AUTOMATIC SECURITY ASSESSMENT OF CONTROL SYSTEMS FOR CRITICAL CYBER-INFRASTRUCTURES

BY

ZAHID ANWAR

B.S., Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, 2001
M.S., University of Illinois at Urbana-Champaign, 2005

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Professor Roy H. Campbell, Chair
Professor Klara Nahrstedt
Professor Carl A. Gunter
Assistant Professor Sam King

Abstract

Government and advisory agencies have issued various critical infrastructure protection standards and best-practice schemes for security-hardening of power and control networks in the wake of several security incidents threatening the reliability of the Power Grid. These security schemes differ in their installation costs and the degree of protection they provide against cyber attacks. Manually determining vulnerabilities and security risks in control networks and their maintenance procedures is a cumbersome process. Furthermore the process of mapping security schemes to vulnerable cyber assets each with a unique criticality in the power network configuration can easily become intractable.

This research attempts to bridge this gap by investigating the automation of security assessment of the static and dynamic properties of critical infrastructures. We describe first-order logic based models of the static elements including power and control devices, services and connectivity, and re-writing logic based models of the dynamic elements such as operating procedure workflows, and the state of a working power grid. We introduce a tool-chain that, with a little manual assistance, can automatically generate these models from specifications, continuously update attributes from online event aggregators, and perform security assessment. Aside from checking whether the system configuration conforms to recommended best-practices for establishing security controls, the assessment also reveals whether the observed anomalies about the system could indicate possible security problems and permits dynamic ranking of alternative recovery procedures to minimize the total risk. Moreover the tool-chain can recommend an optimal selection of security schemes to apply to various vulnerable parts of the Power Grid network to maximize security when faced with a budget constraint. A case study on security hardening the IEEE power system 118-bus test case from a pool of five different best-practice schemes is used to demonstrate the feasibility of the tool chain implementation.

To my parents.

Acknowledgments

This project could not have been possible without the support of my seniors, colleagues and friends. It is a pleasure to acknowledge all those who educated, supported and guided me in the process of my research. I was lucky to be associated with my adviser, Roy Campbell, who has supported me for the past six years. I am very grateful to him for his stimulating conversations, provocative ideas and practical help in the areas of learning. I hope that I could be as lively, enthusiastic, and energetic as Roy and to someday be able to command an audience as well as he can. Also thanks to my committee members, Klara Nahrstedt, Carl Gunter, and Sam King, for their scientific advice and knowledge and many insightful discussions and suggestions. I also have to thank the project leads of my technical research group, William H Sanders and Himanshu Khurana for their helpful career advice and suggestions in general. A good research team is important to surviving and staying sane during paper submission deadlines and preparing for conference presentations. I thank all the members of the Systems Research Group i.e. Ravinder Shankesi who was extremely knowledgeable, helpful, and friendly, Mirko Montanari, who has been helpful with answering Model Checking questions and paper editing, Alejandro Gutierrez, a nice and helpful person who has been supportive throughout the thesis writing, Ellick Chan for all his constructive feedback and Naeem Sheikh for bringing awareness to me and thrashing out complicated mathematical segments in this thesis.

Thanks to the University of Illinois, Department of Computer Science for awarding me a Teaching Assistantship in the early part of my PhD career, a Sohaib and Sara Abbasi Fellowship in the mid-part, providing me with the financial means to complete this project. It would not be just if I do not extend thanks to the Information Trust Institute (ITI) that provided funding for all my training, travel and research involved in the Trustworthy Cyber Infrastructure for the Power Grid (TCIP) Project.[1] And finally, thanks to my parents, who endured this long process with me, always offering support and love.

[1]The research presented in this thesis was funded by the National Science Foundation under grant TCIP NSF CNS 05-24695.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SCADA	Supervisory Control and Data Acquisition.
FERC	Federal Energy Regulatory Commission.
NERC	North American Power and Reliability Council.
NIST	National Institute of Standards and Technology.
ISA	Instrumentation, Systems and Automation Society.
DoS	Denial of Service Attacks.
CIP	Critical Infrastructure Protection.
VLAN	Virtual LAN.
FWALL	Firewall.
FLINK	Firewall plus Link Encryption.
CIM	Common Information Model.
YAWL	Yet Another Workflow Language.
XML	Extensible Markup Language.
PLC	Programmable Logic Controller.
RTU	Remote Terminal Unit.
EN	Enterprise Network.
PCN	Process Control Network.
NFS	Network File System.

1 Introduction

"Techniques to automate security assessment, evaluation with applications to cost-benefit analysis, conformance to best-practices of arbitrary Power Grid configurations and critical infrastructure protection of SCADA systems"

The Power Grid has been designed with the N-1 principle in mind, meaning that it is built to survive at least one failure. The redundant design is such that if a power asset such as a transmission line or generator were to fail the power would be routed from elsewhere without causing major blackouts. However, infrastructure that resists single points of random failure, may not survive malicious, intelligent attacks by disgruntled employees, terrorist networks, etc, especially if this redundancy is in the power network alone, without isolation in the control. Consider two redundant power lines designed to handle the extra load if one or the other goes down but essentially controlled by a common vulnerable relay.

Computerized control systems, also referred to as Supervisory Control and Data Acquisition (SCADA) systems, have become vital in the modern world. SCADA is deployed to control water supply, telecommunications as well as electricity generation and distribution. In this thesis we focus on SCADA systems for the electrical power grid. Relays are a popular choice for protection and control in power utilities. They communicate to constantly monitor the status of equipment, and participate in real-time pilot protection schemes [33] that involve detecting and agreeing on presence of faults, de-energizing equipment to protect against short circuits and reclosing circuits automatically in an attempt to clear faults. Relays are programmed to send alarms to operation personnel in case faults cannot be cleared automatically. Despite their important role, relay configurations are generally set up with convenience and efficiency in mind rather than security. Their open accessibility from the enterprise and office LANs and sometimes even the Internet gives the adversary easy opportunities for attacks. Furthermore, knowledgeable and inside attackers can use the properties of the power grid and its operating procedures to cause cascading failures, power blackouts or damage vital resources which are difficult to replace such as high-power transformers.

The security of SCADA systems made headlines [39] recently when the CIA reported incidents where hackers shut down power to entire cities by breaking into the systems of electricity companies. In a separate incident [37] researchers

at the Department of Energy's Idaho lab launched an experimental cyber attack (codenamed Aurora) on an electrical power plant causing a generator to self-destruct. While the attack details are not explained in depth, what is clear is that researchers were able to remotely hack into the SCADA network and change its configuration to cause significant damage to the generator. A comprehensive report compiled by the Industrial Security Incident Database (ISID) [2], shows an alarming increase in the numbers of security attacks on cyber infrastructures in recent years, with externally generated incidents accounting for 70% of all events between 2001 and 2003. The Slammer Worm infiltration of an Ohio nuclear plant [16] and the Australian sewage spill incident [58] in 2000 are two recent examples. In the latter case an attacker connected through a wireless network used to control sensors for a sewage treatment plant in Queensland, taking control of the main system to drain raw sewage into many of the parks and lakes.

The Federal Energy Regulatory Commission (FERC) recently approved eight cyber security and critical infrastructure protection standards proposed by NERC [15, 40] requiring bulk power system users, owners, and operators to identify and document cyber risks and vulnerabilities both for the cyber critical assets and their configurations as well as for operating procedures and incidents. While these standards represent a major break-through in what security controls *should* be enforced they are fairly flexible in *how* exactly they be implemented. For instance CIP 005 requirement 4 (R4) in the standards document states:

> "*The Responsible Entity shall perform a cyber vulnerability assessment of the electronic access points to the Electronic Security Perimeter(s) at least annually*"

Possible ways of securing access to the electronic security perimeter is through proper deployments of firewalls and access control protocols. Similarly CIP-009-R2 discusses security implications of operating and recovery procedures from disasters, their relative ordering, timeframe and requirements. While the NERC standards do not discuss implementation, other government security organizations such as National Institute of Standards and Technology (NIST) and the Instrumentation, Systems and Automation Society (ISA) have published a large amount of literature detailing guidelines for implementing best security practices for SCADA systems [31, 17, 21, 43, 52]. Most of these guidelines are informal English descriptions of secure SCADA infrastructure configurations, firewall rules, services that should be allowed and security protocols to employ.

Unfortunately incidents keep reoccurring [2] due to nonconformance to best practices resulting in loss of power, revenue and harm to consumers. These best practices can be implemented using security schemes that differ vastly in the kind of protection they provide. For instance firewalls may hinder DoS against control devices but will not prevent eavesdropping attacks. A link encrypter on the other hand may solve the latter attack but might prove ineffective against

DoS. Additionally cost and effort to implement the two schemes will differ; consider just upgrading the firmware on a router to provide firewall services as apposed to buying special hardware to provide 'bump in the wire' encryption for the real-time traffic demands of control devices. Similarly cyber assets for controlling power system resources differ vastly in terms of criticality. For instance it would be worthwhile investing in an expensive security lock down of a valuable asset like a 50,000 MW powerplant as compared to a substation that contributes less than 500 MW to the grid.

Perfect security is ideal but in reality security administrators are usually faced with budget constraints and end up trying to balance cost and security. This kind of manual cyber-security planning for a network the size of the Power Grid can easily become intractable and is actually an NP hard problem.

The contribution of this research is the observation that formalizing these best practices would allow automated conformance checking of arbitrary SCADA network configurations to enable FERC standards compliance. Moreover it recognizes the importance of spending more capital (more powerful security controls) on securing assets that are more critical. Security schemes are presented that use best practice guidelines from NIST [52, 51] and other advisory agencies e.g. firewall, VLAN segregation, link encryption to isolate redundant power network assets in the control networks. Metrics have been proposed to evaluate the protection provided by security schemes, the cost to implement them, and determine the criticality of equipment in terms of revenue loss incurred in the event of their compromise. A pseudo-polynomial time automated solution is proposed that uses these metrics together to determine the optimal scheme selection to maximize security given a fixed budget allocated for Power Grid security hardening.

In our work [8, 6], it is shown that logic-based models of the power grid, and its control elements can be used for automatic conformance checking for adherence to best security practice schemes. Throughout this thesis we model power and SCADA networks in predicate-logic (henceforth called a network model) which consists of a set of devices, services, network connections, known vulnerabilities as well as their attributes. The network model contains complete information regarding the network dependency graph both in terms of physical connections such as links as well as logical connections such as service dependencies. This information is automatically obtained from SCADA specification languages such as the Common Information Model (CIM). Best practices modeled as a set of rules over the facts determine if the network meets certain security constraints.

In addition, the network model is used to generate attack graphs for various SCADA devices to determine vulnerability to external attackers. While evaluating these attack graphs, we compute the security risk for each device depending on both the severity of the isolated vulnerabilities of various nodes, on the path to that device, as well as the topology of the paths. Our device security risk

is formalized as a lattice whose partial order function depends on the type of vulnerability and the calculated severity of the vulnerability (i.e. an execution control vulnerability is rated higher than a denial of service vulnerability and the severity of a vulnerability may be valued higher or lower depending on its exploitability). The device security risk calculated in the network model is used as input for our second model explained below. The second model, called the workflow model, describes the various operating procedures, as workflows encoded in rewriting logic. Operational procedures are usually recovery or maintenance activities that the operators follow in the system, for instance, to recover from a failed component or deal with some contingency. These recovery procedures are made up of an ordered set of tasks that enable or disable SCADA network devices (for instance, a task to allow selection of a backup transformer from a list of idle transformers in a substation). Tasks can be fulfilled in various ways, allowing operators a choice of strategies to perform a particular function. The security risk of a particular recovery procedure is calculated by aggregating the risks derived from all the vulnerabilities of each device used in the recovery procedure as obtained from the network model. Finally, the security model presents its evaluation of the possible recovery procedure options to an operator along with their security risks.

We have developed a tool-chain that semi-automates the generation of the network and workflow models as well as keeps the attributes up-to-date using on-line event aggregators. We demonstrate its feasibility to find complicated attacks on arbitrary network configurations. Besides conformance checking, the tool-chain can use additional attributes from the power network itself such as *overloading violations*, *power flows* and *costs* to automatically evaluate for SCADA network configurations, which combination of security schemes would best protect against a knowledgeable adversary who attempts to maximize the damage inflicted by attacking SCADA that controls the most critical power equipment[7]. An extensive case study evaluates the feasibility of our approach and demonstrates the tool-chain functionality by security hardening the IEEE 118-bus test case from a pool of five different best-practice schemes.

The remainder of this thesis is organized as follows: Chapter 2 puts this work into prospective by describing background and related work on security assessment techniques for the electric power grids and computer networks. Chapter 3 outlines the design of our security model and Chapter 4 describes the architecture and implementation of our tool chain that allows automated construction of these models. Chapter 5 walks the reader through the tool-chain's model checking process using representative scenarios. Chapter 6 introduces metrics for quantifying security and uses them as a basis for a cost-benefit analysis of Power Grid configurations. Chapter 7 describes the details of an evaluation case study of the cost-benefit analysis on the 118-bus test case to demonstrate the tool-chain functionality. The thesis is concluded with a short discussion and possible future work in Chapter 8.

2 Background and Related Work

While assessment and evaluation of the cyber-security properties of SCADA networks for critical infrastructures is a relatively new and emergent area; quantitative analysis of network security in general and of electric power networks has been separately looked at extensively by their respective research communities. This chapter gives a brief introduction to the power system terminology and the logic model checking techniques used in the thesis to the new reader. It also discusses complementary related power and cyber security analysis research.

2.1 Background

This paper does not assume that the reader has a power engineering background and explains all the key power terms used throughout the paper in this section. Horn-clause logic and rewriting logic have been used for the formal analysis and while we assume the reader is reasonably familiar with the former we give a brief explanation of the latter here as well.

2.1.1 A Power System

A conventional electric power system has three main components - generation, transformation and distribution [32]. Typically, energy from fossil fuels or falling water is harnessed to generate steam to drive power turbines that produce electricity, which is then transmitted and distributed to the end user. There are a variety of SCADA controls used throughout the process e.g. turbine control, burner control, and substation local control. This paper focuses on automated security assessment of the SCADA network and switching devices (e.g. relays and circuit and protection breakers) used in the production and distribution of electric power but the security model we use is not restricted to just this particular aspect of SCADA systems and can be easily applied to other supervisor power control systems as well.

A power grid has a large number of switches that affect the way power is routed and distributed within various components system. These switches are often controlled remotely through SCADA (but can also be turned on or off manually). Changing the status of switching devices in a substation allows some interesting attack scenarios from an intruder's point of view. A denial of service attack on the controlling relay would lead to a failure to report the proper state in time (and might require manual intervention). Even more seriously, a

buffer-overflow in a networked device (allowing execution privileges) can allow an attacker to black-out a feeder or overload a transformer. The latter is a very serious attack as transformers are expensive and hard to replace.

A small example of a power-grid system is illustrated in Figure 2.1. A power distribution system can be viewed as a network of electric lines connected via switching devices (represented by small circles) and fed via circuit-breakers (represented by large squares). Switching devices and circuit-breakers are connected to, at most, two lines. They have two possible positions: Open or closed. A circuit-breaker supplies power if and only if it is closed, and a switching device stops the power propagation if and only if it is open. Consumers may be located on any line and are supplied only when their line is supplied. Distribution networks have a meshable structure exploited radially: The positions of the devices are set so that the paths taken by the power of each circuit-breaker form a tree called a feeder. The root of a feeder is a circuit-breaker, and its leaves are whatever switching devices downstream happen to be open at the time.

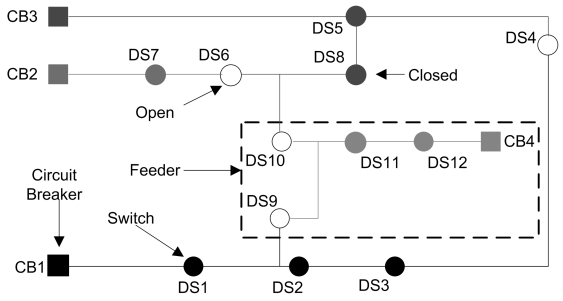

Figure 2.1: Power Distribution System

Power distribution systems are often subject to permanent faults (short circuits) occurring on one or even several lines. Since these short circuits are mainly due to bad weather conditions and lightning, multiple faults are not rare. Upon occurrence of a fault, the circuit-breaker feeding the faulty line opens in order to protect the rest of its feeder from damaging overloads. For instance, if a fault occurs on the line between DS1 and DS2, CB1 will open. As a result, all consumers located on that feeder are left without power. Simply reclosing the circuit-breaker will not help. Since the fault is permanent, the circuit-breaker will still be feeding it and will open again. Instead, using SCADA devices controlling the switches, the faulty lines must be located and the network reconfigured so as to isolate them and restore the supply to the non-faulty lines. Switches are controlled by remote-controlled actuators that sense and change their position and report sensing the presence of faults.

2.1.2 Rewriting Logic

We specify the operating procedures as workflows in rewriting logic. In general, a concurrent system can be specified in rewriting logic as the theory $\mathcal{R} = (\Sigma, E, R)$

where (Σ, E) is the *order-sorted equational theory* [24] such that:

- The signature Σ specifies the sorts, a sub-sort relation, constants and function symbols. The terms T_Σ and $T_\Sigma(X)$ denote, respectively, the terms the set of ground Σ-terms and the set of Σ-terms over variables in X.

- The equations in E are of the form

$$(\forall X)\ u = \ v \ \textbf{if}\ C$$

  where u, v are of the same sort and the (possibly null) condition C is a conjunction of unquantified Σ-equations involving variables (only) in X. We say the Σ-algebra A satisfies the equation $(\forall X)\ u = v$ **if** C iff for each assignment $a : X \to A$, $a(u) = a(v)$

Intuitively the theory (Σ, E) defines the states of the system and has the initial model $T_{\Sigma|E}$, The elements in $T_{\Sigma|E}$ are E-equivalence classes of terms.

The *dynamics* are described by the *rewrite rules* R that specify concurrent transitions that can occur in the system and that can be applied *modulo* the equations in E.

In general rewrite rules are of the form

$$(\forall X)\ u \to v \quad \textbf{if}\ \bigwedge_i u_i = v_i$$

and describes a transition from the term t to term t'. To apply the rewrite rule to the term t, we find a subterm of t which is an instance of u under some substitution σ. We substitute u in t by v, only if all the conditions hold i.e.., $\forall i : \sigma(u_i) = \sigma(v_i)$. Note that multiple rewrite rules may be applicable to a given term.

A concurrent system can be described using rewriting logic [36]. Given a rewrite-theory (Σ, E, R) we can define the transition relation $\to$ over the states (given by the terms in the algebra $T_{\Sigma|E}$) by using the one-step rewrite rule in R. We can label the transition system given by $(T_{\Sigma|E}, \to)$ by using predicates defined using equations P[1] that associate a term in (Σ, E) to a proposition. Therefore, given a rewrite theory, (and appropriate labeling) we can define the Kripke-structure that describes the transition system. The extension from a rewriting logic to the Kripke structures on which the LTL model checking works is described in greater detail in [13]. Therefore, given a system described using term-rewriting logic, we can we can verify if it satisfies a given LTL property by using LTL model checking [12].

Rewrite theories are *executable* (under reasonable assumptions over E, R) and there are several rewriting logic implementations such as Maude [14], CafeOBJ

[1] We require that $(\Sigma, E \cup P)$ be a *protecting* extension of the theory (Σ, E)

[23], ELAN [10]. In this work, we use Maude to implement our *workflow model*. Maude supports LTL model checking by using a on-the-fly model checker [20].

For instance consider the following example, taken from the Maude book [13], of a system describing the classic crossing the river problem [2]

```
1  mod RIVER-CROSSING-2 is
2      sorts Side Group .
3      ops left right : -> Side [ctor] .
4      op change : Side -> Side .
5      ---- shepherd, wolf, goat, cabbage
6      ops s w g c : Side -> Group [ctor] .
7      op _+_ : Group Group -> Group [ctor assoc comm] .
8      op initial : -> Group .
9      vars S S' : Side .
10     eq change(left) = right .
11     eq change(right) = left .
12     eq initial = s(left) w(left) g(left) c(left) .
13     rl [shepherd-alone] : s(S) => s(change(S)) .
14     rl [wolf] : s(S) w(S) => s(change(S)) w(change(S)) .
15     rl [goat] : s(S) g(S) => s(change(S)) g(change(S)) .
16     rl [cabbage] : s(S) c(S) => s(change(S)) c(change(S))
17 endm
```

.

In the example given above, the signature Σ is given by the sorts Side,Group as well as the unary operators change,s,w,g,c; the binary operator + (given in line 7) and the constants left,right (of sort Side) and initial (of sort Group). The equations E of the rewriting theory are given by the three equations in lines 10-12. Essentially, the equations in E give the equivalence classes for the terms in Σ. Therefore the terms s(change(change(left))) w(right), s(right) w(change(left)) and s(left)w(right) are equivalent and describe the state of the system when the shepherd is on the left of the bank and the wolf is on the right.

The real state-changing transitions of the system are given by the rewrite rules R (given in lines 13-16. For instance, the rule in line 14, labelled 'wolf' describes a transition where the system transitions from a state where the shepherd and the wolf are on the same side of the bank and their state changes to the other bank in one atomic transition.

Note that the transitions for a given state can be non-deterministic as multiple rules might be applicable to a given state. For instance, given a state consisting of a shepherd,wolf,goat,cabbage on the same side (s(S)w(S)g(S)c(S)) [3], we can apply any of the four rules on that given state. Also note that transitions are modulo equations in E, therefore you can apply the first rule 'shepherd-alone' to the term s(change(change(S))) to yield s(change(S)).

[2]The problem describes a system where a shepherd, a goat, a wolf and a cabbage need to cross a river, such that the shepherd can take at most one of the items along with him when he crosses the river and he cannot leave a goat alone with a wolf or a goat with a cabbage.

[3]note that we use the variable S for every operator here, signifying that its value (left or right) must be the same for every operator

2.2 Related Work

Our research benefits from related work on surveys of SCADA system security, security assessment techniques such as attack trees for computer networks in general and in particular techniques that target critical infrastructure protection.

2.2.1 Surveys of SCADA Security

The Power Grid's vulnerability to physical disruptions from natural disasters and other causes has long been recognized [45]. This vulnerability has increased in recent years because infrastructure has not expanded as quickly as demand has, thereby reducing the systems 'cushion' against failed, destroyed, or otherwise unavailable system components. A survey of real SCADA systems and the security controls in place by security analysts [34] reveals a startling truth about the lack of authentication mechanisms, little or no software patch updates, and numerous uncontrolled interconnects to the public Internet. The authors were able to break into oil and power production systems by using simple exploits such as SQL Injection. The paper argues that even with knowledge of individual vulnerabilities in the nodes of the system, there are currently no adequate tools for reasoning about the overall security of the system. Our work can be seen as a way of reasoning about such systems.

Mark Grimes [25] presents a detailed survey of vulnerabilities in popular protocols used in SCADA systems such as MODBUS, DNP3, UCA. The author walks the reader through step by step instructions on exploiting each of the protocol vulnerabilities via packet fuzzing, DoS and spoofing attacks demonstrating the easy with which SCADA control devices could be compromised if an attacker were to gain network access.

Risley et al [50] survey a set of security tools suitable for the stringent communication demands of SCADA networks as well as meet the CIP security standards set by NERC and other government agencies. Merits of devices such as crypto modems, secure communication processors and firewall solutions are discussed along with their installation costs. While it is clear from this work that multiple competing tools and schemes exist for protecting control networks, how to map them to individual configurations is unclear. Certainly applying these controls in the entire network would be too cost prohibitive to be feasible.

2.2.2 Security Models for General Computer Networks

Guttmann et al, developed a technique for rigorous automated network security management [27] that builds a formal model of the various devices present in computer networks, reasons about their security and generates the necessary configuration files for the devices in order to deploy the required security strategies. They use the SNMP [3] data model for computer networks and try to

deduce the security of the network.

Attack graphs are a well known technique [47, 46, 57] that represents a chain of exploits as a path, where each exploit in the chain lays the groundwork for subsequent exploits. The pioneering work [57] in this area used model checkers to identify explicit attack sequences. However, this approach suffers from scalability issues because the number of such sequences grows exponentially with the product of the number of vulnerabilities and devices. Later work [46] proposes a new logic based approach where each node in the graph is a logical statement and edges are causality relations between network configurations and attacker privileges. This results in the attack graph size being polynomial in the size of the network.

Dewri et al. [18] use attack trees to model networks and employ evolutionary algorithms to solve the optimization problem of what subset of security measures to use so that the cost of implementing these measures and the cost of residual damage is minimized.

Jajodia and Noel have used automated attack graph generation and processing techniques using vulnerability scanning tools like Nessus [42] for aiding sensor placement for monitoring attack paths to critical network nodes [41].

2.2.3 Security Models for Critical Cyber-Infrastructures

One work on **SCADA attack trees** [11] describes the application of attack trees to the common MODBUS SCADA protocol with the goal of identifying security vulnerabilities inherent in the specification and in typical deployments. The paper shows, using domain relevant examples, how attack trees are a technique to achieve their goal. The scope of this study is, however, limited to the examination of one particular SCADA protocol and the authors have not described any implementation of their idea. We extend attack trees to evaluate the security properties of entire SCADA infrastructures and their operating procedures with a working implementation.

An interesting use of SCADA attack trees is described by **C. W. Ten et al.** [61] where the authors evaluate security improvements based on countermeasures types and password policy enforcement on tree leaves. The structure of their attack tree is however different from ours in that it includes defense nodes with countermeasures and an optimization problem is formulated to determine pivotal leaves in the tree for security improvement. Their tree structure is however limited in that it does not capture the sequence in which attack leaves are penetrated and the paper does not offer a way to build the tree structure in the first place.

The most notable efforts in modeling and reasoning about security of critical cyber-infrastructures has been carried out by the **SINTEF** [1] group of the European Union. Their project, **CORAS**, [9] supports methodologies for risk analysis of security-critical systems by modeling threats to a system as unwanted

features of the system in question. This allows users to model a system and its associated threats as UML diagrams and XML schemas for exchange of risk assessment data in a more formal and standardized language. We improve upon their idea of using UML by employing the standard descriptive language based on Common Information Models (CIM) [30] to automate the generation of our security models. CIM, an object-oriented cyber-infrastructure modeling language developed by the Electric Power Research Institute (EPRI) is better suited for modeling electrical utility enterprises.

Salmeron et al. [53] use bilevel mathematical models to determine the most critical components in a power grid network i.e. those if taken down will cause the most disruption in the network. Their model however does not include the use of any security schemes to protect against attacks.

Researchers have also proposed various methodologies such as compromise graphs [35] and Markov Chains [44] for obtaining a quantitative measurement of the risk reduction achieved when a control system is modified with the intent to improve cyber security defense. No discussion has made regarding generation of these models or of how to mitigate the risk once determined.

The Control System Cyber Security Self-Assessment Tool **(CS2SAT)** [56] is an online, detailed questionnaire consisting of hundreds of questions about the components used by a power utility. The answers are compared to a database of regulations and best-practises and a non-numerical report is generated identifing if the control system meets the requirement or is deficient. It was developed by Idaho National Labs (INL) as part of the DHS Control System Security Program (CSSP).

While it is clear that security assessment of networked systems and the use of attack graphs models to find network vulnerabilities is a fairly mature area, we did not find much work in the automated generation of these models especially for the cyber-infrastructures domain. Moreover there has been little work if any in using the security vulnerabilities calculated for network elements as inputs to finding risks in operating procedures and providing advisories. Similarly theoretical work exists on identifying security controls that would mitigate vulnerabilities discovered by attack trees but again we did not find any research on applications of this work to an actual cost-benefit analysis for a real system such as the Power Grid.

3 Logic- Based Models of the Power Grid

Our formal model is composed of two parts. A *network model* captures the static parts of the Power Grid- comprising the power and control devices, network topology, services, connectivity and vulnerabilities (known software exploits) described in first-order predicate logic. A *workflow model* captures the dynamic parts such as maintenance, recovery activities involved and their ordering and relationships in re-writing logic.

3.1 Network Model, $\mathcal{N}$

A power system is an electric network consisting of a set of power devices
$D_p = \{B \cup E \cup F \cup G \cup L\}$ where

B buses;

T transmission lines where $T \subset B \times B$;

F transformers where $F \subset B \times B$, $\{B \times B\} \setminus \{T \cup F\} = \emptyset$ and $\{T \cap F\} = \emptyset$;

G generators;

L loads;

and substations S where
$S = \{S_i | S_i \subseteq B\}$, $\bigcup_{\forall i} S_i = B$ and $\forall_{\substack{i,j \\ i \neq j}} S_i \cap S_j = \emptyset$;
the relation:

$ConnectedTo\ zCb$ where $z \in T, F, G, L$ and $b \in B$;

and a set of functions:

$$linesin : B \rightarrow \mathbb{P}\{T\} \text{ bus to lines mapping;} \tag{3.1}$$

$$transin : B \rightarrow \mathbb{P}\{F\} \text{ bus to transformers mapping;} \tag{3.2}$$

$$gensin : B \rightarrow \mathbb{P}\{G\} \text{ bus to generators mapping} b_i \in B \text{ and} \tag{3.3}$$
$$\forall_{b_1, b_2} b_1 \neq b_2 \, , gensin(b_1) \cap gensin(b_2) = \emptyset;$$

$$ldsin : B \rightarrow \mathbb{P}\{L\} \text{ bus to loads mapping and} \tag{3.4}$$
$$\forall_{b_1, b_2} b_1 \neq b_2 \, , ldsin(b_1) \cap ldsin(b_2) = \emptyset;$$

$$power : D_p \rightarrow P \text{ device to power mapping where } P \in \mathbb{R}_{\geq 0}; \tag{3.5}$$

A power system is typically depicted as a graph in one-line diagrams where nodes are buses and edges are branches. Branches can be either of type transmission lines or transformers, through which electrical energy is transmitted to

supply customers. Devices in a power network conduct power (lines, buses), generate (generators) or consume it (loads) as depicted in function 3.5.

A set of buses are functionally grouped together to form substations. We distinguish between three types of substations as shown by the predicates 3.6-3.9. (1) A **power plant** characterized by one or more generators connected to atleast one of the buses. (2) A **distribution substation** has no generators and is characterized by one or more loads connected to one of the buses. (3) A **transmission substation** has no generators and loads, connects two or more transmission lines and may have transformers to convert between two transmission voltages.

$$powerplant(s_i) = s_i \in S \wedge \exists b_i[b_i \in s_i \wedge gensin(b_i) \neq \emptyset]; \tag{3.6}$$

$$dist\_substation(s_i) = s_i \in S \wedge \forall b_i[b_i \in s_i \wedge gensin(b_i) = \emptyset] \wedge \exists b_j[b_j \in s_i \wedge ldsin(b_j) \neq \emptyset]; \tag{3.7}$$

$$trans\_substation(s_i) = s_i \in S \wedge \forall b_i[b_i \in s_i \wedge gensin(b_i) \cup ldsin(b_i) = \emptyset \wedge \exists t_i, \exists t_j[t_i, t_j \in linesin(b_i) \wedge t_i \neq t_j]]; \tag{3.8}$$

$$substation(s_i) = s_i \in S \wedge (trans\_substation(s_i) \vee dist\_substation(s_i)) \tag{3.9}$$

For instance the left part of Figure 3.1 shows an example of a simple power system consisting of 4 substations. The buses represent powerplants (Bus 1 and 2), transmission (Bus 4 and Bus 5), and distribution substations (Bus 3).

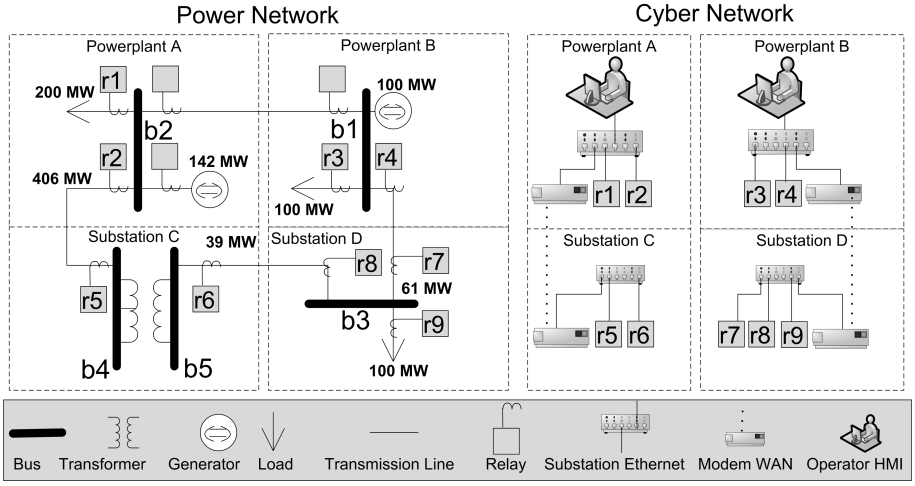

Figure 3.1: A Simple Power and Corresponding Control Network

The network model $\mathcal{N}$ represents the SCADA network as a dependency graph G, a set of best practice rules and a set of logical attack graphs G'.

3.1.1 Dependency Graph G

The dependency graph is given as $G = (D_c, E, S, V, S_{TYPES}, D_{TYPES}, S_p)$ where

D_c	control devices;
E	edges between two physically connected devices where $E \subseteq D_c \times D_c$;
S	services;
V	vulnerabilities;
S_p	security protocols;
K	pre-shared keys;
S_{TYPES}	set of all service types;
D_{TYPES}	set of all device types;

and the following set of functions that give attribute mappings to the various devices and dependencies:

$$devof : S \rightarrow D_c \text{ hosted service to device;} \tag{3.10}$$

$$hostedsvs : D_c \rightarrow \mathbb{P}\{S\} \text{ device to hosted services where } devof(S) = D_c; \tag{3.11}$$

$$defsvs : D_c \rightarrow S \text{ device to the default service;}^{1} \tag{3.12}$$

$$depdtsvs : S \rightarrow \mathbb{P}\{S\} \text{ service to services dependent upon it;} \tag{3.13}$$

$$trusteddevs : D_c \rightarrow \mathbb{P}\{D_c\} \text{ device to its trusted devices;} \tag{3.14}$$

$$secprots : S \rightarrow \mathbb{P}\{S_p\} \text{ service to its installed security protocols;} \tag{3.15}$$

$$keys : D_c \rightarrow \mathbb{P}\{K\} \text{ device to its set of pre-shared keys;} \tag{3.16}$$

$$typeofsvs : S \rightarrow stype \text{ service to its type where } stype \in S_{TYPES}; \tag{3.17}$$

$$typeofdvs : D_c \rightarrow dtype \text{ device to its type where } dtype \in D_{TYPES}; \tag{3.18}$$

$$knownvuls : stype \rightarrow \mathbb{P}\{V\} \text{ service to its known vulnerabilities;} \tag{3.19}$$

$$priv : S \rightarrow privlvl \text{ maps a service to its privilege Level where } privlvl \in \{none \leq user \leq root\}; \tag{3.20}$$

$$exploitability : V \rightarrow likelihood \text{ vulnerability to its exploit likelihood where } likelihood \in \mathbb{R} \text{ (s.t } 0 \leq n \leq 1); \tag{3.21}$$

and the following set of functions that give attribute mappings to the various devices and dependencies.

3.1.2 Best Security Practices

We define best practices and standards by advisory bodies as rules whose terms are a set of constraints on G.

[1] For example, the default service could be the Operating System.

Services between the PCN and the Internet should be strictly allowed on a need basis

IAONA's [29] template for protocols access in industrial environments states:

Security Concern: Incomming DNS, HTTP, FTP, TELNET, SMTP traffic to the PCN should be discouraged unless absolutely required.

$$\begin{aligned} needbasis(G) =& \forall d_1, \forall d_2 \in D_c\ [EN \in typeof(d_1) \wedge PCN \in typeof(d_2) \wedge \\ & \forall s_1 \forall s_2 \in S\ [s_1 \in hostedsvs(d_1) \wedge s_2 \in hostedsvs(d_2) \wedge \\ & depdtsvs(s_1, s_2) \Rightarrow typeofsvs(s_2) \notin \{dns, http, ftp, telnet, smtp\}]]; \end{aligned} \tag{3.22}$$

Explanation: This rule checks to see whether there exist two devices belonging to the enterprise and PCN networks in a substation such that there is service dependency between the former to the latter. If there exists such a dependency then it should not be of the type dns, ftp, telnet or smtp.

Correct Implementation of Access Control

American Gas Association (AGA)'s report 12 on Cryptographic protection of SCADA Communication [5] states:

Security Concern: In a proper access control implementation, a service should provide an authentication scheme and also be capable of using communication protocols which guarantee confidentiality and integrity.

$$\begin{aligned} correct\_ac(G) =& \forall d_{alice}, \forall d_i, \forall d_j, \forall d_{bob} \in D_c\ [depends(d_{alice}, d_{bob}) \wedge \\ & d_i \in path(d_{alice}, d_{bob}) \ \wedge d_j \in path(d_{alice}, d_{bob}) \ \wedge (d_i, d_j) \in E \\ & \Rightarrow (secprots(defsvs(d_i)) \cap secprots(defsvs(d_j))) \neq \emptyset \wedge \\ & (keys(d_{alice}) \cap keys(d_{bob})) \neq \emptyset]; \end{aligned} \tag{3.23}$$

where we define the auxiliary functions:

$$path(d_1, d_k) = \{d_1, d_2, ...d_{k-1}, d_k \in D_c | \forall_{1 \leq i < k-1} (d_i, d_{i+1} \in E)\}; \tag{3.24}$$

$$\begin{aligned} depends(d_1, d_2) =& \forall d_1, \forall d_2 \in D_c, \exists s_1, \exists s_2 \in S[s_1 \in hostedsvs(d_1) \wedge \\ & s_2 \in hostedsvs(d_2) \wedge depdtsvs(s_1, s_2)]; \end{aligned} \tag{3.25}$$

where path (predicate 3.24) is a mapping from a pair of devices to all the set of paths between them.

Explanation: This predicate checks if a given service S_i implements access control, confidentiality and integrity in the correct manner. It does this by

checking whether all its dependent services have a common shared-key mechanism in place. A helper function *path* is used to check that the default service on each pair of devices along the path from the queried service to the dependent services share common security properties. The helper function *keys* checks if the dependent services share a pre-shared key.

Correct Firewall Deployment for SCADA and Process Control Networks

NISCC [43] provides good practice guidelines on Firewall Deployment in SCADA:

Security Concern: Traffic from the office LAN should be separated from the industrial-control LAN by a firewall. Firewall architectures with lowest security rating to highest security rating can be broken down into five general classifications summarized in Table 3.1.

Table 3.1: 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.

$$
\begin{aligned}
dualhomedfirewalled(G) = & \forall d_e, d_p, d_s \in D_c \; [EN \in typeofdvs(d_e) \wedge \\
& PCN \in typeofdvs(d_p) \wedge depends(d_e, d_s) \wedge \\
& depends(d_p, d_s) \wedge \; (d_s \in path(d_e, d_p)) \wedge \\
& \exists s_f \in S[s_f = hostedsvs(d_s) \wedge \\
& firewall = typeofsvs(s_f)]];
\end{aligned} \tag{3.26}
$$

$$
\begin{aligned}
dmz(G) = & \forall d_e, d_p, d_s \in D_c \; [EN \in typeofdvs(d_e) \wedge PCN \in typeofdvs(d_p) \wedge \\
& depends(d_e, d_s) \wedge depends(d_p, d_s) \wedge \exists d_{f1}, d_{f2}, d_{f3} \in D_c \\
& [(d_{f1} \in path(d_e, d_s)) \wedge \; (d_{f2} \in path(d_p, d_s)) \wedge (d_{f3} \in path(d_e, d_p)) \wedge \\
& firewall \in typeofdvs(d_{f1}) \wedge \; firewall \in typeofdvs(d_{f2}) \wedge \\
& firewall \in typeofdvs(d_{f3})]];
\end{aligned}
\tag{3.27}
$$

Explanation: This predicate identifies the shared network devices, for instance the data historians, aggregators and access points. Servers that are accessed by dependent services running on devices in both the enterprise and PCN are characterized as shared. Proper placement of these devices with respect to firewalls determines the firewall architecture in use. For instance the first predicate states that if all the paths between the two dependent PCN and enterprise devices pass through the shared device and that the device is running a personal firewall service then the architecture belongs to the category of 'dual-homed with personal firewall' and has a security level of 2. Similarly the second predicate checks to see that all possible paths between the PCN d_p and enterprise device d_e, d_p and the shared device d_s, and finally d_e and d_s pass through firewalls then the shared device is properly placed inside a isolated dmz and can be assigned the security level of 4. Rules for the other four firewall architectures are not shown here but are constructed along the same lines.

3.1.3 Attack Graphs G'

The security risk of a device is dependent upon an attacker's ability to exploit a vulnerability V on that device or on a device from which it is reachable. Attack graphs are a well known technique [47, 46, 57] that represents a chain of exploits as a *path*, where each exploit in the chain lays the groundwork for subsequent exploits. The pioneering work [57] in this area used model checkers to identify explicit attack sequences. However, this approach suffers from scalability issues because the number of such sequences grows exponentially with the product of the number of vulnerabilities and devices. Later work [46] proposes a new logic based approach where each node in the graph is a logical statement and edges are causality relations between network configurations and attacker privileges. This results in the attack graph size being polynomial in the size of the network. We reuse their technique for generating our attack graphs.

We use two algorithms *ConstructAttackGraph* and *EvaluateAttackRisk* that are executed sequentially for each device whose security risk is being evaluated. The first algorithm constructs a graph depicting all possible ways an attacker could effect a device's safety (e.g. compromise of availability, integrity). The *EvaluateAttackRisk* algorithm then uses this output graph as input and calcu-

lates the overall security risk for the device.

ConstructAttackGraph essentially records a successful Prolog derivation (an implementation of the backward chaining algorithm in horn clause logic) as an attack graph G'. Similar to [46] the logical attack graph G' is a tuple $(N_r, N_p, N_d, E, \tau, \gamma)$ where N_r, N_p and N_d are three sets of disjoint nodes in the graph, $E' \subset (N_r \times (N_p \cup N_d)) \cup (N_d \times N_r))$, τ is a mapping from a node to its label, and $\gamma \in N_d$ is the attacker goal needed to perform the exploit r_l on device D_c [2]. N_r, N_p and N_d are the sets of rule nodes, primitive fact nodes and derived fact nodes respectively. Primitive fact nodes N_p were described earlier in the construction of G. N_r represent predicate rules that describe conditions on N_p to form derived nodes N_d. The root node of the attack graph is the goal we are trying to satisfy e.g. 'is a device vulnerable to DoS' while the primitive facts such as knowledge about exploits form the leaves.

To find the accumulative security risk D_M associated with the root device node we use the following heuristics:

- H1: Security Risk decreases if the 'length of the paths' leading to the victim increases following the analogy that the difficulty accumulated in reaching a target is proportional to the number of locks to be opened.
- H2: Security Risk increases if the 'number of paths' leading to the target is large. The attacker can use the different paths simultaneously to break different locks on each path.

We can translate these heuristics into a graph traversal algorithm (see algorithm 1) that evaluates the security risk. This algorithm is essentially a variant of the recursive depth-first search algorithm over a directed acyclic graph. Intermediate nodes evaluate their security risk by recursively calling *EvaluateAttackRisk* on their children and returning a product of their own risk and that of their children. Individual exploit likelihoods are available from vulnerability databases such as [54, 60]). If there is more than one path to exploit a node then the total node probability is calculated from the respective exploit probabilities of all such paths. For instance, for a device with a vulnerable service with expoitability p_d and i possible paths reaching it each with exploitability p_i, then the total exploit probability is $1 - \pi_i(1 - p_d \times p_i)$. Since probabilities are less than one, multiplication as we go up the tree ensures that longer paths will decrease attack risk.

The algorithms *ConstructAttackGraph*, *EvaluateAttackRisk* together give, for each vulnerable device D_c, the tuple (r_l, D_M), where $r_l \in R_L$ and $(R_L, \leq)$ is the security risk lattice which characterizes the kind of vulnerability that device D_c has, and $D_M \in \mathbb{R}$ s.t($0 \leq D_M \leq 1$) is the severity of the vulnerability. We use D_M to assign a severity label to the original risk as follows $Sev : \mathbb{R} \rightarrow S_L$ where $(S_L, <)$ is the set of labels with a total order[3].

[2] for e.g., *codeExecute*,*DoS* to exploit integrity,availability resp.

[3] For instance, $S_L = \{High, Medium, Low\}$ such that $(Low < Medium), (Medium < High)$

Algorithm 1 Evaluates the risk associated with an input attack graph

```
/*initialize Risk=1 for all nodes*/
double procedure EvaluateAttackRisk(V)
  if Visited(V) then
    return Risk(V)
  end if
  markAsVisited(V)
  /*A rule node's self risk is 1-P(no service is exploited)*/
  if isRuleNode(V) then
    for EACH I ∈ AdjacentPrimitiveNodesSet(V) do
      Risk(V) ← Risk(V) × (1 − Exploitability(I))
    end for
    Risk(V) ← (1 − Risk(V))
  end if
  /*If a Leaf node then just return your self risk*/
  if isLeafNode(V) then
    return Risk(V)
  end if
  /*If an intermediate node then recursively evaluate risks of children*/
  childRsk = 1
  for EACH I ∈ ChildSet(V) do
    childRsk ← (1 − EvaluateAttackRisk(I) × Risk(V)) × childRsk
  end for
  Risk(V) ← (1 − childRsk)
  return Risk(V)
```

Given the risk lattice R_L and the severity label S_L, we define the new extended risk-lattice R_E as follows $R_E \subset \{(R_L \times S_L)\}$. The new security lattice is defined by the partial order operator $\leq_E$ such that (given $R, R' \in R_L$ and $S, S' \in S_L$ and $\forall R \in R_L : min(R_L) \leq R$):

$(R, S) = (R, S')$ if $(R = \min(R_L))$

$(R, S) \leq_E (R', S)$ if $(R \leq R')$

$(R, S) \leq_E (R, S')$ if $(S < S')(and)(R \neq \min(R_L))$

$(R, S) \leq_E (R', S')$ if $(R \leq R')$ and $(S < S')$

The new security-risk for each vulnerability is now given by $(r \times Sev(D_M)) \in R_E$. For instance, the final risk-lattice for a system with $\{NoRisk, Availability, Integrity\}$ as the vulnerabilities and $\{Low, High\}$ as the severity can be depicted in Figure 3.2.

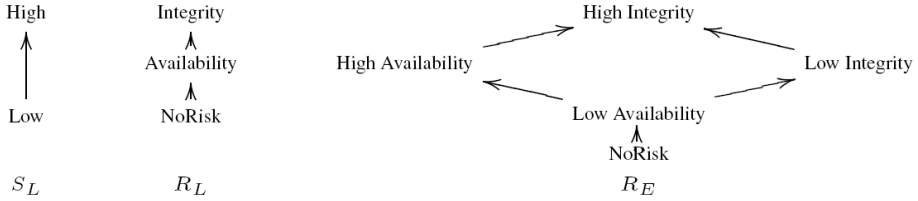

Figure 3.2: Workflows Security Lattice

3.2 Workflow Model $\mathcal{W}$

We formalize the notion of recovery and operating procedures in the form of workflows. There are many different workflow description languages and our model of workflows is a subset of YAWL's basic control flow patterns. One distinction is that we add the notion of "actions" to be performed by a task. Here we have a list of actions that can be performed, and each of which has a security risk. The next task that can be fired as well as the actions that are chosen at any given task can be *context-sensitive*. The *context* consists of the list of (task,action) pairs performed in the workflow so far [4]. At any given task, the workflow execution makes a non-deterministic choice between the list of actions available at that task. Although this can be modelled in terms of basic workflow primitives, our method of allowing non-deterministic choice at a task allows us to use a "generic" workflow description for various different power-grid environments. The individual "environment-specific" choices that can occur at any given task can be modeled as actions possible at that task.

Formally, we define the workflow as the tuple, $\mathcal{W} = (T, C, F, A_{id}, RL, \leq, A, R, S_t, J_t)$, where each element in the tuple is defined in Table 3.2

Table 3.2: Workflow Definition

Element	**Definition**
T	set of Tasks
C	set of Conditions.
$F \subseteq (T \times C) \cup (C \times T)$	transition between Tasks and Conditions.
A_{id}	set of all actions possible.
RL	set of security risks and $\leq$ is the partial-order over elements in RL, such that $(RL, \leq)$ is a lattice.
$A : T \rightarrow \mathbb{P}(A_{id})$	is the set of actions associated with any given task.
$R : A_{id} \rightarrow RL$	maps a risk with each action.
$S_t : T \rightarrow \{AND, XOR\}$	is the split condition[5].
$J_t : T \rightarrow \{AND, XOR\}$	is the join condition.

The semantics of the workflow are defined in terms of a transition system over workflow states. A *workflow state* for a given workflow $\mathcal{W}$ is defined by the tuple $W_s = (Tk, H, R_a)$ where:

- $Tk \subseteq \{(T \times Tk_s) \cup (C)\}$ is the set of tokens present at any of the tasks or conditions. A token at a task $t \in T$ can be in one of two states $Tk_s = \{ENABLED, FINISHED\}$.
- $H \subseteq \{(T \times A_{id})\}$ stores the set of actions performed at all the finished tasks.

[4] However, we do not show the context in the model here to simplify the model description.

[5] Similar to the actions, we support conditional XOR splits, i.e., the next task to be fired depends on the (task,action) performed at an earlier stage in the workflow. However we do not describe it here to simplify the model description.

- $R_a \in RL$ refers to the accumulated risk over all the actions performed at all the finished tasks, ie., $R_a = \cup_a(R(a)), \forall a \in H$.

The transition relation between two consecutive workflow states in the system depends on the workflow transition relation F and the split, join conditions S_t, J_t. We define the state transition relation $\delta : W_s \rightarrow W_s$ over the workflow states W_s corresponding to a workflow $\mathcal{W}$ as follows:

Join Processing

- $\delta(\{Pred(T) \cup Tk, R_a, H\}) = \{(T, ENABLED), R_a, H\}$ if $J_t(T) = AND$, where $Pred(t) = \{c \mid (c,t) \in F\}$ and F is the flow relation in workflow $\mathcal{W}$. $Pred(T)$ defines the set of all predecessor condition nodes for task T in the workflow $\mathcal{W}$.

- $\delta(\{T_p \cup Tk, R_a, H\}) = \{(T, ENABLED), R_a, H\}$ if $J_t(T) = XOR$, where $T_p \in Pred(T)$ for task T in the workflow $\mathcal{W}$.

Task Processing

- $\delta(\{(T, ENABLED, R_a, H)\}) = \{(T, FINISHED), R_a + R(A_i), (T, A_i) \cup H\}$ where $(A_i) \in A(T)$ and $+$ is the *least upper bound* function for the risk-lattice $(RL, \leq)$ of the workflow $\mathcal{W}$.

Split Processing

- $\delta(\{(T, FINISHED) \cup Tk, R_a, H\}) = \{Succ(T) \cup Tk, R_a, H\}$ if $S_t(T) = AND$, where $Succ(t) = \{c \mid (t,c) \in F\}$. $Succ(t)$ defines the set of all successor condition nodes for task t in the workflow $\mathcal{W}$.

- $\delta(\{(T, FINISHED) \cup Tk, R_a, H\}) = \{T_s, R_a, H\}$ if $S_t(T) = XOR$, where $T_s \in Succ(T)$ for task T in the workflow $\mathcal{W}$.

. We define a *workflow-run* as the sequence of workflow states $w_1, w_2, w_3, \cdots, w_n$ such that $w_{i+1} = \delta(w_i)$. Given a workflow W with the transition relation δ, (W_s, δ) is a transition system. From this we can derive a Kripke-structure (W_s, δ, L) by defining a labeling function. This allows us to perform model-checking on the system. The labeling function can depend on any of the information present in the workflow-state W_s. In particular, this allows us to reason about the accumulated risk as well as the tasks and actions fired in the system.

4 Tool Chain Architecture

Fig 4.1 shows a high level architectural diagram of the security assessment tool-chain detailing how the various components sit with respect to each other. We give a detailed description of each of the various modules:

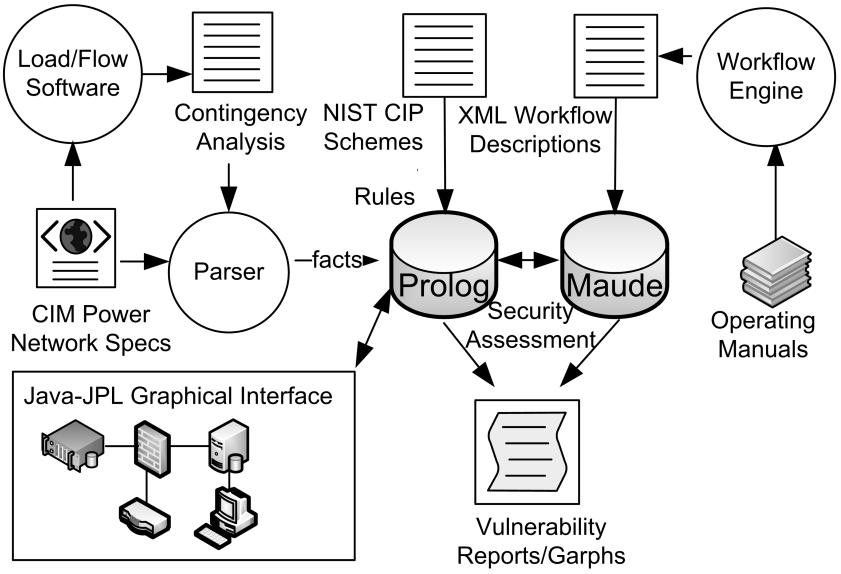

Figure 4.1: Tool Chain: High-Level Architectural Diagram

4.1 CIM Parsing

The network model $\mathcal{N}$ of the power and control networks is auto-generated from annotated specifications written in the standard descriptive language based on Common Information Models (CIM) [30] with the help of a parser tool and stored in a Prolog database. CIM's comprehensive packages cover everything from equipment, topology, load data, generation profiles to measurement and scheduling. The CIM RDF schema is documented as a self-describing XML-based IEC standard. The fact that CIM is becoming increasingly popular in the Power industry as a means of exchanging Power System information is evident from the fact that even load-flow software e.g. InterPSS[65] can digest and export CIM descriptions directly for power analysis. We create a mapping of the classes in the RDF model to entities in our security model. Table 4.1 shows the XML description and the Prolog version of a specific instance of an

actuator for a disconnect switch (DS3). Various XML attributes give detailed information regarding the switch's function, and of the SCADA elements that control it.

The parser starts by identifying the main entities such as devices, connectivity and services and then proceeds to populate the attributes of the entities. The attributes can be easily populated by looking at the properties and associations for each object in the CIM model. Some attributes such as inter-service data dependencies and the security protocols used by services are not covered by the basic CIM data model and therefore need to be merged in by parsing firewall configuration logs or manually annotating the CIM XML or doing a look-up from a services to security properties table whenever a services entity is encountered. Services running on a device can be easily determined by running *nmap* port scans while two communicating services can be identified by parsing firewall logs. To populate vulnerabilities we used models of popular exploits such as buffer overflows quoted in attack graph literature and open vulnerability databases such as CERT.

Table 4.1: CIM XML and Prolog Description of a Substation

```
1  <!-- Describes our Substation Architecture -->
2   <SubstationArchitecture>
3   <class name="CIM_LogicalSwitch"
4   Superclass="CIM_LogicalDevice">
5    <cim:CIM_LogicalSwitch ID="ActDS3"        device( ActDS3,     //ID
6    cim:type="DisconnectSwitch"                DisconnectSwitch,  //Type
7    cim:State="Closed"                         Closed,            //State
8    cim:PowerSystemResourceName=               FwareActDS3        //Services
9    "Disconnect Switch No 3 Actuator"         ).
10   cim:Manufacturer="General Electric"
11   cim:Controllerforresource="#DS3">
12     <class name="CIM_SerialLink"
13     Superclass="CIM_Link">                   connected( PLC2,    //Start Node
14      <cim:CIM_SerialLink ID="SlinkActDS3"     ActDS              //End Node
15      cim:source="PLC2"                       ).
16      cim:dest="ActDS3"/> </class>
17     <class name="CIM_Firmware"
18     Superclass="CIM_Service">                service(FwareActDS3,//Service ID
19      <cim:CIM_Firmware ID="F.wareActDS3"      1.0,               //Version #
20      cim:ver="1.0" cim:type="ModbusSlave"     ModbusSlave,       //ServiceType
21      cim:PowerSystemResourceName=             [PLC2Master],      //Dependence
22      "Actuator Service for ActDS3"            [TLS],             //SecProtocol
23      cim:secprops="TLS"                       502                //ConnectPort
24      cim:dependsupon="PLC2Master"            ).
25      cim:port="502"/> </class>
26     </cim:CIM_LogicalSwitch>
27   </class>
28    .
29    .
30  </SubstationArchitecture>
```

4.2 Event Aggregator

Modeling a "live" system such as SCADA involves inherently dealing with incomplete and imperfect information that is continually subject to change and revision. Knowing the up-to-date properties of Power system assets and its configuration is important for accurate security assessment. We added support

Table 4.2: Control Device Models as Prolog Facts

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

% service(ID,TYPE,VER,PRIV_LEVEL,PROTOCOL,ACL)
service(sqlclt1,database_client,2003,user,odbc,_).
service(sqlclt2,database_client,2000,user,odbc,_).
service(pstgresqld,dbms,1998,root,odbc,[sqlclt1,sqlclt2]).

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

.

to our tool-chain to interface to Power flow analysis software and online event aggregators to modify prolog attribute information to reflect the current state of the system. We used PowerWorld [48]-a power system visualization, simulation, and analysis tool that allows clients to take snapshots of a real Power System. A powerworld client provides a graphical view of the power system states and the information used to drive the display is obtained via TCP/IP from the retriever. This mimics a control room display that is obtaining data from the power grid over a communications network. The retriever interfaces to real devices in a SCADA network and provides data to all connected clients such as bus voltages, phase angles, flows and lines and generator status. SimAuto runs on the same host as the client and provides a COM API interface to third-party applications to access the clients datastructures. Our Tool interfaces to Powerworld through SimAuto to update its attributes about SCADA devices allowing it to analyse the system at interesting periods of time such as when there is a fault.

4.3 Implementation in Predicate Calculus

We implemented our predicate calculus security model as a form of Horn Clause logic in Prolog using SWI-Prolog version 5.6. The various devices, services, connectivity and dependencies identified by the parser were asserted as 'facts' in the Prolog knowledge base. Table 4.2 shows how Prolog facts describe interconnections and service dependencies of a sensor reporting its readings to a historian and a AdminPC accessing the same through a firewall. Facts in our knowledge base can be thought of as relational tables for example service IDs serve as foreign keys in the device entities and primary keys in the services entities. The *connected* predicate shows a bidirectional link between two devices.

Table 4.3: 'Access Control' Concern: Prolog Implementation

```
16 path(A,B,Path) :-
17     travel(A,B,[A],Q),
18     reverse(Q,Path).
19
20 travel(A,B,P,[B|P]) :-
21     connected(A,B).
22
23 travel(A,B,Visited,Path) :-
24     connected(A,C),
25     C \== B,
26     \+member(C,Visited),
27     travel(C,B,[C|Visited],Path).
28
29 is_access_control(DevA,DevB) :-
30     keys(DevA,AuthMechA),
31     keys(DevB,AuthMechB),
32     match(AuthMechA,AuthMechB).
33
34 is_end2end_conf_integ(SecPropsList, [Head]) :-
35     defsvs(Head, dservice),
36     secprots(dservice,SPList),
37     SecPropsList = SPList.
38
39 is_end2end_conf_integ(SecPropsList, [Head|Tail]) :-
40     defsvs(Head, dservice),
41     secprots(dservice,SPList1),
42     is_end2end_conf_integ(SPList2,Tail),
43     intersection(SPList1,SPList2,CommonSPList),
44     nth0(0,CommonSPList,_),
45     SecPropsList=SPList1.
46
47 ck_ConformanceTo_CIP002-08(DevA,DevB,Path) :-
48     path(DevA,DevB,Path),
49     is_access_control(DevA,DevB),
50     is_end2end_conf_integ(Path).
```

4.4 Implementation of Best Practice Rules in Prolog

We implemented various rules to check for CIP conformance in a cyber-infrastructure. Prolog rules are essentially goals that check for other sub-goals to hold true. Sub goals maybe other rules or primitive facts such as those described in the last section. Table 4.3 describes the implementation of the 'Correct implementation of Access Control' concern. Note that some of the attributes have been taken out for brevity.

We explain the listing bottom up. The *ck_conformanceTo_CIP002-08 rule* (line 47) takes three arguments: the two communicating devices and a Path variable. It then calls a helper rule *path* (line 16) which finds a path namely a list of nodes through which one must travel to get from node A to node B. Path uses the recursive travel rule to do this. A declarative reading for line 20 is: "A path from A to B is obtained if A and B are connected". The second clause (line 23) amounts to "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 part of the path, and one continues finding a path from C to B". Avoiding repeated nodes ensures that the program will not cycle endlessly. Once the Path is known then access control is checked (line 29) by comparing if both communicating nodes

Table 4.4: A Prolog Rule for an Attack Graph

```
52 execCode(Principal,Victim,Priv) :-
53     device(Victim,_,_,Svslst),
54     containsVul(Svslst,remoteExploit,VSrvc),
55     service(VSrvc,_,_,Priv,AllowedHosts,AllowedSvs),
56     hasaccount(Principal,Source,PrincipalPriv),
57     isIncluded(Source,AllowedHosts),
58     existsServiceType(Source, PrincipalPriv, AllowedSvs),
59     path(Source, Victim,Path).
```

Table 4.5: Java-JPL Code to Query Path Information

```
60 Variable X = new Variable("X");
61 Variable Y = new Variable("Y");
62 Variable P = new Variable("P");
63 Term arg[] = {X,Y,P };
64 Query q = new Query("path", arg);
65
66 while (q.hasMoreElements()){
67  Term bound_to_x= (Term)((Hashtable) q.nextElement()).get("P");
68  String[] strarray = jpl.Util.atomListToStringArray(bound_to_x);
69  for(int i=0;i<strarray.length;i++)
70   System.out.println(strarray[i]);
71 }
```

use a pre-shared key or PKI authentication. Confidential communication (line 34,39) amounts to checking if every pair of consecutive nodes on a path share a encryption channel (e.g. IPSec, TLS) with each other.

4.5 Implementation of Attack Graph Rules in Prolog

Generic Prolog rules search for facts derived from the CIMs to determine whether an attack is possible. For instance the prolog rule shown in Table 4.4 says that a *Principal* can execute code on a *Victim* device with a privilege *Priv* if a service *VSrvc* running on that device contains a *remoteExploit* vulnerability and it allows connections from the *Source* device that the *Principal* has an account on. Furthermore *Source* should have a service from the set of allowed *AllowedSvs* types, there should be a network path from *Source* to *Victim* and *Source* should be in the ACL of *Victim's* allowed hosts.

4.6 Graphical Control User Interface

In order to allow the tool to be used by security analysts and have easy and fast use we added a Java based user interface frontend to the Prolog engine. JPL is a set of Java classes and C functions providing an interface between Java and Prolog by the embedding of a Prolog engine within the Java VM. By annotating each device in our CIM specification with x and y coordinates we can easily import and display the SCADA network in a Java grid panel.

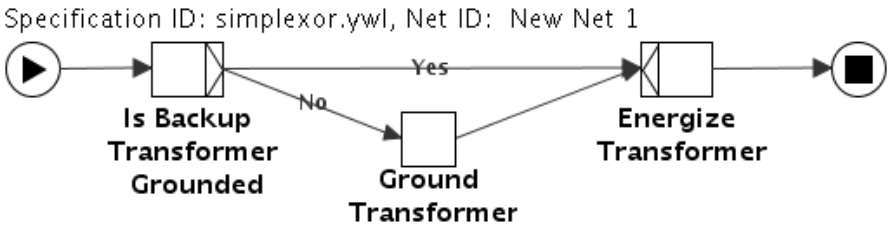

Figure 4.2: Generic Workflow Example in the YAWL Editor

The advantage of this approach over other network visualization tools such as Graphviz [49] is that it allows more user interaction. For instance a security engineer can hover a mouse over a device icon to see a detailed listing of its security attributes or point and click on devices of interest and formulate a query. Table 4.5 demonstrates JPL code that imports information about all possible paths between two devices in the form of lists from Prolog for display. This predicate implementation was described in Table 4.3

Three JPL variables (lines 60,61 and 62) are created to hold the two devices and the list of all possible paths between them. A query is then formulated and sent to the Prolog engine which populates these values. The *'atomList-ToStringArray'* (line 68) is one of many utility functions provided by the JPL library to convert between Java and Prolog types.

4.7 Representing Recovery Workflows

We represent operating procedures of a SCADA power-grid as "generic" control-flow workflows that are not constrained by the detailed architecture of a specific power-grid implementation. For example, the workflow described in figure 4.2 could be applied to activate any transformer.

The transformer workflow involves grounding the transformer (if it isn't already grounded) and energizing it afterwards. The first task of the workflow "Is Transformer Grounded" has a mutually exclusive conditional split to two other tasks. The transformer chosen for the first task (there can be multiple transformers to choose from) is left undecided. We call the possible ways a given task can be bound to specific entities as leading to a choice of "actions". This list of actions depends on the specific substation (some substations might have 1 backup transformer and some might have more) as well as their individual states (some might already be grounded, for instance). However, the "action" available at any given task may be conditional on the action done at an earlier task. Furthermore, the choices taken after the task may be dependent on the actions done earlier or at the current task.

We describe the operating procedures as workflows using Yet Another Workflow Language (YAWL [63]). YAWL is a workflow description language that

Table 4.6: XML Output of YAWL

```
<task id="Is_Backup_Transformer_Grounded_3">
<name>Is Backup Transformer Grounded</name>
<flowsInto>
    <nextElementRef id="Energize_Transformer_5" />
    <isDefaultFlow />
</flowsInto>
<flowsInto>
    <nextElementRef id="Ground_Transformer_4" />
    <predicate ordering="0">true()</predicate>
</flowsInto>
<join code="xor" />
<split code="xor" />
</task>
```

supports common workflow patterns and its Editor [64] allows the construction of appropriate control-flow descriptions for the workflows and the export of this information as an XML file. (Note that our tool chain uses only these XML files and we do not use either YAWL's data-flow aspects or validation engine.) For instance, the XML output by the YAWL Editor for "Is Transformer Grounded" shown in table 4.6 illustrates a split to "Energize Transformer" as well as "Ground Transformer". Furthermore, it splits mutually exclusively (<split code="xor">), i.e., exactly one of the next two tasks is fired, and the task itself is fired when any one of the previous tasks incoming task is finished (<join code="xor">)[1]. Similarly we can describe, splits and merges requiring parallel firing of tasks (AND-splits, AND-join) or multiple choice of tasks (OR-splits, OR-join). We currently support only the basic control flow patterns as described in ([63]).

4.8 Mapping Workflows to Term-rewriting Logic

We use the "XML to Maude" converter tool to read the XML based workflow description generated by YAWL and generate a term-rewriting description of the workflow in Maude. As described earlier in Section 3.2 the workflow description can be split into two parts. The generic configuration independent workflow that describes the control-flow of the recovery procedure is given in YAWL as described in Section 4.7. The configuration specific aspects of the workflow (such as the set of actions that can be done at a given task, the conditions for choosing the next task(s) to be fired) are given to the converter tool by the "Risk Calc" tool (as shown in Figure 4.1).

To help with the workflow analysis, we developed a term-rewriting module in Maude. This generic module consists of rules which describe the valid workflow transitions for any generic workflow. For instance, Table 4.7 shows the rule that

[1]This incoming condition is irrelevant because there is exactly only one incoming task to "Is Transformer Grounded" in our example, but it will be relevant when there are multiple incoming transitions to a given task.

Table 4.7: Maude Rule for Conditional Split

```
1  rl  [conditionalsplit] : ——Rewrite Rule for Conditional split
2  [
3  < TkId , FirCond , CndActs , { [ ( TkId1 ? ActId1 ) = TkId2  ] ;
4      CondSpltLst }>  TkList ,ActList ,
5      ( TkId1 ? ActId1 ) ActCxt ,
6      TkIdList1 ,
7      TkId * TkIdList2 ,
8      TkIdList3 ,
9      SecRsk
10 ]
11 =>
12 [
13 < TkId , FirCond , CndActs , { [ ( TkId1 ? ActId1 ) = TkId2  ] ;
14      CondSpltLst }>  TkList ,ActList ,
15     ( TkId1 ? ActId1 ) ActCxt ,
16     TkIdList1 ,
17     TkIdList2 ,
18     TkId2 * TkIdList3 ,
19     SecRsk
20 ] .
```

allows Maude to evaluate a task with a conditional split [2].

The lines 2-10 describe the configuration before and the lines 12-20 describe the configuration after the transition rule is fired. Line 2 contains the conditional action ($[(\mathsf{TkId1?ActId1}) = \mathsf{TkId2}]$) which determines whether while evaluating task TkId, the next task chosen should depend upon the action *ActId1* chosen earlier at task TkId1. Note that TkId1, ActId1 etc., in the rules are variables and Maude can match the instantiation of the configuration to the left-hand-side of the rule and transform the current configuration as per the rewriting rule. Maude performs the transition by looking at whether the action-context (given in line 5) contains the tuple that says that the required condition to perform the particular task $TkId2$ was satisfied. If so, it picks the next task TKId2 and puts it in the tasks to be fired (in line 18), while simultaneously removing the current task $TkId$ from the evaluation queue (line 16). We emphasize that these rules are generic and will work for any workflow description under consideration.

It is important to emphasize that these rules are generic and will work for any workflow description under consideration. Once a concrete workflow description is given to Maude, along with these workflow transition rules, Maude's engine will check if the possible transitions could be applied to the system.

The "XML to Maude" converter populates the terms to instantiate a *specific* workflow that was described generically using YAWL (and given in XML to the tool). The input includes the various actions possible at any given task and the evaluated security risk for each task.

[2] A conditional split is similar to an XOR split except that the next task chosen depends on the context

4.9 Analyzing Workflows in Term-rewriting Logic

Given the workflow description (the generic term-rewriting theory along with the specific workflow instantiation), we can use Maude's LTL Model Checker to verify any LTL property on the workflow. For instance, we can verify that any task initiated in the system should never deadlock (starting from the given initial state). This could be specified as the following LTL property:

$$\Box(\mathsf{initialstate} \rightarrow \Diamond(\mathsf{finalstate}))$$

Furthermore, we can verify other hard constraints that should always hold on any execution of the workflow (for instance if we open a switch in a given recovery workflow, we should never close it).

Given that each task might be satisfied by multiple actions, we can have multiple ways of executing a given workflow. Since each action (such as turning on a device), can have a different security risk, the risk taken to complete a given task is dependant on the choices of actions taken for any given tasks. Hard security constraints based on actions cannot thus be model checked because we do not have any idea in advance about the accumulative security risk of all the choices.

However, we can find the workflow path (along with the choices taken at every task) that minimize the security risk for the given procedure by model checking iteratively. This can be performed by iteratively checking for the following LTL formula:

$$\Box(\mathsf{initialstate} \rightarrow (\mathsf{finalrisk} \geq \mathsf{maxrisk}).$$

A simpler mechanism for finding the path with minimal security risk is to use Maude's *search* command to find a final-state in the workflow with the minimal risk. We search to find a path from initial state to final state such that the final state contains no security risk, failing which, we iteratively query for a path with the next higher risk. Note that there might be multiple solutions because our security risk lattice is a partial order.

This effectively checks that in all the paths that we are taking to finish the procedure, the total risk taken is $\geq$ the maximum risk computed so far (maxrisk is initialized to the upper bound of the security-risk lattice that we have). In case this isn't true, it means that there is a set-of-choices that could be taken that contain a security-risk which is lesser than maxrisk and Maude will return the counter-example. Note however, that this needn't be the path with the *least* cumulative risk, only lesser than the maxrisk. However, by doing this iteratively by substituting maxrisk with the value returned in the previous iteration, we will eventually reach a value where the formula holds. We then present the last counter-example returned by Maude as the set of actions that the user should take to minimize his risk while performing this recovery procedure.

5 The Model Checking Process

Our implementation involves approximately 2,200 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 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 with scenario B showing ranking of recovery procedures during a substation fault.

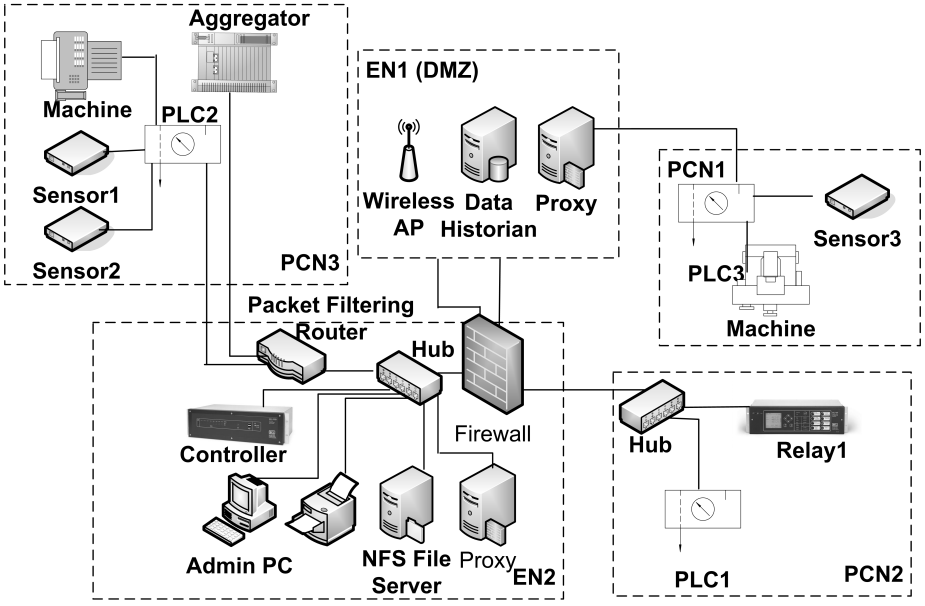

Figure 5.1: Assessment of a Simple Control Network Configuration

5.1 Scenario A: Security Best Practices Conformance Checking

Figure 5.1 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 relation-

ships 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.

5.1.1 Test 1: Access Control Implementation

Figure 5.2 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 5.1 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.

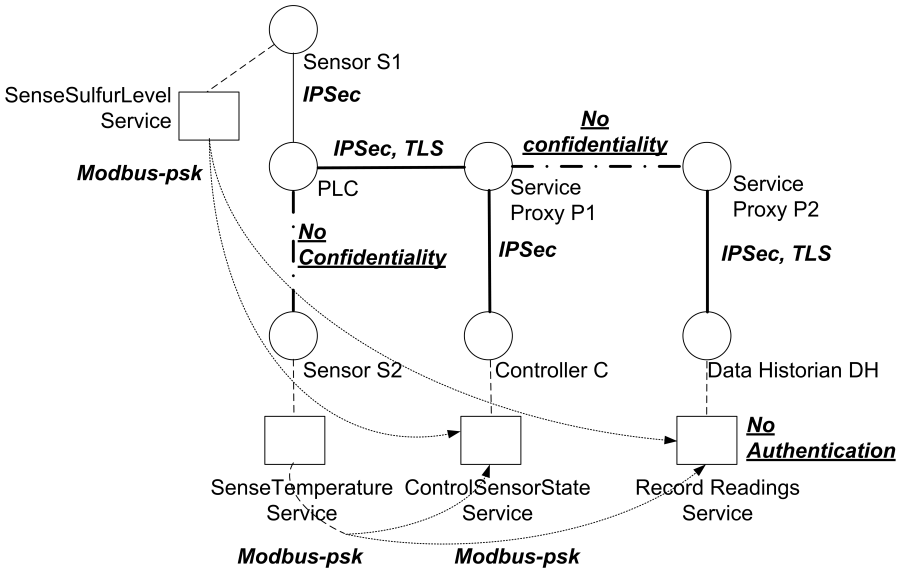

Figure 5.2: Access control Conformance Checking

Table 5.1: 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 → DH	No	No	No
S2	C3	S2 → PLC → P1 → C	No	Yes	No
S2	DH	S2 → PLC → P1 → P2 → DH	No	No	No

The **ck_ConformanceTo_CIP002(sensors,sinks,Paths)** reveals that the

Table 5.2: Correct Firewall Deployments Results

Num	Substation Architecture	Rating	Offending Link
1	Original Configuration as shown in Fig 5.1	2	DH-PCN1
2	Same as 1 but with DH-PCN1 link removed	2	S2-PacketFilteringRouter-C
3	Same as 2 but with router replaced with a stateful firewall	3	C-AdminPC
4	3 but with C moved to the DH subnet	4	None

data association channel between the sensors and the controller enforces integrity because there is an end-to-end pre-shared key. However the channel between sensor S2 and the controller does not have confidentiality because the hop between sensor 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 mainly because the sensor supports IPSec as an encryption protocol.

5.1.2 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 5.2 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 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.

5.2 Scenario B: Security Ranking of Recovery Procedures

We describe a scenario to illustrate our security assessment tool chain functionality to rank recovery procedures according to security risk when a fault occurs

in a substation. Vulnerable services were hypothetically introduced by us for the purposes of this paper. Figure 5.3 shows a 275 kV substation with three transformers and two bus bars. The PowerWorld tool provides a snapshot of the system when a fault occurs on the 77kV bus-bar causing the protection system to operate, trip two of the transformers (TR2 and TR3) causing the remaining transformer to suffer from a severe overload condition. Operators in the control center have to reduce the load of the transformer in ten minutes otherwise TR1 overheats and burns out causing a blackout of the entire 77kV system in the substation.

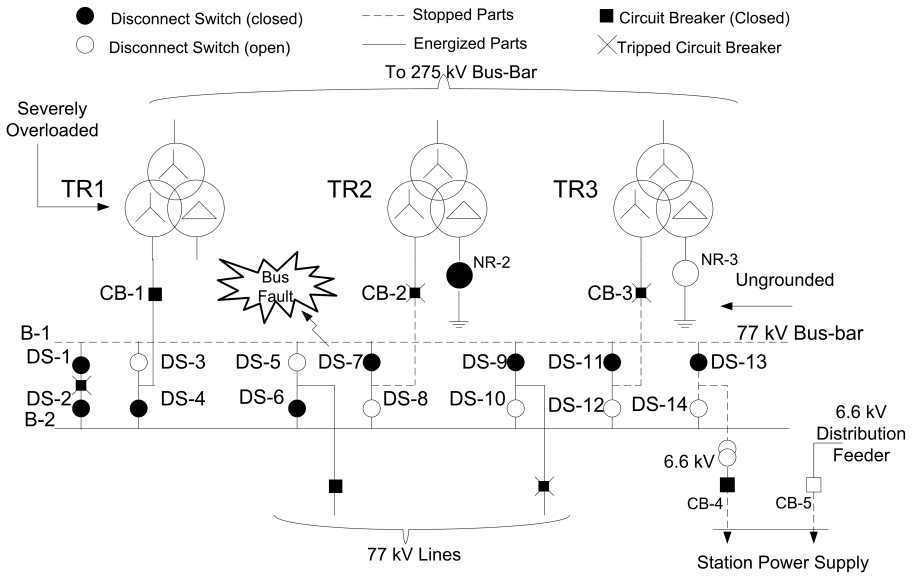

Figure 5.3: A 275 kV Substation with a Fault

Figure 5.4 shows the SCADA network that controls the energizing of the various power elements. The left hand side of the figure shows the enterprise network with commodity desk tops and servers while the right hand side shows the PCN involved with taking sensor measurements and flipping switches to control the power flow. Top and bottom rows of actuators correspond to Bus 1 and Bus 2 control respectively. Groups of actuators are controlled by a PLC which is in turn controlled by a relay. Mostly serial connections (RS-485) are used to link these SCADA devices together, however some of the more upgraded devices (for instance PLC4) do use Ethernet as well. Finally the relays report their values to a data aggregator that communicates these to a data historian on the EN. The firewall contains rules to allow modbus communication between the aggregator and the historian.

We ran our security assessment toolkit on a model of this substation infrastructure. Logical attack graphs were generated for each candidate device in the various recovery procedures. The attacker was assumed to be an outsider with network access to the EN and no privileges to any machine except for his own.

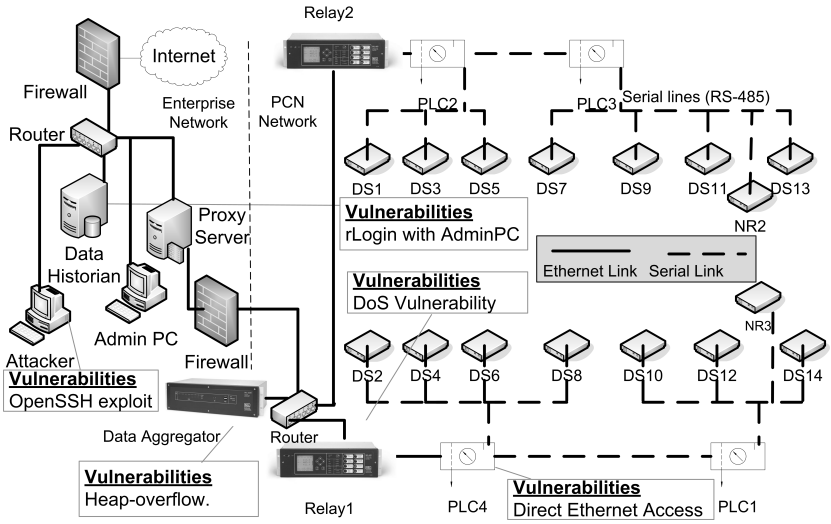

Figure 5.4: Control Network for the 275kV Substation

Figure 5.5 shows a DoS attack graph for one of these devices: DS12. DS12 is controlled by PLC1 which is in turn controlled by Relay1 that forwards it instructions for controlling the re-energizing of the transformer. If an attacker were able to successfully deny service to this device then the underload operation for TR1 could be delayed causing it to overload.

```
<0>||--doSattack(attacker, DS12)
  <r0a> Rule: Modbus Invalid Header
   []-connectivity(aggregator,Relay1)
   []-vulExists(modbus_srvce, 'MODBUS_INVALID_HEADER', remoteExploit, doS, 0.9)
    <1>|--execCode(attacker, aggregator, root)
     <r1> Rule : Remote Heap over-flow of modbus server
     []-connectivity(historian, aggregator, modbus)
     []-vulExists(modbus_srvce, 'HEAP_OVERFLOW', remoteExploit, Integrity, 0.7)
      <2>||--execCode(attacker,historian,root)
       <r2> Rule : Trust Relationship with AdminPC
        []-connectivity(adminPC,historian, rlogin)
        []-trust(adminPC,historian)
         <3>|--execCode(attacker,adminPC, root)
          <r3> Rule : Remote Buffer Overflow
           []-connectivity(attacker,adminPC)
           []-service(adminPC,OpenSSL)
           []-vulExists(OpenSSL, 'GLSA 20071030', remoteExploit, Integrity,0.49)
  <r0b> Rule: 08 Diagnostics function code with a sub-function of 01
   []-connectivity(aggregator,Relay1)
   []-vulExists(modbus_srvce, '08DiagnosticSubf01', remoteExploit, doS, 0.87)
    |--execCode(attacker, aggregator, root) ==> <1>
```

Figure 5.5: A Logical Attack Graph for Device DS12

The root node is the attack goal; in this example it is *doSattack(attacker,DS12)*, meaning "the attacker can cause the DS12 to be unresponsive". Every rule node is labeled with the rule name that is used for the derivation step. Rule nodes *r0a* and *r0b* illustrate that there are two possible DoS vulnerabilities (depicted by square brackets) in the Modbus protocol the Relay is using. Sending a Modbus TCP message with an invalid header or a *08Diagnostic* message would cause the Relay to be unresponsive. The aggregator that logs events, talks directly

to all the relays and a root privilege on it can be achieved if the attacker can exploit a remote heap-overflow vulnerability in its modbus service. Finally the attacker can cross over the firewall to get to the aggregator by compromising the historian on the EN that has a service relationship with it. The attacker can gain access to the historian via its rlogin trust relationship with the admin PC which has an OpenSSL service with a buffer overflow vulnerability. Notice that in this scenario the attacker does not need to be an insider with user or root access to any of the machines. Not shown due to space limitation are graphs for DS8, NR2 and NR3. NR3's risk tree is the same as that of DS12 because they are connected to the same vulnerable relay. DS8 on the other hand has a direct Ethernet access from the compromised aggregator and its integrity can be easily breached by the attacker. A high integrity risk unlike an availability risk is very dangerous as we will see in the workflows later because it means that the attacker has complete control and can thwart an operator's attempts to open or close a switch whether he does it manually or via SCADA commands.

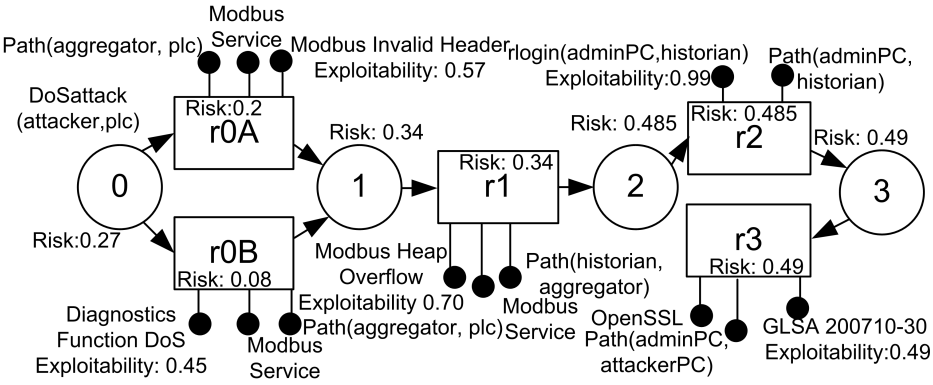

Figure 5.6: Logical Attack Graph for a DoS attack on a PLC

A more compact version of the attack graph is shown in Figure 5.6 that uses circles to represent derived facts, squares to represent derivation rules and filled circles to show primitive facts. This graph shows the calculation of the security label for DS12. The leaf nodes representing exploits are labelled with exploitability probabilities obtained from the CERT Vulnerability calculator [54].

Figure 5.7 shows the recovery procedure that an operator has to follow to bring up a transformer. These procedures are independent of the specific configuration of the sub-station and the actual running of the workflow depends on the possible actions at any given task (not depicted in the picture). For instance, the workflow task "Select Transformer" consists of two actions "ChooseTR2" and "ChooseTR3". Depending on the current configuration, the task "Transformer Grounded" can either split to "Ground Transformer" or directly to "Energize Transformer". Both splits and actions depend on the context, i.e., which transformer was chosen earlier at "Select Transformer" and whether according to the

Table 5.3: Device Risk Evaluated and Assigned to Actions

Task	Context Required	Action	Risk
Select Transformer		ChooseTR2	No Risk
Select Transformer		ChooseTR3	No Risk
Ground Transformer	ChooseTR3 at Select Transformer	GroundTR3	Low Availability NR3:0.27
Close Manually	ChooseTR3 at Select Transformer	Close Switch DS12 Manually	No Risk
Close Manually	ChooseTR2 at Select Transformer	Close Switch DS8 Manually	High Integrity DS8:0.34
Close Via SCADA	ChooseTR3 at "Select Transformer"	Close Switch DS12 by SCADA	Low Availability DS12:0.27
Close Via SCADA	Choose TR2 at "Select Transformer"	Close Switch DS8 by SCADA	High Integrity DS8:0.34

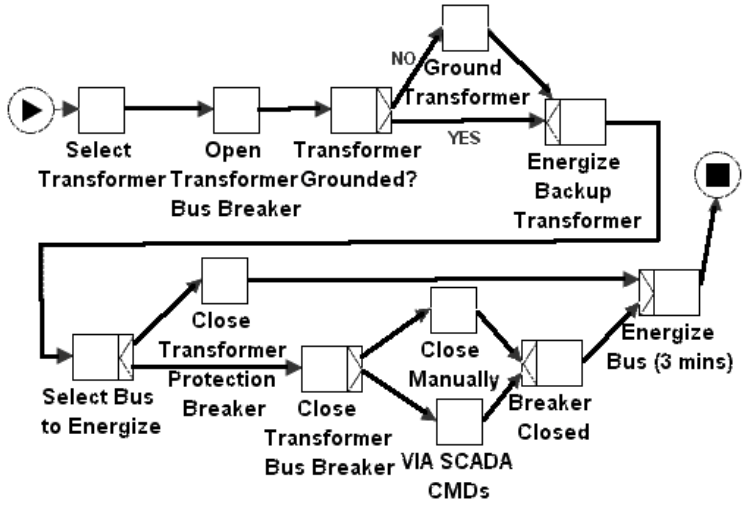

Figure 5.7: Procedure to Enable a Backup Transformer

current configuration the transformer chosen was grounded or not. A subset [1] of actions possible at any given task along with their given risks is calculated by the network-analysis tool and is given in Table 5.3. The list of possible actions also depends on the current state of the system. For instance, considering the configuration described in Figure 5.3 (TR3 is not grounded while TR2 is not), the "workflow analysis" tool can model-check if the final state is reachable from the initial state (i.e., there are no deadlocks etc.,). Furthermore, we can check if there is any path that can finish the workflow with "No Risk". Searching for the paths that fulfill the workflow in Maude gives us a list of paths each with its own risk. For instance, searching for the path with "No Risk" fails in this scenario because there is no set of choices that can fulfill the workflow with no risk at all. Searching for paths with Availability and Integrity Risks for the configuration in (Fig:5.3) gives us the paths as shown in in Table 5.2. Essentially we see that

[1] We do not show other actions possible at other tasks for instance "Close Transformer Protection Breaker"

if we choose TR2, then we have a potential integrity risk [2] (because device DS8 has an integrity risk), whereas if we choose the TR3, we have a potential availability risk (because Grounding the Transformer, uses switch NR3 which has an availability risk). There are also other possible evaluations (such as using SCADA to close the switch at the task "Close Transformer Bus Breaker", which we haven't shown).

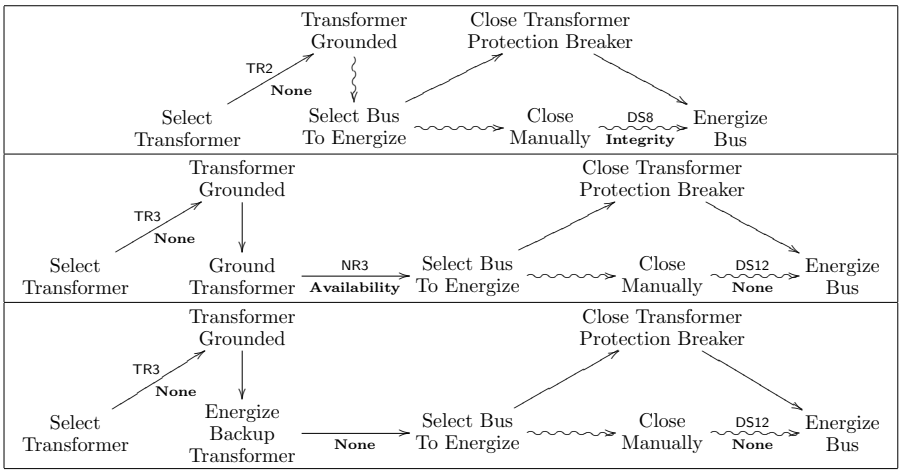

Table 5.4: Possible Ways to Fulfill the Workflows

Since, the minimal risk for any workflow run is "Availability" , we recommend that the operator choose "TR3" and (due to the dependency on the transformer chosen), close switch "DS12" manually. The third workflow shown, shows the evaluation after a change in the configuration (where TR3 is grounded). Note that because of the fact that the Transformer TR3 is already grounded, we no longer need to run the task "Ground Transformer" and therefore the minimal risk run is the one that chooses "TR3" transformer (as before) and there is a set of choices which the operator can take which have no risk at all.

[2] Recall that the evaluated risk of a given workflow run is the least upper bound of all the risks of the individual actions performed during that run

6 Optimal Security Hardening of the Power Grid

The preceding discussion has so far shared the firm posture adopted by CIP advisory agencies such as FERC and NERC regarding best-practice compliance meaning that a power utility will either comply or will not comply to a security standard. For instance a security assessment by the tool-chain on a control network will return a report indicating all the access points that should have been protected by firewalls but were not. While the notion that all vulnerable holes should be plugged is ideal, in reality security comes at a cost (time, resources, efficiency etc) and security administrators will often find themselves faced with the challenge of having to conduct a cost-benefit analysis before they can comply with a best-practice.

In 2007 when NERC issued an advisory to 1,800 power operators and owners outlining immediate and longer term steps they should take to address cybersecurity vulnerabilities, one of them being the protection against an Aurora[37]-style attack, compliance with the advisory was voluntary. A recent FERC audit of 30 utilities found that most were not in compliance with the advisory [26, 28]. Discussion with the security engineers showed that the majority was still unsure about the nature and direction of the possible attack vector that could exploit this vulnerability as well as whether the criticality of their generators merited such an exercise.

This chapter extends the logical models presented earlier to try and quantify the cost of implementing a security scheme and the benefit it provides in terms of protection against possible attacks. To this effect the scope of the control network models from earlier have been narrowed to a particular set of control device type i.e. relays R where $R \subset D_c$ and a realistic threat model for relay networks is presented. Best practice schemes as applied to relays are presented and algorithms are described that automate the cost-benefit analysis exercise required when up against multiple factors: several power network vulnerabilities, and different schemes with their varying degrees of protection and implementation costs and lastly a budget constraint.

6.1 Role of Relay Networks in the Power Grid

Power flow (energized or deenergized) in a power device is controlled by breaker/relay combinations, henceforth called just relays at the point the power device connects to a bus and is governed by the *controls* relationship rNd where $r \in R$

and $d \in D_p \setminus B$ and can be queried by the following function 6.1.

$$controls : R \rightarrow (D_p \setminus B, B) \text{ relay to device, bus mapping;} \tag{6.1}$$

Relays belonging to the same substation (pred 6.8) communicate in real-time pilot protection schemes[33] via multicast protocols (for example 61850 GOOSE [4] messages) using the publish-subscribe paradigm over a broadcast medium such as Ethernet (pred 6.2). Relays across different substations can communicate if there is a wide area network (WAN) connection via modem lines between the two substations (pred 6.3). A WAN network access usually exists between an unmanned substation and a control center for purposes of remote engineering access, monitoring and alarms. Unless otherwise indicated on the power network schematic, we assume that a substation's control center is the powerplant with the largest generation connected to it via transmission lines (pred 6.6). The logical predicates below dictate when network access exist between two relays. Note that network access for instance for TCP/IP communication should not be confused with electrical power connections.

$$\begin{aligned} ethernetlink(r_i, r_j) =& r_i, r_j \in R \wedge \exists s \in S \\ & [s \in belongsto(r_i) \wedge s \in belongsto(r_j)]; \end{aligned} \tag{6.2}$$

$$modemlink(r_i, r_j) = r_i, r_j \in R \wedge (modem(r_i, r_j) \vee modem(r_j, r_i)) \tag{6.3}$$

$$netaccess(r_i, r_j) = r_i, r_j \in R \wedge (ethernetlink(r_i, r_j) \vee modemlink(r_i, r_j)); \tag{6.4}$$

where we have the helper functions:

$$\begin{aligned} modem(r_i, r_j) =& r_i, r_j \in R \wedge \exists s_i \in S[s_i \in belongsto(r_i) \wedge \\ & substation(s_i) \wedge \exists s_j \in S[s_j \in controlctr(s_i) \wedge \\ & s_j \in belongsto(r_j)]] \end{aligned} \tag{6.5}$$

$$\begin{aligned} \underset{s_i \in S}{controlctr(s_i)} =& \{s_j \in S | s_j \in adjacentpplants(s_i) \wedge \forall s_k \in S \\ & [s_k \in adjacentpplants(s_i) \wedge (power(s_j) \geq power(s_k))]\}; \end{aligned} \tag{6.6}$$

$$\begin{aligned} \underset{s_i \in S}{adjacentpplants(s_i)} =& \{s_j \in S | powerplant(s_j) \wedge \exists b_i \in s_i, \exists b_j \in s_j \\ & [linesin(b_i) \cap linesin(b_j) \neq \emptyset]\}; \end{aligned} \tag{6.7}$$

$$\underset{r_i \in R}{belongsto(r_i)} = \{s_i \in S | \underset{d_i \in D_p, b_i \in B}{\exists d_i \exists b_i} [(d_i, b_i) \in controls(r_i) \wedge b_i \in s_i]\}; \tag{6.8}$$

6.2 Contingency Losses: A Metric to Quantify Cyber-Attack Damage

Definition 1 *A contingency is a condition where a set of devices D_i are taken out of service during power system operation by misconfiguration of their controlling relays R_i that causes a violation in a set of other devices D_j where $D_i \cup D_j = \emptyset$ i.e. the set of devices D_j exceed their maximum operating limits.*

In any electric network, current and voltage are governed by Kirchhoff's current and voltage law and the current flow through the branches is governed by a generalization of Ohm's resistive law. Therefore there is a current limit on each line. If one line is not operational due to some contingency, the current flow will take a different path in the network. This may cause a current in a branch to increase to a dangerous level and might cause a heating and melting of the wires. Similarly if a transformer or a generator is forced to operate beyond its intended capacity violations may occur causing malfunctioning, for instance burnt insulation and wiring. Since relays can be operated remotely, the ability to cause contingencies offers the malicious adversary an excellent avenue of attack. Function 6.9 returns the device violations which occur when a set of devices have a contingency.

$$conting(R_i) : R_i \rightarrow D_j \text{ where } R_i \subset R, D_j \subset D_p; \tag{6.9}$$

Definition 2 *Loss is a metric to estimate the extent of the damage caused by violations and is directly proportional to the product of the cost (per hr) of unmet demand $cost_{umd}$, the load shed (per hr) and the time (per hr) to repair the violations of all n devices under violation:*

$$\underset{D'_p \subset D_p}{loss(D'_p)} = \sum_{d_i \in D'_p} power(d_i).cost_{umd}.time_{repair}(d_i); \tag{6.10}$$

6.3 Threat Model

We assume a non-global, partial adversary who does not monitor all links. He is limited to one tap which can be put on any substation ethernet and between modem-to-modem links but not powerplants. He is familiar with the power network schematics and knows the exact contingencies that will cause the maximum loss, which is also his objective. He is however limited by a cost that he has to pay every time he compromises a relay needed for a contingency and this cost is deducted from the number of resources he has available. He must have network access to all relays needed to cause contingencies. Finally we assume that he has two kinds of attacks at his disposal- DoS and Masquerade attacks. The former can be protected against by firewalls and traffic segregation and

latter via encryption. We assume the attacker has 2 resources and each relay compromise has a cost of 1. Under these assumptions predicate 6.12 returns all the pairs of relays possible to attack for an unprotected substation while predicate 6.11 determines the contingency relay pair with the maximum loss.

$$\begin{aligned} \underset{s\in S}{maxattack}(s) =& \{(r_k, r_l) \in attack(s) | \forall (r_i, r_j) \in attack(s) \\ & [\exists D_n \in conting(r_k, r_l) [\forall D_m \in conting(r_i, r_j) \\ & [loss(D_n) \geq loss(D_m)]]]\}; \end{aligned} \tag{6.11}$$

where we have the helper predicate

$$\begin{aligned} \underset{s\in S}{attack}(s) = \{(r_1, r_2) \in R \times R | & netaccess(r_1, r_2) \wedge \\ & \exists (d_1, b_1) \in controls(r_1) [\exists (d_2, b_2) \in controls(r_2) \\ & [((b_1 \in s \wedge b_2 \in s) \vee \\ & (b_1 \in s \wedge b_2 \in controlctr(s)) \vee \\ & (b_1 \in controlctr(s) \wedge b_2 \in s))]]\}; \end{aligned} \tag{6.12}$$

We will now present a short discussion on why such an adversary model is realistic. Substations are unmanned, with little or no enclosure because of their large size. Modems relay substation messages across large distances mostly using telephone cables or the public internet infrastructure. Taping into either one is trivial for a determined attacker. Powerplants on the other hand are harder to infiltrate because they are usually manned with physical security in place to protect generators and the fuel. Pilot protection schemes used by relays have stringent demands such as low-latency communication and high susceptibility to replay and error propagation (small blocks of data transmitted in real-time) so traffic manipulation attacks can severely impact system reliability. Once the adversary taps into a substation he only employs cyber-attacks as apposed to physical because a) cyber attacks have the potential to cause considerably more damage (as will be evident later in the paper) and b) they are more subtle. Consider the attention drawn by attempting to damage a transformer using a shotgun as apposed to causing it overload by a simple command sent to a relay.

6.4 Security Best-Practice Schemes for Attack Damage Mitigation

Given the adversary model we describe the options available to the security engineer to mitigate the damage caused.

Definition 3 *A security scheme l_i can be applied to a substation s_j to limit the*

adversary's network access. Each scheme once applied to a particular substation has an associated implementation cost c_{ij} and attack coverage a_{ij}. a_{ij} provides an estimate of the average revenue loss if the substation gets attacked despite the security scheme in place.

The next couple of sections elaborate on the terms used in this definition by describing some schemes formalized from NIST's security best practices and how their costs and attack coverages are determined.

6.4.1 Intrasubstation Traffic Segregation via Virtual LANs

Ethernet on its own provides little security from malicious intruders and segregating it into multiple IP subnets is one approach to narrowing an electronic security parameter. The NIST [59] guide on SCADA security states that:

> *"VLANs allow switches to enforce security policies and segregate traffic at the Ethernet layer mitigating the risks of broadcast storms that may result from port scanning or worm activity."*

The vlan predicate 6.13 maps a substation ethernet into n multiple broadcast lan segments separated by vlan switches such that no groups of relays whose contingencies will together cause a violation belong in the same segment.

$$\begin{aligned} \underset{s\in S}{vlan(s)} \geq min\{n | \underset{X_i\in\mathbb{P}\{R\}}{X_1,...X_n}, \bigcup_{\forall i} X_i = relaysin(s), \underset{i\neq j}{X_i \cap X_j} = \emptyset \\ \forall r_l \underset{l\neq k}{\forall r_k} \in X_i \; [conting(r_l, r_k) = \emptyset]\}; \end{aligned} \tag{6.13}$$

where we define the helper function:

$$relaysin(s) = \{r \in R | s \in belongsto(r)\}; \tag{6.14}$$

Figure 6.1 illustrates how a vlan supporting switch and router combination can be used (right) to replace a simple hub-spoke ethernet configuration (left) to segment the network into multiple broadcast domains such that dependent relay combinations needed to cause violations are isolated.

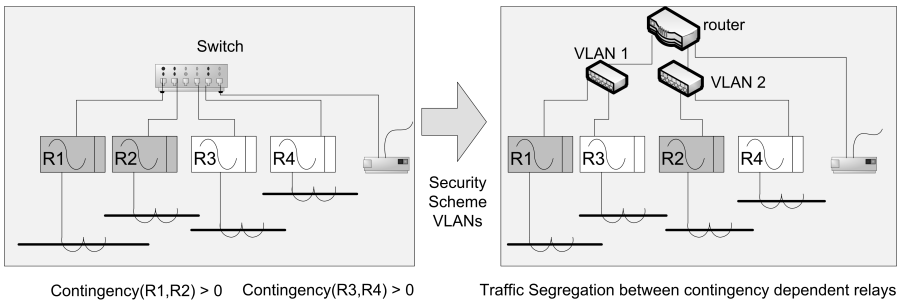

Figure 6.1: Intrasubstation Traffic Segregation using VLANs

$$\begin{aligned} \underset{s\in S}{attack_{vlan}}(s) =& \{(r_1, r_2) \in attack(s)| \\ & \exists(d_1, b_1) \in controls(r_1)[\exists(d_2, b_2) \in controls(r_2)[\\ & (ethernetlink(r_1, r_2) \wedge \exists X_i \in vlan(s)[r_1, r_2 \in X_i \wedge b_1, b_2 \in s])\vee \\ & (modemlink(r_1, r_2) \wedge ((b_1 \in s \wedge b_2 \in ctrlcenter(s))\vee \\ & (b_2 \in s \wedge b_1 \in ctrlcenter(s))))]]\}; \end{aligned} \tag{6.15}$$

As shown in pred 6.15 the attacker is restricted in the network access he has, and can only compromise devices if they are in the same vlan as his initial tap or accessible via a modem link.

6.4.2 Intersubstation Traffic Segregation via Firewalls

While the vlan scheme limits the adversary from attacking multiple targets within a substation it provides little or no protection against attacks traversing multiple connected substations. According to NIST's CIP best security practices rules [43, 22] firewalls should be used to segregate traffic between process control networks (PCN), and engineering and monitoring access. Predicate 6.16 shows that in a firewall protected modem link the only avenue of attack is the substation ethernet.

$$\begin{aligned} \underset{s\in S}{attack_{firewall}}(s) = & \{(r_1, r_2) \in attack(s)|ethernetlink(r_1, r_2)\wedge \\ & \exists(d_1, b_1) \in controls(r_1)[\exists(d_2, b_2) \in controls(r_2)[\\ & (b_1, b_2 \in s)]]\}; \end{aligned} \tag{6.16}$$

Figure 6.2 illustrates how a three-port firewall segregates control traffic into multiple domains; HMI/local generation control, transmission and remote monitoring preventing the attacker from using a compromised modem connection to exploit a cross substation contingency i.e. between relays 1 and 4.

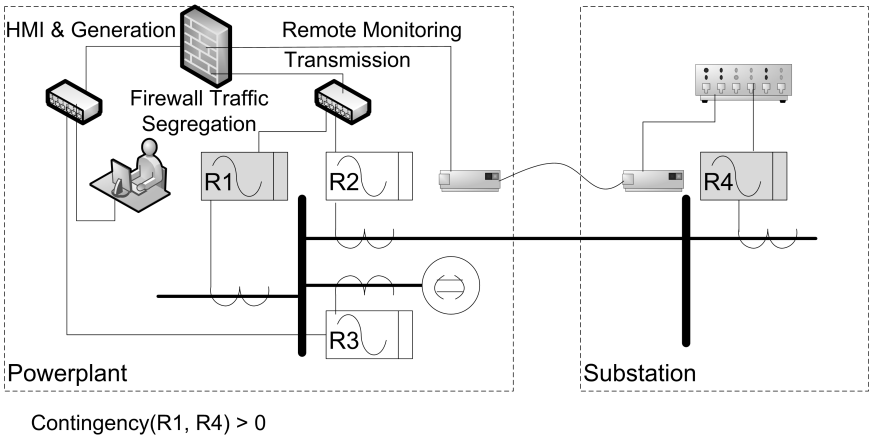

Figure 6.2: Intersubstation Traffic Segregation using Firewalls

6.4.3 Intersubstation Traffic Encryption via Link Encryption

Security best-practice guides recommend encrypting traffic when sending control messages over a WAN connection. Since most relays do not support encryption schemes, special devices may be installed next to modems to encrypt all outgoing traffic. A serial encrypting transceiver [55] is an example of such a device which acts as a 'bump in the wire' standalone cryptographic module designed to protect latency-sensitive devices. A logic function to determine where to place an encrypting transceiver would be similar to the one above shown for firewalls except that there would be one placed on either side of a WAN link to perform both encryption and decryption functions. Note that a encryption transceiver does not duplicate the functionality of a firewall in fact both devices are complementary. This fact becomes clearer in the evaluation section where we show how a composite firewall plus link encryption scheme provides more security then any one scheme used in isolation.

6.5 Implementation Costs and Attack Coverage of Security Schemes

For the purpose of the security analysis, the result of the implementation of the security schemes in a substation is the determination of attack coverage reduction and implementation cost. We determine implementation cost of a scheme to be the sum of the cost of security control devices used in implementing that scheme. For instance the cost of implementing VLANs in a substation s_j would use the predicate 6.13 to determine the number of switches needed.

For a scheme l_i implemented on substation s_j, predicate 6.17 gives a set of relay pairs V that are vulnerable to attack because they cause contingencies and $V' = attack_{l_i}(s_j) \cap vuldevpairs(s_j)$ gives the pairs that are not protected

by the security scheme.

$$vuldevpairs(s_j) = \{(r_x, r_y) \in attack(s_j) | conting(r_x, r_y) \neq \emptyset\}; \tag{6.17}$$

Since the adversary will always try to exploit the contingencies in order of greatest to least damage, we can sort the relay pairs $R_1, R_2, ..., R_n$ where $R_1, R_2, R_n \subset R$ returned by predicate 6.17 according to loss damage (predicate 6.10) $loss(D_1), loss(D_2), ..., loss(D_n)$ where $D_1 = conting(R_1)$, $D_2 = conting(R_2)$, and $D_n = conting(R_n)$. We can associate to every contingency D_i a parameter $\alpha_i \in \mathbb{R}^+_{\leq 1}$ that represents the scheme's inability of preventing the exploiting of the contingency. Then the attack coverage a_{ij} is given by equation 6.18.

$$\begin{aligned} a_{i,j} =& \alpha_1 \cdot loss(D_1) + (1-\alpha_1) \cdot \alpha_2 \cdot loss(D_2) + (1-\alpha_1) \cdot (1-\alpha_2) \cdot \alpha_3 \cdot loss(D_3) \\ &+ \ldots + (1-\alpha_1) \cdots (1-\alpha_{n-1}) \cdot \alpha_n \cdot loss(D_n) \end{aligned} . \tag{6.18}$$

Generally the set of relay pairs V' would have an $\alpha = 1$ indicating that an attack can exploit this contingency successfully every time, while the set $V \setminus V'$ would have a lower α value. A value of 0 indicates that the scheme completely protects against attacks. The computation of a_{ij} can be considered as a probability that the attacker is able to exploit the given contingency during the attack. The attacker tries to exploit the contingency associated with the maximum coverage. The success of this action is determined by the probability α_1. In case of failure (due to network access denied by a security scheme), the attacker would try to exploit the next contingency with the maximum loss and so on, until one exploitable contingency is found.

6.6 Optimal Security Hardening Algorithm

Given a set of substations and a set of independent strategies each with its unique implementation cost and coverage against malicious attacks the budget problem is to search for the optimal combination of strategies to apply at each individual substation so as to maximize the overall network security while remaining within a fixed budget.

6.6.1 Reduction from Multiple-Choice Knapsack

Assuming that only one strategy can be applied at each substation it is easy to see that enumerating all possibilities is an NP-hard problem. The proof lies in a straightforward reduction from the Multiple-Choice 0-1 Knapsack problem (MCKS). The goal of a general 0-1 Knapsack problem is to select a set of objects,

each with an associated weight and a revenue, such that the sum of the weight is below a predefined bound and the total revenue is maximized. In the Multiple - Choice variation of this problem, the objects are partitioned into n groups, and only a single object can be chosen from each group. This problem can be reduced to our formulation by considering each object to be a different security scheme and each substation to be one of the groups in which the objects are partitioned. The weight of each object becomes the cost of the implementation of the security scheme and the revenue is the opposite of the attack coverage (i.e., such that the maximization of the revenue can be expressed as a minimization of the attack coverage). By selecting the security schemes that minimize the attack coverage under a specified budget, we are solving the general MCKS problem.

The formulation of the optimal security hardening problem is defined as follow. Given the schemes $l_1, \ldots, l_m$ and given a set of substations $s_1, \ldots, s_n$, we define a variable $x_{ij} \in \{0, 1\}$ to be equal to 1 if the security scheme l_i is applied to the substation s_j, 0 otherwise. Each security scheme l_i, when applied to a substation s_j, has an associated implementation cost c_{ij} and an associated attack coverage a_{ij}. The budget allocated for security hardening is expressed as *budget*. The problem can be expressed as in equation 6.19.

$$\begin{aligned} & & \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} \cdot x_{ij} \\ & & & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \cdot x_{ij} \leq budget \\ \text{For } x_{ij} \in \{0,1\} \text{ where} \quad & \forall j \quad & & \sum_{i=1}^{m} x_{ij} = 1 \end{aligned} \tag{6.19}$$

6.6.2 Dynamic Programming Solution

However by assuming that the *principal of optimality* holds i.e. the optimal security strategy decided at a particular substation only depends upon the budget spent so far and is independent of all previous strategy decisions at other substations, a recursive dynamic programming solution can be formulated.

We divide the recursive solution into multiple states x_j where j denotes the substation number currently under consideration in that state and x is the remaining budget. Similarly $a(l_j)$ denotes the attack coverage for strategy l_j, and $c(l_j)$ the corresponding cost. If $f_j(x_j)$ is final attack coverage for the state x_j then we have the dynamic programming solution given by the recurrence relations in equation 6.20.

$$\begin{aligned} f_1(x) &= \min_{l_1 : c(l_1) \leq x_1} \{a(l_1)\} \\ f_j(x_j) &= \min_{l_j : c(l_j) \leq x_j} \{a(l_j) + f_{j-1}(x_j - c(l_j))\} \text{ for } j > 1 \end{aligned} \tag{6.20}$$

The first equation in 6.20 is the base case for one substation which returns the security scheme with the minimum attack coverage whose cost is below the budget. The second equation depicts the forward recursion returning the minimum $a(l_j)$ of the current state plus the minimum of the last state. It is easy to see that the $f_j(x_j)$s can be stored in a table (or *memorized* in logical programming as will become apparent in our Prolog implementation later) preventing a state explosion and allowing a polynomial time evaluation.

6.7 Implementation Details

This section describes how the proposed metrics for the cost-benefit security analysis were easily incorporated in the Logic-based tool chain described in Chapter 4.

6.7.1 Contingency Analysis

Various power-flow simulation software both commercial e.g. Powerworld[48] and opensource e.g. InterPSS[65] exist that allow contingency analysis of power networks. A contingency analysis via a power-flow simulation software will take out of service each device, one-by-one, resolve the power flow, and then check that no *violations* have occurred e.g no lines have exceeded their rated capacity. Industry planning and operating criteria often refer to the $n-1$ rule, which holds that a system must operate in a stable and secure manner following any single transmission or generation outage. We scripted the Powerworld software to do an $n-1$ and an $n-2$ contingency analysis on each substation and its associated control center in a power network schematic essentially returning the tuples $(contingentdevices, violations)$ to be stored in the Prolog knowledge base. Predicate 6.9 essentially searches through this table for its solution.

6.7.2 Security Analysis Implementation as Logical Rules

This section describes how the various logic predicates are implemented as Prolog rules using two representative examples that of VLAN security scheme selection (Eq. 6.13) and the max-security fixed budget optimization problem (Eq. 6.20.

Device to VLAN Assignment

The cost of implementation of this scheme varies depending on the number of VLANs used to secure the substation: high number of VLANs requires more equipment and higher set up costs. For this reason, it is necessary to provide an estimate of the number of VLANs needed. In order to minimize the number of VLANs, the scheme proposes to segregate only devices that, if exploited together, can create a violation. This problem can be mapped into a graph

Algorithm 2 Greedy algorithm for assigning VLANs to devices

```
/* At the start, all nodes do not have a label associated with them */
l(n) = -1, ∀n
/* Start the algorithm by assigning label = 1 */
c = 1
Order nodes in non-increasing degree order
/* until all nodes have labels */
while ∃n : l(n) == −1 do
  for each node n do
    if l(n) == −1 and ∀ neighbor nodes j, l(j) ≠ c then
      Assign c to n
    end if
  end for
  c = c + 1
end while
```

coloring problem. In this model, each device is a node in the graph and edges are created according to the contingency analysis: if two devices can be used together to create a violation, then an edge exists between the them. The goal of this problem is to find the minimum number of labels (i.e., VLANs) that we need to assign to each node such that two connected nodes do not share the same label. For determining the solution to this problem, we used a greedy algorithm for graph coloring, shown in Figure 2.

Optimal Selection of Security Schemes

Table 6.1 describes part of the implementation of the optimal security scheme selection algorithm described previously. Note that some of the predicates and attributes have been taken out for brevity. We describe the listing bottom up. Line 29 is the base case and line 34 the recursive case of equations 6.20. They take as arguments the state J and the total budget constaint X_j and return the overall Loss in *Result* and a list representing the schemes applied at each substation. The *mklist* rule is called (line 37) in the recursive case to merge the list of current attack coverages with that of the most optimal results in the last state subject to the budget constaint. Lines 16 and 22 detail the implementation of the *mklist* rule, the former dealing with the boundary condition of the budget running out while the latter deals with the regular case. Once the attack coverages list has been completely processed the *mklist* predicate on line 14 unifies the under process lists *Acc* and *Acc3* with the result *MergeAj* and *SLst*. Note the memorization function used in line 24 and declared on line 2 that allows computed solutions to be stored in a look-up table enabling the dynamic programming optimization.

Table 6.1: Optimal Security Scheme Selection: Prolog Implementation

```
:- dynamic stored/1.
memo(Goal) :-
  ( stored(Goal) -> true ;
    Goal, assertz(stored(Goal)) ).

% cmin(Aj,Cj,Result,Xj,Index)
% returns min of List Aj and its
% index s.t. corresponding
% Cj[index] < Xj

% substation(ID,List_of_Scheme_costs,
% List_of_scheme_attck_covgs)

mklst(_,[],[],_, L2, L2, L3, SLst) :- !.

mklst(J,[CH|CT],[_|AT],Xj,Acc,MergeAj,Acc3,SLst):-
 Y is Xj-CH, Y < 0,
 append(Acc,[-1], List1),
 append(Acc3,[[0]], List2),
 mklst(J,CT,AT,Xj,List1,MergeAj,List2, SLst).

mklst(J,[CH|CT],[AH|AT],Xj,Acc,MergeAj,Acc3,SLst):-
 J1 is J-1, Y is Xj-CH, Y > -1,
 memo(f(J1,Y,Fjminus1,PrevSchemeLst)),
 NewH is AH+Fjminus1,
 append(Acc,[NewH], List1),
 append(Acc3,[PrevSchemeLst], List2),
 mklst(J,CT,AT,Xj,List1,MergeAj,List2, SLst).

f(1,X1,Result,[Index]) :-
 substation(1,C1,A1),
 cmin(A1,C1,Result,X1,Index).


f(J,Xj,Result,[Index|Scheme]) :-
 J > 1,
 substation(J,Cj,Aj),
 mklst(J,Cj,Aj,Xj,[],MergeAj,[],SLst),
 cmin(MergeAj,Cj,Result,Xj,Index),
 element_at(Scheme, SLst, F).
```

.

7 Case Study: Optimal Security Hardening of the 118-Bus

The IEEE 118 Bus test case was used to test and analyze the optimal security hardening problem formulation. Data for the IEEE 118 bus test case representing a portion of the American Electric Power System in the Midwestern US, was downloaded from the University of Washington Power System [62]. The system consists of 118 buses, 186 transmission elements, 19 committed generators with a total capacity of 5,859 MW, and 99 load buses with a total load of 4,519 MW. The complete power flow simulation along with the line limits that we used can be found at the Powerworld website [19]. Since our threat model assumes the adversary has two resources, we did a N-1 and an N-2 contingency analysis. While our tests were run on the entire network whose results are presented at the end, we initially walk the reader through just a portion of the analysis (south-west part of the network) shown in figure 7.1 for easy understanding.

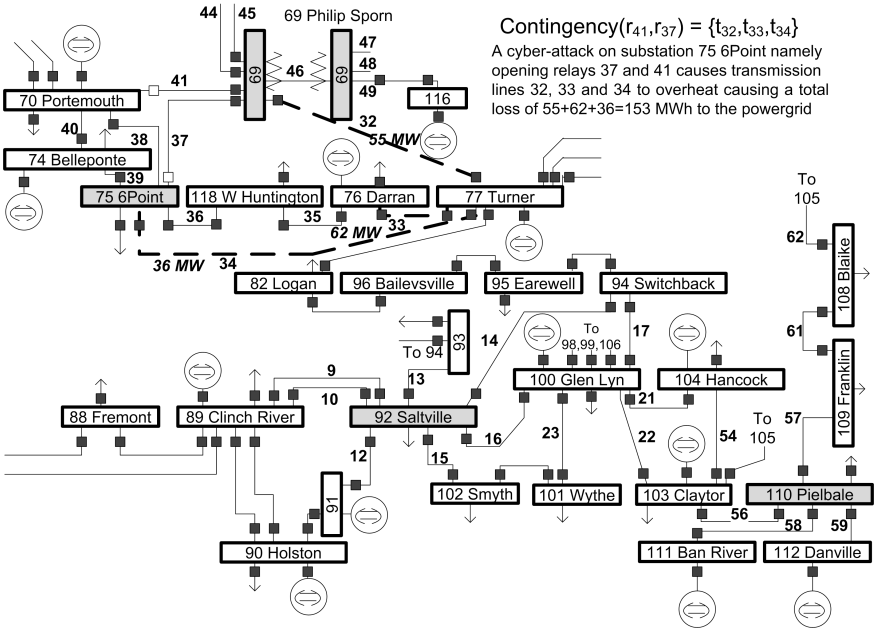

Figure 7.1: Contingency Analysis of A Portion of the 118-Bus Test Case

7.1 Security Schemes Employed

Suitable candidate devices were picked for each of the security strategies identified. We use the label *VLAN* to indicate the intra-substation traffic segregation via Virtual LANs scheme, the label *FWALL* to indicate the intersubstation traffic segregation via firewall scheme, and *FLINK* for the composite scheme of firewall and link encryption combined. Table 7.1 gives a descriptive summary of each of security schemes applied along with its coverage and an estimate of individual device cost required to implement each one. In the case of composite strategies, the devices shared among the two security solutions are counted only once. For example, in the case of the *VLAN* + *FWALL* strategy, the CISCO 1760 router can be used, at the same time, for implementing both the *VLAN* and the *FWALL* strategies. Hence, its cost is counted only once.

Table 7.1: A Cost-Benefit Comparison of Security Schemes

Security Scheme	**Scheme *VLAN***	**Scheme *FWALL***	**Scheme *FLINK***
Description	Intra-substation traffic segregation using VLANs	Inter-substation traffic segregation using firewalls	Firewall and link encryption across substation boundary
Mitigated Threats	Traffic Manipulation	Traffic Manipulation	Traffic Manipulation + Masquerade Attacks
Typical Deployment Budget	•Cisco 1760 router $1k •Cisco Catalyst switch 2950 each VLAN ($500)	•Cisco 1760 router $1k •Stateful Firewall Integrated Router ($200)	•Cisco 1760 router $1k •Stateful Firewall Integrated Router ($200) •Serial Transceiver SEL-3021 ($540)

Once the security strategies have been defined, the first step of the analysis is determining the attack coverages and overall loss for each substation. This process is split into two parts. In the first part of the analysis each contingency is analyzed to determine the degree of protection that a given security scheme provides. During this process, each security scheme associates a value of α to each contingency. In the second phase of the analysis, the substation attack coverage is computed using Eq 6.18.

7.2 Case Study Results

Tables 7.2 (a), (b), (c) and (d) show the results of the analysis. In each table, the devices that are part of each contingency are shown on the first column, *Contingency*. The second column, *Loss*, reports an estimation of the loss caused by an exploitation of the associated contingency, as defined in Section 6.2. In this subset of the power grid, all violations affect lines. In the estimation of the loss we assume an unmet demand cost of $1000 per hour and a time to repair of 10 hours for all violations. Different choices of coefficients would affect the losses of all contingencies in the same way, hence not affecting the final results.

Coefficients would have been different for violations that affect transformers, but the overall analysis procedure would not change. The attack coverage is expressed in ten thousand dollar units. The last columns of each table report the degree of exploitability of each contingency using different security schemes.

The results for the analysis of substation 92 Saltville are reported in Table 7.2a. For each security scheme, each consistency has been analyzed using the logic rules described in Section 6.4. We consider an α value of 1 for contingencies that are not protected by the security scheme, a value of 0.5 if protected by the *VLAN* or *FWALL* scheme. A value of 0 is assigned if the contingency is protected by the *FLINK*: this security scheme provides more protection than *FWALL* alone as it protects against both traffic manipulation and masquerade attacks.

In substation 92 Saltville, the security scheme *VLAN* does not protect against the contingency with the highest loss. Hence, the overall attack coverage for the *VLAN* strategy is \$1,777,000, as if no security scheme were used. Security scheme *FWALL* partially protects against the contingencies with more losses, and we obtain an overall attack coverage of \$1,502,000. Security schemes $VLAN + FWALL$ and *FLINK* and $VLAN + FLINK$ improve the protection, leading to lower overall attack coverages.

Table 7.2 (a) also reports the cost of applying each strategy and the substation attack coverage, obtained using Eq 6.18.

The same process has been applied to substations 69 Philip Sporn, 75 6Point, 92 Saltville and 110 Pielbale (the greyed substations shown in Figure 7.1). Results of this analysis are reported in Tables 7.2 (b),(c) and (d).

Given the values of Tables 7.2 (a), (b), (c) and (d) and a finite budget, the optimal security hardening algorithm finds the optimal assignment of security schemes to substations that minimizes the overall attack coverage.

Without the application of any security scheme, the overall attack coverage is \$5,964,000. As an example, the results for a budget of \$7000 are elaborated in Table 7.3. The symbol $'-'$ represents the choice of not implementing a security scheme.

As Table 7.3 shows, the optimal allocation of the budget (method A) is to implement scheme $VLAN + FLINK$ in substation 75 6Point, scheme *FLINK* in substation 92 Saltville and scheme *VLAN* in substation 110 Pielbale. No security scheme has been applied to substation 69 Philip Sporn. This solution has an overall attack coverage of \$3,681,000, the minimum value obtainable using this budget. The fact that no security scheme has been applied to one of the substation might seem counterintuitive, however this is a consequence of having a finite budget. If we were to apply any security scheme to substation 69, the cost of the solution would go over the budget. On the other hand, if we define a new security scheme assignment method (method C) where some security scheme must be applied to each substation, we obtain the solution reported in Table 7.4. The overall attack coverage of this solution is \$4,073,000,

Contingency	Loss [$10k]	*VLAN*	*FWALL*	*VLAN + FWALL*	*FLINK*	*VLAN + FLINK*
6,13	112.5	1	0.5	0.5	0	0
6,14	177.5	1	0.5	0.5	0	0
6,15	128	1	0.5	0.5	0	0
9,10	100.9	0.5	1	0.5	1	0.5
13,16	57.3	0.5	1	0.5	1	0.5
13,15	57.3	0.5	1	0.5	1	0.5
14,16	62.8	0.5	1	0.5	1	0.5
14,15	112.5	0.5	1	0.5	1	0.5
6	65.2	1	0.5	0.5	0	0
Scheme cost		$2000	$1200	$2200	$1740	$2740
Overall Attack Coverage		177.7	150.2	148.2	112.5	94.7

(a) Substation 92 Saltville

Contingency	Loss [$10k]	*VLAN*	*FWALL*	*VLAN + FWALL*	*FLINK*	*VLAN + FLINK*
41,37	153.8	0.5	1	0.5	1	0.5
46	94.4	1	1	1	1	1
26	52.5	1	0.5	0.5	0	0
32,46	7.8	0.5	1	0.5	1	0.5
48,46	7.8	0.5	1	0.5	1	0.5
Scheme cost		$2000	$1200	$2200	$1740	$2740
Overall Attack coverage		124.1	153.8	124.1	153.8	124.1

(b) Substation 69 Philip Sporn

Contingency	Loss [$10k]	*VLAN*	*FWALL*	*VLAN + FWALL*	*FLINK*	*VLAN + FLINK*
41,37	153.8	1	0.5	0.5	0	0
36	62.4	1	0.5	0.5	0	0
37,34	62.4	0.5	1	0.5	1	0.5
41,38	51.9	1	0.5	0.5	0	0
Scheme cost		$2000	$1200	$2200	$1740	$2740
Overall Attack coverage		153.8	108.1	103.5	62.4	31.2

(c) Substation 75 6Point

Contingency	Loss [$10k]	*VLAN*	*FWALL*	*VLAN + FWALL*	*FLINK*	*VLAN + FLINK*
56,58	111.3	0.5	1	0.5	1	0.5
57,58	59.7	0.5	1	0.5	1	0.5
Scheme cost		$ 2000	$1200	$2200	$1740	$2740
Overall Attack coverage		70.6	111.3	70.6	111.3	70.6

(d) Substation 110 Pielbale

Table 7.2: Attack Coverage Quantification of Security Schemes

Substation	Selected scheme	Overall Attack Coverage [$10k]	Scheme Cost Total Budget $7k
69 Philip Sporn	-	153.8	$0
75 6Point	*VLAN + FLINK*	31.2	$ 2740
92 Saltville	*FLINK*	112.5	$1740
110 Pielbale	*VLAN*	70.6	$2000

Table 7.3: Method A: Optimal Assignment of Security Schemes

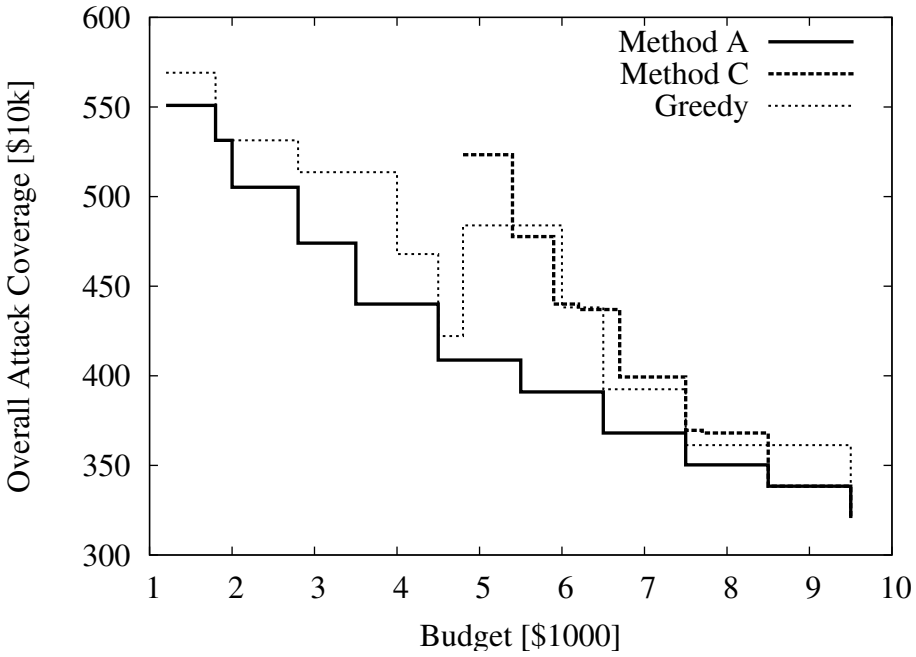

Figure 7.2: Cost-Benefit Comparison of Scheme Selection Methodologies.

which is higher than our original solution.

Substation	Selected scheme	Overall Attack Coverage [$10k]	Scheme Cost Total Budget $7k
69 Philip Sporn	*VLAN*	124.1	$2000
75 6Point	*FLINK*	62.4	$1740
92 Saltville	*FWALL*	150.2	$1200
110 Pielbale	*VLAN*	70.6	$2000

Table 7.4: Method C: Optimal Complete Assignment of Security Schemes

Given no constraints over the budget, the best allocation of security schemes is reported in Table 7.5. We call this security scheme allocation method B and, in this case, this allocation is possible only with a budget of $9480.

Figure 7.2 reports the total attack coverage for all substations when different budgets are allocated for security hardening. Method C results start only after a budget of $4800: before that value there is no assignment that provides a security scheme to each substation. It can be seen that method C provides an higher overall attack coverage for the same budget. A and C are also compared to a greedy scheme selection methodology, namely making the best local decisions at each state without considering the previously computed results. It is clear from the traces that not only does this scheme have an overall higher attack coverage than the rest e.g. $3925k for a budget of $7000 but it also sometimes ends up making incorrect decisions so that for an increase in budget the attack coverage increases as well. This is apparent from the spike at a budget

Substation	Selected scheme	Substation Attack Coverage [$10k]	Scheme Cost
69 Philip Sporn	*VLAN*	124.1	$2000
75 6Point	*VLAN + FLINK*	31.2	$2740
92 Saltville	*VLAN + FLINK*	94.7	$2740
110 Pielbale	*VLAN*	70.6	$2000

Table 7.5: Method B: Unlimited Budget Assignment of Security Schemes.

of $4,800 to $5,900 resulting from a lucrative local decision that doesn't leave enough revenue for reasonably securing other states.

Security Schemes	Substation Deployment	Attack Coverage [$10k]	Cost
VLAN	18	2436.1	$16000
FWALL	0	0	$0
VLAN+FWALL	22	2546.7	$43360
FLINK	14	654.2	$15920
VLAN+FLINK	28	3322.8	$49580

Table 7.6: Security Hardening Results of the Entire 118-Bus

The results of the security hardening exercise of the entire 118-Bus system are shown in table 7.6. We wanted to see the total protection that could be achieved without any budget constraints, hence the reason for FLINK scheme always being the preferred over FWALL. 82 total substations were secured with a overall cost of $124,860 and a total attack coverage of $89,598,000.

8 Conclusion and Future Work

8.1 Conclusion

In this work we propose a security model that incorporates descriptions of the SCADA infrastructure and its workflow activities. Using Logic-based models of security schemes we can evaluate the compliance of the infrastructure to security best-practices. In addition by extending existing techniques for scalable attack graph generation we evaluate risks and give advisories on which workflows are safer than others based on a cost-lattice. We implemented a tool-chain that automates most of the process of generating our models from CIM specifications. Moreover the tool chain updates configuration information dynamically from an event aggregator allowing our security model to give accurate results.

Besides security assessment, we also addressed the Power Grid security administrators' dilemma, namely, how to select, an optimal combination of security hardening schemes from a set of choices so that not only the total attack coverage is minimized but also the total installation cost remains within a certain budget. The security-budget optimization was formulated as a dynamic programming knapsack problem allowing it to be solvable in pseudo-polynomial time. The tool-chain was evaluated on the IEEE 118-bus test case with five different security schemes.

8.2 Future Work

In the future the scalability of this approach can be evaluated by using bigger models and seeing the impact on the model checking time. Currently actions are limited to security properties associated with devices. A more realistic model would be to incorporate side effects of actions and their impact on the state of the system. That model could potentially allow feedback modelling (e.g. via SMTP monitoring software) of the impact of firing certain actions in a workflow and recalculation of the new security risks. The feedback mechanism would allow detection of changes in not only the status of devices but also of security provisions. Finally the cost-lattice could be expanded to include more properties of interest.

Our model for estimating cost of installing a security scheme is a little simplistic taking into account only the total monetary value of the security devices required. In reality there may be a lot more cost factors involved such as that

of installation, system downtime, incompatibility, training and effort. In our case study, our threat model only covered cyber attacks on relays, because the IEEE Power System Test case archive did not contain any other control device information. In actuality there are a host of cyber devices e.g. PLCs, RTUs, Data Historians etc. in a power network that control power assets in a variety of ways. We believe our model can encompass these devices but more detailed networks are needed for evaluation. The assignment of degree of protection to the various security schemes for the 118-bus test case is course grained based only on the type of attacks the scheme protects. For Power Grid schematics with more detailed control network information this calculation would involve more complicated attack trees.

Finally modelling distributed attacks that exploit temporal dependencies, would enable the tool-chain to evaluate possibilities of cascading failures akin to the series of timed electric surges seen across the Power Grid during the 2003 blackout[38].

References

[1] The Foundation for Scientific and Industrial Research at the Norwegian Institute of Technology. *www.sintef.no/*.

[2] British Columbia Institute of Technology Industrial Security Incident Database. *http://www.bcit.ca /appliedresearch/ security/*, 2001.

[3] RFC 1157. Simple network management protocol (snmp).

[4] IEC 61850-SER. IEC 61850 Communication networks and systems in substations parts. *http://www.61850.com*, 2004.

[5] American Gas Association (AGA). Cryptographic Protection of SCADA Communications, Part 1: Background, Policies and Test Plan. *www.aga.org*, March 2006.

[6] Z. Anwar and R. H. Campbell. Automated Assessment of Critical Infrastructures for Compliance to Best Practices. *IFIP WG 11.10 International Conference on Critical Infrastructure Protection*, March 2008.

[7] Z. Anwar, M. Montanari, A. Gutierrez, and R. H. Campbell. Budget Constrained Optimal Security Hardening of Control Networks for Critical Cyber-Infrastructures. *International Journal of Critical Infrastructure Protection*, 2008.

[8] Z. Anwar, R. Shankesi, and R. H. Campbell. Automatic Security Assessment of Large-Scale Cyber-Infrastructures. *International Conference on Dependable Systems and Networks*, June 2008.

[9] B. Axel, R. Fredriksen, and A. Thunem. *An Approach for Model-based risk management*. Springer LNCS, 2004.

[10] P. Borovanský, C. Kirchner, H. Kirchner, and P.-E. Moreau. ELAN From a Rewriting Logic Point of View. *Theoretical Computer Science*, 285:155–185, 2002.

[11] E.J. Byres, M. Franz, and D. Miller. The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems. *International Infrastructure Survivability Workshop*, 2004.

[12] Edmund .M. Clarke, Orna. Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 2001.

[13] M. Clavel, F. Durán, S. Eker, J. Meseguer, P. Lincoln, N. Martí-Oliet, and C. Talcott. *All About Maude – A High-Performance Logical Framework*. Springer LNCS Vol. 4350, 2007.

[14] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and Carolyn Talcott. The Maude 2.0 System. In *Rewriting Techniques and Applications*, LNCS, pages 76–87. Springer-Verlag, June 2003.

[15] Federal Energy Regulatory Commission. Mandatory Reliability Standards for Critical Infrastructure Protection. *Docket No. RM06-22-000; Order No. 706 http://ferc.gov/whats-new/comm-meet/2008/011708/E-2.pdf*, January 2008.

[16] United States Nuclear Regulatory Commission. Potential Vulnerability of Plant Computer Network to Worm Infection. *NRC Information Notice 2003-14*, 2003.

[17] R. F. Dacey. Critical Infrastructure Protection: Challenges in Securing Control Systems. 2004. GAO-04-140T.

[18] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. *Proceedings of the 14th ACM conference on Computer and communications security*, 2007.

[19] POWERWORLD Educator. The Visual Approach to Analyzing Power Systems. *http://www.powerworld.com/cases/118bus.zip*, 2007.

[20] Steven Eker, José Meseguer, and Ambarish Sridharanarayanan. The Maude LTL model checker. In F. Gadducci and U. Montanari, editors, *Proc. 4th. Intl. Workshop on Rewriting Logic and its Applications*. ENTCS, Elsevier, 2002.

[21] D. L. Evans, P. J. Bond, and A. L. Bement. Standards for Security Categorization of Federal Information and Information Systems. Feburary 2004. FIPSPUB199.

[22] D. L. Evans, P. J. Bond, and A. L. Bement. Standards for Security Categorization of Federal Information and Information Systems,. *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology*, september 2006.

[23] K. Futatsugi and R. Diaconescu. *CafeOBJ Report*. World Scientific, AMAST Series, 1998.

[24] Joseph Goguen and José Meseguer. Order-Sorted Algebra I: Equational Deduction for Multiple Inheritance, Overloading, Exceptions and Partial Operations. *Theoretical Computer Science*, 105(2):217–273, 1992.

[25] Mark Grimes. SCADA Exposed. *TOORCON*, September 2005.

[26] Juliana Gruenwald. Power grid vulnerable to cyberattacks, committee told. *CongressDaily*, September 2008. http:// www.govexec.com/ story_page.cfm? articleid= 40940.

[27] J. D. Guttman and A. L. Herzog. Rigorous automated network security. *International Journal for Information Security*, 2004.

[28] Dave Hendricks. Fear About Cyberattacks Against Power Grid Prompts Proposal for New Regulation. *InfoZine*, September 2008. http://www.infozine.com/news/stories/op/storiesView/sid/30602/.

[29] Industrial Automation Open Networking Association (IAONA). Draft/RFC v0.4. *The IAONA Handbook for Network Security*, 2003.

[30] Distributed Management Task Force, Common Information Model (CIM). *DSP 0004- CIM Infrastructure Specification 2.14*, October 2005.

[31] Systems Instrumentation and Automation Society (ISA). Security Technologies for Manufacturing and Control Systems. March 2004. ISATR99.00.012004.

[32] R. L. Krutz. *Securing SCADA Systems*. WILEY, 2006.

[33] J. Lewis and T. Domin. Protective Relaying: Principles and Applications. 2006.

[34] D. Maynor and R. Graham. SCADA Security and Terrorism: We're Not Crying Wolf! *Black Hat*, 2006.

[35] Miles A. McQueen, Wayne F. Boyer, Mark A. Flynn, and George A. Beitel. Quantitative Cyber Risk Reduction Estimation Methodology for a Small SCADA Control System. *39th Annual Hawaii International Conference on System Sciences*, 2006.

[36] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theor. Comput. Sci.*, 96(1):73–155, 1992.

[37] J. Meserve. Sources: Staged Cyber Attack Reveals Vulnerability in Power Grid. *CNN Article*, September 2007. http:// www.cnn.com/ 2007/ US/ 09/ 26/ power.at.risk/.

[38] H. Michael and D. Klaidman. Blackout 2003: What Went Wrong. *Newsweek 25 Aug. 2003*, 2004.

[39] E. Nakashima and S. Mufson. Hackers Have Attacked Foreign Utilities. *Washington Post*, January 2008.

[40] Electric Energy Industy News. NERC Cyber Security Standards to Become Mandatory in United States. *http:// www. electricenergyonline. com/ IndustryNews*, 2008.

[41] Steven Noel and Sushil Jajodia. Attack Graphs for Sensor Placement, Alert Prioritization, and Attack Response. *Cyberspace Research Workshop (AirForce Cyberspace Symposium)*, 2007.

[42] Steven Noel, Eric Robertson, and Sushil Jajodia. Correlating Intrusion Events and Building Attack Scenarios through Attack Graph Distances. *Computer Security Applications*, 2004.

[43] British Columbia Institute of Technology (BCIT). NISCC Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks. February 2005.

[44] R. Ortalo, Y. Deswarte, and M. Kaaniche. Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security. *IEEE Trans. Software Eng*, 25(5): 633-650, 1999.

[45] Physical Vulnerability of Electric Systems to Natural Disasters and Sabotage. *OTA-E-453*, 1990.

[46] X. Ou, W. F. Boyer, and M. A. McQueen. A Scalable Approach to Attack Graph Generation. *13th ACM conference on Computer and communications security*, 2006.

[47] C. Phillips and L. Swiler. A Graph-based System for Network-Vulnerability Analysis. *New Security Paradigms Workshop*, 1998.

[48] POWER WORLD The Visual Approach to Analyzing Power Systems. *http://www.powerworld.com/*, July 2007.

[49] ATT Research. Graphviz - Graph Visualization Software. *http:// www.graphviz.org/*, 2006.

[50] A. Risley and K. Carson. Low or No-Cost CyberSecurity Solutions for Defending the Electric Power System Against Electronic Intrusions. *Schweitzer Engineering Laboratories, Inc.*, 2008.

[51] R. Ross, M. Swanson, G. Stoneburner, S. Katzke, and A. Johnson. Recommended Security Controls for Federal Information Systems. Feburary 2005. NIST Special Publication 800-53.

[52] R. Ross, M. Swanson, G. Stoneburner, S. Katzke, and A. Johnson. Guide for Assessing the Security Controls in Federal Information Systems. June 2007. NIST Special Publication 800-53A.

[53] J. Salmeron, K. Wood, and R. Baldick. Analysis of Electric Grid Security Under Terrorist Threat. *IEEE Transactions On Power Systems, Vol. 19, No. 2*, 2004.

[54] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright, and S. Romanosky. CVSS: A Common Vulnerability Scoring System. *http:// www.first.org /cvss/ cvss-guide.html*, 2007.

[55] Inc Schweitzer Engineering Laboratories. SEL-3021: Serial Encrypting Transceiver Security Policy. *https:// www.ee.washington.edu/ research/ pstca/ index.html*, 2008.

[56] Homeland Security. CS^2SAT Control System Cyber Security Self-Assessment tool. *http:// www.us-cert.gov/ control_systems/ pdf/ CS2SAT.pdf*, 2008.

[57] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated Generation and Analysis of Attack Graphs. *IEEE Symposium on Security and Privacy*, 2002.

[58] Stephanou and Tony. Assessing and exploiting the internal security of an organization. *The SANS Institute*, 2001.

[59] K. Stouffer, J. Falco, and K. Kent. Guide to Supervisory Control and Data Acquisition (SCADA) and Industrial Control Systems Security. *NIST Special Publication 800-82*, september 2006.

[60] United States Computer Emergency Readiness Team. Us-cert vulnerability note field descriptions. *http://www.kb.cert.org/vuls/html/fieldhelp*, 2007.

[61] C. W Ten, C. C. Liu, and M. Govindarasu. Vulnerability Assessment of Cybersecurity for SCADA Systems Using Attack Trees. *Power Engineering Society General Meeting*, July 2007.

[62] Dept of Electrical Engineering University of Washington. Power System Test Case Archive. *https:// www.ee.washington.edu/ research/ pstca/ index.html*, 2007.

[63] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: Yet Another Workflow Language. *Inf. Syst.*, 30(4):245–275, 2005.

[64] QUT BPM Research group and Eindhoven University's YAWL Editor. *http://www.yawl-system.com/*, 2007.

[65] M. Zhou and S. Zhou. Internet, Open-source and Power System Simulation. *IEEE Power Engineering Society General Meeting*, June 2007.

Author's Biography

Zahid Anwar was born in Rawalpindi, Pakistan in 1979. He graduated on the Dean's honor role from Ghulam Ishaq Khan Institute of Engineering Sciences and Technology in 2001 with Bachelors in Computer Systems Engineering. After working for a couple of years in the software industry, Zahid rejoined academia and completed his graduate studies in Computer Science at the University of Illinois at Urbana-Champaign, receiving the M.S. and Ph.D. degrees in August 2005 and December 2008, respectively. During the course of his graduate studies, Zahid received several research honors including a Sarah and Sohaib Abbasi Graduate Fellowship from the Department of Computer Science, an Intel Corporate Services Award, and a Teachers and Researchers Fellowship from the Ministry of Science, Government of Pakistan. Zahid has several successful publications in leading ACM and IEEE International Conferences on network security and dependability such as DSN and MMCN. He also received the Graduate Teaching Certificate award from the University of Illinois at Urbana-Champaign. In the final years of his PhD, Zahid got a chance to exchange ideas and apply his research while working for some of the largest leading research labs such as IBM T.J Watson Research, NY, Intel Research Labs, OR, Motorola Labs Wireless Center of Excellence, Schaumburg, IL and the National Center for Supercomputing Applications (NCSA), Urbana, IL.

Zahid's research interests lie in the fields of the computer security, networks, wireless and distributed systems. He is particularly interested in formal methods of automated security assessment for large networks and protection of real-time protocols such as Voice over IP.