Chapter 13

AUTOMATED ASSESSMENT OF COMPLIANCE WITH SECURITY BEST PRACTICES

Zahid Anwar and Roy Campbell

Abstract Several standards and best practices have been proposed for critical infrastructure protection. However, the scale and complexity of critical infrastructure assets renders manual compliance checking difficult, if not impossible. This paper focuses on the automated assessment of security compliance of electrical power grid assets. A security model based on predicate calculus is used to express infrastructure elements (e.g., devices, services, protocols, access control implementations) as "facts" and security standards and best practices as "rules" that specify constraints on the facts. A tool chain is applied to automatically generate the security model from specifications and to check compliance with standards and best practices. The tool chain also supports the visualization of network topology and security assessment results to reveal possible points of attack.

Keywords: Security best practices, compliance assessment, first order logic

1. Introduction

The Industrial Security Incident Database (ISID) [4] reveals an alarming number of attacks on cyber infrastructures, more than half of them originating from external sites. The electrical power grid is especially vulnerable to attack. An experimental cyber attack on an electrical power plant generator made headlines in September 2007 [14]. While the details of the attack have not been released, it is clear that researchers were able to hack into the SCADA network and change its configuration to cause significant damage to the generator.

To address security problems, the Federal Energy Regulatory Commission (FERC) has approved eight cyber security and critical infrastructure protection standards proposed by NERC [9, 20]. However, there is considerable flexibility

Please use the following format when citing this chapter:

Anwar, Z. and Campbell, R., 2008, in IFIP International Federation for Information Processing, Volume 290; *Critical Infrastructure Protection II*, eds. Papa, M., Shenoi, S., (Boston: Springer), pp. 173–187.

with regard to their implementation. For example, CIP-005 Requirement 4 (R4) states:

> "*The responsible entity shall perform a cyber vulnerability assessment of the electronic access points to the electronic security perimeter(s) at least annually.*"

Obviously, there are several ways of securing the electronic perimeter. Two of the most popular techniques are firewall deployment and access control.

Similarly, CIP-009 Requirement 2 (R2) discusses the security implications of operating procedures and disaster recovery procedures, their relative orderings, timeframes and requirements. While the NERC standards do not discuss implementation, several entities have published guidelines for implementing security best practices for SCADA systems [3, 7, 11, 15, 17]. Most of these guidelines are informal English descriptions of SCADA infrastructure configurations, firewall rules, allowable services and security protocols. Our strategy is to formalize these best practices and to use them in automated conformance checking of SCADA network configurations.

We use predicate logic to model SCADA and enterprise networks along with their security properties. The model creates a comprehensive network dependency graph containing information about physical connections (e.g., links) and logical connections (e.g., service dependencies). This information is automatically obtained from SCADA specification languages such as the Common Information Model (CIM) [8]. Best practices modeled as rules defined in terms of facts are used to determine whether or not the dependency graph satisfies the security constraints.

2. Related Work

A survey of SCADA security implementations reveals a lack of authentication mechanisms, limited patch protection and uncontrolled Internet connections [13]. This situation exposes SCADA systems to a variety of exploits, including simple SQL injection attacks. However, even when the vulnerabilities of individual components are known, no adequate tools are available for reasoning about the overall security of a system.

The SINTEF CORAS Project [19] has developed a risk analysis methodology that models threats as unwanted system features. The system and its associated threats are modeled using Unified Modeling Language (UML) diagrams that support security risk assessments. An XML schema is available for communicating risk and vulnerability assessment data in a standardized format.

Masera and Nai Fovino [12] have proposed a service-oriented architecture for conducting security assessments. A service-oriented description of a system (where components and subsystems are interconnected by "service chains") is used to identify attacks and reason about the propagation of faults and failures. Our model is not limited to checking for particular attacks; instead, it allows for conformance checking against security standards and best practices that defend against a variety of attacks simultaneously.

Chandia and colleagues [5] have proposed a standards-based security services suite that provides security functionality at different levels of the network infrastructure. Their approach involves message monitoring, protocol-based solutions, tunneling services, middleware components and cryptographic key management, along with a network forensic system for analyzing attacks. Our approach is different in that it involves the static analysis of security conformance of implementations instead of implementing security solutions.

While the use of attack graph models to identify vulnerabilities in large-scale networks is fairly mature, little work has focused on the automated generation of these models, especially for cyber infrastructures. Also, few researchers have investigated the use of dependency graphs for vulnerability assessment and automated checking against security standards and best practices.

3. Security Model

Our security model captures the static aspects of SCADA systems, including network topology, devices, services, connectivity and security properties. A SCADA system is expressed as a dependency graph G and a set of rules expressing security standards and best practices.

3.1 Dependency Graph

A dependency graph G is defined as a tuple $(D, E, S, V, S_T, D_T, S_p)$ where D is the set of devices, $E \subseteq D \times D$ is the set of edges between two physically connected devices, S is the set of services, V is the set of vulnerabilities, S_T is the set of service types, D_T is the set of device types and S_p is the set of security protocols.

The following functions provide attribute mappings to the various devices and dependencies:

- $devof : S \rightarrow D$ maps a service to the device that hosts it.
- $hostedsvs : D \rightarrow \mathbb{P}\{S\}$ s.t. $devof(S) = D$ maps a device to the services it hosts.
- $defsvs : D \rightarrow S$ maps a device to its default service.
- $depdtsvs : S \rightarrow \mathbb{P}\{S\}$ maps a service to the set of services on which it depends.
- $trusteddevs : D \rightarrow \mathbb{P}\{D\}$ maps a device to a set of trusted devices.
- $secprots : S \rightarrow \mathbb{P}\{S_p\}$ maps a service to a set of security protocols it uses.
- $typeofsvs : S \rightarrow stype$ where $stype \in S_T$ maps a service to its type.
- $typeofdvs : D \rightarrow dtype$ where $dtype \in D_T$ maps a device to its type.
- $knownvuls : stype \rightarrow \mathbb{P}\{V\}$ maps a service to its set of known vulnerabilities.
- $priv : S \rightarrow privlvl$ where $privlvl \in \{none \leq user \leq root\}$ maps a service to its privilege level.
- $exploitability : V \rightarrow likelihood$ where $likelihood \in \mathbb{R}$ $(0 \leq n \leq 1)$ maps a vulnerability to the likelihood it will be exploited.

The dependency graph G is modeled as facts in first order predicate logic.

3.2 Security Standards and Best Practices

Security standards and best practices are expressed as rules whose terms are constraints on G.

Intranet Services. Services between a process control network (PCN), an enterprise network (EN) and the Internet should be allowed strictly on a need basis. IAONA's template for protocol access in industrial environments states that incoming DNS, HTTP, FTP, telnet and SMTP traffic to a PCN should be discouraged unless absolutely required [10].

We express this best practice as:

- $\forall d_1, \forall d_2 \in D\ [typeof(d_1, EN) \wedge typeof(d_2, PCN) \wedge$
  $\forall s_1, \forall s_2 \in S\ [devof(s_1, d_1) \wedge devof(s_2, d_2) \wedge depends(s_1, s_2)$
  $\Rightarrow s_2 \notin \{dns, http, ftp, telnet, smtp\}]]$

with the auxiliary functions:

- $typeof : \forall d \in D, \forall x \in D_T(typeofdvs(d) = x) \Rightarrow typeof(d, x)$
- $devof : \forall s \in S, \forall d \in D(devof(s) == d) \Rightarrow devof(s, d)$
- $depends : \forall d_1, \forall d_2 \in D, \exists s_1, \exists s_2 \in S[devof(s_1, d_1) \wedge$
  $devof(s_2, d_2) \wedge depdtsvs(s_1, s_2) \Rightarrow depends(d_1, d_2)]$.

The rule checks if a service dependency exists from an EN device to a PCN device, where both devices are in a substation. If a dependency exists, then it should not be of the type DNS, HTTP, FTP, telnet or SMTP.

Access Control Implementation. The American Gas Association's document on cryptographic protection of SCADA communications [1] states that in a proper access control implementation, a service should provide an authentication scheme and also use communication protocols that guarantee confidentiality and integrity.

We express this best practice as:

- $\forall d_{alice}, \forall d_i, \forall d_j, \forall d_{bob} \in D\ [depends(d_{alice}, d_{bob}) \wedge$
  $d_i \in path(d_{alice}, d_{bob}) \wedge d_j \in path(d_{alice}, d_{bob}) \wedge (d_i, d_j) \in E$
  $\Rightarrow (secprots(defsvs(d_i)) \cap secprots(defsvs(d_j))) \neq \emptyset \wedge$
  $(keys(d_{alice}) \cap keys(d_{bob})) \neq \emptyset]$

with the auxiliary functions:

- $path : path(d_1, d_k) = d_1, d_2, ...d_{k-1}, d_k$ s.t. $\forall_{1 \leq i < k-1}(d_i, d_{i+1} \in E)$
- $keys : D \rightarrow \mathbb{P}\{K\}$

where *path* is a mapping from a pair of devices to the set of paths between them and *keys* is a mapping from a device to the set of pre-shared keys it has with other devices.

Table 1. Firewall architectures.

Type (Rating)	Description
Dual-Homed Server (1)	This design installs two network interface cards on devices requiring access to both networks, which violates the principle of no direct Internet access from the PCN. This configuration was severely affected by the Slammer worm in January 2003.
Dual-Homed Host Firewall (2)	The host-based firewall on a dual-homed machine prevents traffic from traversing the PCN-EN boundary. However, it offers low granularity with multiple shared servers when remote PCN management is required.
Packet Filtering Router (2)	This design uses a Layer 3 switch with basic filters to block unwanted traffic. It offers limited protection because it is not stateful and assumes that the EN is highly secure.
Two-Port Dedicated Firewall (3)	This aggressively configured stateful firewall provides considerable security. The shared device is positioned in the PCN or EN and the firewall is configured with the appropriate rules.
Two-Zone Firewall-Based DMZ (4)	This design positions shared devices in their own DMZs, which eliminates direct communication between the plant floor and the EN. Multiple DMZs ensure that only desired traffic is forwarded between zones. However, compromised entities in the DMZs may be used as staging points for attacks against PCN devices.
Firewall and VLAN Design (4.5)	This design partitions PCNs into subnets so that devices that require little or no communication are placed in separate networks and only communicate via Layer 3 switches.

The best practice predicate checks if a service s_i implements access control, confidentiality and integrity correctly. It does this by checking if all its dependent services have a common shared-key mechanism. The helper function *path* checks if the default service on each pair of devices along the path from the queried service to the dependent services share common security properties. Function *keys* checks if all the dependent services use a pre-shared key.

Firewall Deployment. NISCC's document on SCADA firewall deployment [3] states that traffic from an enterprise LAN must be separated from an industrial control LAN by a firewall. Table 1 presents five firewall architectures and their security ratings. We express the firewall best practices as:

- $\forall d_e, \forall d_p, \forall d_s \in D\ [typeof(d_e, EN) \wedge typeof(d_p, PCN) \wedge depends(d_e, d_s) \wedge depends(d_p, d_s) \wedge (d_s \in path(d_e, d_p)) \wedge \exists s_f \in S[devof(s_f, d_s) \wedge typeof(s_f, firewall) \Rightarrow dualhomedfirewalled]]$
- $\forall d_e, \forall d_p, \forall d_s \in D\ [typeof(d_e, EN) \wedge typeof(d_p, PCN) \wedge depends(d_e, d_s) \wedge depends(d_p, d_s) \wedge \exists d_{f1}, \exists d_{f2}, \exists d_{f3} \in D\ [(d_{f1} \in path(d_e, d_s)) \wedge (d_{f2} \in path(d_p, d_s)) \wedge (d_{f3} \in path(d_e, d_p)) \wedge typeof(d_{f1}, firewall) \wedge typeof(d_{f2}, firewall) \wedge typeof(d_{f3}, firewall) \Rightarrow dmz$

with the auxiliary predicate:

- $typeof : \forall s \in S, \forall x \in S_T(typeofsvs(s) = x) \Rightarrow typeof(s, x)$.

This predicate identifies the shared network devices (e.g., historians, aggregators and access points). Servers accessed by dependent services running on devices in the PCN and EN are characterized as shared. The proper placement of these devices with respect to firewalls determines the architecture to be used. The first predicate states that if all the paths between the two dependent PCN and EN devices pass through the shared device and the shared device is running a personal firewall service, then a dual-homed host firewall with a security rating of 2 should be used. The second predicate checks if all possible paths between PCN and ECN devices d_p and d_e, d_p and the shared device d_s, and d_e and d_s pass through firewalls, in which case, the shared device should be placed in a isolated DMZ with a security rating of 4. Predicates for the other four firewall architectures are specified along the same lines.

4. Tool Chain Architecture and Implementation

This section discusses the architecture and implementation of the security assessment tool chain (Figure 1).

4.1 Parsing Specification Files

The dependency graph of the SCADA network is generated from annotated specifications written in CIM [8] with the help of a parser tool and stored in a Prolog database. CIM is an object-oriented cyber infrastructure modeling language developed by EPRI for representing objects encountered in electric utilities. The objects are represented as classes that have attributes and relationships (with other classes). The CIM RDF schema, which is documented as IEC Standard 61970-501, is self-describing because it is based on XML.

CIM RDF classes are mapped to entities in our security model. Figure 2 shows the XML description and the Prolog version of an actuator for a disconnect switch (DS3). The XML attributes provide detailed information about switch functions and the SCADA elements that control it.

The parser begins by identifying the principal entities such as devices, connections and services, and populates their attributes based on the properties and relationships of CIM objects. Some attributes (e.g., inter-service

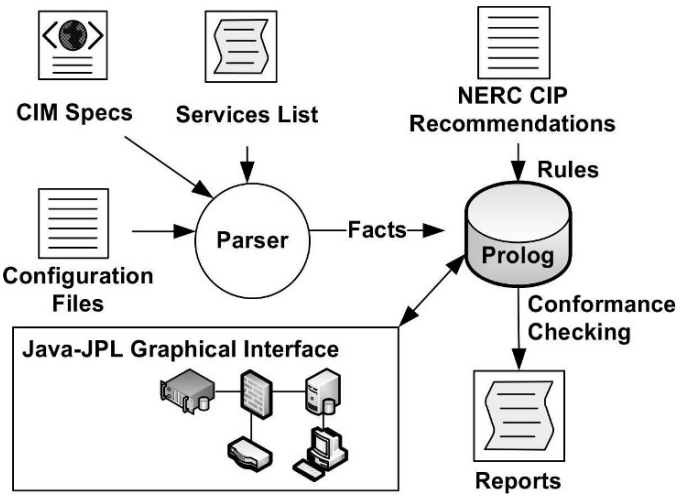

Figure 1. Security assessment tool chain.

data dependencies and security protocols used by services) are not covered by CIM. These attributes are incorporated by parsing the firewall configuration logs, manually annotating the CIM XML or looking-up a services-to-security-properties table when a services entity is encountered. Services running on a device may be determined by running `nmap` port scans; communicating services are identified by parsing firewall logs.

Discovering service dependencies from firewall logs is not a new technique. Several open source tools are available for traffic analysis and network dependency visualization based on firewall logs (e.g., `psad` [16] and `afterglow` [18]).

4.2 Predicate Calculus Implementation

The predicate calculus security model is implemented as a form of Horn Clause Logic using SWI-Prolog (version 5.6). The devices, services, connectivity and dependencies identified by the parser are asserted as Prolog facts. Table 2 lists the Prolog facts that describe the interconnections and service dependencies of a sensor, which reports readings to a historian and an administrator's computer through a firewall. The connected predicate shows a bidirectional link between two devices.

4.3 Rules Implementation

The rules and helper functions were implemented to check for conformance with best practices. Prolog rules are essentially goals that check for other subgoals to hold true; subgoals are other rules or primitive facts.

Table 3 summarizes the Prolog implementation of an American Gas Association access control best practice. We explain the Prolog listing from the bottom up. The `ck_ConformanceTo_CIP002-08` rule (Line 32) takes three arguments:

CIM XML Specification

```
<!-- Describes our Substation Architecture -->
<SubstationArchitecture>
 <class name="CIM_LogicalSwitch"
  Superclass="CIM_LogicalDevice">
  <cim:CIM_LogicalSwitch ID="ActDS3"
   cim:type="DisconnectSwitch" cim:State="Closed"
   cim:PowerSystemResourceName=
     "Disconnect Switch No 3 Actuator"
   cim:Manufacturer="General Electric"
   cim:Controllerforresource="#DS3">
 <class name="CIM_SerialLink" Superclass="CIM_Link">
  <cim:CIM_SerialLink ID="SlinkActDS3" cim:source="PLC2"
   cim:dest="ActDS3"/> </class>
 <class name="CIM_Firmware" Superclass="CIM_Service">
  <cim:CIM_Firmware ID="F.wareActDS3"
   cim:ver="1.0" cim:type="ModbusSlave"
   cim:PowerSystemResourceName=
     "Actuator Service for ActDS3"
   cim:secprops="TLS" cim:dependsupon="PLC2Master"
   cim:port="502"/> </class>
  </cim:CIM_LogicalSwitch>
 </class>
.
.
</SubstationArchitecture>
```

Prolog Primitive Facts

```
device( ActDS3,       //ID
 DisconnectSwitch,    //Type
 Closed,              //State
 FwareActDS3          //Services
).

connected( PLC2,      //Start Node
 ActDS                //End Node
).

service( FwareActDS3,//Service ID
 1.0,                 //Version Number
 ModbusSlave,         //Service Type
 [PLC2Master],        //Dependent Upon
 [TLS],               //Security Protocols
 502                  //Connecting Port
).
```

Figure 2. CIM XML and Prolog specifications of a substation.

two communicating devices and a *Path* variable. It then calls a helper rule *path* (Line 1), which finds a path (list of nodes) from A to B. This is accomplished using a recursive travel rule: a path from A to B is obtained if A and B are connected (Line 5) and a path from A to B is obtained provided that A is connected to a node C different from B that is not on the previously visited

Table 2. SCADA device models described as Prolog facts.

```
% device(ID,TYPE,GROUP,SERVICES_LIST,COORDX,COORDY)
device(adminpc,pc,en,[ssh1,sqlclient1],10,20).
device(historian,pc,en,[rlogind1,postgresqld],10,30).
device(sensor,pc,pcn,[rlogin2,sqlclient2],30,10).
device(serviceproxy,router,firewall,[firewalld],10,10).

% service(ID,TYPE,VER,PRIV_LEVEL,PROTOCOL,ACL)
service(sqlclient1,database_client,2003,user,odbc,_).
service(sqlclient2,database_client,2000,user,odbc,_).
service(postgresqld,database_server,1998,root,odbc,[sqlclient1,sqlclient2]).

% bydirectionlink(SRC,DEST)
connected(adminpc,serviceproxy).
connected(historian,serviceproxy).
connected(sensor,serviceproxy).
```

Table 3. Access control implementation.

```
1  path(A,B,Path) :-
2          travel(A,B,[A],Q),
3          reverse(Q,Path).
4
5  travel(A,B,P,[B|P]) :-
6          connected(A,B).
7
8  travel(A,B,Visited,Path) :-
9          connected(A,C),
10         C \== B,
11         \+member(C,Visited),
12         travel(C,B,[C|Visited],Path).
13
14 is_access_control(DevA,DevB) :-
15         keys(DevA,AuthMechA),
16         keys(DevB,AuthMechB),
17         match(AuthMechA,AuthMechB).
18
19 is_end2end_conf_integ(SecPropsList,[Head]) :-
20         defsvs(Head,dservice),
21         secprots(dservice,SPList),
22         SecPropsList = SPList.
23
24 is_end2end_conf_integ(SecPropsList,[Head|Tail]) :-
25         defsvs(Head,dservice),
26         secprots(dservice,SPList1),
27         is_end2end_conf_integ(SPList2,Tail),
28         intersection(SPList1,SPList2,CommonSPList),
29         nth0(0,CommonSPList,_),
30         SecPropsList=SPList1.
31
32 ck_ConformanceTo_CIP002-08(DevA,DevB,Path) :-
33         path(DevA,DevB,Path),
34         is_access_control(DevA,DevB),
35         is_end2end_conf_integ(Path).
```

part of the path, and a path is found from C to B (Line 8). Avoiding repeated nodes ensures that the program halts. Once the path is known, checking access control (Line 14) is a matter of comparing if both the communicating nodes use

Table 4. Java-JPL code for querying and importing path information.

```
1  Variable X = new Variable("X");
2  Variable Y = new Variable("Y");
3  Variable P = new Variable("P");
4  Term arg[] = { X,Y,P };
5  Query q = new Query("path", arg);
6
7  while (q.hasMoreElements()) {
8      Term bound_to_x = (Term) ((Hashtable) q.nextElement()).get(↩
           "P");
9      String[] strarray = jpl.Util.atomListToStringArray(↩
           bound_to_x);
10     for (int i=0; i<strarray.length; i++) {
11         System.out.println(strarray[i]);
12     }
13 }
```

a pre-shared key or PKI authentication. Checking confidentiality and integrity (Lines 19 and 24) amounts to checking if every pair of consecutive nodes on a path share an encryption channel (e.g., IPSec or TLS).

4.4 Graphical User Interface

A Java-based GUI front-end to Prolog facilitates user interaction with the system. The implementation leverages JPL, a set of Java classes and C functions, which provides a Java-Prolog interface by embedding a Prolog engine in Java VM. Annotating each device in a CIM specification with (x, y) coordinates enables SCADA network data to be imported and viewed in a Java grid panel. This approach allows for more user interaction than other network visualization tools (e.g., Graphviz [2] and CAIDA [6]). For example, a user can hover over a device icon to see a detailed listing of its security attributes or click on devices of interest and formulate a query. Table 4 presents JPL code that imports information about all possible paths between two devices in the form of Prolog lists for display (Table 3 describes the predicate implementation).

Three JPL variables (Lines 1–3) are created to hold the two devices and the list of possible paths between them. A query is then formulated and sent to the Prolog engine, which populates these values. The JPL library provides several utility functions such as `atomListToStringArray` to perform conversions between Java and Prolog types.

5. Evaluation

Our implementation involves approximately 1,620 lines of Prolog code (not including network and workflow encodings) and 3,500 lines of Java code. The twelve best practices rules took roughly 30 hours to encode in Prolog. The implementation was tested with several substation network-level scenarios (involving less than 100 machines). Each scenario executed in a few seconds on

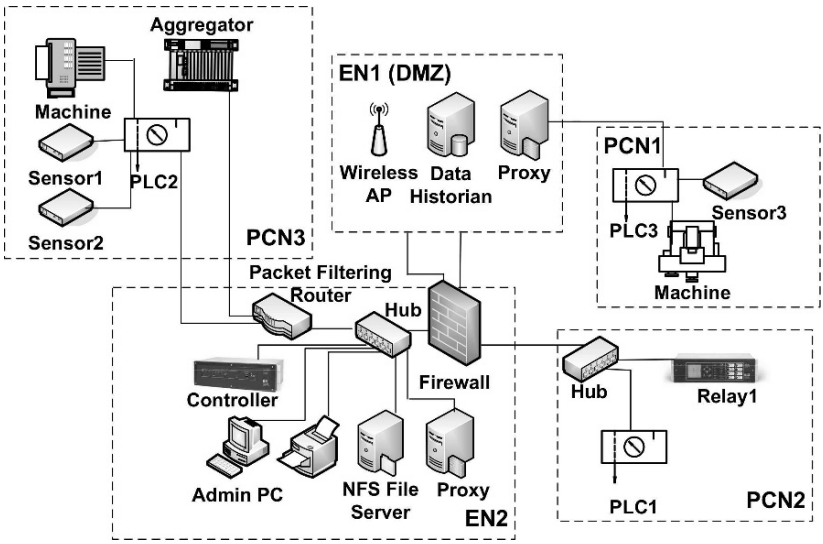

Figure 3. SCADA architecture with two ENs and three PCNs.

an Intel Core2Duo 2.0 GHz machine running Ubuntu Linux 7.10. This section presents the results of access control and firewall deployment evaluations for one of these scenarios.

Figure 3 presents a typical SCADA architecture containing two subnets (EN1 and EN2) with several enterprise machines and devices, and three subnets (PCN1, PCN2 and PCN3) with process control devices. EN1 has two important devices, the Wireless AP (access point) and Data Historian. The data relationships are as follows. The Data Historian is a shared device that logs events in several SCADA devices. It is accessed by local and remote users for supervisory purposes. The Data Historian connects directly to devices in PCN1 via a proxy server; this configuration enables the vendor to maintain the machine remotely via the Internet. The Data Historian also logs events from Relay1 in PCN2 and is accessed by the Admin PC and NFS File Server in EN3. Sensor1 and Sensor2 in PCN3 are managed by the Controller in EN3 and their events are logged by the Data Historian. Services provided by the Controller are accessed by the Admin PC.

5.1 Test 1: Access Control Implementation

Figure 4 shows the dependency graph of the two sensors in PCN3 that report their readings to two enterprise devices (Controller and Data Historian) through several proxy servers and PLCs, not all of which support IPSec or TLS stacks for confidentiality.

Table 5 summarizes the results of running a "correct implementation of access control" query for confidentiality and integrity (C/I), authentication (Auth) and CIP conformance (Conf) on the sensors.

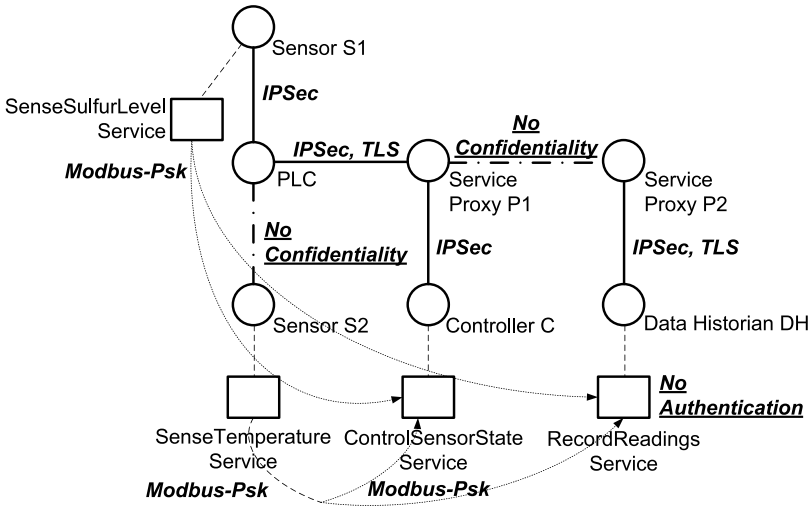

Figure 4. Access control conformance of the SCADA architecture.

Table 5. Access control implementation results.

Source	Sink	Path	C/I	Auth	Conf
S1	C	S1 → PLC → P1 → C	Yes	Yes	Yes
S1	DH	S1 → PLC → P1 → P2 → DH4	No	No	No
S2	C	S2 → PLC → P1 → C	No	Yes	No
S2	DH	S2 → PLC → P1 → P2 → DH	No	No	No

The invocation `ck_ConformanceTo_CIP002(sensors,sinks,Paths)` reveals that the data association channel between the sensors and controller enforces integrity because of an end-to-end pre-shared key. However the channel between sensor S2 and the controller does not have confidentiality because the hop between S2 and the PLC does not support a confidentiality protocol. A similar problem occurs along the paths between the sensors and the data historian where the hop between the two proxies is not confidential. The only channel that passes the test successfully is between S1 and the controller because the sensor supports IPSec as an encryption protocol.

5.2 Test 2: Firewall Deployment

Firewall deployment was evaluated by starting with the original configuration, identifying the offending link, incorporating the appropriate firewall, and repeating the conformance checking of the new configuration. Table 6 presents the results. The original configuration has a security rating of 2 due to the direct historian-PCN1 link that was incorporated for vendor convenience. This

Table 6. Firewall deployment results.

Num	Substation Architecture	Rating	Offending Link
1	Original configuration (Figure 3)	2	DH-PCN1
2	Same as 1 with DH-PCN1 link removed	2	S2-PacketFilteringRouter-C
3	Same as 2 with router replaced with a stateful firewall	3	C-AdminPC
4	Same as 3 with C moved to the DH subnet	4	None

link poses a serious threat to the substation as it potentially allows direct Internet access to the plant floor. Note that the security rating of the entire substation is dependent on the security rating of the weakest link. Removing this link (by incorporating a new firewall or relocating PCN devices) and repeating the analysis produces a security rating of 2. This is due to the presence of a packet filtering router that separates devices in PCN3 from the controller in EN3. Upon replacing the router with a stateful firewall, the new configuration has a security rating of 3 with all the shared devices positioned behind the proper firewalls. Finally, moving the shared controller to same subnet as the historian produces a DMZ configuration (security rating 4) with all the shared devices located in a separate subnet.

6. Conclusions

The predicate-calculus-based security model described in this paper expresses infrastructure elements as facts and security standards and best practices as rules that specify constraints on the facts. The implemented tool chain automatically generates a security model from SCADA infrastructure specifications and checks it for compliance with security standards and best practices. The tool chain provides a rich front-end to the predicate logic engine, which enables security administrators to compose their own queries during security assessments. It also supports the visualization of network topology and security assessment results to reveal possible points of attack.

Although the approach has been tested on infrastructures with less than 100 nodes, it is scalable to large infrastructures because best practices are typically specified at the substation level. Moreover, checking system conformance against best practices is a static process that is typically performed offline.

The model allows the implementation of checks against other standards and best practices with minimal changes; however, the generation of the security model from CIM specifications could be improved. Our future work will focus on integrating security tools such as Nessus that automatically provide details of

services running on devices along with device dependencies and vulnerabilities. Also, the security model will be extended to include SCADA infrastructures that are complementary to the power grid (e.g., water supply and telecommunications systems).

References

[1] American Gas Association, Cryptographic Protection of SCADA Communications; Part 1: Background, Policies and Test Plan, AGA Report No. 12 (Part 1), Draft 5, Washington, DC (www.gtiservices.org/security/AGA12Draft5r3.pdf), 2005.

[2] AT&T Research, Graphviz – Graph Visualization Software, Florham Park, New Jersey (www.graphviz.org).

[3] British Columbia Institute of Technology, Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks, National Infrastructure Security Co-ordination Centre, London, United Kingdom, 2005.

[4] British Columbia Institute of Technology, Industrial Security Incident Database, Burnaby, Canada.

[5] R. Chandia, J. Gonzalez, T. Kilpatrick, M. Papa and S. Shenoi, Security strategies for SCADA networks, in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 117–131, 2007.

[6] Cooperative Association for Internet Data Analysis, The CAIDA web site, La Jolla, California (www.caida.org).

[7] R. Dacey, Critical Infrastructure Protection: Challenges in Securing Control Systems, Report GAO-04-140T, United States General Accounting Office, Washington, DC (www.gao.gov/new.items/d04140t.pdf), 2004.

[8] Distributed Management Task Force, Common Information Model (CIM) Infrastructure Specification, Document DSP0004 Version 2.3 Final, Portland, Oregon (www.dmtf.org/standards/published_documents/DSP0004V2.3_final.pdf), 2005.

[9] Federal Energy Regulatory Commission, Mandatory Reliability Standards for Critical Infrastructure Protection, Docket No. RM06-22-000; Order No. 706, Washington, DC (ferc.gov/whats-new/comm-meet/2008/011708/E-2.pdf), 2008.

[10] Industrial Automation Open Networking Association, The IAONA Handbook for Network Security, Version 1.3, Magdeburg, Germany (www.iaona.org/pictures/files/1122888138–IAONA_HNS_1_3-reduced_050725.pdf), 2005.

[11] Instrumentation Systems and Automation Society, Security Technologies for Manufacturing and Control Systems (ANSI/ISA-TR99.00.01-2004), Research Triangle Park, North Carolina, 2004.

[12] M. Masera and I. Nai Fovino, A service-oriented approach for assessing infrastructure security, in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 367–379, 2007.

[13] D. Maynor and R. Graham, SCADA security and terrorism: We're not crying wolf! presented at the *Black Hat Federal Conference*, 2006.

[14] J. Meserve, Sources: Staged cyber attack reveals vulnerability in power grid, Cable News Network, Atlanta, Georgia (www.cnn.com/2007/US /09 /26/power.at.risk), September 26, 2007.

[15] National Institute of Standards and Technology, Standards for Security Categorization of Federal Information and Information Systems, FIPS Publication 199, Gaithersburg, Maryland, 2004.

[16] M. Rash, `psad`: Intrusion detection for `iptables` (www.cipherdyne.com /psad).

[17] R. Ross, A. Johnson, S. Katzke, P. Toth, G. Stoneburner and G. Rogers, Guide for Assessing the Security Controls in Federal Information Systems, NIST Special Publication 800-53A, National Institute of Standards and Technology, Gaithersburg, Maryland, 2008.

[18] SourceForge.net, AfterGlow (afterglow.sourceforge.net).

[19] Y. Stamatiou, E. Skipenes, E. Henriksen, N. Stathiakis, A. Sikianakis, E. Charalambous, N. Antonakis, K. Stolen, F. den Braber, M. Sodal Lund, K. Papadaki and G. Valvis, The CORAS approach for model-based risk management applied to a telemedicine service, *Proceedings of the European Medical Informatics Conference*, pp. 206–211, 2003.

[20] K. Ziegler, NERC cyber security standards to become mandatory in United States, *Electric Energy Industry News*, January 21, 2008.