115	20,30	push	20,60	auth1	encrypt1	nil	lontalk				
Collector Class	**ID**	**Type**	**Patch Info**	**Buffer Info**	**Reporting Mode (to Headend)**	**Schedule (to**	**Pull Schedule (from meter)**	**ConnectedMeters**	**Connected Headend**	**Link (to Meter)**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**
collector	c0003	rev03	pc213, pc012	data, 9000, 1	pull	nil	nil	m00003,5; m00123,4	hs001	powerline1	auth1, auth2	encrypt1, encrypt2	22, 53, 161, 222	lontalk, ip
collector	c0005	rev02	pc012	data, 8000, 1	push	300, 14400	nil	m00003,5; m00129,5	hs001	powerline1	auth1, auth2	encrypt1, encrypt2	22, 53, 161, 222	lontalk, ip
Headend Class	**ID**	**Type**	**OS**	**Patch Info**	**Pull Schedule (from Collector)**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**					
headend	hs001	nil	win2010	p357, p254	180, 2880, c0003	auth2, none	encrypt2, none	22, 53, 161	ip					
Backend Class	**ID**	**OS**	**Patch**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**							
backend	bs001	fedora14	p357	none	none	22	ip							
Home Host Class	**ID**	**OS**	**Auth Property**	**Encrypt Property**	**Ports in Service**	**Comm Protocol**								
home host	h0001	win_vista	none	none	_	ip								

Fig. 2. An example data of AMI topology configuration.

network consisting of (1) heterogeneous devices, such as meters, collectors, firewalls, routers, IPSec gateways, etc, (2) heterogeneous links of power lines, wired, and wireless, and (3) heterogeneous protocols, such as LonTalk (meter-IC) and TCP/IP (IC-headend). Secondly, AMI network involves varieties of data stream types, such as power usage data, control commands, s/w patches, which exhibit different priorities and resource requirements. Thirdly, unlike policy based Internet forwarding, data delivery in AMI is time-driven or request-driven that follows specific schedules. For example, as shown in Fig. 2, meter $m00003$ reports data periodically every 40 seconds starting at 15 seconds from the base time. AMI has to be configured carefully to synchronize the data delivery without overflowing the network or its devices. Moreover, AMI network must be accessible from the utility network for different purposes like management, patching, etc. The energy users from Home Area Network (HAN) can also access AMI via Internet or smart meters.

The correct functioning of AMI stands on consistent and secure execution of tasks in time. The safe security configuration depends not only on the local device parameters but also on the secure interactions and flows of these parameters across the network. There is a significant number of logical constraints on millions of configuration parameters, which need to be satisfied to ensure secure interactions among AMI components. These constraints represent system invariants and user-defined security controls (i.e., the organizational security requirements). Implementing these constraints in a scalable manner is one of the major challenges in smart grid security. An effort has been made by the government, DHS AMI Task Force [2] and NIST [3], to develop guidelines for more than 300 security controls. NISTIR 7628 provides guidelines for ensuring trusted path, resource availability, boundary security protection, etc, towards controlling different security threats. However, the manual analysis and enforcement of these controls can be overwhelming and potentially inaccurate due to high potential of human errors.

Link	Src	Dest	Status	Link Type				
link	zc101	v1	up	gprs				
link	zhs101	r2	up	ethernet1				
link	v1	r1	up	fiber1				
Link Profile	**ID**	**Media**	**Mode**	**Shared Status**	**BW**			
link profile	powerline1	power_line	halfduplex	yes	5			
link profile	wifi1	wireless	fullduplex	no	150			
Auth Profile	**ID**	**Algo**	**Key**					
auth	auth0	sha1	96					
auth	auth1	sha1	160					
auth	auth_ipsec1	sha1	160					
Encrypt Profile	**ID**	**Algorithm**	**Key**					
encrypt	encrypt1	rc4	64					
encrypt	encrypt2	rc4	128					
encrypt	crypt_ipsec1	md5	64					
Fw Policy	**Node**	**Src**	**Src Port**	**Dest**	**Dest Port**	**Protocol**	**Action**	**Limit**
fw policy	f1	150.0.0.0/8	161	172.16.0.0/16	161	udp	allow	_
fw policy	f1	150.0.0.0/8	22	172.16.0.0/16	22	tcp	allow	_

Fig. 3. An example data of AMI (security) policy configuration.

Researchers proposed different security analysis tools for analyzing misconfiguration problems in traditional networks [12][8]. These tools do not model complex heterogeneous configurations like time-driven data forwarding and different security controls specific to a smart grid. The objective of this work is to develop an automated tool, SmartAnalyzer that will allow energy providers to objectively assess and investigate AMI security configuration for identifying and mitigating potential security threats and to enforce AMI operational and organizational security requirements. The major technical novelty of the tool lies in its capability of analyzing various safety critical constraints on AMI network, such as (i) data overwrite protection; (ii) device scheduling and cyber bandwidth constraint; (iii) assured data delivery; and (iv) data freshness. Apart from these, the tool is capable of verifying various basic security properties like trusted path, data integrity, confidentiality, etc. Most importantly, the tool uses SMT based formal analysis engine as the core that provides proof-based threat report as the outcome, which can be comprehensively used for fixing the errors.

The rest of the paper is organized as follows. We present the architecture and brief functional description of SmartAnalyzer tool in Section II. The formal modeling of AMI components, network topology, and data delivery modes is presented in Section III. The following section presents the modeling and analysis of various security controls in AMI. Section V presents the evaluation of our tool. Section VI describes the related works on smart grid security analysis. Finally, we write the conclusion and the future work in Section VII.

II. SmartAnalyzer Architecture

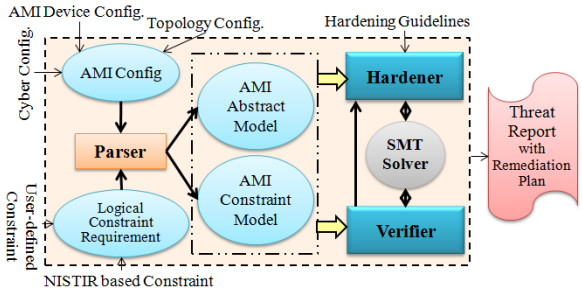

Fig. 4. The architecture of SmartAnalyzer.

SmartAnalyzer is an automated security analysis tool for AMI smart grid that has the following functionalities:

- Providing an extensible global model abstraction capable of representing millions of AMI device configurations.
- Formal modeling and encoding of various constraints into SMT logic.
- Verifying the satisfaction of the constraints with AMI configuration using SMT solver.
- Identifying potential security threats from the constraint violations and providing remediation plans for security hardening by analyzing the verification results.

SmartAnalyzer architecture is shown in Fig. 4. First, the tool parses given *AMI configuration template* (e.g., Fig. 2 and Fig. 3) and encodes it into SMT logic. The configuration template is a CSV file. It consists of the device configurations (based on abstraction), topology, communication between the devices, data delivery schedules in the network, etc. The model abstraction is done by exploiting the correlation between the configuration parameters of different AMI devices. SmartAnalyzer formally models the organizational requirements and various security guidelines (such as from NISTIR) as AMI invariant and user-driven constraints; and encodes these constraints into SMT logic. Then, the tool (*verifier* module) uses Yices SMT solver [13] to verify these constraints with the configurations. A comprehensive threat report is generated based on the verification results. Finally, the tool (*hardener* module) creates a remediation plan by systematically analyzing the *unsat-core* traces produced by the SMT solver (when the verification is unsatisfied), which helps the administrators in reconfiguring AMI by directly fixing the configuration values or further incorporating new security alternatives.

III. Modeling AMI Configuration

A. Modeling AMI Physical Components

In this subsection, we present the formalizations of different AMI device configurations.

Smart Meter: A meter class is identified by an *Id* and its profile *SM* is represented as a conjunction ($\wedge$) of different parameters shown in Table I. The vendor type (i.e., *Echelon*, *GE*, etc.) is represented by the parameter *Type*. We represent the sampling information of a meter using *SInfo* that consists of two components, sampling size (*SSize* in KB) and sampling time (*STime*). A meter can deliver data to a collector in two different modes: *pull* and *push*. In *pull* mode, the meter reports data based on the request from the collector that follows a specific *pull schedule* of the collector. On the other hand, in *push* mode, the meter reports data to the collector (without waiting for request) based on its own *report schedule*. This reporting mode is captured by *Mode*. The reporting time schedule of a meter (in *push* mode) is modeled using *RSche* that consists of *RScheBase* and *RScheInt*. This indicates that the meter will report periodically in a regular interval of *RScheInt* starting from *RScheBase* after the base time. To achieve end-to-end security, the communicating devices must agree in their authentication and encryption properties. We model the authentication properties of a meter using the parameter *Auth* as conjunction of algorithm (*AAlgo*) and key length (*AKey*). A meter may support multiple authentication properties. Encryption property is modeled similarly as *Encr*. The running services and communication protocols associated to a meter are represented by *Serv* and *CommProto* respectively. The parameter *Patch* denotes the patches that are installed in the meter. The maximum transmission rate (in Mbps) of a meter is denoted by the parameter *TRate*. The formalization of a meter class is shown in Table I.

TABLE I
Formal Definition of AMI Meter and Collector

Smart Meter:

$SM_i \Rightarrow Type_i \wedge Patch_i \wedge SRate_i \wedge Mode_i \wedge RSche_i \wedge Auth_i \wedge Encr_i \wedge Serv_i \wedge CommProto_i \wedge TRate_i$

$Patch_i \Rightarrow \bigwedge_{j=0\ldots} Patch_{i,j}$

$SRate_i \Rightarrow SSize_i \wedge STime_i$

$RSche_i \Rightarrow RScheBase_i \wedge RScheInt_i$

$Auth_i \Rightarrow \bigwedge_{j=0\ldots} (AAlgo_{i,j} \wedge AKey_{i,j})$

$Encr_i \Rightarrow \bigwedge_{j=0\ldots} (EAlgo_{i,j} \wedge EKey_{i,j})$

$Serv_i \Rightarrow \bigwedge_{j=0\ldots} SPort_{i,j}$

$CommProto_i \Rightarrow \bigwedge_{j=0\ldots} CommProto_{i,j}$

Intelligent Data Collector:

$IC_i \Rightarrow Type_i \wedge Patch_i \wedge BufSize_i \wedge Mode_i \wedge RSche_i \wedge PRSche_i \wedge Auth_i \wedge Encr_i \wedge AttachSM_i \wedge LinkToSM_i \wedge AttachHS_i \wedge Serv_i \wedge CommProto_i \wedge TRate_i$

. . . . . .

$PRSche_i \Rightarrow \bigwedge_{j=0\ldots} (PScheBase_{i,j} \wedge PScheInt_{i,j} \wedge RDev_{i,j})$

$ConnSM_i \Rightarrow \bigwedge_{j=0\ldots} (CSMId_{i,j} \wedge CSMNum_{i,j})$

Intelligent Data Collector: A collector class profile *IC* is represented as a conjunction of different parameters, which include all parameters of meter class profile except the sampling information. In addition, each collector may have a pull schedule that is represented by the parameter *PRSche*. It has three components: *PSBTime*, *PInt*, and *RDev*, which denote that the collector periodically pulls data from reporting device (*RDev*, a meter) starting at *PScheBase* with interval *PScheInt*.

TABLE II
MODELING OF ZONE ITS RELATION WITH AMI DEVICES

Zone:
$Zone_i \Leftrightarrow ZSn_i \wedge ZMem_i \wedge ZGw_i$
$ZSn_i \Rightarrow Ip_{i,j} \wedge Mask_{i,j}$
$ZMem_{i,j} \Rightarrow ZMId_{i,j} \wedge ZMNum_{i,j}$
. . . . . .
$ZMem_i \Rightarrow \bigwedge_{j=0...} ZMem_{i,j}$

Representation of a source:
$(S \Leftrightarrow Id \wedge ZId) \Rightarrow (Id \Leftrightarrow ZMId)$

A collector has a buffer for storing the report data from different meters. *BufSize* represents the buffer size (in KB). The parameter *ConnSM* is a conjunction of the meter classes (*CSMId*) and their numbers (*CSMNum*), which are connected to the collector. *LinkToSM* represents the ID of the link (refer to Section III-B) that connects collector to the meter. The parameter *AttachHS* represents the headend system to which data is reported by the collector.

Headend System: A headend system class profile *HS* is a conjunction of the parameters: *Type*, *OS*, *Mode*, *TRate*, *Patch*, *PRSche*, *Auth*, *Encr*, *Serv* and *CommProto*. These properties are modeled as similar to those of meter/collector.

Host Devices: AMI network contains different type of hosts, such as (1) hosts of home area network (enterprise clients), (2) enterprise internal hosts, (3) enterprise application servers (backend systems), and (4) external hosts from Internet. Hosts have considerably less parameters. For example, an enterprise client host class profile has *OS*, *Auth*, *Encr*, *Serv*, *CommProto* and *TRate* parameters only.

B. Modeling AMI Network Topology

AMI topology defines the physical and logical connectivity between different AMI devices.

Router, Firewall and IPSec: We model router (*R*), firewall (*F*), and IPSec (*IS*) devices similar to [12]. We only introduce the traffic limiting capability of a firewall in the model using the parameter *FwLim* along with its action (*FwAct*) in its policy (*FwPolicy*). Router selects the next-hop (*RNext*) for a particular traffic based on its forwarding policy (*RPolicy*).

Link: A link is identified by an ID (*LId*). Its profile is a conjunction of *NodePair* (i.e., the node-pair connected by the link) and *LinkStatus* (i.e., up or down). *LId* binds the specified link type to the predicate *LinkProp* that represents the properties of that link including *MediaType* (i.e., wireless, ethernet, etc), *SharedStatus* (i.e., shared or not), *CommMode* (i.e., half-duplex, full-dulex, etc) and *LinkBw* (in Mbps).

Zone: We model logical zone as a collection of similar AMI devices. Each zone has an ID (*ZId*). The profile of a zone (*Zone*) comprises of three parameters: *ZSn*, *ZMem* and *ZGw*. The parameter *ZSn* denotes an IP-address (with subnet Mask) that covers all devices in that zone. *ZMem* represents the IDs of different device classes and the number of devices under each class that belong to the zone. *ZGw* denotes the gateway router ID for that zone. The formalization of a zone, $Zone_i$ is represented in Table II. Any source/destination node of a traffic is represented as a conjunction of its *Id* and *ZId*. The number of traffic source/destination depends on the number of zones and the number of classes in the zones. For example, if there are 50 zones and 4 collector classes per zone on average, then there are 200 possible source/destination collectors. In the rest of the paper, we refer to a source/destination (especially in traffic) as a node associated with its zone. It is to remember that meters are directly associated to a collector.

IV. AMI THREAT ANALYSIS MODEL

This section describes the potential security threats on AMI and the modeling of associated security constraints. Finally, we present the constraint verification methodology in AMI.

A. Threats on AMI

It is well documented that configuration errors cause 50%-80% of vulnerabilities in cyber infrastructure [7]. There are various potential threats on AMI networks due to noncompliance/violation of different constraints.

Reachability and Integrity Threats: To achieve successful data delivery, reachability must hold between the sender and the receiver. Data Integrity is important, since its violation not only can cause incorrect billing but also can launch malicious control commands towards AMI that may result in massive power outage. So, data should be delivered satisfying end-to-end integrity. Moreover, inconsistencies in authentication and encryption parameters (say, algorithm) may cause service disruption. Reachability intuitively ensures the correctness of any IPSec based tunnel existing in the path.

Availability Threats: Improper scheduling of data delivery between meters and collectors can lead to buffer overflow and data loss in the collector side. Moreover, this can cause delay in data delivery, even data loss at the endpoints due to limited link bandwidth. For example, if UDP protocol is used between a collector and a headend, then improper scheduling may allow a large number of nodes transferring data to the headend, which can flood links on the path and consequently cause data loss. In this case, use of TCP protocol will create congestion, which in turn lead to delay in reporting, data loss in the sender side (if data rate is higher than the delay). The main purpose of AMI is to deliver clients' power usage data to the provider's side. Hence, the resource availability threats can be very devastating.

Common Network Threats: Common network threats are endpoint DoS, link flooding, wireless link jamming. In AMI, a large number of compromised collectors can launch DoS attack to headend. It is infeasible to provide protection against any number of compromised collectors. Constraints can limit the possibility of such attack. Controlling these threats require exploring the use of network limiters.

B. Modeling AMI Security Constraints

Appropriate modeling of the constraints is required to protect AMI from different security threats. We classify these

TABLE III
FORMALIZATIONS OF REACHABILITY AND PAIRING CONSTRAINTS

Reachability Constraint:

$Forward_{X,Y,Tr_{S,D},TrR} \Rightarrow$
$Link_{X,Y} \wedge$
$(((X \Leftrightarrow S) \wedge (ZGw_S \Leftrightarrow Y)) \vee (Y \Leftrightarrow D) \vee$
$(R_X \wedge RPolicy_{X,Tr_{S,D}} \wedge (RNext_X \Leftrightarrow Y)) \wedge$
$((F_X \Rightarrow FwPolicy_{X,Tr_{S,D}} \wedge FwAct_X \wedge$
$(FwLim \Rightarrow (TrR \Leftrightarrow min(LimVal, LinkBw_{X,Y})))) \vee$
$(\neg F_X \Rightarrow (TrR \Leftrightarrow LinkBw_{X,Y})))$

$Reachable_{A,B,Tr_{S,D},TrR} \Rightarrow$
$Forward_{A,B,Tr_{S,D},TrR} \vee$
$(\exists C, Forward_{A,C,Tr_{S,D},TrR1} \wedge Reachable_{C,B,Tr_{S,D},TrR2} \wedge$
$(TrR \Leftrightarrow min(TrR1, TrR2))$

$ReachabilityConstr_{Tr_{S,D},TrR} \Rightarrow Reachable_{S,D,Tr_{S,D},TrR}$

Pairing Constraint:

$AuthPairings_{S,D} \Rightarrow$
$(AAlgo_S \Leftrightarrow AAlgo_D) \wedge (AKey_S \Leftrightarrow AKey_D)$

$EncrPairing_{S,D} \Rightarrow$
$(EAlgo_S \Leftrightarrow EAlgo_D) \wedge (EKey_S \Leftrightarrow EKey_{m,D})$

$ProtoPairings_{S,D} \Rightarrow$
$(Proto \in CommProto_S) \wedge (Proto \in CommProto_D)$

$PairingConstr_{S,D} \Rightarrow$
$AuthPairings_{S,D} \wedge EncrPairing_{S,D} \wedge ProtoPairings_{S,D}$

TABLE IV
FORMALIZATIONS OF SCHEDULE AND RESOURCE CONSTRAINTS

Schedule Constraints:

$MeterSampConstr_M \Rightarrow$
$SM_M \wedge (Mode_M \Rightarrow ((STime_M \leq RScheInt_M) \wedge$
$(RScheBase_M \leq RScheInt_M))) \wedge$
$((SSize_M / STime_M) \leq TRate_M)$

$CollectorPullScheConstr_C \Rightarrow$
$IC_C \wedge (((M \Leftrightarrow CSMId_C) \wedge \neg Mode_M) \Rightarrow PRSche_C)$

Resource Constraints:

$(TotalSData_C \Leftrightarrow \sum_M SData_M) \Rightarrow$
$(M \Leftrightarrow CSMId_C) \Rightarrow$
$(SData_M \Leftrightarrow (SSize_M \times CSMNum_C)))$

$CollectorBufConstr_C \Rightarrow$
$IC_C \wedge (BufSize_C \geq TotalSData_C)$

$(TotalSRate_C \Leftrightarrow \sum_M SRate_M) \Rightarrow$
$(M \Leftrightarrow CSMId_C) \wedge$
$(SRate_M \Leftrightarrow ((SSize_M / STime_M) \times CSMNum_C)))$

$CollectorTrRConstr_C \Rightarrow$
$IC_C \wedge (TrR_C \geq TotalSRate_C)$

$CollectorBwOutConstr_C \Rightarrow$
$IC_C \wedge (TotalSRate_C \leq LinkBw_{C,ZGw_C})$

constraints into *invariant* and *user-driven* constraints. Many of these constraints are mapped to the NISTIR [3] guidelines.

1) Invariant Constraints: There are various invariant constraints based on connectivity, data delivery schedule, resource availability, etc, between AMI components. These constraints must be satisfied for any successful communication.

Reachability Constraint: Reachability must hold between a pair of devices, if data is required to be transmitted between them. For example, a meter should be able to reach a collector to deliver the report to the collector. Similarly, there should be reachability from collector to headend, so that the collector can deliver the report to the headend. This constraint intuitively checks the links between a pair of devices along with satisfaction of routing/security device policies. The formalization of general reachability constraint is shown in Table III. We first define the *Forward* constraint that checks whether a specific traffic $Tr_{S,D}$ (i.e., from S to D) can be transferred from a node (X) to another node (Y) (like state transition). Then, we define $Reachable$ and the reachability constraint (*ReachabilityConstr*) on top of this. In the constraint formalization, we also model the maximum possible transmission rate (TrR) by taking the minimum bandwidth of the links across the path along with the limits that may imposed by a firewall.

Connectivity Pairing Constraint: Consistent pairing between a meter and a collector is required over reachability for successful communication. This constraint is considered as conjunction of security pairing and protocol pairing. In words, it states that the authentication and confidentiality properties of the communicating devices should match and they have a common protocol to communicate. For example, in Fig. 2, although there are 4 meters of class $m000123$ are connected with the collector $c0003$, they are not allowed to communicate as the violation of security pairing will occur due to mismatch in their authentication properties (i.e., $auth0$ and $auth1$). Similarly, a host from HAN will not be able to communicate with a meter, if that host does not support the LonTalk protocol, which is the only protocol supported by a meter. *PairingConstr* in Table III checks these issues.

Schedule Constraint: The schedule constraints (refer to Table IV) ensure the basic correctness of report or pull schedule configuration. The *MeterSampConstr* constraint states that the sampling time and the reporting base-start time of a meter must be less than (or equal to) its reporting interval, such that no reporting is done without new data. It also verifies that the sampling rate cannot be more than its maximum transmission rate. If a meter is in push mode ($Mode$ is true), then it should have a reporting schedule. A similar constraint (*CollectorPullScheConstr*) is true for a collector. If a collector is connected with some meters, whose reporting mode are pull ($Mode$ is false), then the collector should have a pull schedule for them.

Resource Constraint: There are different resource constraints (refer to Table IV), which are often relate to report/pull schedules. The *CollectorBufConsrt* constraint states that the buffer size of a collector should be greater than or equal to the cumulative sampled data size of all the connected meters to that collector. Otherwise, data loss must occur in collector buffer under any report schedule. Similarly, the *CollectorTrRateConstr* constraint states that the cumulative sampling rate of the connected meters cannot be more than the transmission rate of the collector. The *CollectorBwConstr* constraint states that the bandwidth of the link from the collector to its gateway

TABLE V
FORMALIZATIONS OF DIFFERENT USER-DRIVEN CONSTRAINTS

Data Loss (Collector Buffer Overwritting) Constraint:

$(TotalRData_C \Leftrightarrow TotalSRate_C \times Period) \Rightarrow$
$((Mode_C \Rightarrow (Period \Leftrightarrow RSInt_C)) \vee$
$(\neg Mode_C \Rightarrow (H \Leftrightarrow AttachHS_C) \wedge (Period \Leftrightarrow PRSInt_H)))$
$OverwriteProtectConstr_C \Rightarrow$
$IC_C \wedge (BufSize_C \geq TotalRData_C)$

Cyber Bandwidth Constraint:

$(Num_C \Leftrightarrow \sum_Z ZMNum_Z) \Rightarrow (MId_Z \Leftrightarrow C)$
$(TotalRRate_{H,Sche} \Leftrightarrow \sum_C (TotalSRate_C \times Num_C)) \Rightarrow$
$(H \Leftrightarrow AttachHS_C) \wedge Mode_C \wedge (RSche_C \Leftrightarrow Sche)$
$LinkBwConstr_{H,X,Y} \Rightarrow$
$HS_H \wedge (LinkBw_{X,Y} \geq TotalRRate_H)$

Assured Data Delivery:

$AssuredDelivery_{M,C,H} \Rightarrow$
$SM_M \wedge IC_C \wedge HS_H \wedge$
$Pairing_{M,C} \wedge (M \Leftrightarrow CSMId_C) \wedge Reachable_{M,C} \wedge$
$Pairing_{C,H} \wedge (H \Leftrightarrow AttacheHS_C) \wedge Reachable_{C,H} \wedge$
$ResourceConstr_{M,C,H} \wedge CyberConstr_{M,C,H}$

Quality of Delivery (Data Freshness) Constraint:

$FreshnessConstr_{M,C,H,T} \Rightarrow$
$AssuredDelivery_{M,C,H} \wedge$
$((Sum_{T1,T2} \leq T) \Rightarrow (((T1 \Leftrightarrow RSInt_M) \wedge Mode_M) \vee$
$((T1 \Leftrightarrow PRSInt_C) \wedge \neg Mode_M)) \wedge$
$(((T2 \Leftrightarrow RSInt_C) \wedge Mode_C) \vee$
$((T2 \Leftrightarrow PRSInt_H) \wedge \neg Mode_C))$

Availability Protection Constraint (Limit DoS Attack):

$(MaxTrR_{H,X,Y} \Leftrightarrow \sum_C TrR_C \times Num_C) \Rightarrow$
$Compromise_C \wedge (AttachHS_C \Leftrightarrow H) \wedge$
$Forward_{X,Y,Tr_{C,H},TrR}$
$AvailProtectionConstr_{H,X,Y} \Rightarrow$
$IC_C \wedge (LinkBw_{X,Y} \geq MaxTrR_{H,X,Y})$

must be greater than or equal to the accumulated sampling rate of all the meters connected to it. Otherwise, no schedule will be possible without data loss.

2) User-driven Constraints: To achieve correct and secure functioning of AMI network, there can exist different user-driven constraints. We focus on AMI specific constraints. Formalizations of these constraints are shown in Table V.

Data Overwrite Protection Constraint: This constraint states that the aggregate report data of all the meters connected to a specific collector must not flood the collector buffer within the reporting interval. For example, in Fig. 2, $c0005$ collector receives reports from 5 meters of $m00129$ class (sampling rate: 20KB/30secs) and 5 of $m0003$ class (sampling rate: 18KB/40secs). Therefore, $c0005$ will receive 335KB (average) data in every 60 seconds, which is to be stored in its buffer. Now, based on the report schedule, collector pushes the data to headend in every 1440 seconds. Thus, during this period, in an aggregate 8040KB data will be sent to the collector by these meters. This amount of data will flood the collector buffer (size 80000 KB), which will in turn cause data loss (i.e., initial 40KB report data will be overwritten).

Cyber Bandwidth Constraint: The *LinkBwConstr* constraint is to conform that the aggregate report rate of the collectors reporting simultaneously due to matching report schedule should not exceed the bandwidth limitation of the network path (considering a link from X to Y) connecting to headend (H). Violation of this constraint will cause link congestion/DoS.

Assured Data Delivery: This constraint requires checking the end-to-end data delivery (from a meter to a headend through a collector) to satisfy the AMI global functionality. This constraint intuitively implies the satisfaction of the following constraints: (1) reachability, (2) trusted path (successful security pairing), (3) availability of resources (conjunction of all resource constraints including data overwrite constraint as *ResourceConstr*), and (4) synchronous reporting without flooding the cyber towards the headend (conjunction of the *LinkBwConstr* constraints across the path as *CyberConstr*). A violation of these constraints can create failure in data delivery.

Quality-of-delivery Constraints: There are user-driven constraints for ensuring the quality of delivery. For example, the report freshness constraint (*FreshnessConstr*) restricts the delivery of data within a specific time window, say, T along with assured data delivery. A user can have constraint on the quality of the trusted path. For example, This requirement can be defined as the satisfaction of (i) end-to-end encryption level based on key length (say, 256 bits); and (ii) specific single or nested tunnel(s) (say, 2-level of nested tunnels) requirement.

Availability Protection Constraint: This constraint (*AvailProtectionConstr*) ensures that if there are X number (or portion) of AMI devices being compromised, assured data delivery constraint is still preserved. It intuitively verifies that DoS attack is not possible on links or endpoints, when number of compromised nodes is no more than X (say, 5% collectors).

C. AMI Constraint Verification

The formalizations of AMI device configurations and security properties are presented in Section III. We use Boolean terms to encode the Boolean configuration parameters and some integer parameters that can have very small range of values. Remaining parameters are modeled as integer terms. We normalize the parameters into integers that may take real values (e.g., bandwidth). We use *bit-vector* terms for encoding IP addresses. In some of the computations, we require multiplying/dividing two variables. But, Yices [13] SMT solver does not support such non-linear operations. Thus, we encode such operations by normalizing one of the variables to a small set of possible values and applying the operation on the other variable with one of those values by matching to the former variable. After encoding the configuration parameters into SMT variables, we model the configuration rules associated with all AMI components. The complete AMI configuration model represented by $Model_{Conf}$. Finally, during verification, we encode each AMI constraint under the same formalism.

SmartAnalyzer creates a verification query that checks the satisfaction of the constraint, Q_c, with the configuration model,

TABLE VI
A SIMPLE EXAMPLE OF RESOURCE CONSTRAINT VERIFICATION

```
(assert+ (M 0)) ;1
(assert+ (M 1)) ;2
(assert+ (= (MId 0) 0)) ;3
(assert+ (= (SSize 0) 15)) ;4
(assert+ (= (SInt 0) 30)) ;5
(assert+ (= (MId 1) 1)) ;6
(assert+ (= (SSize 1) 25)) ;7
(assert+ (= (SInt 1) 60)) ;8

(assert+ (IC 0)) ;9
(assert+ (= (CId 0) 10)) ;10
(assert+ (= (BufSize 0) 300)) ;11
(assert+ (= (CSMId 0 0) 0)) ;12
(assert+ (= (CSMId 0 1) 1)) ;13
(assert+ (= (CSMNum 0 0) 6)) ;14
(assert+ (= (CSMNum 0 1) 6)) ;15

(assert+ (⇒ CollectorBufConstr ;16
   (forall (c::(subrange 0 0))
    (⇒ (IC c)
     (let ((a::int (CSMId c 0)) (b::int (CSMId c 1)))
      (and (M a) (M b)
       (⇒ (= (SSize a) 15) (= (SData c 0) (* (CSMNum c 0) 15)))
       (⇒ (= (SSize a) 25) (= (SData c 0) (* (CSMNum c 0) 25)))
       (⇒ (= (SSize b) 15) (= (SData c 1) (* (CSMNum c 1) 15)))
       (⇒ (= (SSize b) 25) (= (SData c 1) (* (CSMNum c 1) 25)))
       (>= (BufSize c) (+ (SData c 0) (SData c 1)))))))))

(assert+ CollectorBufConstr) ;17
(check) ; SAT
```

TABLE VII
UNSAT-CORE OF THE EXAMPLE AND CORRESPONDING REPORT

SAT output:
unsat
unsat core ids: 1 2 3 4 6 7 9 11 12 13 14 15 16 17

Max-SAT output:
(Weights are only given to the assertions on the numbers of connected meters and the collector buffer size as 1 and 2 respectively)
sat
unsatisfied assertion ids: 15
. . . . . .
(= (BufSize 0) 200)
(= (CSMId 0 0) 0)
(= (CSMId 0 1) 1)
(= (CSMNum 0 0) 6)
(= (CSMNum 0 1) 1)
. . . . . .

Corresponding Report:
Collector Buffer Constraint fails.
Buffer size (200) is less than the total sampling size (90 + 150) of ...
. . . . . .
If number of connected meters decreases, the constraint will be satisfied.
. . . . . .

$Model_{Conf}$. This query is encoded as the Boolean clause: $Q_c \Rightarrow Model_{Conf}$.

The main portion of the SMT-LIB encoding of AMI configuration and the resource constraint (*CollectorBufConstr*, refer to Table IV) is shown in Table VI. In order to keep the example easy to grasp, we consider a tiny AMI configuration with two meters, one collector, and one headend. In the example, we show the verification of collector buffer constraint.

D. Verification Result Analysis and Hardening

SmartAnalyzer generates the verification results as either satisfiable (*sat*) or un-satisfiable (*unsat*). In case of unsat, SmartAnalyzer verification engine (*verifier*) provides an *unsat-core* that basically represents the constraint violation traces in the configuration. Then it systematically analyzes these violation traces and generates a comprehensive threat report for the overall AMI configuration. This report includes threat sources, targets, violating rules, threat reasonings, and the associated configuration values. Example of unsat-core for a constraint violation and associated threat report (partial) is shown in Table VII. This result corresponds to the example in Table VI, except that the buffer size is set to 200.

In case of unsat, the *hardener* module checks *max-sat* by assigning suitable weights to the assertions of the configuration parameters that can be determined from the organizational guidelines. Hardener creates remediation plan from the max-sat output. It is worth mentioning that we use quantifiers for the purpose of verifying some constraints. In such cases, Yices often returns *unknown* instead of *sat*. This implies that there is no constraint violation found by the solver. Hence, if the result is not unsat, we consider that the model is satisfied with the given constraints.

V. EVALUATION AND DISCUSSION

We evaluate SmartAnalyzer in terms of *accuracy*, *usability* and *scalability*. We analyze the tool by evaluating different constraints under real and synthetic configuration data.

A. Accuracy

Firstly, the accuracy of our tool is ensured by the use of formal constraint satisfaction checking method. In addition, we evaluate our tool with ground truth scenarios by deploying it in a small AMI testbed created in our university [14]. The testbed setup typically represents a small subset of the network shown in Fig. 1. We analyze some of the security constraints, especially, data overwrite protection and cyber bandwidth constraints. The results of our tool are cross-validated with the real scenario. For the purpose of analyzing the constraints, we slide the values of different configuration parameters, such as (i) taking very low and high pull schedule intervals for the headend, and (ii) changing the bandwidth of the links from high to very low. We find some constraint violations that lead to link flooding and data loss. In addition, we inject high amount of data through simulation (by adding multiple simulated collectors in the testbed) to observe its effect on cyber bandwidth constraint. After observing the constraint violations, we reconfigure the setup according to the remediation plan and reevaluate the constraint to see the effect. For example, in case of cyber bandwidth constraint violation, we add traffic limit in firewall rules and observe the resolving of link flooding. These tests significantly help us in verifying the accuracy of the tool.

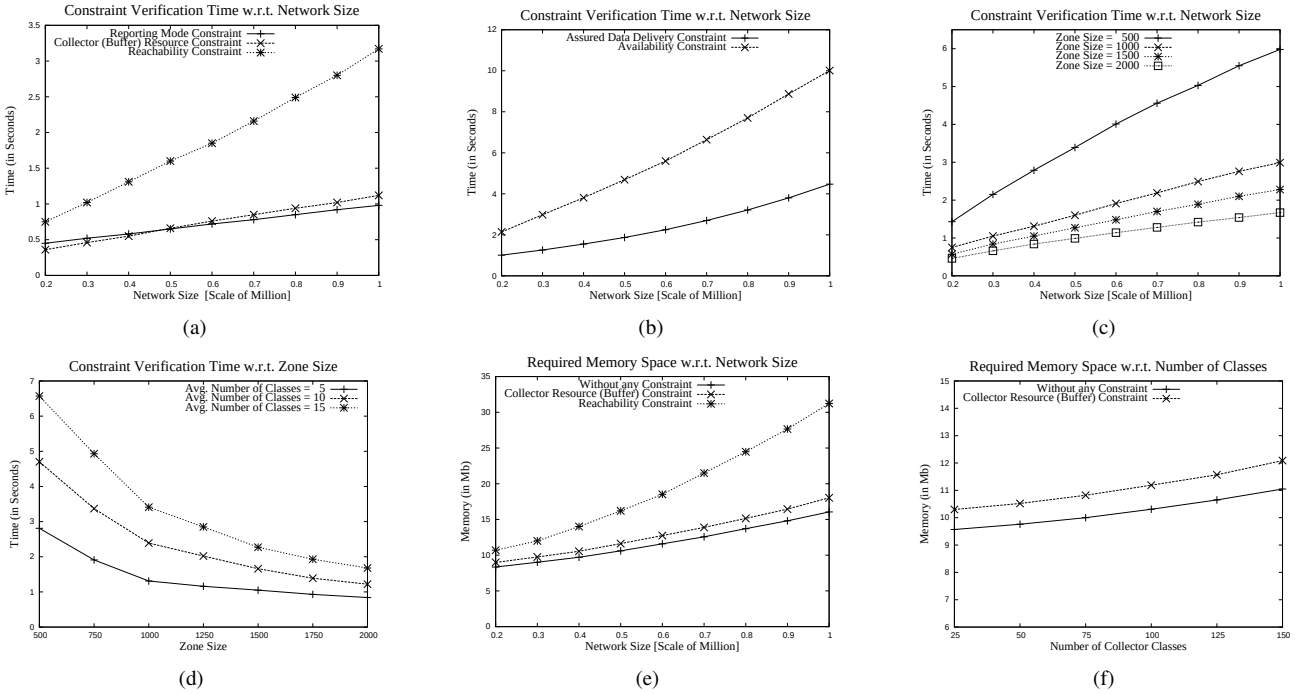

Fig. 5. (a) Impact of network size (varying number of zones) on invariant constraint verification time, (b) Impact of network size on user-driven constraint verification time, (c) Impact of zone size on constraint verification time, (d) Impact of number of collector classes per zone on constraint verification time, (e) Impact of network size on memory requirement, and (f) Impact of number of classes on memory requirement.

B. Usability

The usability of SmartAnalyzer is evaluated by providing it to different real-life experienced users and considering their feedback. The main usability of our tool lies in the operational efficiency. It allows users (i) to evaluate new AMI configuration, and (ii) to modify and reevaluate the configuration within few seconds (particularly in 5-7 mouse clicks). The input (configuration template) and output (threat report) of the tool are simple to understand and are easy to use. In addition, remediation instructions in the threat report allows a user to reconfigure the data accordingly.

C. Scalability

We evaluate the scalability of SmartAnalyzer by analyzing the *time* and *space* required in constraint verification by varying the AMI network size. We consider the network size as the total number of collectors in AMI (the number of meters are proportional to the number of collectors). The number of collectors depends on the number of collector zones and their sizes. We consider only a single headend zone (10 headends of two headend classes) in the network. We take 100 and 50 meter and collector classes respectively, while each collector is connected with 10 meters (of 2 random meter classes) on average. Each collector zone consists of around 1000 collectors (of 5 random classes). We keep the values of these parameters fixed in most of the experiments, except those cases where their impacts on the scalability are analyzed.

Impact of network size: Fig. 5(a) and Fig. 5(b) show constraint verification time w.r.t. network size. We show the verification time for different invariant constraints (i.e., reporting mode, collector resource, and reachability) and user-driven constraints (i.e., assured data delivery and availability protection constraint). A significant part of the constraint analysis time is covered by the modeling (SMT logic encoding) time, which is almost linearly dependent on the network size that varies with number of zones. Verifications of some constraints involves all (or a portion of) possible potential source/target (refer to Section III-B) nodes that implicitly increase with number of zones. Thus, the verification time of such kind of constraints (e.g., reachability) increases more with the size of the network than that of the constraints (e.g., collector resource), which are involved with the class size only. Usually, the user-driven constraint analysis time is more than the invariant constraint analysis time (see Fig. 5(b)), since most of the former constraints subsume the later constraints.

Impact of zone size and member classes: We evaluate constraint verification time w.r.t. network size for different network zone sizes. This analysis is shown in Fig. 5(c) with respect to the reachability constraint. We observe that the analysis time significantly reduces with the increase in the number of collectors in the zone. This is due to the fact that the number of zones decreases as the zone size increases, which in turn decreases overall model size and the potential sources/targets. Fig. 5(d) shows the constraint verification time taking a fixed zone size and varying the number of average classes per zone. We find that the time increases, if variation of classes increases.

SMT space requirement: The space requirement (memory

used) of the SMT solver [13] is evaluated by changing the network size (i.e., number of zones) and the number of classes. Such analysis results are shown in Fig. 5(e) and Fig. 5(f). We observe that the space requirement increases linearly with the network size. Similar to the analysis time, the space for constraint verification is sum of the space for modeling of AMI configuration and that of for modeling a constraint. The figures justify this by showing that less space is required when no constraint is verified. The constraints involving more quantifiers require larger memory space for encoding. Fig. 5(e) shows such comparison between collector resource and reachability constraints.

D. Discussion

SmartAnalyzer can successfully identify possible threats on AMI by constraint satisfaction checking. It is highly scalable with the network size. However, there is a couple of limitations of the tool. First, we have used device and property level abstraction for achieving scalability under large scale smart grid configuration, which in turn may not provide fine-grained attack path. Second, due to the use of Yices SMT solver as the core analysis engine , we had to consider different bounds in arithmetic computations. Moreover, the tool does not provide the functionality for analyzing some of the inherent smart grid security properties like LonTalk protocol configuration.

VI. Related Work

Throughout the last decade, significant amount of works [1][3][4] have been initiated on describing the interoperability among heterogeneous smart grid components including security issues based on different attack scenarios. These works also describe the operational functionalities of AMI components and energy providers internal system with guidelines for secured communication between them. McDaniel et al. present the security and privacy challenges in smart grid in [5]. This work reports that appropriate security policies need to be enforced for communication between the home users and the energy providers internal system. The authors in [6] propose an artificial intelligent based approach for analyzing risks in smart grid networks. However, in their analysis, they do not consider network link capacity, bandwidth and different communication modes of AMI components. Anwar et al. propose a framework [10] for modeling power grid and its control elements using first order logic. This framework is capable of evaluating power flows, overloading violations in smart grid. Liu et. al. [9] present a study on false data injection attacks in power grid. McLaughlin et. al. [11] presents an approach for penetration testing on AMI systems. They develop archetypal and concrete attack trees for energy fraud, denial of service and targeted disconnect attacks. However, these works do not analyze various misconfiguration problems and security controls on power grid networks.

Analyzing an AMI system requires modeling of its configuration and various security controls, which are more complex than the traditional network. The survey reveals that no significant research has been done on formal modeling of the complex AMI configuration and analyzing various security constraints on the configuration. Therefore, SmartAnalyzer is a novel and useful tool for provably analyzing operational consistency and security controls in AMI. Moreover, the tool provides possible remediation plans for the constraint violations, which can be used by the energy providers for reconfiguration planning towards security hardening.

VII. Conclusion

Due to the heterogeneity in AMI device configurations and emerging security threats on it, automated analysis of AMI configuration is an important but challenging problem. In this research, considering this challenge, we built SmartAnalyzer, an automated tool for AMI configuration verification. First, we analyze and identify different invariant and user-driven constraints, violation of which can cause various threats on AMI. Then based on these constraints, we develop SmartAnalyzer using SMT based formal logic. Under any constraint violation, the tool generates a threat report that includes the reasoning of the violation and the possible remediation plan. The accuracy of the tool is evaluated in an AMI testbed through cross-validation with the ground truth. We evaluate the scalability of SmartAnalyzer in different test configurations. We achieve significantly high scalability by applying the property level abstractions in the model. We observe that the constraint verification time lies within 10 seconds for a network of 1 million collectors. Our tool is highly usable as it requires only few steps to analyze new configurations. In future, we plan to explore AMI configuration synthesis problem that will satisfy the necessary security constraints.

References

[1] *Security in the Smart Grid*, ABB White Paper, ABB Inc. Cary, 2009.

[2] B. Brown et al. *AMI system security requirements: V1.01* AMI-SEC Task Force, 2008.

[3] *NISTIR 7628: Guidelines for Smart Grid Cyber Security*, Smart Grid Interoperability Panel- Cyber Security Working Group, August 2010.

[4] *AMI System Security Requirements: V1.01.* AMI-SEC Task Force, Available in http://osgug.ucaiug.org/utilisec/amisec/.

[5] Patrick McDaniel and Sean W. Smith, *Security and Privacy Challenges in Smart Grid*, IEEE Security and Privacy, June, 2009.

[6] Y. Wang, D. Ruan, J. Xu, M. Wen and L. Deng. *Computational Intelligence Algorithms Analysis for Smart Grid Cyber Security*, Lecture Notes in Computer Science, Vol. 6146, p. 77-84, Springer, 2010.

[7] Richard Alimi, Ye Wang, and Y. Richard Yang. *Shadow configuration as a network management primitive*. ACM SIGCOMM conference on Data communication, 2008.

[8] X. Ou, S. Govindavajhala, and A. Appel. *MulVAL: A Logic-based Network Security Analyzer*, 14th USENIX Security Symposium, 2005.

[9] Y. Liu, P. Ning and M. K. Reiter. *False Data Injection Attacks Against State Estimation in Electrical power Grids*, The 16th ACM Conference on Computer and Communications Security, November 2009.

[10] Z. Anwar, R. Shankesi and R. H. Campbell. *Automatic Security Assessment of critical cyber-infrastructures*, 38th Annual IEE/IFIP International Conference on Dependable Systems and Networks, 2008.

[11] S. McLaughlin, et al. *Multi-vendor Penetration Testing in the Advanced Metering Infrastructure*. ACSAC, USA, 2010.

[12] E. Al-shaer , W. Marrero , A. El-atawy and K. Elbadawi, *Network Configuration in A Box: Towards End-to-End Verification of Network Reachability and Security*. ICNP, 2009.

[13] Bruno Dutertre and Leonardo De Moura, *The Yices SMT Solver*, Technical Report, 2006, Available in http://yices.csl.sri.com/tool-paper.pdf

[14] *AMI Smart Grid Testbed at UNC Charlotte*, http://www.cyberdna.uncc.edu/events.php.