Analyzing the Security and Survivability of Real-time Control Systems

Paul Oman, Senior Member, IEEE, Axel Krings, Senior Member, IEEE, Daniel Conte de Leon, and Jim Alves-Foss, Member, IEEE

Abstract: Many problems found in complex real-time control systems can be transformed into graph and scheduling problems, thereby inheriting a wealth of potential solutions and prior knowledge. This paper describes a transformation from a real-time control system problem into a graph theoretical formulation in order to leverage existing knowledge of graph theory back into the real world network being analyzed. We use a five-step transformation that converts an example electric power SCADA system into a graph model that allows for solutions derived from graph algorithms. Physical and logical characteristics of the SCADA system are represented within the model in a manner that permits manipulation of the network data. System vulnerabilities are identified and compared via graph transformations prior to transformation back into the real-time control system problem space. The SCADA system analysis serves as an example of exploiting graph representations and algorithms in order to encapsulate and simplify complex problems into manageable and quantifiable models.

Keywords: Security Analysis, Cyber Vulnerabilities, Cyber Attack, Critical Infrastructure Protection, SCADA Systems

I. INTRODUCTION

Every nation's critical infrastructures, such as telecommunications, finance, electric power, water supplies and transportation systems, are increasingly reliant on remotely accessible microprocessor-based controls. We now live in a global digital society where day-to-day operations are optimized by complex real-time control systems. Electricity generation and load is adjusted by time-of-day parameters, as are water and sewage controls; large financial transactions are predominately electronic; air transportation is optimized by digital bookings and load information, and controlled by electronic air-traffic control systems; and our surface transportation infrastructure has evolved to a level of complexity where intelligent transportation systems are essential. Thus, our critical infrastructures have become highly complex real-time control systems subject to time-of-day and day-of-week influences, stresses from special events, and damage from natural disasters, cyber attack, and both physical and electronic sabotage.

The increased use of computer-based systems for Supervisory Control And Data Acquisition (SCADA) has created vulnerabilities within real-time control systems similar to those seen in traditional computer networks. In many cases the remote access is over dedicated communication networks, but several utilities have begun to use IP-based network communication for monitoring and control. In some cases these SCADA systems are implemented with connection points to public communications networks. Because of the nature of the activities controlled by these systems, misuse of SCADA devices and actuators could have disastrous consequences. Physical intruders have been known to randomly and maliciously push buttons and operate switches, and there is increasing evidence that electronic intruders are just as curious and malicious. Such attacks have already occurred in water and sewage controls, natural gas pipelines, nuclear power station controls, and electric utilities, so it is inevitable that refineries and manufacturing plants will be the target of malicious cyber attacks sometime in the future. Safeguarding the electronic access points, detecting intrusions, and rapidly isolating the attacker(s) are crucial to maintaining integrity of the control system, regardless of whether the initiating attack is mounted through public or private access points.

Much attention has been made to optimize the reliability of real-time control systems under normal conditions in both congested and non-congested regimes, but little has been done to model extreme events, contingencies, massive or cascading failures, and malicious attacks. In our research of electric power system networks we completed eleven on-site visits to power companies. Lack of security awareness can be found at all levels of the industry from developers of systems and software that control the power grid to the operators of the power control systems, the power engineers, and the utility...
executives themselves. A comparison of vulnerabilities documented in old assessment reports against those found in recent security and survivability assessments shows that the problem is increasing rather than abating. Table 1 shows a checklist of the known vulnerabilities documented in [1] that still exist and have been observed in recent assessment visitations conducted by us. It can be seen that all prior vulnerabilities still exist, and new ones, associated with emerging technologies and business needs, have come to bear.

Table 1. Power Grid Vulnerabilities

<table>
<thead>
<tr>
<th>Documented SCADA Vulnerability</th>
<th>1997 NSTAC</th>
<th>2002 Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak Passwords Used</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Default Passwords Not Changed</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Passwords Posted Visibly</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shared Logins</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Inconsistent or Non-existent Warning Banners</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Personnel Unaware of Hacking Threat</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-existent Security Policies</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unsecured Modern Access</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IT Network Interconnectivity</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Non-existent or Inadequate Intrusion Detection</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Internet Connectivity</td>
<td>Non-existent</td>
<td>✓</td>
</tr>
<tr>
<td>Wireless Networks</td>
<td>Non-existent</td>
<td>✓</td>
</tr>
<tr>
<td>Commercialization of Utility Telecomms</td>
<td>Non-existent</td>
<td>✓</td>
</tr>
</tbody>
</table>

We have also conducted security and survivability assessments of water supply and sewage treatment facilities, rail transportation systems, methane production and power generation, Internet service providers, and intelligent traffic control systems. We feel that our on-site experience with the power industry is fairly typical of utilities as a whole. By and large, they are not aware of their cyber vulnerabilities and have not implemented the rudimentary safeguards needed to protect their control systems from malicious and/or accidental cyber intrusions and denial-of-service attacks.

To date, few models exist that allow formalizing the effects of malicious acts, in either host-based or networked systems. Evaluating prior research involving the modeling of attacks and vulnerabilities within a system we found several disparate approaches. In [2, 3] the authors use attack trees, while in [4] the authors use graphs to represent the network attack space. Other approaches by [5, 6, 7] include the use of formal description languages or other mechanisms for modeling network attacks. Despite all these vulnerability models, the increasing number of intrusions and vulnerabilities evident in the CMU CERT logs suggests that we are losing ground when it comes to hardening our computer networks, much less our real-time control systems. The lack of success in securing networked computer systems may be attributed to missing theoretical groundwork and mathematical models [8]. That is, most approaches to security and survivability are ad hoc and, in the absence of standardized security test procedures, claims of security improvements cannot be replicated or verified. Furthermore, it is not possible to compare relative empirical results from security analyses, because there is no common basis for comparison.

In an attempt to increase the scientific rigor in certain real-time control problems, we explored the transformation of security and survivability real-world problems into the mathematics and computer science disciplines. Problem transformations have been used extensively in mathematics and engineering in order to solve hard problems. Well known examples include exponentiation or Laplace transformation. The general strategy is to transform the original problem into a different problem space in which known solutions exist, or solutions can be found at lesser cost. After a solution has been derived in the new problem space, a reverse transformation is used to translate the proposed solution back to the original problem space.

This paper presents a transformation to formalize real-time control system survivability and security problems into graph or scheduling theory so that graph and scheduling algorithms can be brought to bear. The transformation enables solutions to be based on graph and scheduling theoretical concepts. The next section gives an overview of the five-step transformational process. Section 3 contains a simple example using the transformation to convert a SCADA vulnerability analyses to a graph problem where Dijkstra's shortest-path algorithm is used to quantify the most vulnerable access paths. Section 4 expands the application of the transformation process by discussing examples of graph and scheduling problems. Finally, Section 5 provides a short summary and concludes the paper.
II. A TRANSFORMATIONAL MODEL

A transformational model for mapping network survivability analysis into graph or scheduling problems can be found in [9], which is partially restated in this section. In the next section (Section 3), we will apply that process to an electric power SCADA system, but first we review the basic steps for the reader’s convenience. The transformation process is shown in Figure 1. The process starts at the bottom-left (Application A) and proceeds clockwise through model generation, parameterization, representation, optimization, and reverse transformation resulting in either the same application or a revised version of the application (denoted by the ellipse to Application X). Not obvious from the figure is that the transformation is usually iterative, so researchers can explore other representations and algorithmic solutions.

![Figure 1. Transformation Overview](image)

We now describe through the five-step transformation process in clockwise progression, from Model Generation to Reverse Transformation.

A. Model Generation

The real-world application is transformed into a task graph together with the task model specification, if applicable. The general model is based on a directed graph $G = (V, E)$, where $V$ is a finite set of vertices $v_i$ and $E$ is a set of edges $e_{ij}$, with $i \neq j$, representing precedence relations between $v_i$ and $v_j$ within $V$. The most important feature of the model generation process is matching of the system’s security and survivability requirements and objectives with the graph or scheduling model’s capabilities and/or potential.

B. Parameterization

Once the application is mapped to vertices and edges of $G$, a mapping of application specific parameters to generic parameters is needed. Examples of such parameters are power transmission, network throughput, communication cost, sensitivity or confidentiality, relative importance based on the cost of loss of services, etc. The vertices and/or edges of the generated graph need to be assigned weights representing the characteristics under study. The results can be generalized by integer or real valued weights. Thus, for each vertex in $V$ and edge in $E$, vertex and edge weights are defined respectively. Let $w_{vi}$ denote the vertex weight of $v_i$ and let $w_{eij}$ denote the weight of edge $e_{ij}$. If multiple parameters need to be considered simultaneously, these scalar weights may be insufficient and multiple weights may need to be defined for vertices and/or edges. In this case $w_{vi}$ and/or $w_{eij}$ are vectors, where $w_{vi}[k]$ and $w_{eij}[l]$ represent the $k^{th}$ and $l^{th}$ parameters, respectively.

C. Model Abstraction and Representation

Once a weighted graph $G$ is defined, the graph can be considered in the context of standard graph or scheduling problems. A graph theoretical formulation can be represented by the graph itself, along with the manipulative objectives, such as max-flow or min-cut. On the other hand, a scheduling theoretical formulation requires the specification of the scheduling model, i.e., the processing environment and the optimization criteria. In order to avoid lengthy descriptions of scheduling model $S$, a compact description of the form $S = (\alpha|\beta|\gamma)$ is commonly used, where the fields $\alpha$, $\beta$, and $\gamma$ indicate the processor environment, the task and resource characteristics, and the optimization criteria, respectively. Again, the important aspect of the model abstraction and representation process is fitting or matching your security and survivability objectives within the graph or scheduling model’s capabilities and/or potential.

D. Graph and Scheduling Algorithms

Graph $G$ or schedule model $S$ are now subjected to graph and scheduling theoretical algorithms, respectively. A plethora of algorithms and heuristics of varying space and time complexity already exist. That is, because of the transformation to known graph and queuing structures, researchers are empowered with a wealth of known algorithms with which to explore, probe, and test the security and survivability characteristics of their real-time control systems. The goal is to find optimal or sub-optimal solutions for the sought-after security and survivability criteria, by applying the best suitable algorithm(s) and observing the effect on the characteristics of interest. Of course, appropriate
algorithms need to be identified that suit the optimization criteria (e.g., a specific survivability criteria), including considerations for response time, computation requirements, and/or costs. One of the desired aspect of using graph or scheduling models is that the time or space complexity may be inherited from the set of known algorithms. For example, many problems have been shown to be intractable (e.g., NP-complete or NP-hard), which may provide valuable information about the solution space for real world problems. However, it should be noted that intractability in the general case does not necessarily imply that the real-time control problem cannot be solved efficiently. In fact, for specific problems of limited size, solutions may be obtainable with efficient or acceptable cost, despite of the problem of being computationally hard. After the application of graph or scheduling algorithms or heuristics, optimal or sub-optimal solutions may present themselves.

E. Reverse Transformation

The solutions of the graph or scheduling algorithms must now be translated back to the original problem domain or specific application. This requires a reverse transformation analogous to the transformation used in the model generation step. This last step represents the transformation from the solution space back to the application space.

III. A SCADA Vulnerability Analysis Example

We now demonstrate the transformation process with an example taken from our work in critical infrastructure protection. The electric power system in most industrialized nations is a complex real-time control system with a variety of remote access points used for Supervisory Control And Data Acquisition (SCADA). Thus, the physical infrastructure of the power transmission and distribution system is dependent upon the communications network and SCADA devices used to monitor and control the electric power generation and delivery. At various places throughout the system are substations used to step-up or step-down the power being delivered through the system.

A. The Problem – Substation Vulnerability

Figure 2 shows an example electric power substations configuration with a variety of electronic access points scattered around the periphery and the physical system control actuators shown at the bottom. Note that several means of remote access are included here, even though all those access mechanisms would rarely be employed in a single site. Remotely accessible devices include any mechanism capable of circuit switching, analog or digital metering, calculating data values for protective functions, transmitting data to and from control power apparatus, and communications devices for remote access. Examples of remotely accessible SCADA devices include digital protective relays, telemetry devices, Remote Terminal Units (RTUs), Data Processing Units (DPUs), Programmable Logic Controllers (PLCs), Intelligent Electronic Devices (IEDs), and microprocessor-based substation controllers.

![Figure 2. Example Substation Control System](image)

Figure 2. Example Substation Control System

Assume we have an attacker who wishes to gain control of the circuit breaker (shown at the bottom-middle of Figure 2) in order to disrupt power to a geographic region. There are several ways an intruder can gain remote access to that breaker: They can dial-up and attempt to directly connect to RTUs or IEDs that offer dial-up access; they can wiretap telecom, LAN or WAN transmission; they can attack through the corporate Information Technology (IT) system and gain backdoor access to interconnected SCADA systems; or they can attack...
through a telecomm or ISP provider. In [10], the authors describe how electric utilities are dependent upon their corporate IT systems and how interconnected SCADA systems greatly increase the vulnerability of the electric power grid. As shown by the colored connections in Figure 3, Internet connectivity provides three different access paths to the circuit breaker within the target system.

From a SCADA security and survivability point of view we need to determine the relative vulnerabilities and corresponding mitigation costs for each of that attack paths shown in Figure 3. We can do that using a transformational model that allows us to manipulate the relative weights of multidimensional graph paths.

B. An Example Graph Model

Figure 4 depicts a graph representation of our hypothetical SCADA system introduced in Figure 2. Figure 5 shows the attack paths corresponding to Figure 3 in color. In both graphs we label the vertices with names that will allow convenient recognition during the analysis phase to come: SubstationController, SCADA-Master, SCADA-Interface, Corporate-Network, Local-Console, IED1, IED2, IED3, Internet, Transformer, CircuitBreaker, Relay, Remote-Control, Telephone-Network.

Thus far our mapping is trivial, but a device is more or less vulnerable to attacks and intrusions depending upon its properties and how it is connected to the network. Properties we are interested can be derived from questions like:

- What are the communication media properties (e.g., copper wire, optic fiber, radio, microwave)?
- What are the communication channel properties (e.g., speed, accessibility, bandwidth)?
- Does it offer public dial-in or ISP access?
- Is the access password protected?
- Does the device connect to a LAN or WAN?
- Does the LAN or WAN have public access points?
- Is the device TCP/IP enabled?
- Are communications to and from the device encrypted?

This additional information needs to be incorporated into the model, so we now expand the model to incorporate connectivity and security parameters.

C. SCADA Graph Model Parameters

Thus far we have constructed a model capable of representing and identifying device access
paths, but we cannot say anything about the vulnerability of the accessible devices. Note that the vulnerability of an attack object is not a static property, it is dynamic. It depends on device properties, network topology, network usage, enabled protocols, and other attributes and devices on the network. Other researchers have defined vulnerability based on continuous or probability distribution functions, and while we recognize the value of those approaches, we use a much simpler approach here for purposes of illustration.

We need path vulnerabilities, but static edge weights do not give us the flexibility we need for our research so we need to separate the edge properties from the edge weights. Edge properties are multidimensional, including aspects for physical media, logical connectivity, authentication and encryption mechanisms, so we define our graph edges using a 4-tuple containing predefined values or codes: <physical, logical, authentication, encryption>. Some examples consistent with the previous figures include:

\[
\begin{align*}
\text{Edge(SubstationController, SCADAMaster)} &= \langle\text{Fiber, SONET, None, None}\rangle \\
\text{Edge(SubstationController, LocalConsole)} &= \langle\text{CAT-5, Proprietary, None, None}\rangle \\
\text{Edge(SubstationController, IED1)} &= \langle\text{Coax, TCP/IP, None, None}\rangle \\
\text{Edge(SubstationController, Internet)} &= \langle\text{CAT-5, TCP/IP, Password, None}\rangle \\
\text{Edge(IE1, RemoteControl)} &= \langle\text{TwistedPair, QAM, Password, None}\rangle \\
\text{Edge(IE2, Wireless)} &= \langle\text{RF, WAP, Password, RC4}\rangle
\end{align*}
\]

This permits us to create a separate, dynamic table of edge vulnerabilities (i.e., weights) based on the edge properties. Every combination of physical connection, logical connection, authentication, and encryption characteristics can be represented in the table of weights. The weights increase with vulnerability, based on some arbitrary a priori data. Some examples corresponding to the above list of edge properties include:

\[
\begin{align*}
\text{edgeweight(Fiber, SONET, None, None)} &= 2 \\
\text{edgeweight(CAT-5, Proprietary, None, None)} &= 1 \\
\text{edgeweight(Coax, TCP/IP, None, None)} &= 3 \\
\text{edgeweight(CAT-5, TCP/IP, Password, None)} &= 6 \\
\text{edgeweight(TwistedPair, QAM, Password, None)} &= 4 \\
\text{edgeweight(RF, WAP, Password, RC4)} &= 9
\end{align*}
\]

We now have a parameterized graph model of a SCADA system with multidimensional edges that characterize device connectivity in a manner that can be conveniently manipulated.

D. Analyzing and Manipulating the SCADA Graph Representation

We define access path vulnerability as the sum of weights of the edges that must be traversed in order to get to the target device (i.e., graph vertex). Thus, the edge weight table is used to "color" the graph edges and then Dijkstra's shortest-path algorithm is used to order the access paths from highest to lowest vulnerability. We used Prolog to implement a program that represents and manipulates our model to find the most vulnerable access paths within our SCADA system. The most vulnerable access path is defined to be the highest vulnerability path with respect to a hypothetical attack from vertex i to vertex j in the graph. Figure 6 shows the output from the Prolog program analyzing the vulnerability of all paths from Internet to CircuitBreaker.
solutions, restrictions, and limitations garnered from the graph or scheduling application. For example, many critical infrastructure protection problems have topological maps that can be represented by directed or undirected graphs. Typical examples are transportation networks, electrical power grids, pipelines, water lines, and the communication networks controlling these infrastructures.

Many security problems can also be mapped to scheduling problems expressed as relations on the processor environment, the task and resource characteristics, and the optimization criteria. In computer science, scheduling theory is usually seen in the traditional sense of tasks and machine resources, but in the fields of security and survivability we can interpret this more loosely (e.g., software patches, agents, or recognition events). In this way, security issues can be mapped to scheduling problems in order to formalize our research and exploit the wealth of knowledge accumulated in scheduling theory. Our current research efforts are exploring ways in which real-time system security and survivability can be expressed in terms of graph and scheduling problems.

V. CONCLUSION

We have presented a simple example of transforming a real-time complex system into a graph model that permits exploration and manipulation of the security and survivability characteristics of the real world problem. We used the graph model as a first step to recognize the security characteristics of a remotely accessible electric power substation system. Further work enabled the development of tools and methods to mitigate vulnerabilities identified through the graph manipulations, and we have also used the graph model approach to conduct survivability analysis of complex systems. The modeling and evaluation of real-time control systems is much aided by the five-step transformational process described in this paper.

While our work is focused on critical infrastructure protection, specifically SCADA vulnerabilities, electric power substations are just one example of complex control systems where disparate devices are connected to each other and to other information systems and corporate networks [13, 14, 15]. This interconnectivity is becoming more and more prevalent because it offers system operators and administrators a convenient way of managing their apparatus and infrastructure. Unfortunately, it also exacerbates the security and survivability concerns within those systems. Researchers need a more uniform approach to analyzing security and survivability characteristics of complex systems. The transformational process, where real-time systems are reduced to graph or scheduling problems, permits researchers to address their problems in with a consistent approach that not only provides a wealth of defined algorithms, it should permit replication and comparison of results within the research community.

VI. ACKNOWLEDGMENT

We acknowledge and appreciate the financial support of the Critical Infrastructure Protection Grants Program of the National Institute of Standards and Technology and the Scholarship for Service Program of the National Science Foundation.

REFERENCES


