A Holistic Approach for Measuring the Survivability of SCADA Systems

A thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

Carlos Alexandre Queiroz Batista da Silva

B.Comp.Sc.

MSc.Eng (Dist.Comp.)

School of Computer Science and Information Technology
College of Science, Engineering and Health
RMIT University
August 2012

Declaration

Carlos Alexandre Queiroz Batista da Silva
College of Science, Engineering and Health
RMIT University
31th August, 2012

Acknowledgments

First, I would like to thank all my family. My parents (Edwaldo and Carmita) that have been always supportive, my brothers(Fred and Vinicius) that supported me with words and sometimes financially and my wife (Tais) that supported me during most part of this journey. I could not have done without her. Recently, my dad passed away, and it is with great gratitude that I dedicate this thesis to him, Edwaldo Batista da Silva. Papai, thanks for everything, I know you will be with me always. I love you.

I also want to thank Zahir Tari, who has been a great supervisor during my candidature and many times helped me to keep the track on my research. His advices were very helpful. I would like also to thank all my friends that directly or indirectly have supported me. It has been a long journal since the first day I started this PhD. I learnt a lot all way through. I will never forget such an amazing experience, a mix of feelings. Sometimes you think it will never get it done and suddenly you are here writing the acknowledgements with the feeling of *I made it*.

Credits

Portions of the material in this thesis have previously appeared in the following publications:

- Building a SCADA security testbed. C Queiroz, A Mahmood, J Hu, Z Tari, X Yu. Third International Conference on Network and System Security, 2009.
- An analytical framework for evaluating survivability of SCADA systems. C Queiroz, A Mahmood, Z Tari. IEEE 10th International Computer and Information Technology, 2010.
- Survivable SCADA Systems: An analytical framework using performance modelling. C Queiroz, A Mahmood, Z Tari. IEEE Global Telecommunications Conference - GLOBECOM 2010.
- A Framework to Quantify Survivability of SCADA Systems. C Queiroz. International Conference on Dependable Systems and Networks Supplemental - PhD Symposium. 2010.
- SCADASimA Framework for Building SCADA Simulations. C Queiroz, A Mahmood, Z Tari. IEEE Transactions on Smart Grid 2 (4), 589-597. 2011.
- A Probabilistic Model to Predict the Survivability of SCADA Systems. C Queiroz, A Mahmood, Z Tari. IEEE Transactions on Industrial Informatics. To be published. 2012.

The thesis was written in the `Sublime Text 2` editor on Mac, and typeset using the LaTeX 2_ε document preparation system.
All mathematical calculations with probabilistic graphical models and otherwise were made using *MATLAB*, *Mathematica* and *BRML* graphical model toolkit [Barber, 2012].
All trademarks are the property of their respective owners.

Contents

List of Figures

List of Tables

Abstract

Supervisory Control and Data Acquisition (SCADA) systems [Gaushell and Darlington, 1987; Krutz, 2006] are responsible for controlling and monitoring Industrial Control Systems (ICS) and Critical Infrastructure Systems (CIS) among others. Such systems are responsible to provide services our society relies on such as gas, electricity, and water distribution. They process our waste; manage our railways and our traffic. Nevertheless to say, they are vital for our society and any disruptions on such systems may produce from financial disasters to ultimately loss of lives.

SCADA systems have evolved over the years, from standalone, proprietary solutions and closed networks into large-scale, highly distributed software systems operating over open networks such as the internet. In addition, the hardware and software utilised by SCADA systems is now, in most cases, based on COTS (Commercial Off-The-Shelf) solutions. As they evolved they became vulnerable to malicious attacks.

Over the last few years there is a push from the computer security industry on adapting their security tools and techniques to address the security issues of SCADA systems. Such move is welcome however is not sufficient, otherwise successful malicious attacks on computer systems would be non-existent.

We strongly believe that **rather than trying to stop and detect every attack** on SCADA systems it is imperative to focus on providing critical services in the presence of malicious attacks. Such motivation is similar with the concepts of survivability, a discipline integrates areas of computer science such as performance, security, fault-tolerance and reliability.

In this thesis we present a new concept of survivability; *Holistic survivability* is an

analysis framework suitable for a new era of data-driven networked systems. It extends the current view of survivability by incorporating service interdependencies as a key property and aspects of machine learning. The framework uses the formalism of probabilistic graphical models to quantify survivability and introduces new metrics and heuristics to learn and identify essential services automatically.

Current definitions of survivability are often limited since they either apply performance as measurement metric or use security metrics without any survivability context. Holistic survivability addresses such issues by providing a flexible framework where performance and security metrics can be tailored to the context of survivability. In other words, by applying performance and security our work aims to support key survivability properties such as recognition and resistance. The models and metrics here introduced are applied to SCADA systems as such systems insecurity is one of the motivations of this work. We believe that the proposed work goes beyond the current status of survivability models. Holistic survivability is flexible enough to support the addition of other metrics and can be easily used with different models, for instance a hybrid of performance and security. Because it is based on a well-known formalism its definition and implementation are easy to grasp and to apply. Perhaps more importantly this proposed work is aimed to a new era where data is being produced and consumed on a large-scale. Holistic survivability aims to be the catalyst to new models based on data that will provide better and more accurate insights on the survivability of systems.

Chapter 1

Introduction

This chapter introduces our motivation to pursue this research. It highlights the current issues affecting SCADA systems and also gives an overview of the main ideas developed in this thesis. It goes further presenting the research questions this thesis aims to address and how the introduced concepts provide a better solution for evaluating the survivability of SCADA systems.

1.1 Motivation

Supervisory Control and Data Acquisition (SCADA) systems [Gaushell and Darlington, 1987; Krutz, 2006] are responsible for controlling and monitoring Industrial Control Systems (ICS) and Critical Infrastructure Systems (CIS) among others. Such systems are responsible to provide services our society relies on such as gas, electricity, and water distribution. They process our waste, manage our railways and our traffic. Nevertheless to say, they are vital for our society and any disruptions on such systems may produce from financial disasters to ultimately loss of lives.

In the past, SCADA and consequently the systems monitored and controlled by them were somehow protected because they relied on proprietary technologies, and with a very close industry nearly no information spread beyond the community. However SCADA systems have evolved over the years, from standalone, proprietary solutions and closed networks into large-scale, highly distributed computing systems operating over open networks such as the internet. In addition the hardware and software utilised by SCADA systems are now, in most cases, based on COTS (Commercial Off-The-Shelf) solutions.

Although such changes increased the efficiency and sophistication of the services provided, it also increased their vulnerability to malicious attacks. The once closed, proprietary software and hardware infrastructure is now vulnerable to attacks originating from external (internet) and internal corporate networks. The attacks plaguing such systems are the same ones that have been affecting ordinary systems over the years (e.g. Virus, Trojans, Worms). Additionally, the network protocols used by SCADA systems were not designed with security requirements in mind. For instance, the majority of protocols does not support any type of cryptography. The industry however, shows signs that it is changing.

Over the last few years there is a push from the computer security industry on adapting their security tools and techniques to address the security issues of SCADA systems [Coutinho et al., 2008; Gula, 2007; Kolesnikov and Lee, 2006]. DigitalBond [dig, 2012] initiatives such as *QuickDraw SCADA IDS*, which extends the existing Intrusion Detection Systems (IDS) by adding signatures to SCADA-based protocols, devices and vulnerabilities and project *Basecamp*, which aims to assess the security of SCADA field devices by hacking them and making the exploits publicly available, show that the ICS industry is not ignoring the problem. In addition to such initiatives, the U.S. government together with the ICS industry has put in place a set of standards and regulations that provide recommendations on how to protect SCADA systems [Knapp, 2011]. Those initiatives are on the right track to probably reach the level of security currently deployed on enterprise and personal computer systems. However as we all known this is not sufficient, otherwise successful malicious attacks on computer systems would be non-existent.

The fight against malicious attacks seems to be endless. Security experts appear to agree that the best approaches against such attacks are [Anderson, 2010]:

- **Management** - keep systems up-to-date and configured to minimise the attack surface. In other words, keep an endless scanning mechanism looking for vulnerabilities in applications running in the network.
- **Filtering** - use firewalls to stop trojans and network exploits, and to detect signs of attack and compromise, in case anything gets through.

- **Intrusion detection** - have programs monitoring your networks and machines for signs of malicious behaviours.
- **Encryption** - use protocols such as *TLS* and *SSH* that enable the protection of specific parts of the network against some type of attacks.

Such *approaches* might be effective to minimise the number of attacks, but they do not avoid them. For SCADA systems, some of these concepts cannot even be applied. For instance, we do not know how sensors and actuators will behave when using encrypted channels for communicating with each other. SCADA systems cannot afford be off-line even for a short periods of time, therefore, upgrades that demands restarts are not an option in many cases. Current firewalls and intrusion detection systems still need to improve their support to the protocols used in SCADA systems, a significant part of these protocols utilised by SCADA systems are not recognised by current firewalls and IDS tools [East et al., 2009; Huitsing et al., 2008; Zhu and Sastry, 2010].

In the case of SCADA systems, the possibility of attacks are numerous. Most exploits currently used to attack corporate networks may also be applied to SCADA networks. For instance, virus, worms, and trojans used to infect and gain control of Windows machines on corporate networks can also infect SCADA networks. Many SCADA applications such as MTUs and protocol gateways run on Windows. Below we describe some possible attack scenarios on SCADA networks.

- **Denial of Services (DoS) attacks** - Zombie computers that are part of a *botnet* are programmed to send requests to SCADA machines exposed to the internet. (e.g. gateways, firewalls). The attacks will degrade the network and as a direct consequence the SCADA controllers will receive late or not data from the devices deployed in the field (e.g RTUs, PLCs). As a consequence, system operators would not rely on any information acquired in the field, making any decision possibly erroneous.
- **Virus infection** - A corporate computer is infected by a virus because a user clicked on a bogus link and downloaded a contaminated file, a user plugged to the computer a USB dongle contaminated with a virus. Those are examples of initial virus infections. Once a computer in the network is infected the virus infection would spread quickly. For instance, for the case where the SCADA machine used to control and monitor field devices is in the same network of the

infected computer it is not hard to image this machine would also be infected, as the virus could easily copy itself to the SCADA machine. Another case is when the initial infected machine is the one used by the system operator to deploy applications to the SCADA system. The virus could reprogram the application to give new instructions. In fact such attack vectors were used by *Stuxnet*, a virus that targeted exclusively SCADA systems [Farwell and Rohozinski, 2011]. *Stuxnet* changed the original instructions to increase the operating frequencies of the centrifuges in a nuclear plant.

- **Internal attack** - SCADA operators have unrestricted access to the SCADA system they are supervising. An unhappy employee could perform an internal attack without being even noticed. In 2001 an employee caused 800,000 litres of raw sewage to spill out into local parks and rivers in Australia [Stephanou, 2001].

- **Exploitation through software vulnerability** - New vulnerabilities are discovered every day. A well-informed attacker may write exploits [Foster, 2006] to target new vulnerabilities that are not patched by the critical infrastructure company. Even for the cases there are patches available critical systems have a slower than the normal policy to apply patches. By exploiting the vulnerability the attacker can gain access to the machine, and consequently perform any operation desired. Figure 1.1 shows the number of vulnerabilities affecting SCADA systems over the last ten years. The numbers are only increasing.

- **Social engineering** - Attacks through social engineering are common. In [Mitnick, 2011], the author describes how he fouled employees of telecommunication companies to get access to user credentials he could use to penetrate their systems. In the case of SCADA systems, companies that provide critical infrastructure services such as power grid distributors, have multiple remote sites (substations) whose employees need to visit from time to time. A smart attacker could try to foul the company employees to have access to the remote network by calling the central office saying she or he is an employee and needs to connect to the field network but forgot the network access credentials. The central office could ask questions to certify that the attacker is really an employee, however as shown in [Mitnick, 2011] this can be easily circumvented. Once connected to the network the attacker could perform any authorised operation.

From the attacks highlighted above the most commons are the ones performed by exploiting vulnerabilities on software components. Typical SCADA systems have two type of software components: embedded (which runs on field devices) and COTS (which runs on SCADA servers). Despite such systems provide different characteristics of development and deployment, they share the same type of security problems, they all are vulnerable to attacks.

In general, an embedded software is developed to a specific hardware that means the software is customised to that device. Usually, the software is proprietary and the security is achieved through obscurity. Security researchers do not have access to the software, therefore they do not know what the possible vulnerabilities are. COTS software means the software is not customised to a certain environment or application. For instance, the software that we use on our personal computers is the same one that runs on SCADA systems. The Windows operating system is an example. Of course, we and the company that runs the SCADA system have different needs for security. We can afford to be off-line for couple hours to apply an update that patches a vulnerability. This is not the case to a power grid company, their systems run twenty-four-by-seven. Because they cannot apply patches immediately they remain vulnerable to attacks for a longer period of time.

The security of critical systems is of enormous importance. Unfortunately when compared with the general computer security industry they are very behind. Even though both industries are sharing the same software and hardware, in most cases.

The computer security industry knows that security is hard [Garfinkel, 2012; Hypponen, 2012; Moitra and Konda, 2000]. As mentioned earlier, for many years the security industry has tried to improve and fix the security on computer systems. For a matter of fact the security has improved immensely over the last decade, however we are no where close to totally secure systems. Statements made by people from the security industry corroborate such view. Recently, in an article to a technology magazine, Mikko Hypponen (CTO of F-Secure, a security company) wrote about why anti-virus companies did not catch virus such as *Stuxnet* and *Flame* [Farwell and Rohozinski, 2011; Nicolas Falliere and Chien, 2011], worms built to attack SCADA systems. The writer clearly says that anti-virus products made for regular consumers will not protect against well-resourced adversaries [Hypponen, 2012]. This means many things. First, the use of COTS hardware and software in critical systems is not a good idea. Second, anti-virus companies will never reach the level of sophistication of a well-resourced adversary. And Third, anti-virus products will always be behind trying to keep up to date.

Because of the widely spread use of COTS solutions researchers and adversaries [Knapp, 2011] are more aware of the internals of SCADA systems, the once proudly used 'security through obscurity' mantra does not apply any more. To further corroborate our statement we did a search for the keyword *scada* on the Open Source Vulnerability DataBase (OSVDB) [OSV, 2012], an open source database initiative that catalogues vulnerabilities on computer systems. The search returned 362 vulnerabilities. Figure 1.1 shows vulnerabilities targeting SCADA systems grouped by category. We can see vulnerabilities such as XSS and SQL injection, which are common on web systems. Clearly the adoption of COTS solutions by SCADA systems helped on increasing the number of vulnerabilities targeting them. The database of vulnerabilities can be downloaded from [Queiroz, 2012].

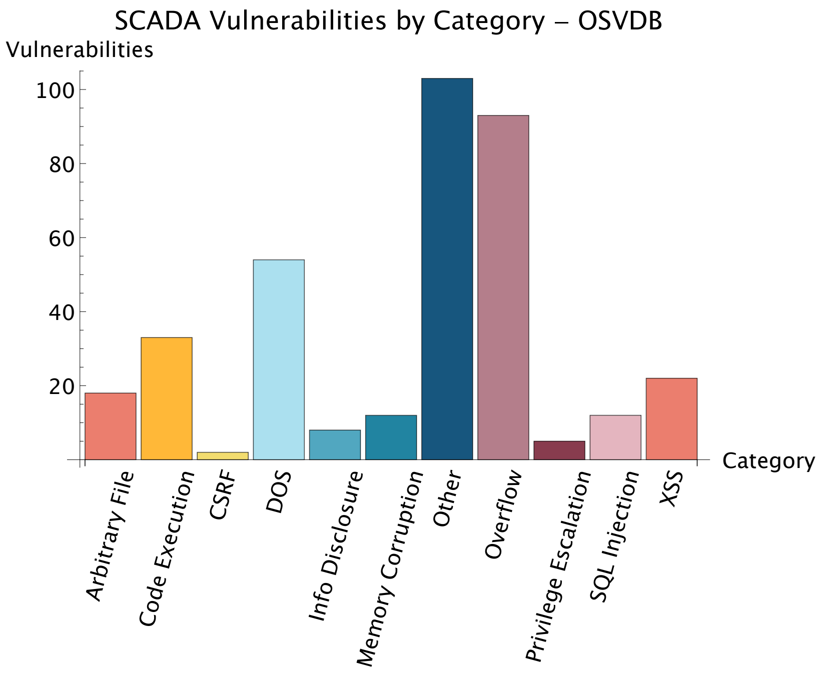


Figure 1.1: ICS/SCADA Vulnerabilities reported since 2001 grouped by category

From the same database [OSV, 2012], Figure 1.2 illustrates the number of vulnerabilities targeting SCADA systems since 2001. We can see that the number of vulnerabilities has double from 2009 to 2010 and more than double from 2010 to 2011. The database is from mid-2012 and the number of vulnerabilities already reported is on the right track to maintain the growth. It is important to say that the years that vulnerabilities were not reported it was not because they did not exist. They just were not catalogued. The 362 vulnerabilities reported are only the point of the iceberg. There is a much higher number of vulnerabilities not reported either because they derives from the ones already reported or because the vendors fear for their market share and public trust.

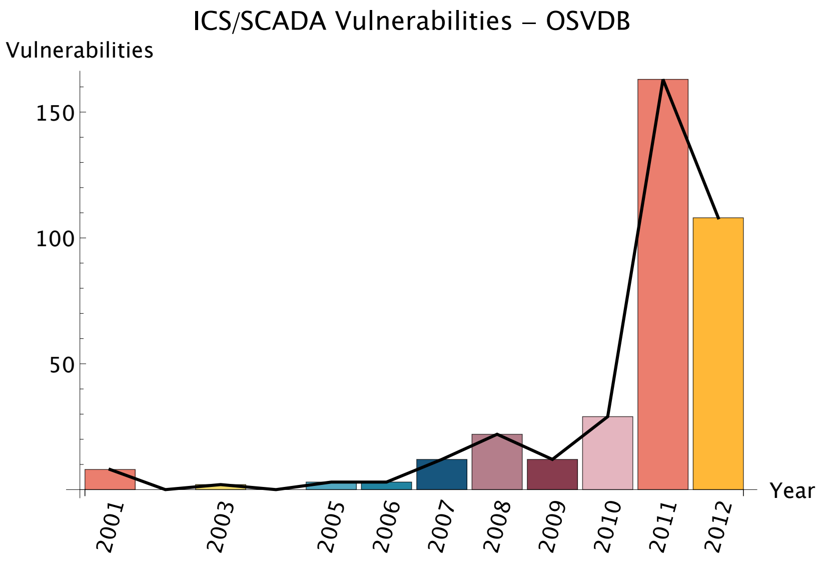


Figure 1.2: ICS/SCADA Vulnerabilities reported since 2001

On reporting and cataloguing attacks on SCADA systems there is the Repository of Industrial Security Incidents (RISI) initiative [RIS, 2012], an organisation that aims to collect and sell information about attacks on SCADA systems. Going in a opposite direction from [OSV, 2012], RISI sells subscriptions to individuals and organisations to have access to their database of incidents. RISI is a re-branding of the so called Industrial Incident Database (ISID) that for many years was controlled by a group of researchers at British Columbia Institute of Technology (BCIT). According to [Byres and Leversage, 2007], one of the founders of the ISID database initiative, ISID had catalogued 105 incidents by 2006. In 2009 RISI reported that they had 175 incidents in their database [RIS, 2012]. Recently ICS-CERT released a report that summarises malicious activities against ICS systems [Team, 2012]. The report corroborates the trend of an increasing number of attacks against ICS systems. From the attacks on SCADA systems, *Stuxnet* is the most important one for few reasons. *Stuxnet* is a worm/virus/malware that was created with the specific purpose of infecting and damaging the nuclear plants in Iran [Sanger, 2012]. Many say that it introduced the concept of cyber-weapons, software made with the specific purpose of attacking enemies [Sanger, 2012]. Yet according to [Sanger, 2012], *Stuxnet* was a jointly work between the US and Israel governments, examples of well-resourced adversaries discussed by [Hypponen, 2012].

Also as is the case to the vulnerabilities database, the number of attacks reported by [Team, 2012] is only a small fraction of the actual amount. Many companies fear that by disclosing an attack they will lose the confident of their customers and shareholders. Another problem is that network traffic datasets, widely used to simulate and identify attacks, are not publicly available for SCADA traffic [Carlson, 2002; Lüders, 2005; Stamp et al., 2003].

From what has been discussed so far it is clear that the current state of affairs in security systems that provide mechanisms based on prevention and interdiction [Coutinho et al., 2008; Goseva-Popstojanova et al., 2001; Gula, 2007; Kolesnikov and Lee, 2006; Krugel and Toth, 2001; Long et al., 2005] are not offering the desired level of security. We believe that such security level is not enough for SCADA systems. SCADA systems requirements are quite different from general corporate systems, they are widely spread, they rely on multiple technologies, they have limited resources, they are a mixture of real-time and not real-time operations and more importantly they have different needs regarding their availability, reliability and security, among other things. We strongly believe that **rather than trying to stop and detect every attack** on SCADA systems it is imperative to focus on providing critical services in the presence of malicious attacks. Such motivation is similar with the concepts of survivability [Ellison et al., 1997], a discipline that aims to integrates other areas of computer science such as performance, security, fault-tolerance and reliability.

We believe that the security of SCADA systems should be approached from a survivability perspective. Rather than trying to achieve an attack-free system, survivability focuses on provisioning of an acceptable level of services even in the presence of malicious attacks. Models have been used to analyse survivability attributes qualitatively, focusing on the process to build survivable systems, and quantitatively, focusing on measuring the survivability of current systems. Survivability can help on improving the security of SCADA systems by offering a model that includes security as a key attribute to keep the quality of service of such systems.

1.2 Existing solutions

Survivability is a concept that originated in the design of military communications networks where every operation should be reliable even if some communication links were destroyed [Frank, 1974]. It can be broadly defined as the capacity of keeping essential services of a system running in case of undesired events. It was later introduced to computing systems as a framework for developing requirements and strategies to build more effective systems [Ellison et al., 1997]. Since then, survivability has been actively studied. [Dai et al., 2007] catalogued a fair amount of work on design, architectures, and models to analyse survivability. In this thesis we are looking at the security perspectives that survivability offers. The undesired events in which survivability should act upon are here defined as malicious attacks. Current survivability models implementations are applied as a tool that provides a cost and performance analysis of the system. Their goal is to measure the system both in terms of the degree of functionality after an undesired event as well as the resultant cost of applying changes on the architecture and design of the system.

The seminal work of [Ellison et al., 1997], which was the first complete document on describing survivability to computer systems, did not get to the point of providing a quantification of survivability as the one proposed by the ANSI T1A1.2 group [T1A1.2, 2001]. The ANSI committee defines survivability as the measure of the interest value V that has a value V_0 just before the failure happens. Then, the survivability behaviour is represented by V_a, the value of V just after the failure occurs, V_u, the maximum difference between V and V_a after the failure, V_r is the value of V when the system is restored after time T_r, and t_R is the time used to the system recovery to the original value V.

However [Ellison et al., 1997] described (i) what is survivability, in what domains it is valid, (ii) what are the characteristics of survivable systems, and (iii) how survivability is related to Engineering. They put survivability together with security and fault-tolerance, and introduced the concepts of essential and non-essential services and the design and architecture of survivable systems. Since then, survivability has turned different paths as we can see by comparing the definitions of survivability presented by [T1A1.2, 2001] and [Ellison et al., 1997].

Today, the state of the practice on survivability is better illustrated by Figure 1.3. The research can be categorised into three groups: *Design and architecture*, *Methodologies for development and implementation of survivable systems*, and *Analysis models*.

Design
&
Architecture
Methodologies
Analysis
Models

Figure 1.3: Current approaches on survivability

Due to the proliferation of definitions, there is not much agreement on what survivability is and how it should be applied. [Westmark, 2004] catalogued more than 50 definitions of survivability. Definitions that encompass the quantification of metrics on critical services by comparing them before and after failures, accidents or attacks to the analysis of network services. In some cases, they do not differentiate much from concepts such as performability, a measure that unifies performance and reliability [Meyer, 1980].

The design and architecture of survivable systems aimed to enhance the survivability of information systems by providing mechanisms which allow the detection and treatment of various types of undesired events. The idea is to have a blueprint that could be used to build survivable information systems. It would allow the support of survivability requirements such as resistance, detection, recovery and adaptation.

The methodologies would provide guidelines and an entire process on how to analyse the survivability of organisations managed by systems through their development life-cycle [Ellison and Woody, 2010].

We believe that current survivability definitions and models are limited. First, even though some definitions of survivability recognised security as an important aspect [Ellison et al., 1999] there is not much survivability models evaluating the security aspects of systems. Usually, the analysis is limited to the performance of a specific service. Second, services are often evaluated individually. No attention or importance is given to the fact that services may depend on other services, and that some of these other services may affect the behaviour and the performance of the current service under evaluation. Third, most current research on survivability analysis is aimed at analytical analysis.

The analytical models are used as a measurement tool to evaluate the impact of failures on networked systems [Heegaard and Trivedi, 2009; Jha and Wing, 2001; Liu and Trivedi, 2006; Moitra and Konda, 2000].

In this thesis we are interested in analysis models, however not the analytical ones. We believe there is already a fair amount of research on analytical models to quantify survivability. We catalogued a reasonable amount of scientific papers focusing on the analytical analysis of survivability. They are discussed in Chapter 2.

Therefore, this research is focused on data-driven analysis. More specifically, we are interested on quantifying the survivability of SCADA systems based on data. A holistic model based on a data-driven approach.

1.3 Research questions

In this thesis we shifted the focus from analytical models based on performance analysis of individual services to a model that takes into account the entire system (holistic view), it is based on the systems security aspects (security as key attribute) and uses empirical data (data-driven) to quantify the survivability of SCADA systems.

More specifically, this thesis aims to tackle the following issues afflicting current survivability analysis models:

- *Single service analysis*. Current models used to analyse the survivability of systems often focus on the analysis of single essential services.
- *Essential service identification*. In general essential services are manually defined by system experts. Such an approach is appropriate when evaluating either parts of a system or small systems. However, it is intractable for complex systems with hundreds of services. An automatic approach to identify essential services is necessary.
- *Lack of security aspects*. A fair amount of current models are based on the performance aspects of a system. This behaviour hides an important issue, the lack of security-based aspects to evaluate survivability. Currently, systems are measured in terms of processing time, throughput, and so forth.
- *Analytical analysis*. Current models are aimed to provide analytical analysis of

systems' survivability. We believe that is time to provide models that could leverage the data acquired from computing systems in order to provide more intelligent analysis about these computing systems.

Based on the issues described above and on the idea of holistic survivability being built on the concept that the entire system should be assessed and evaluated to provide an accurate quantification of survivability, this research aims to answer the following research questions:

- **How the multiple services that are part of a SCADA system should be evaluated together? What formalism should be used to represent the interdependencies of services in a holistic analysis?**
- Performance analysis is a key aspect of quantifying survivability on the current survivability models. Although we believe that performance is an important attribute it is not the only one. If a model aims to quantify survivability considering its security aspects, metrics that represent the security of the system should be used. **What metrics should be added to the model to represent security?**
- We are proposing a data-driven model, where data generated by the system is an important part of the model. We know that systems generate a variate of data in specific formats and they are stored in different ways. **How do we acquire, transform and make the data available to be fed to the model? What parts of the data are important to the proposed model?**

1.4 Summary of the contributions

This thesis provides five main contributions to the field of survivability and SCADA systems.

1. To address the question of analysing SCADA systems taking into account the entirety of services, their functionalities and relationships we propose a holistic approach. **Holistic Survivability** extends current view of survivability by incorporating both service interdependencies (as a key property) and aspects of computational intelligence that uses the formalism of probabilistic graphical models [Koller and Friedman, 2009] to represent and quantify survivability.

2. We believe security is a key attribute of survivability and should be regarded as such. We introduce **metrics** that aims at the **security** characteristics of SCADA systems. Such metrics are integrated into the core of the holistic survivability modelling process. Any survivability quantification that uses the proposed metrics is considering security as a survivability attribute.

3. Currently the proposed survivability models do not offer techniques to determine the essential services of a system. Usually the essential services are determined by system experts before the quantification takes place. Usually, the service under evaluation is the essential service. However, we are convinced that such approach does not work on services with multiple essential services that need to be evaluated together, a key feature of holistic survivability. We propose an algorithmic approach to find essential services that is based on graph theory, network analysis [Landherr et al., 2010] and clustering analysis [Anderberg, 1973].

4. There are two main issues with the security analysis of SCADA systems. The first one is the scarcity of data available for analysis. Corporations that own the data do not disclose it fearing any security vulnerability discovered on the data could be used to perform attacks against them. The second one is the insufficiency of proper modelling tools to evaluate the security of SCADA systems. As it is widely accepted in academic and industrial communities, it is impractical to conduct security experiments on live systems. A modelling simulation tool would enable the simulation of SCADA systems with the benefit of testing different attack and security solutions. In this thesis we introduce a **simulator** aimed at building **simulations of SCADA systems**. A key benefit of the simulator tool is the ability of testing the effect of attacks on SCADA systems. In addition to that we also can generate SCADA network traffic to perform analysis on it.

5. Even though it is possible to perform analytical analysis using the holistic survivability model, holistic survivability focuses on **analysis** based on **data** acquired from deployed SCADA systems, or generated from simulations. Because holistic survivability is built on the formalism of probabilistic graphical models we can use it to perform two types of analysis. More specifically, we can perform inferences such as:

    - **Prediction** - The prediction analysis works by reasoning from causes to effects. In other words, given the causes the tool predicts the effects. It analyses the services and then quantifies the probability of survivability of the SCADA system.

- **Diagnosis** - The diagnostic analysis works the other way around; from symptoms to causes. By changing the overall probability of survivability of the system the tool shows which services are impacted more by the change. For instance, an analyst can use this tool to identify the affected services when the probability of survivability of the system is changed.

1.5 Organisation of the thesis

This thesis is organised in 8 chapters. We tried to map each core chapter to one of the main contributions of the thesis. In addition to these core chapters we have an introduction, background and related work and conclusion chapters. The thesis is organised as follows.

Chapter 2 gives an introduction about SCADA systems and their architectures. It also presents a short introduction to probabilistic graphical models, which is the mathematical formalism utilised by holistic survivability. In addition to that, the chapter provides a literature review on the subject divided into four main topics: Definitions, Design and Architecture, methodologies and analysis models. However, the main focus is on the analysis models proposed in the literature. The chapter describes the current models, the type of formalism adopted and their shortcomings.

Chapter 3 introduces SCADASim, a simulator for creating simulation of SCADA systems. SCADASim implements some protocols used by SCADA systems and also provides SCADA devices implementations that can be used by simulations. It is used throughout the thesis to perform the simulations used by the other chapters.

Chapter 4 introduces holistic survivability, a new concept of survivability. It provides the definition and the concepts of holistic survivability. It introduces the attributes, the means and the type of threats holistic survivability is aimed for. A formal definition is provided together with the formalism necessary to quantify holistic survivability.

Chapter 5 introduces an algorithmic based technique to find essential services in SCADA systems. Further the algorithm is compared with other existing techniques and finally an evaluation using a simulated SCADA system is performed.

Chapter 6 introduces holistic survivability in terms of performance modelling. In other words, it introduces a quantification of holistic survivability based on performance. It also defines performance metrics used to analyse survivability and how these metrics are applied to holistic survivability. It further shows an evaluation of the performance model using a simulated SCADA system.

Chapter 7 introduces the security metrics used to measure SCADA systems. It further shows how the metrics are applied to holistic survivability and finally provides an evaluation of the security model using a simulation of a SCADA system.

Chapters 6 and 7 show how holistic survivability can be applied using monitoring data acquired from the SCADA systems using historical and real-time data. They also show how machine learning techniques can be used to assess the state of the services of the system and how they can be used to generate the probabilistic model that represents the SCADA system under evaluation.

And finally, Chapter 8 concludes by summarising the work presented in the thesis and discussing some aspects for future research in the field.

Chapter 2

Background and related work

This chapter introduces SCADA systems, their properties and entities. Such introduction is important as it will further be used in the analyse of such systems. As also part of background concepts, the chapter gives an overview of the formalism of Probabilistic Graphical Models (PGM). Two types of models are presented: Bayesian networks and Markov Random Fields. They will be used to analyse the survivability of SCADA systems in chapters 6 and 7. In addition, a thoroughly literature review on survivability and security on SCADA systems is presented in this chapter. Survivability is discussed in different aspects: definitions, design and architecture, analysis modelling, development methodologies and a view of survivability and security as a research domain.

2.1 Background

This section introduces SCADA systems. It gives a brief overview on what constitutes a SCADA system. Describes the properties that make SCADA systems 'different' when compared with conventional IT systems, how they are organised, what are the common existing services. It also gives a example of a common network topology utilised by a variety of SCADA system deployments.

2.1.1 SCADA systems

SCADA systems are used to monitor and control critical infrastructures. They are composed by computers, networks, and sensors, which are used to control the processes running in the critical infrastructure sites. They sense, collect and analyse information produced by those processes. However, most of analysis and decision making processes are made by the so called system operators. Such operators are usually engineers with deep understanding on the processes monitored by the system. In this thesis we use the terms SCADA, Distributed Control Systems (DCS) and Industrial Control Systems (ICS) interchangeably, even though DCS systems may present different characteristics. For instance, usually SCADA systems are geographically disperse, which it is not always the case for DCS. However, those systems interact with each other, and often are seen by practitioners as only one system.

SCADA systems have special application demands and configurations; they may communicate using different protocols and diverse transmission medium. For example, protocols that are popular on systems used by power plants may not be used by the oil and gas industry. The SCADA infrastructure (software, hardware, and network) is different from regular IT infrastructure in terms of its network topology, its protocols diversity (hundreds of different SCADA protocols have been catalogued [Igure et al., 2006]), diverse communication medium (Serial, Ethernet, WiFi), the type of information exchanged, and its security priorities. To the regular IT industry the priorities are based on Confidentiality, Integrity and Availability (CIA). On SCADA systems the priorities are different, Availability is the most important aspect for SCADA [Weiss, 2010].

Figure 2.1 shows the three main components of a SCADA system deployment: system operators that controls and monitors the process, the SCADA system that provide software and hardware for data acquisition and intervention, and the critical infrastructure system that is monitored by these tools and system operators. In the example illustrates in Figure 2.1, the systems being under monitoring and control are critical infrastructure systems. However, it is not always the case. Some SCADA systems are deployed at factories and industries that are not as critical. Even for such deployments it is important to protect the system from malicious attacks. The main and only difference between these type of deployments is the scope of damage that a malicious attack can produce.

SCADA systems consist of some properties that may make them quite unusual when compared with conventional networked and IT systems. Below we describe some properties that together make SCADA systems unconventional.

- **Determinism** - SCADA systems interact with Distributed Control Systems (DCS) deployed in the field. Such systems are developed to run in real-time supported

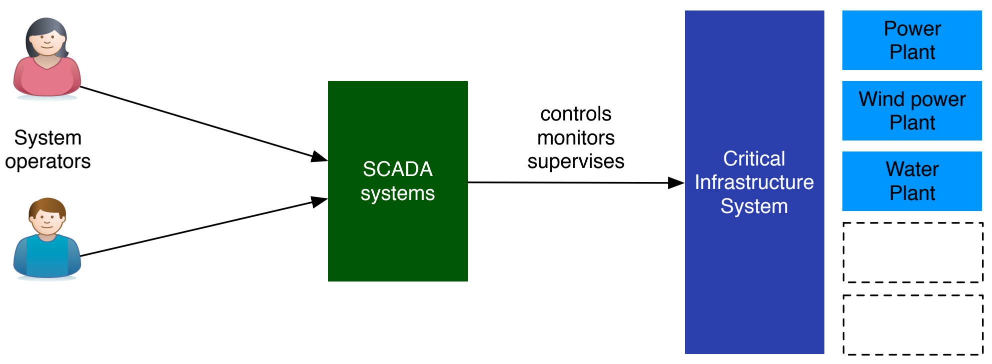


Figure 2.1: SCADA and Critical Infrastructure system

by Real-Time Operating Systems (RTOS). Such features make determinism and real-time characteristics of SCADA systems.

- **Heterogeneity** - It is presented in many parts of the system. SCADA systems integrate different operating systems, RTOS and GPOS (General Purpose Operating Systems) such as Windows and Linux and different hardware and software. A common deployment has embedded devices deployed at the field (usually at remote sites) and PC servers at the corporation site. Operators use CTOS computers to configure, program, monitor and control the systems. Different protocols are used by different applications. For instance, a deployment may have a PC with a HTTP server installed, where clients do requests using the HTTP protocol. Then, it may have embedded devices using MODBUS [Swales, 1999], a common protocol utilised by SCADA systems, to communicate with SCADA servers within the corporation. At the first glance, this configuration it is not much different from regular IT systems with Web and database servers installations, as they also use different protocols. However, these systems are underlined by the same protocol and characteristics: TCP/IP protocol and non-deterministic interactions. In the other hand, SCADA systems mix together deterministic and non-deterministic interactions. In addition to that, the hardware utilised by SCADA systems is comprehensive. It includes from sensors to PC servers, with each one requiring different types of support.

- **Distributed** - SCADA systems are geographically disperse. Control centres are often located far away from the field devices, usually located at remote sites. The communication between them are usually made by different medium; satellite, cellular, and so forth.

- **Criticality** - As mentioned earlier, in general SCADA systems are responsible for controlling critical infrastructures that our society relies on. For instance, power grids, water treatment plants, nuclear plants, traffic signalling, among others. Such critically may have dramatic impact. For instance, the accident at Chernobyl nuclear power plant [Ginzburg and Reis, 1991; Hatch et al., 2005] caused the direct deaths of 250 people and more than 800,000 people suffered radiation exposure which may result in cancer deaths over those exposed [Hatch et al., 2005]. Nevertheless to say, failures on SCADA systems that control critical processes may cause serious catastrophes. Besides, there is no shortage of incidents intentionally caused by humans (malicious attacks), which is one of the points this thesis addresses.
- **Availability** - SCADA systems need to be available at all times. Availability is its most important attribute. System managers normally trade off security for availability. They leave the system without applying patches for months, sometimes years, because they cannot afford any downtime to the system.
- **Reliability** - Field devices operate for long period of time without any human intervention. They are designed to be fault-tolerant and to run and in adverse environmental conditions. In addition, a common practice is to provide redundant communication links between the sites, in order to provide better fault-tolerance.

We categorise SCADA systems in layers based on their components; *communication links*, *hardware*, *protocols* and *services*. The network layer handles the communication infrastructure such type of connections, network topology and so forth. The hardware layer handles the system devices. Field devices, PC servers and clients, network hardware and so forth. The protocols layer consists of the protocols utilised by SCADA systems. And finally the services layer, which is the layer where the applications lives.

We understand a SCADA system as a group of services working together in order to accomplish their functionalities in a service-oriented approach. Figure 2.2 illustrates the layers of a SCADA system and its relationship with a SCADA entity.

Communication links It can be categorised into two different types of networks that generally use different protocols. The corporate network that is inside corporate offices

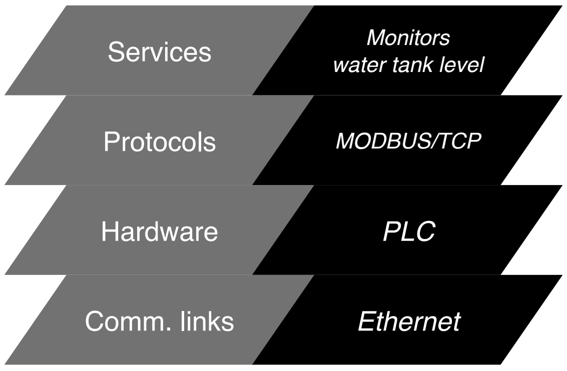


Figure 2.2: On the left, a SCADA system in layers. On the right, a deployed SCADA entity.

(local network), which is usually built over Ethernet and TCP/IP protocols running on standard computer hardware (PC servers and clients). The remote network, which runs on remote sites and is typically connected to the corporate network via Satellite, Radio, Leased-lines and more recently, the internet.

Hardware Most of the hardware used in the field (remote sites) is dedicated. It is designed for operating under extreme environmental conditions in order to require less maintenance and be more durable. A field device such as a RTU (Remote Terminal Unit) generally consists of a real-time clock and operating system (RTOS), input and output interfaces (I/O), electrical spike protectors, restart timer to reboot the device in cased it become unresponsive, power supply with extra batteries, communication ports such as Ethernet, radio and serial (RS-232 and RS-485), RAM and flash memory. On the other hand, the hardware used by the corporate networks is the common COTS (Commercial off-the-shelf), PCs with Ethernet cards running TCP/IP and generic operating systems such as Windows and Linux. A typical MTU (Master Terminal Unit) runs on a PC with Windows or Linux as operating system. It uses a common Ethernet card which runs TCP/IP as communication protocol. HMI (Human Machine Interface) servers and Historian databases usually run on powerful PC servers with more memory, faster CPU, more disk space and faster communication cards. Generally speaking there are two types of hardware: *PC servers* based on COTS hardware and *embedded devices*, custom made for specific tasks.

Protocols According to [Igure et al., 2006] there are over 150 protocols utilised by SCADA systems catalogued, mostly proprietary. However, only a small group is widely used. Protocols such as MODBUS [Swales, 1999], DNP3 [Majdalawieh et al., 2006], and OPC [Knapp, 2011] are very popular and broadly deployed. Recently MODBUS and DNP3 have been integrated with TCP/IP. The MODBUS protocol offers a variant version called MODBUS/TCP [Knapp, 2011] that uses the TCP/IP as transport and network protocols. The DNP3 protocol can also run over TCP and UDP [Knapp, 2011]. Frames at the link layer are encapsulated into TCP/IP packets, so that DNP3 can take full advantage of the internet technology. Such integration and adaptation make possible to use COTS hardware to support DNP3 and MODBUS protocols. The majority of protocols utilised by SCADA systems were not created with security in mind. Most protocols are vulnerable to simple *man-in-the-middle* attacks as they exchange messages in plain text.

Services The service layer is where our research is concentrated. A SCADA system consists of services interconnected through network links. A typical SCADA deployment may consist of hundreds of services running together. Figure 2.3 illustrates a typical SCADA deployment with four networks. The corporate network used by system experts to upload new software releases, monitor and control the critical systems. Remote networks that deploys field devices that are required to acquire data and perform local actions to the local system under monitoring. And a remote corporate network, whose users (system operators) use to work from remote locations.

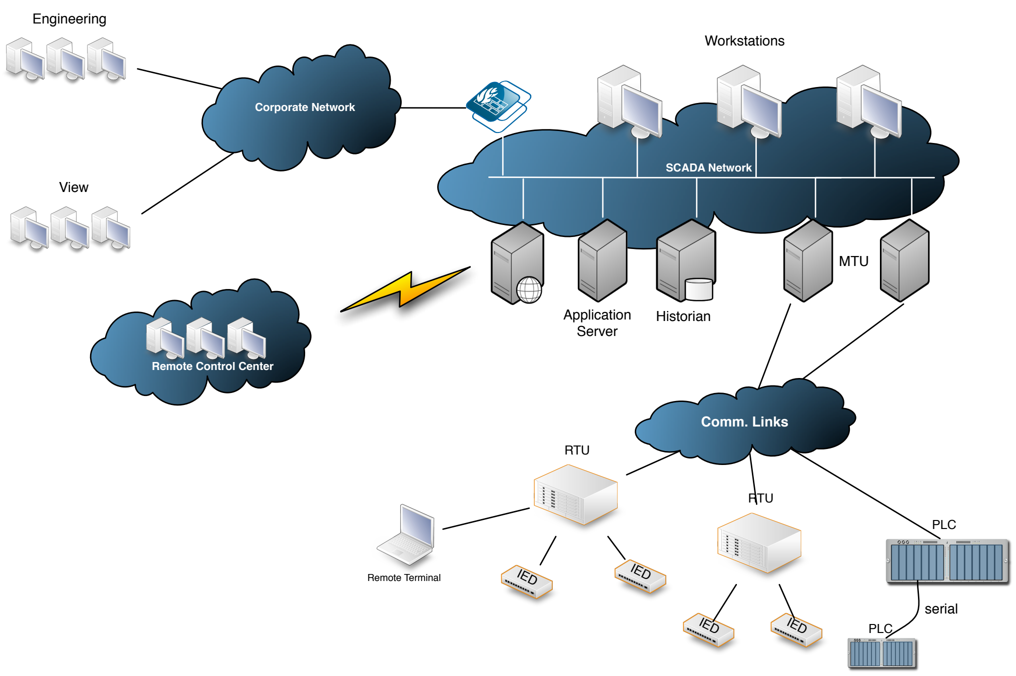


Figure 2.3: Typical SCADA Deployment. The monitored critical system is not illustrated.

A typical SCADA deployment contains four type of services:

- **HMI (Human Machine Interface)** - HMI the service that system operators (experts) use to control and monitor SCADA systems. More recently, such services became web-based. Users interact with the service through web browsers. The HMI service is often deployed with redundancy to avoid downtimes.

- **MTU (Master Terminal Unit)** - MTUs are the bridge between the corporate network and its users (system experts) and the field networks, where critical systems are located. They are responsible for gathering and aggregating data acquired from the field services. In addition, they also provide interfaces for performing actions on the field services. For instance, lets say a system expert wants to stop a motor that is functioning on a remote site. The request is made through the HMI service that communicates with the MTU that controls the remote service, which then sends the request action to the intent service. A normal deployment consists of multiple MTUs.

- **Field services** - They are the services that interact with the critical systems. They are responsible for gathering information from sensors, usually connected to the critical devices, to perform actions through commands sent by MTUs. These services are deployed on field devices such as PLCs (Programmable Logic Controller), RTUs (Remote Terminal Unit) or IEDs (Intelligent Electronic Device). It depends of the functionality of the service. It is a common deployment to have a field device running more that one service. Figure 2.2 shows a PLC that has one service (water tank level monitoring) running. This functionality is provided through a sensor that is attached to the water tank. However, this same PLC could also be responsible for another service. For instance, it could control an motor (actuator) attached to the water tank that pumps water in and out. Normally, PLC, RTU and IED have different functionalities in the field. RTUs usually monitor field parameters and send them back to MTUs. PLCs are used for controlling real-time processes, therefore they are designed for efficiency. IEDs are controlled remotely via PLCs or RTUs, they usually are part of control systems such as transformers, circuit breakers, and so forth.

- **Historian** - It is a database that stores all data gathered from the system. The data is further used for historical and insightful purposes. Usually the data is stored as tags. A tag can be the frequency of a motor or the boiling water temperature of a tank. Human generated information could be also stored. For instance, the targets for some process in production.

Most SCADA services uses the client and server model of computation. MTUs perform requests to field devices (RTUs, PLCs, IEDs), HMI servers receive requests from HMI clients. Historians receive requests from HMI servers and MTUs and so forth. There is a strong interdependency between these services. A service failure may cascade into the failure of other services and sometimes incapacitate the entire system, making the services-oriented assessment of SCADA services an important aspect of system evaluation.

2.1.2 Probabilistic graphical models

Probabilistic Graphical Models (PGM) [Koller and Friedman, 2009] are a framework that combines uncertainty and independence constraints to represent very complex problems compactly. They represent a full joint distribution over a set of variables compactly. Given a model with m variables where each can take n values it would take m^n parameters to represent such model, making it computationally intractable. However, PGMs take advantage that a given variable in a model is only dependent on a small group of variables, and this groups make it independent from the others. Such assumption makes PGM models to remove the redundancy by modelling the conditional independences between the variables and consequently providing a much compact representation of the original model.

PGMs use graph theory to represent a complex probabilistic distribution in a compact way. Variables are represented as nodes, the edges represent the dependencies between the variables in the graph and the variable values are represented by distributions.

The two most common types of PGMs are Bayesian networks [Perl, 1988] and Markov Random Fields [Kinderman and Snell, 1980]. Both representations have the same basic notion of conditional independence, a key aspect of PGMs. Conditional independence is defined as follows.

Definition 1 *Given three random variables A, B and C. We say that A is conditionally independent from B given C in a probability distribution p If $p(A = a, B = b|Z = z) = p(A = a|Z = z) \times p(B = b|Z = z)$ for all values $a \in A$, $b \in B$ and $c \in C$. This can also be written as $A \perp B|C$*

Another theorem that is used by PGM models and highly relevant to the work proposed in this thesis is the Bayes' rule. The theorem can be shown by applying two other probability theorems: Conditional probability and the the total probability theorem (see Definition 2). Then, by applying both definitions the Bayes' theorem is derived, $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$.

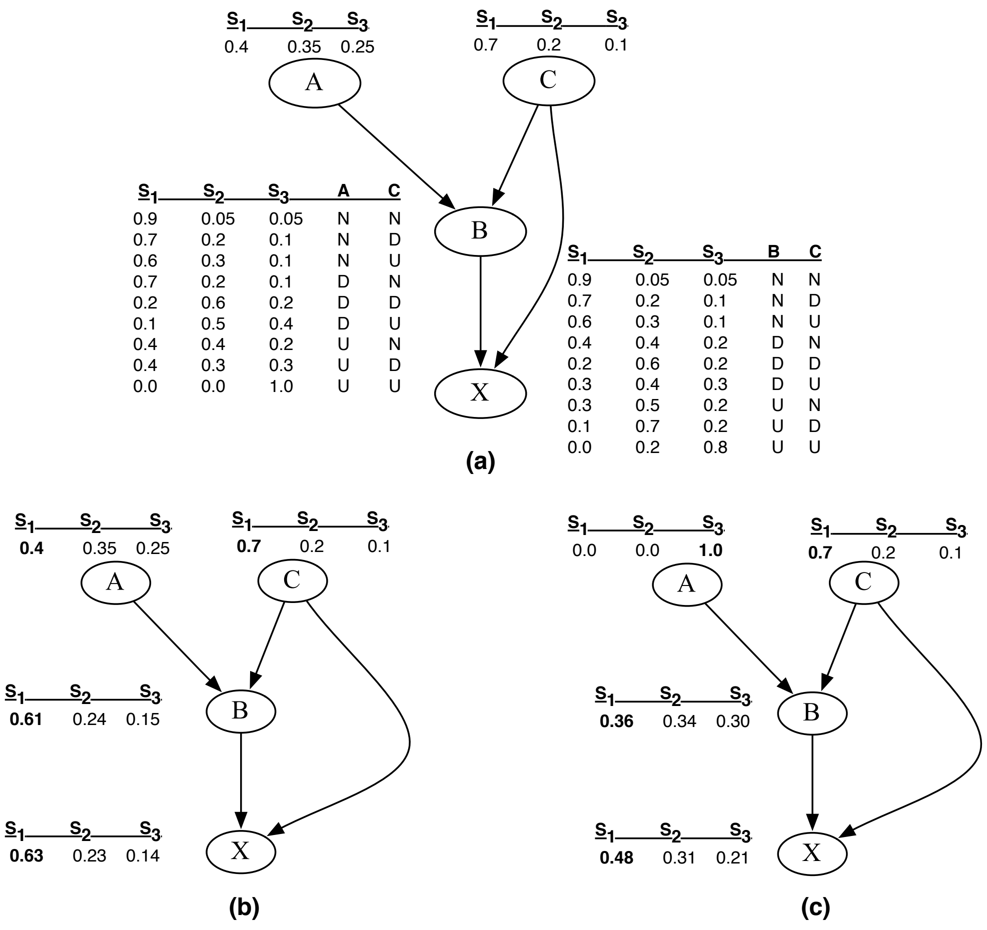
Definition 2 *Given $X_1, X_2, ..., X_n$ is a partition of the sample space Ω, that is X_i ($i \in n$) are pairwise disjoints and $\bigcup_{i \in n} x_i = \Omega$, then $P(Y) = \sum_{i=1}^{n} P(Y|X_i)P(X_i)$.*

PGMs represent a complete model for the variables and their relationships, therefore they can be used to answer probabilistic queries about the variables. For instance, a graphical model can be used to find out about the state of a group of variables when other variables are observed (evidence). This is called probabilistic inference, the process of computing the posterior distribution of variables given evidence. We will used it throughout this thesis to compute the survivability of SCADA systems.

2.1.2.1 Bayesian networks

Bayesian networks are Directed Acyclic Graphs (DAG) that consist of a network structure G that encodes a set of conditional independence assertions about the variables and a set P of local probability distributions associated with each variable. The variables are connected through direct links that represent the dependencies between the variables. For instance, if there is an arrow linking A to B ($A \rightarrow B$), A is said to be the parent of B and that B is dependent on A. $P(B|A)$ represents the Conditional Probability Distribution (CPD) that quantifies the effect of A on B. More generally, For each X_i, $P(X_i|Pa(X_i))$ quantifies the effects of X_i parents in the network.

Figure 2.4 (a) illustrates an example of a Bayesian network with four variables (nodes) $\{A, B, C, X\}$ and three states $\{S_1, S_2, S_3\}$. It shows the CPD of all variables represented as Conditional Probability Tables (CPT), another form of representing discrete variable states. Each row in the CPTs represent the conditional probability of each node for conditioning situation.



(a)

A:

S_1	S_2	S_3
0.4	0.35	0.25

C:

S_1	S_2	S_3
0.7	0.2	0.1

B:

S_1	S_2	S_3	A	C
0.9	0.05	0.05	N	N
0.7	0.2	0.1	N	D
0.6	0.3	0.1	N	U
0.7	0.2	0.1	D	N
0.2	0.6	0.2	D	D
0.1	0.5	0.4	D	U
0.4	0.4	0.2	U	N
0.4	0.3	0.3	U	D
0.0	0.0	1.0	U	U

X:

S_1	S_2	S_3	B	C
0.9	0.05	0.05	N	N
0.7	0.2	0.1	N	D
0.6	0.3	0.1	N	U
0.4	0.4	0.2	D	N
0.2	0.6	0.2	D	D
0.3	0.4	0.3	D	U
0.3	0.5	0.2	U	N
0.1	0.7	0.2	U	D
0.0	0.2	0.8	U	U

(b)

	S_1	S_2	S_3
A	**0.4**	0.35	0.25
C	**0.7**	0.2	0.1
B	**0.61**	0.24	0.15
X	**0.63**	0.23	0.14

(c)

	S_1	S_2	S_3
A	0.0	0.0	**1.0**
C	**0.7**	0.2	0.1
B	**0.36**	0.34	0.30
X	**0.48**	0.31	0.21

Figure 2.4: Example of a Bayesian network

Figure 2.4 (b) shows the marginals of all variables. And Figure 2.4 (c) show the type of inferences are looking for by using Bayesian networks. In this case we set the variable A as the evidence of being in the S_3 state and then we recompute the marginals of the other variables. We can see the that probability of variable X being in state S_1 decreased to 48% from 63%, even though they are not directly connected.

As mentioned earlier, Bayesian networks and graphical models in general represent the joint probability distribution over all variables in the graph. A Bayesian network can be more formally defined as:

Definition 3 *Given a set of variables* $\{x_1, x_2, ...x_n\}$*, a graph G and a set of probabil-*

ities P, the Bayesian network that represents the joint probability distribution is

$$P(x_1, x_2, ...x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i))$$

.

By applying the definition above to the Bayesian network example illustrated in Figure 2.4 we have the factorisation defined in Equation 2.1.1. We can clearly see that Equation 2.1.1 represents a much compact representation of the network.

$$P(A, B, C, X) = P(A) \times P(C) \times P(B|A, C) \times P(X|B, C) \tag{2.1.1}$$

2.1.2.2 Markov networks

Markov networks are built on undirected graphs. As in Bayesian networks, the graph nodes are defined as random variables and the edges as the relationship between nodes. Markov networks differ from Bayesian networks in terms of inference and independence structure.

As illustrated in Figure 2.4 and also mentioned earlier, the parameters of a Bayesian network are represented as CPDs, where distributions over nodes given other nodes are defined. On Markov networks there is no directional interaction defined by the graph structure. Rather than define CPDs for representing nodes distributions, Markov networks use the concept of *potentials*. Given x as a random variable, a *potential* is a function from $Val(x)$ to $\mathbb{R}^+$.

Potentials are used to parametrise the distribution represented by the network. The overall joint distribution considers all potentials defined in the graph structure. A Markov network is defined as a product of potentials. More formally,

Definition 4 *Given a set of random variables $X = \{x_1, ..., x_n\}$, a Markov network is defined as the product of maximal cliques, which are defined as potentials, on subsets of $X_c \in X$, $p(x_1, ..., x_n) = \frac{1}{Z} \times \prod_{c=1}^{C} \phi_c(X_c)$.*

Z is a constant known as partition function. It ensures that the distribution p is normalised [Barber, 2012]. Therefore, in Markov networks the CPDs are attached to the potentials.

2.2 Related work

This section presents an overview of the field of survivability. It starts by describing the multiple definitions of survivability and how they are related with each other. Then it provides a literature review on survivability segmented by three categories: **Design and Architecture**, **Performance models** and **Security models**. To conclude the section describes some relevant work on the field of security on SCADA systems.

The research on *survivability* is divided into two sub-fields; Network communication and Information systems. Network communication addresses the aspects of the network survival, links, speed, routes, and so forth. Information systems address software. It is related to performance, reliability and security aspects of applications and their services. This thesis is interested on how one can analysis and measure the survivability of networked systems (information systems survivability). Therefore, when it refers to *survivability* it is talking about information systems *survivability*.

2.2.1 Survivability definitions

There is no agreement on a common definition of survivability. [Westmark, 2004] catalogued more than 50 definitions about survivability. [Avizienis et al., 2001, 2004b] argue further that survivability and dependability are equivalent. Clearly, such confusion on survivability concepts makes difficult to have a general definition, one that could be applied across domains and with a common set of attributes. Many [Al-Kuwaiti et al., 2009; Westmark, 2004] assign such issues to the cross-domain use of survivability itself. There are other issues with the definition and use of survivability as well, we approach them later on this thesis. In fact, one main contribution of this thesis is the proposal of a new concept of survivability, which fixes issues such as interdependencies of services, recognition of essential services and predictability.

From the definitions described in [Westmark, 2004], and other ones found in the literature, we classified the definitions of survivability in three groups:

- *Group 1* - The definitions in this group can be summed up as: *the capacity of the system to continue to fulfil its mission despite intrusions, failures and accidents.*

Implicitly, the essential services are responsible for the mission, and the type of threats (intrusions, failures and accidents) may be malicious and non-malicious.

- *Group 2* - In this group the definitions are related to network survivability [Gokhale, 2006]. *the network is still able to perform part of its functionality after a failure by restoring from a degraded condition*. Note that the type of failure is not defined, and also they assume that the network recoveries are part of its functionality.

- *Group 3* - This group clusters definitions that are not part of groups 1 and 2, and are restricted to a particular issue. For instance, 'Service stream over time' [Sullivan et al., 1999], or 'Where the integrity is not compromised at the occurrence of unexpected disasters' [Liew and Lu, 1992].

We are interested in Group 1 definitions, as they are related to the concept of survivability presented in our research. From all definitions surveyed two deserve more attention; [Knight and Sullivan, 2000] was the first formal definition on survivability and [T1A1.2, 2001], which is widely used by survivability practitioners. They are part of a small group of definitions that aim to quantify survivability, closely related to our goal in this thesis that is to measure survivability.

[Knight and Sullivan, 2000] definition states that *A system is survivable if it complies with its survivability specification*. The specification is based on a four-tuple (E, R, P, M), where: E, environment, is a definition of the environment in which the system has to operate. R, specification set, is the set of specifications of tolerable forms of service for the system. P, probability distribution, is associated with each member of the set R with the sum of these probabilities being one. M, a finite state machine, is denoted by four-tuple (S, S_0, V, T), where S is a finite set of state, S_0 is the initial state of the machine, V is the finite set of customer values and T is the state transition matrix. Yet according to [Knight and Sullivan, 2000] a probability is assumed with each state and the system survivability is defined by the sum of the probabilities that the system is in one of the preferable states. There are some issues with this definition. First, it was intended to be used during the development of a survivable system, as it defines a specification in which developers need to agree upon. Second, it is labour intense, demanding a huge amount of time to define all attributes required by the definition. And finally, it does not take into account service (inter)dependencies.

[T1A1.2, 2001] was introduced by the working group on network survivability performance of the American National Standards Institute (ANSI). This definition is more related with group 2 however due to its wide usage, we present some discussion about it. The definition states that survivability can be quantified as the measure of the interest value V that has a value V_0 just before the failure happens. Then, the survivability behaviour is represented by V_a, the value of V just after the failure occurs, V_u, the maximum difference between V and V_a after the failure, V_r is the value of V when the system is restored after time T_r, and t_R is the time used to the system recovery to the original value V. In other words, survivability is measured before and after some undesired event happen, the difference will quantify the survivability of the system or network. The main drawback with this definition is it is completely tighten with performance and also does not provide any insight on how it is measured, it is measured in the service itself, in the system, if so how the services are related. In the literature many have adopted this definition, usually applying to a specific service entitled to be measured. A metric is defined and measured (V), and then the difference between before and after events is computed. The time (t_R) to recovery is also important to the definition. The shorter is the time to recovery the better is the survivability.

We can also notice that both definitions make no mention about essential services either they assume the service being evaluated is of interest and therefore essential, or they do not have such concept. [Knight and Sullivan, 2000] definition has the concept of acceptable services, which is close to the idea of essential services but it is not the same. For them acceptable services are the ones that provide the best they could under undesired conditions.

As survivability definitions are the catalyst for proposals on new architectures, methodologies, and analysis models for survivable systems, over the next sections we provide more discussion on these topics.

2.2.2 Design and architecture of survivable systems

The idea of design survivable architectures for information systems is to provide mechanisms which allow the detection and treatment of various types of failures, faults and malicious attacks in design time. It then facilitates the implementation of survivable systems as they provide a blueprint on the development of survivable systems. A survivable architecture will provide run-time support for survivability requirements, such as availability, reliability, resistance to failures, and so forth. [Heimbigner et al., 2002; Knight and Strunk, 2004] proposed an architecture in which survivable systems should be built upon. The architecture is based on reactive control. It provides a set of components for monitoring, diagnosis, synthesis, coordination, and response that are used to adapt the system in real-time, in the case of systems components not performing as expected. [Knight et al., 1997] proposed another architecture for survivable systems. The innovation was the introduction of wrappers or 'shells'. The idea was to wrap services with layers of certain functionalities in order to achieve survivability. For instance, in order to provide localised security a 'shell' could be implemented on a specific service denoted important to the system. By doing that, they argue that the localised policy could be easily enforced, rather than on the entire system.

As one could expect there is also some research on architectures inspired by the adaptive aspects of the human immune system. After all, adaptation is a key attribute of living organisms. [Sheldon et al., 2004] proposed an architectural model that uses principles such as regeneration to achieve survivability. It uses aspects of autonomic computing [Kephart and Chess, 2003] to provide the vision of an architecture that manages and adapts by itself. Even though they only have shown ways of implementing without providing any test implementation, the architecture ideas are well aligned with the main idea of creating survivable computer systems based on adaptation and regeneration. [Hiltunen et al., 1999] presented an approach to survivable architectures that is based on the customisation of and adaptation of services. Through customisation one could create variants of a service with different QoS, for instance. The customisation also enables service to be changed in runtime, therefore, providing them with dynamic adaptation, a key aspect of survivability.

[Hiltunen et al., 2001] proposed an architecture to enhance the survivability of systems through redundancy. Such approach is widely used nowadays with most of the deployed architectures.

[Yurcik, 2002] proposed an adaptive multi-layer framework to improve survivability looking at the security aspects of the system. They argued that layering the system reduces complexity and increases flexibility for adaptation, and consequently survivability. [Browne et al., 1999] proposed middleware based architecture that addresses the mobile IP protocol for military use. The architecture proposes changes on the hand-off mechanism of the mobile IP. More specifically, they adapted the session layer of the OSI model to handle the hand-off, moving this functionality from the original layer, network layer. They argue that the session layer is more suitable to handle recovery, replication and migration of resources.

There is some work on proposing architectures based on distributed systems technologies such as CORBA. In [Narasimhan et al., 1999], the authors proposed an architectural mechanism to support survivability on CORBA applications. The immune system mechanism replicates CORBA objects on the system to avoid undesired events. [Wells et al., 2000] provides a survivable architecture using the loosely coupled aspects of service-oriented architecture. They also show how a survivable system could be implemented on CORBA and Java/RMI.

Rather than proposing architectures for survivable systems, [Kazman et al., 1998] proposed a method to evaluate architectures looking at their quality requirements. More specifically, they look at different architectural scenarios based on the quality attributes and then proposes an architecture that tries to maximise all quality attributes even though trade-off between the attributes are needed. They examine attributes such as performance, security, availability, and so forth, all of them essential for the survivability of a system.

Despite the fact that there is not much work in the literature on methodologies to developing survivable systems, we briefly discuss it here. [Fisher and Lipson, 1999] proposed to use emergent algorithms to enhance the survivability of systems on unbounded systems. They argue the emergent algorithms are best suitable for implementing survivable systems because their characteristics are similar with survivable systems. For instance, emergent algorithms produce global effects through cooperative local actions distributed throughout the system. Computer systems cannot achieve survivability at the level of system components because the component itself represents a single point of failure for its own survival. Therefore, distribution and implementation in a distributed fashion, is essential to achieve survivability. As we can see survivability and emergent algorithms are very close on their characteristics and forms of functioning.

[Ellison and Moore, 2001] proposed a process for systematically refining an enterprise system architecture to resist, recognize, and recover from intended malicious attacks by applying reusable design primitives that help ensure the survival of the enterprise mission. [Mead et al., 2001] discussed the changes that project management environments aimed at development and and evolution of survivable systems brings to regular software development environments. How the activities are tailored to such systems impact the current development process.

More recently, [Ellison and Woody, 2010] proposed a framework to analyse the operational aspects of distributed software development in large corporations with dynamic environments. The framework works by analysing the complexity and integration issues of the development life cycle. It is mostly used by project managers that want to ensure that the development is proceeding as expected operationally. They call it Survivability Analysis Framework (SAF). The intuition of using survivability is that the software should survive to this distributed, and dynamic environment of people, and software components during. More specifically, they want to assert that the development and integration operations work under these conditions.

In [Linger and Moore, 2001] the authors created the bridge between development of survivable systems and models to analyse them. Unfortunately, they did not provide quantification techniques for analysing survivability. However, their intuition for analysing survivability where systems should be quantified 'before' and 'after' undesired events to then compute survivability is the base for many analytical models that were devised afterwards [Heegaard and Trivedi, 2009; Liu and Trivedi, 2006; Moitra and Konda, 2000; T1A1.2, 2001].

2.2.3 Performance models

The types of processes to analysing survivability can be defined into two groups: qualitative and quantitative. In the current literature most attempts on measuring survivability have been qualitative, focusing more in the process of building survivable systems, the software engineering aspects. The Survivable Network Analysis (SNA) [Mead et al., 2000] method is such an example. SNA assumes that selected intrusion scenarios are used to address the survivability of essential functionalities (services). Therefore, the quality and comprehensiveness of the set of intrusion scenarios depends on the skills of the analysts conducting the process. Possibly we can categorise the processes of evaluating survivability qualitatively into four types:

1. Model-based approach - constructing mathematical models that abstract the system by encapsulating key features of the system and then analysing these models.
2. Tests - Part of software development process, the components should go through

tests to validate their functionalities.

3. Fault injection - Creation of situations (intrusion injections) that mimic possible attacks but may be difficult to arise without a real attack.

4. Read teams - Deploys teams of attackers that try to break the system. The attackers are usually consultants hired to perform this type of tasks.

In this thesis we are interested in analysis models, models used to quantify survivability. Therefore, we refer to analysis models as models that quantify survivability.

As mentioned before, this research is interested in quantifying the survivability of SCADA systems, therefore, we are interested in quantitative processes to measure survivability. Over the years the analysis models to quantify survivability has increased rapidly. However, most models seem to be an adaptation from the ones used in the dependability domain. *Dependability* [Avizienis et al., 2001] has a long and rich history of research on techniques for quantitative and model-based evaluation of dependable computer systems. Therefore, it seems logical to adapt these models to survivability. However, it requires further attention. In dependable models the faults are clearly defined and statistically predictable. By contrast, statistical predictability of fail rates in systems subject to malicious attacks is arguably difficult. [Nicol et al., 2004] presented a survey on the existing model-based techniques for evaluating computer systems dependability and summarise how they are being extended to evaluate system security and also survivability. [Qian et al., 2008] also presented a survey on the existing model-based techniques for evaluating system dependability and summarise how they could be extended to evaluate system security and also survivability. The work was concentrated on surveying model techniques and on challenges of adapting them to security and survivability.

A fair amount of analytical models in the literature aimed at measuring survivability [Goseva-Popstojanova et al., 2001; Heegaard and Trivedi, 2008, 2009; Liu and Trivedi, 2004, 2006; Liu et al., 2004; Trivedi et al., 2008], they describes survivability as a variation of other related domains: sometimes performability, sometimes as a composite of performability and availability, sometimes as an attribute of dependability, and sometimes as part of resilience. In other words, they consider survivability as an attribute of other domains that is applied to specific aspects of the system under evaluation.

This work sees survivability differently. A key aspect that differentiates survivability from the other domains is the provisioning of an acceptable level of service and the definition of essential services. In other words, survivability is about keeping the essential services running under any condition. A key attribute to evaluate the quality of service is performance. Therefore is expected that survivability analysis models focus on evaluating system performance. This practice is responsible for the confusion between survivability definitions, and the other concepts related to it. Domains such as availability, performability, reliability, and dependability also use performance as a metric to evaluate systems.

Indeed the use of performance measurement to quantify survivability has been applied widely. [Moitra and Konda, 2000] defined a model to quantify the survivability of information systems based on system performance. The model analyses system survivability based on costs of the services and their importance to the system. It quantifies survivability as the ratio between the system performance when under attack and its performance in normal state. The services have weights, which are defined according to the type of the service. The higher are the weights the more essential are the services. They also measure the worst degree of survivability that is possible to achieve without having the system to stop working. They have used Internet based incidents to quantify survivability of the systems such as Web and Mail servers. The two main limitations with this model are the manual definition of essential service and the failure to account for services interdependencies.

In [Sullivan et al., 1999] the authors have used control theory as a model to control an information system. They address the reactive element of survivability. If performance disruptions occur actions are taken to ensure the the system continue to meet their survivability performance requirements. The system must be adjustable when needed.

[Krings and Azadmanesh, 2005] presented a graph-based model suitable for helping the analysis of survivability of computer systems. The model transforms the system into a graph, and from the graph some scheduling models are applied in order to compute the survivability of the system. However, it was not clear how survivability was represented. Even though the examples provided are based on performance aspects of the system.

[Liu and Trivedi, 2006] proposed an analytical framework that combines performance and availability to quantify survivability. The performance model is a homogeneous CTMC model based on the $M/M/m/m$ queuing model [Ross, 2007]. It assumes memoryless inter-arrival and service times with m servers or processors and m jobs (requests). While the performance model handles performance measurements the availability model uses the notion of failures replacing arrival-time and service rates with failure and repair rates, respectively. The composite model evaluates the entire system as if the service model worked uniformly across all its services. Sadly, this assumption does not hold for most systems (SCADA and otherwise). Since services may behave differently based on their characteristics and roles in the system. Evaluating the system in terms of performance and availability without considering the system's heterogeneity gives an inaccurate system picture. Therefore, it is not flexible enough for use on different types of SCADA systems with a very diverse class of services.

[Heegaard and Trivedi, 2009] proposed another performance-based model that also takes into account dependency between services. The model uses techniques similar to the ones presented in [Heegaard and Trivedi, 2008; Liu and Trivedi, 2006] with the difference that it considers interdependencies between services. However, the proposed approach has an issue with scalability, as the state space becomes intractable as the network size increases. To overcome such limitation they applied a space decomposition approximation that assumes independence between the network nodes (services).

[Gokhale, 2006] surveyed the current models and evaluations techniques used to evaluate survivability. One of the author's conclusions was the lack of survivability models targeting information systems. The majority of work was intended to network survivability. This is not a surprise as the concept of survivability on computer systems is relatively recent [Ellison et al., 1997] when compared with the network field, which is traced back to the 1970s [Frank, 1974; Frank and Frisch, 1970]. [Bing-Lin et al., 2009] also presented a survey in the literature review of survivability and one of their conclusions and the need for further research on models that analyse survivability.

2.2.4 Security models

The analysis of survivability based on security aspects is addressed in the literature from different aspects: there are the models that use attack scenarios to compute survivability, the ones that look at the effects of attacks on the services and finally the ones that model attacks and attackers in order to attain survivability. In this section we discuss some relevant work covering these different approaches.

[Zhang et al., 2007] presented a quantitative analysis model for network systems based on attack graphs. The survivability depends of the system itself as well as the environment it is running on. The model is defined in three steps: the preparation step which gathers vulnerabilities information from Internet and set difficult parameters for them; the generation of the attack graph based on the vulnerabilities gathered; and the quantification of the network system survivability as the final assessment. For them survivability is the minimal cost function to compromise the system with respect to all possible intrusion scenarios via a system administrator, a human that confirms the level that system cannot survive. [Jha and Wing, 2001] use a model to quantify survivability which visualise the effects of attacks through scenario graphs. They aim the model helps systems architects on the design phase of the software life-cycle. The model is based on *Constrained Markov Decision Process* (CMDP) [Bolch et al., 1998], a generalisation of Markov chains, where the transition probabilities are dependable of the past history, it contains six steps: model the network, inject faults into it, specify survivability properties (fault, service), generate scenario graphs, perform analysis on reliability and latency and finally, perform a cost-benefit analysis.

[Bowen et al., 1999] proposed a model that rather than detect attacks after they occurred they detect malicious attacks in real-time. They provide a real-time event monitoring and comparison with events known to be unacceptable. They look at the system call level to detect not permitted behaviour on the sequence of system calls in the operating system. Then, they compare any deviation with the specified behaviour.

[Moore et al., 2001] proposed an approach for documenting attack information in a structured way. The intuition is that by having this information catalogued in common structure it would be easier to practitioners to identify and catalogue new attacks.

[Moitra and Konda, 2004] have devised a model to quantify survivability by evaluating the trade-offs between the cost of defence mechanisms and the expected survivability after a malicious attack. They have used the idea of *expected survivability* to measure the survivability of a system. They suggest three metrics for computing survivability. The first one is a ratio of the system's performance under attack to the normal system performance. The main issue is how to measure performance levels. A suggestion is to consider the different functionalities and services of the system separately, then perform an assessment comparing to what extent each functionality has survived after an attack. The second one uses weights to define the expected survivability. It defines weights to services due to their importance and criticality, then a summation of the weighted values with the degree of a service or function is compromised is computed. The third and last one considers only the worse degree of compromise that has occurred across all the functions and services of the system.

[Wang and Liu, 2009] proposed a new technique to evaluate survivability. They extended the definition of availability to propose a new definition of survivability. According to them, survivability is the capacity of the system maintain the integrity and availability of essential data, information and services, even when under attack. This definition is tailored to the system being evaluated, in this case a database system.

A different approach is described by [Merideth and Narasimhan, 2003]. They subdivided survivability taking into account how a system reacts to attacks. They argue that proactive survivability systems differ from reactive ones as pro-actively survivable systems may act to increase survivability, may to initiate recovery and even to adapt to survive. They use different metrics to evaluate such systems. For reactive survivable systems they use *window of vulnerability* - time period which an additional fault will result in an epidemic, *fault-detection latency, recovery latency, reactive fault-detection accuracy* - likelihood fault detections involve neither false positives nor false negatives. For proactive survivable systems, in addition to the equivalents to the reactive systems there is the proactive bonus metric which is the benefit of pro-actively knowing the propagation of faults, instead of waiting for their detections. They aimed to apply these metrics to the starfish system [Kihlstrom and Narasimhan, 2003], a proactive intrusion detection and intrusion tolerance system. [Jajodia et al., 1999] presented a model to analyse the recovery of database transactions compromised by malicious attacks.

[Trivedi et al., 2008] presented some stochastic techniques that took into account security and survivable systems. Security and survivability are seen as dependability attributes, even though they recognised the relationship and importance of survivability and security together.

In [Pal et al., 2009] the authors have used a more practical approach. They defined a group of five metrics based on the system under evaluation, the Joint Battlespace Infosphere (JBI) concept. Even though the metrics are created with the system in mind, they might be applied for different systems. They focus on the performance attributes of the system such as detection of 95% of large-scale attacks within ten minutes of attack initiation. To evaluate the system against these metrics they defined two attack teams (read and blue) to perform the actual attacks.

[Wang et al., 2003] have proposed a model based on Fuzzy Matrix Game (FMG) [Bector and Chandra, 2005] to analyse the survivability of network systems. They integrate fuzzy matrix game theory with strategy selection. Attacker and attacked select different fuzzy strategies to maximise its benefit (objective). Both sides do not have access to the each other's intentions.

More recently, [Chen et al., 2010] devised a model to evaluate the survivability of Service-Oriented Architecture (SOA) based systems. The model also introduced a new definition of survivability. The model is based on Hidden Markov Models and aims to provide a holistic evaluation of survivability. Even though they described the steps necessary to compute the state of each service, they did not provided a mechanism to compute the overall survivability.

2.2.5 SCADA and industrial control systems security

The security of SCADA systems and consequently the critical infrastructure monitored and controlled by them is paramount. Over the last few years the research community and the computer security industry started to recognise such importance. The growing number of vulnerabilities detected over the decade (see Figure 1.2) is a good sign that the industry is becoming aware of the challenges on the security of ICS/SCADA systems.

[Knapp, 2011] gives a good overview of the ICS/SCADA industry providing all necessary vocabulary to understand the domain. More importantly, he raises awareness for the security issues that SCADA systems are facing currently. Actually, there is a fair amount of research on introducing and raising awareness to the security issues of SCADA systems, more recently.

[Barnes and Johnson, 2004; Fernandez and Fernandez, 2005] presented an overview on SCADA systems and their vulnerabilities. In addition, they also provided some best practices on protecting systems from malicious attacks. In the field of attack and anomalies detection, [Mahmood et al., 2009] presents a model to analyse network traffic of SCADA systems in order to detect suspicious behaviours and consequently possible malicious attacks. [East et al., 2009] defined a taxonomy of attacks on DNP3 protocol, a protocol widely used on SCADA systems [Clarke and Reynders, 2006]. [Huitsing et al., 2008] defined a taxonomy of attack on MODBUS [Swales, 1999], a protocol also widely used on SCADA systems. [Fleury et al., 2009] defined a taxonomy of attacks against energy control systems, a type of monitoring and controlling system very similar to SCADA systems. Such taxonomies are important as they are used for creating new rules on firewall systems.

[Kobayashi et al., 2009] presented an analysis of malicious traffic on the MODBUS/TCP protocol. [Svendsen and Wolthusen, 2009] presented some statistical analysis techniques to detect anomalies on SCADA systems. They tested their technique on liquefied natural gas production. [Cheung et al., 2007] described a model-based approach for intrusion detection. The idea was to define models that represent the expected and acceptable behaviour of services and then detect changes on the behaviour. Because a SCADA system tends to have fix topologies and regular traffic patterns, and a limited number of protocols deployed it seems natural to use the behavioural modelling technique as the number of false positives have a tendency to be very low. In Chapter 4, we presented a model to detect variations on SCADA systems by monitoring services performance in terms of service processing time. Even though this is not the main intuit of the model, it can detect certain types of attacks that affect the performance of system services. Also, in Chapter 5 we used the same intuition to find essential services by considering network traffic patterns.

From a more practical aspect, as mentioned earlier there is the QuickDraw SCADA IDS project, which extends the existing Intrusion Detection Systems (IDS) by adding signatures to SCADA-based protocols such as MODBUS and DNP3 [dig, 2012].

For security assessment, [Ten et al., 2008] proposes a vulnerability assessment framework to evaluate the vulnerabilities of SCADA systems at three different levels: system, scenarios, and access points. They also provided an assessment model based on attack trees [Ten et al., 2007]. [El-Sharkawi, 2002; Holmgren, 2004; Parks and Rogers, 2008] also provide vulnerability assessment of critical infrastructure systems.

[Anwar and Campbell, 2009] proposed an automated assessment of security compliance of power grids monitored by a SCADA system. The security model proposed uses predictive calculus to express the entities (e.g. devices, services) of the system (power grid). Then, it matches the entities with the best practices (rules) that defines constraints on the entities. [Cheminod et al., 2009] described their model of detecting vulnerabilities in industrial systems (such as SCADA systems). The model aims to detect chain reactions in the system.

[Fovino et al.] and [Masera et al., 2008] provided a case study of security assessment of a critical infrastructure system. They used a service-oriented approach that focuses on the analysis and interactions between services [Masera and Fovino, 2010]. The general idea of capturing the (inter)dependencies between services is similar to the one we introduce in Chapter 4, however their process is manual and only works in terms of vulnerabilities. In addition to that, they do not support any type of inference. [Fovino and Carcano, 2009] proposed an architecture to make SCADA systems security and survivable. Essentially they proposed changes on current protocols; encryption, filtering, checking for malformed packets, and the deployment of firewalls with SCADA-based rules. They also presented a prototype to validate their architecture.

[Genge et al., 2011] provided a model to analyse attacks on cyber-physical systems. They provided a framework that incorporates simulators of physical devices based on MATLAB/Simulink and software components simulated on a network test-bed. This work is similar to the simulator built for creating SCADA simulations presented in Chapter 3.

Chapter 3

SCADASim - A simulator for SCADA systems

SCADA systems control and monitor industrial and critical infrastructure systems. As mentioned earlier it is important to analyse the security risk and develop appropriate security solutions to protect such systems. However, a key problem is the lack of proper modelling tools to evaluate the security of SCADA systems. As widely accepted in academic and industrial communities, it is impractical to conduct security experiments on live systems. A modelling simulation tool would enable the simulation of SCADA systems with the benefit of testing different attack and security solutions. This chapter introduces a simulation tool for building SCADA simulations that supports the integration of external devices and applications. A key benefit of such tool is the ability of testing the effect of attacks on real devices and applications, even though using a simulated environment. The chapter also describes two case studies that demonstrate how the tool can be efficiently used to create SCADA simulations and inject malicious attacks.

3.1 Introduction

There are few instances of active SCADA test-bed and simulation development [san, 2012; Cheung et al., 1997; Christiansson and Luiijf, 2007; Davis et al., 2006; Fovino

Part of this work appeared in IEEE Transactions on Smart Grid et al., 2009]. However, these tools are either proprietary, used by researchers within the organisation, and the software is not released for external use or it is not generic enough to support different architectures, protocols and systems. Another important issue regarding the successful modelling of a SCADA system is how to control real devices, such as sensors and actuators from within the simulation environment.

In this chapter we introduce SCADASim, a framework for building SCADA simulations. It provides a modular SCADA modelling tool that allows real-time communication with external devices using SCADA protocols. This work builds on the basic simulator we proposed previously [Queiroz et al., 2009]. It adds value on top of our previous work by making SCADASim truly flexible for connecting real and simulated devices, and also offering a plug-n-play solution that supports multiple protocols and external applications for building SCADA simulations.

SCADASim aims to provide:

1. *A modular, extensible and flexible tool to model SCADA simulations*. SCADASim provides a set of modules that represent SCADA components such as RTU, PLC, MTU and protocols (such as Modbus/TCP and DNP3). Such modules and protocols can be easily extended, compounded into other modules and used into any SCADASim simulation.

2. *The integration of external components into the simulation simultaneously*. SCADASim introduces the concept of gates. A gate is an object that links the external environment with the simulation environment. SCADASim simulations may have multiple gates communicating with different external environments at the same time.

3. *The possibility of testing attack scenarios seamlessly*. SCADASim supports four main type of attacks: *Denial of service, Man-in-the-middle, Eavesdropping and Spoofing*. Users can easily create attacks to run inside the simulated environment or they can use tools such as *trinno* [Dittrich, 1999] to run attacks against the external components. SCADASim also provides a library with some well-known attacks such as worm attack and Distributed Denial of Service (DDoS) attack.

A key benefit of SCADASim is its high scalability while integrating both real hardware and prototype software modules. In addition to that, the integration of existing external components into simulations simplifies evaluation of these components since topologies, traffic patterns, and other parameters can be changed easily within the tool. This paper evaluates SCADASim by simulating two smart grid scenarios. The first scenario simulates a network of smart meters connected to an energy utility provider. The second one demonstrates a wind power farm plant. The plant is monitored by a SCADA system that controls the generation and distribution of electricity. Malicious attacks are evaluated in both scenarios.

SCADASim aims to provide an environment to build flexible SCADA system simulations efficiently. As shown in Table 3.1, it allows multi-protocol integration with external devices and applications.

Table 3.1: SCADA simulation functionalities across different implementations

	Req. 1	Req. 2	Req. 3	Req. 4
OMNET++ [Varga and Hornig, 2008]	√	–	√	–
PowerCyber [Hahn et al., 2010]	–	√	–	–
VPS [Bergman et al., 2009]	–	–	√	–
SCADASim	√	√	√	√

SCADASim is built on top of OMNET++ [Varga and Hornig, 2008], a discrete event simulation engine. OMNET++ consists of modules that communicate with each other through message passing. The communication occurs either through input and output gates or through direct messages. Modules can also be combined in a hierarchy of levels making possible to build complex simulation components. It also provides a set of tools for designing network topologies (NED language and editor) and supports architecture of plugins extensions. Plugins can be used to modify the default behaviour of the simulation engine. For example, a different message scheduler can be added to a simulation, which changes the default behaviour of the scheduling of messages.

To choose OMNET++ as simulator engine we evaluated few other engines such as NS2/NS3 [ns 3 maintainers, 2011]. OMNET++ was chosen because it is a generic simulation engine and it allows plug-n-play through its NED WYSIWYG editor and also allows integration to external applications and devices.

According to [Mayer and Gamer, 2008] there are three ways of integrating an OMNET++ simulation with external applications: *sockets, source code, and shared li-*

braries.

- *Source code* - In the case of source code integration, the application source code would have to be compiled together with the simulation, so that a final simulation binary would be generated.
- *Shared library* - In this case a dynamic or static library previously compiled for the target platform would have to be linked to the simulation code.
- *Sockets* - Using sockets we can create an object that acts as a server proxy to the external world within OMNET++. The proxy would maintain a listening socket waiting for external connections and would be responsible to delivering messages from the simulation components to the external world and vice-versa.

From these three options a socket based integration is the only one capable of integrating external both devices and applications. The socket integration is seamless without requiring changes to the external components, does not require shared library files or source code that are either not practical or available for commercial and legacy SCADA systems. However, it suffers from one major drawback, only socket-based protocols are supported. This also means that we could not have gates communicating to devices through serial connections, for instance.

Because of these issues we introduced a new integration option. Gates allow the integration with different type of protocols (such as socket-based, serial, or many others). This flexibility allows users to create simulations using socket-based protocols such as Modbus/TCP, serial ones such as Modbus Serial and so forth.

3.2 Architecture

The SCADASim architecture is based on three main components: *SSScheduler*, a real-time scheduler; *SSGate*, a communication port that implements protocols for communication to the external environment; *SSProxy*, a simulation object that represents an external component within the simulation environment.

Figure 3.1 illustrates the various components of the SCADASim architecture. Gates are the only components able to communicate with the external environment. Proxies communicate to the external environment through Gates. SCADASim allows multiple gate instances each representing communication connections to the external environment. For instance, one may have two gates one listening to socket connections and another one listening to serial connections. The gates are managed by the *SSScheduler*. Next, we describe the operation of each of these components in more detail.

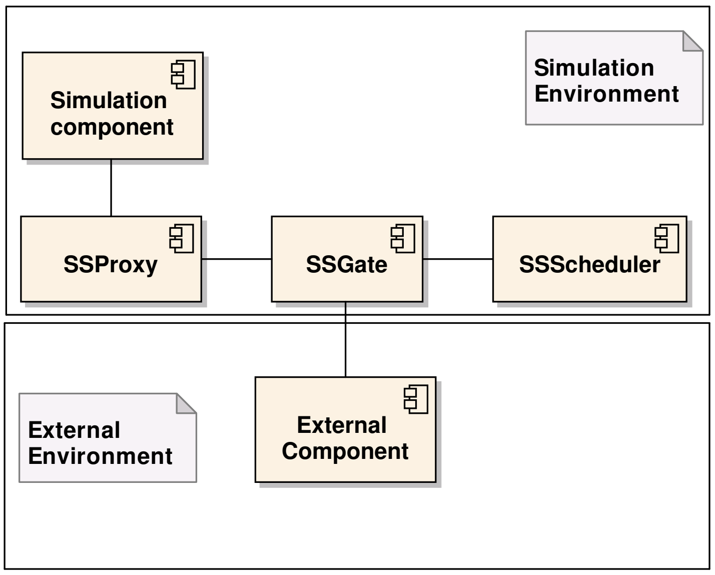


Figure 3.1: SCADASim component architecture

OMNET++ offers a plug-in scheduler architecture. Users can add new schedulers that address needs not provided by OMNET++. New scheduler implementations can be added by implementing the *cScheduler* abstract class. The *SSScheduler* extends *cScheduler* and adds new functionalities to control and synchronise messages received from the external environment. The class diagram described in Figure 3.2 shows the relationship between the *SSScheduler* and the other components in the SCADASim architecture. The *SSScheduler* manages a list of *SSGate* instances that are responsible for sending and receiving messages to and from the external environment. Every time the simulation engine will process an event the scheduler loops through all active gates getting their messages and adding them to the simulation queue to be processed and consequently delivered to their destination. The synchronisation process guarantees that the messages between the two environments will be synchronised.

The *SSGate* provides the connection to the external environment by implementing a protocol, which is used to communicate with external SCADA components. At the moment, SCADASim provides three gate implementations: *ModbusGate*, *DNP3Gate*,

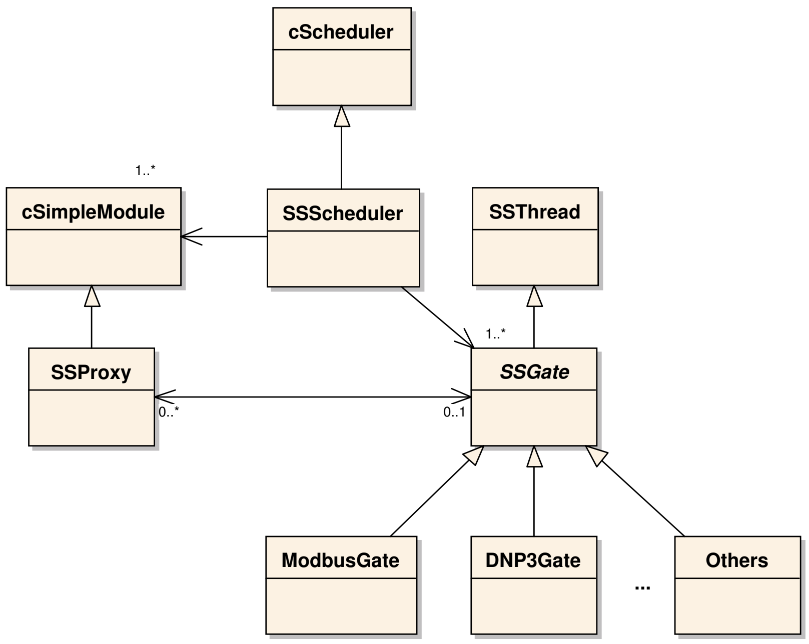


Figure 3.2: SCADASim class diagram

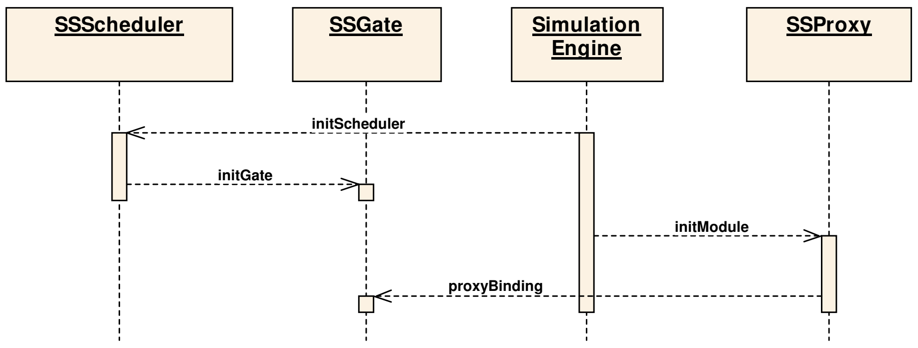
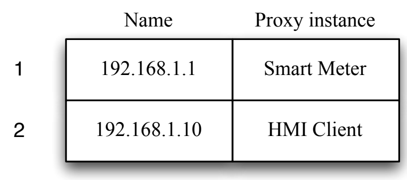


Figure 3.3: Sequence diagram - Gateway startup

HTTPGate. However, programmers can add implementation of other protocols by extending the *SSGate* abstract class (see Figure 3.2).

Gates run as system threads so that the framework can handle multiple instances simultaneously. Consequently, satisfying one of the framework requirements.

A *SSGate* instance maintains a list of registered proxies. When a proxy is initialised, it registers itself with the gate by calling the *proxyBinding* method on the gate class (see Figure 3.3). The gate then adds the proxy to an internal table as illustrated in Figure 3.4). So that, the gate keeps track of the external address and objects associated with the proxy during the simulation. *Row 1* in Figure 3.4 shows that a smart meter proxy is representing an external component that has IP address 192.168.1.1.



	Name	Proxy instance
1	192.168.1.1	Smart Meter
2	192.168.1.10	HMI Client

Figure 3.4: Proxy binding table

For a gate instance to be used by the simulation it need to be registered with the simulation engine. The attribute *extgates* must be added to the simulation configuration file (*omnetpp.ini*) where all registered gates are separated by commas. The *SSScheduler* reads this attribute when it initialises.

The *SSProxy* represents a real device or external application in the simulated environment. They interact with simulated objects such as simulated MTUs and with a *SSGate*, which is responsible for routing their messages to external components. There is a relation of one to one between a real component (hardware device or application) and a proxy object.

Proxies are linked to external objects through their ‘names’. Every proxy must have a name that identifies which external object it is representing. For instance, a proxy with name 192.168.1.1 represents an external object with this IP address. It is responsibility of the simulation creator to configure the proxies names in the *omnetpp.ini* configuration file. The gate matches the source IP of an external request to the proxy name of an object in its internal proxy table. This is how the gate knows how to route their messages to the right proxies.

It is easy to create proxies and simulated objects within SCADASim over OMNET++. Objects are created by a composition of modules. Figure 3.6 illustrates a smart meter simulation object built through module composition. The device has PPP and Ethernet interfaces, supports IP, TCP and Modbus/TCP. The entire smart meter component is built without writing any code thanks to the OMNET++ NED editor.

SCADASim provides built-in modules such as RTU, PLC, MTU and modules that represent type of attacks, for instance a DDoS module.

SCADASim handles protocols in two distinct ways: the ones used within SCADASim/OMNET++ environment to communicate to other simulation components in the same environment, here called *simulated protocols*, and the ones used by external devices and applications to communicate to *SSGates*, called *original protocols*. Figure 3.5 illustrates an example of a typical SCADASim simulation. External devices communicate to *SSGates* using the original version of SCADA protocols, for instance, an external PLC could use Modbus/TCP to communicate to a *SSGate* that implements Modbus/TCP (*ModbusGate*). In the simulated environment all communication among the components use the simulated versions of the SCADA protocols. These changes are necessary because the simulation does not provide the environment where the original protocol was designed to run.

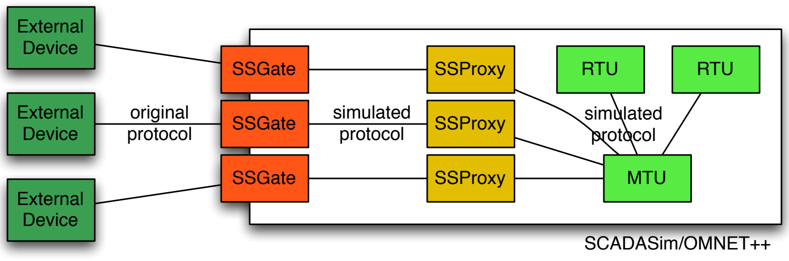


Figure 3.5: Simulation architecture example

For instance, the Modbus/TCP simulated protocol uses sockets, however the socket stack cannot be used to send messages among the components of the simulation, as OMNET++ uses another technique. Thus the adapted version of Modbus/TCP had the socket stack replaced by INET [INET, 2012], a OMNET++ library that simulates the TCP/IP protocol.

SCADASim provides a library of SCADA protocols (simulated and original) to be used in simulations. In the current version we have ported Modbus/TCP (adapted from

[Raimbault, 2010]), DNP3/TCP (adapted from [Crain, 2011]) and HTTP protocols to SCADASim.

Figure 3.6 shows how a protocol can be used as part of a simulation component. For example, the Modbus/TCP protocol module is connected to the TCP module provided by the INET [INET, 2012] library, making the smart meter capable of communicating with other components using the Modbus/TCP protocol. All protocols available in the protocol library can be used by any SCADA simulation without the need of writing any code.

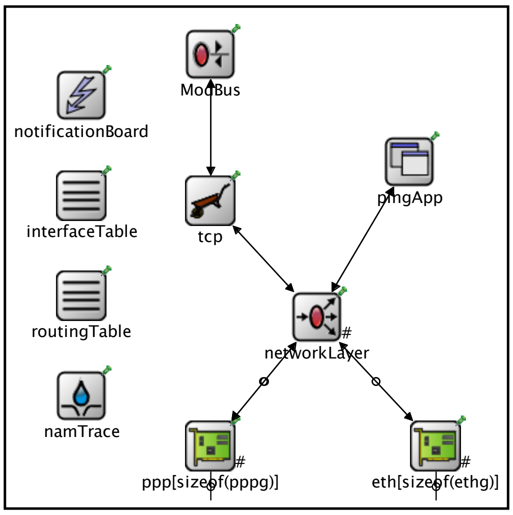


Figure 3.6: Simulated SCADA device

It is possible to use proprietary protocol implementations (with no changes) to exchange messages between external devices and applications to SCADASim gates, however as mentioned earlier communication protocols need to be adapted to run within SCADASim simulations. Proprietary protocols are no different, they would need to be adapted in order to be used in SCADASim simulations.

Figure 3.3 illustrates the startup process of SCADASim. The *SSScheduler* class is initialised by the OMNET++ simulation engine and registered as the simulation scheduler. During its initialisation the scheduler instantiates all *SSGate* implementations registered to it. Then, the simulation engine initialise all proxies and simulation components. During the initialisation proxies register themselves to the *SSGate* they would like to bind to.

The proxy bootstrap process is initiated when a proxy starts communicating with an external component for the first time. There are two ways for a proxy to start communication with external components: *it sends a request or it receives a request*. These two processes are handled quite differently by the *SSGate*, as described below.

When the *SSGate* receives a request from an external component to route a message to a proxy it needs to know to whom the message is addressed to. It is important to remember that external components may only have the IP address and port (in case of sockets) of the *SSGate*.

The *SSGate* lookups its *Proxy Binding Table* and finds out which proxy name matches to the IP address of the message sender. If a match is found the *SSGate* creates a message using the simulated protocol of the original version and sends the message to the proxy. Protocols that do not rely on IP addresses have to use another technique to match proxy names.

The *SSProxy* creates a message using the simulated protocol and delivery it to its *SSGate*, the gate creates a request using the original protocol and sends the message to the final destination.

3.3 Simulating malicious attacks

In addition to be a framework to create SCADA system simulations, SCADASim also supports the creation of malicious attacks targeting SCADA systems.

In [Sun Microsystems, 2000], authors categorised attacks (threats) into four main types as illustrated in Figure 3.7. Denial of services are attacks where the attacker (Bob) denies source (Alice) access to the destination (Mary). Man-in-the-middle are attacks where Bob impersonates Mary, consequently, receiving all requests from Alice to Mary. Bob then, changes the requests and relays them on to Mary. Spoofing are attacks where Bob masquerades as Alice, so that it can create and send requests to Mary. Eavesdropping are attacks where Bob receives all requests sent to Mary by Alice, even though Mary and Alice have no knowledge of that.

SCADASim supports the all four type of attacks. However, SCADASim does not provide implementations of all type of attacks to all SCADA protocols. There are more than 150 protocols used by SCADA systems [Igure et al., 2006]. We have decided to initially support the most used protocols such as Modbus/TCP [Swales, 1999] and

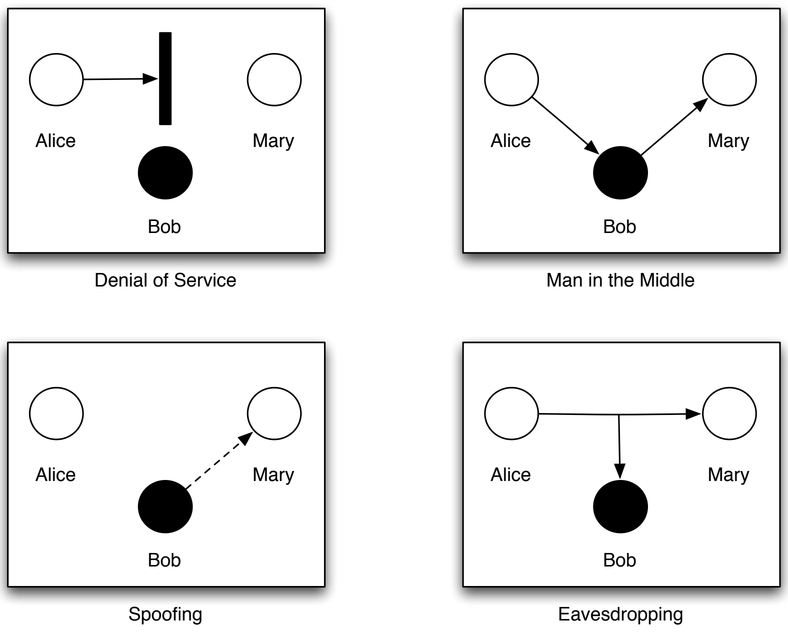


Figure 3.7: Attack categories

DNP3 [Group, 2011]. According to [Clarke and Reynders, 2006] DNP3 is predominant in the US energy sector with over 75% of electric utilities using it in their SCADA applications. As mentioned earlier SCADASim users can add their own protocol implementations.

As SCADASim supports the integration of external applications and devices into simulations, there are three possible attack scenarios supported by SCADASim:

1. Attacks launched from within SCADASim targeting SCADASim internal components - it is the most common one, where you have all components within SCADASim, even though they may represent external components. An user can launch attacks and monitor their behaviour easily from the controlled environment.

2. Attacks launched from the external environment targeting an internal SCADASim components - the only reason to launch these type of attacks is when the user has no access to the source code of the attack making it impossible to adapt the code to run within SCADASim.

3. Attacks launched from the external environment targeting an external component.

There are so many sorts of attacks targeting SCADA systems with some researchers listing more than 50 different type of attacks targeting DNP3 [East et al., 2009] and Modbus [Huitsing et al., 2008]. These are attacks against protocols specification, technically supported by SCADASim, even though we have not implemented all of them.

3.4 Evaluation

To evaluate SCADASim we demonstrate two common attack scenarios that show how malicious attacks can be implemented in SCADASim. The first scenario is a *denial of service attack*. Attackers launch flood attacks against a smart meter network. The second one is a *spoofing attack*. An intruder connects to the local field network to send fake messages to the PLCs as they were coming from a real MTU.

3.4.1 Denial of service attack scenario

In this scenario a "smart" house is connected to the grid distribution company. The home updates its energy consume frequently to the company servers. With this information the utility provider can predict consuming scenarios and consequently provide more efficiently, and in same cases cheaper, electricity to their clients. Figure 3.8 illustrates the scenario. The SCADA system controls and monitors smarter meters installed at homes. For the sake of simplicity the scenario presented only shows one home connected to the grid. The scenario consists of a smart meter gathering information from energy consume in a "smart" home. The house is equipped with a smart meter that informs home spending to the utility company automatically. To simulate the smart meter we have implemented an application that simulates the electricity consume and sends it to the SCADASim simulation.

3.4.1.1 Simulation environment

The network topology used in in the simulation is based on a real SCADA network. The same components are utilised; RTUs (Smart meter), MTU, HMI Server and HMI client. The network topology consists of Local Area Networks (LAN) connected with each other through the Internet. Different network topologies arise simply from the differences in the number of "smart" houses being used. Figure 3.9 shows a generic SCADA topology used in our simulation. The upper left part of the figure shows the

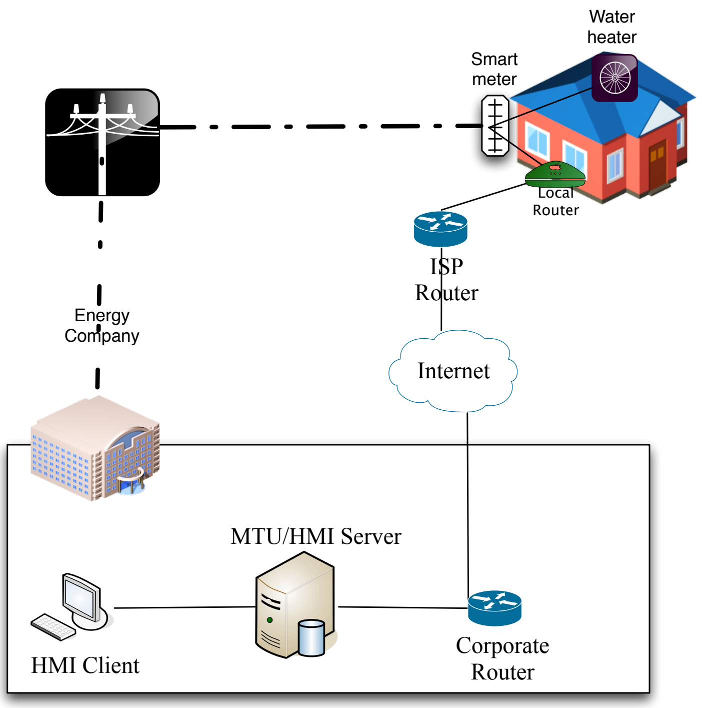


Figure 3.8: SCADA system network of a smart meter deployment

“smart” house with its RTU (smart meter) and its router connected to the ISP router. The right part of the figure shows the corporate network consisting of a MTU, an HMI server and an HMI client. The smart grid network and the corporate network are connected to each other via routers over the Internet.

In this environment all messages exchange between the MTU and the smart meter proxy are implemented by Modbus/TCP protocol [Swales, 1999], a widely used SCADA protocol. The Modbus/TCP version used in the simulation is a port from the LibModbus library [Raimbault, 2010]. The adapted version is part of the SCADASim library of simulated protocols.

The external application communicates to the SCADASim simulation through Modbus/TCP, therefore, the *ModbusGate* translates the request to the simulated Modbus/TCP before to delivery it to the proxy.

3.4.1.2 Components

The simulation consists of six main components; *Smart Meter, Smart Meter Proxy, SSScheduler, MTU/HMI, HMI Client Proxy, and HMI Client*.

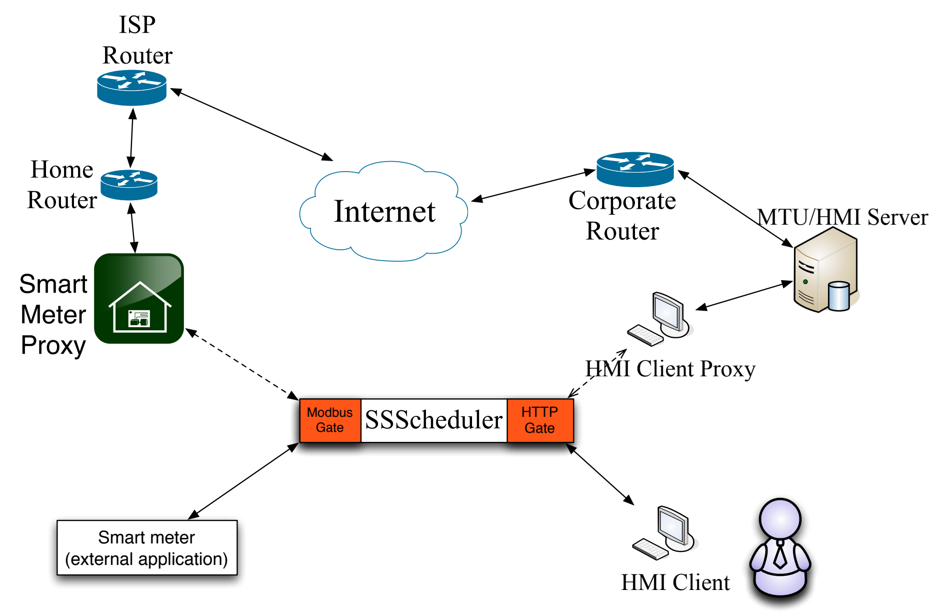


Figure 3.9: Smart meter network topology

3.4.1.3 Smart meter

The Smart meter is represented by the external application. It measures the power consumption of a fictitious home. However, for the sake of simplicity, the house only has a water heater being monitored.

3.4.1.4 HMI client

In real SCADA systems, the HMI client is used by operators to monitor and control the system. We have developed a real but simple HMI web application that runs as part of the *HTTPGate*. The operations executed on the web application are real control actions performed by human operators. In order to communicate to the simulator the HMI client (represented by a Web browser) sends HTTP requests to the *HTTPGate*, which then translates them into simulated HTTP messages and passes them on to the HMI Client proxy that then passes it on to the HMI Server.

3.4.1.5 Smart meter proxy

The Smart Meter Proxy represents the smart meter (implemented by the external application) inside the SCADA simulation. It contains all the information provided by the smart meter, in this case the energy consumption. It reads its information by receiving Modbus/TCP requests from the external application (smart meter).

3.4.1.6 HMI client proxy

The HMI Client Proxy represents the HMI Client application inside the simulation. Every request coming from the HMI Client goes through the *HTTPGate*, and then it is passed on to the HMI Client Proxy, which then sends it to the HMI Server. Following the reverse route, the responses are routed from the HMI Client Proxy to the *HTTPGate* then to the real HMI Client.

3.4.1.7 SSScheduler

The *SSScheduler* module is responsible for synchronising the clocks between both environments (SCADASim and external) and to control the active gates (*SSGate*) running in the simulation.

3.4.1.8 MTU/HMI server

The MTU is responsible for controlling and monitoring the SCADA system. It performs the same activities performed by the real MTU device. It communicates with the RTUs to gather information about the network and to request actions to be performed on the network. The HMI Server is responsible for providing reports on the SCADA system for human operators. The HMI server receives requests from and provides responses to the HMI client. In this simulation the RTU is represented by a smart meter.

3.4.1.9 Results

In this scenario a TCP SYN flooding attack is performed. The intent is to disrupt the normal operation of the smart metering network. For instance, operators would get delayed information about power consumption of the monitored houses. Based on such delayed information the operator is likely to take wrong actions that could adversely affect the operation of the SCADA system. To perform the attack we have used a modified version of the TFN (Tribe Flood Network) tool [Dittrich, 1999].

Figure 3.10 shows the effect of the DDoS attack on the smart meter. The attack occurs within in the SCADASim environment. It starts at time = $700s$ in the simulation timeline by ten attackers, each sending 1000 TCP SYN packets to the smart meter. The time between 0 and 700 seconds $(0 \le T \le 700)$ displays the normal behaviour of the smart meter. As we can see in Figure 3.10 during this time the smart meter receives a constant number of requests and process them normally. At *Time = 700* (red circle) the smart meter starts to receive an enormous amount of packets requesting new TCP connections. From this time on the smart meter starts to show an unusual behaviour. There is a huge drop on the number of processed requests, even though the processing time does not change. Such behaviour indicates that many legitimate requests are not being processed at all, they are just being dropped by the smart meter.

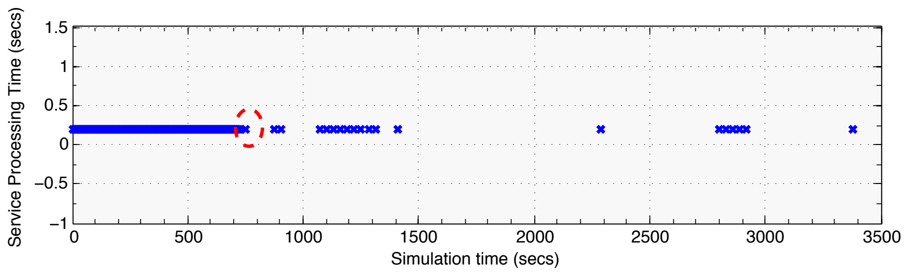


Figure 3.10: Smart meter under attack

Looking at the behaviour of the MTU, Figure 3.11 shows that when the attack starts the service perceived time spikes up dramatically. Clearly, it demonstrates that the attack is affecting the smart meter responses to the MTU.

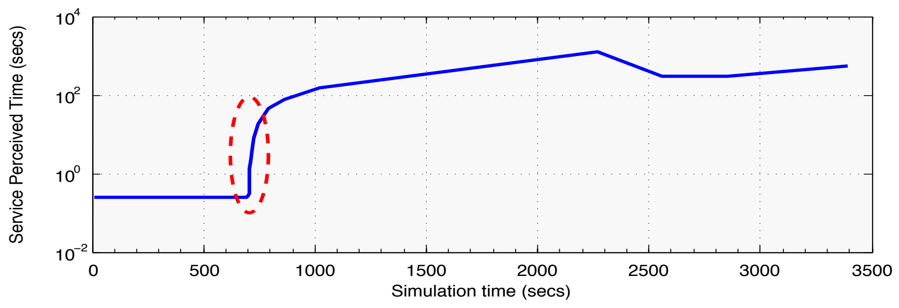


Figure 3.11: MTU perceived service time

3.4.2 Spoofing attack scenario

This scenario shows how a wind power plant simulation can be integrated to SCADASim. We also shows how a spoofing attack can compromise the wind power plant by injecting fake requests to the PLCs that control the power plant. The wind power plant is represented by a MATLAB/Simulink model (adapted from Simulink software demo - see Figure 3.12). The model consists of a wind farm of six 1.5 (MW) wind turbines connected to a 25 (kV) distribution system that exports power to a 120 (kV) grid through a 25 (kV) feeder. The turbines are simulated in pairs, therefore, in total the model has 3 pairs of turbines that together generates 9 (MW) of electricity. Each wind turbine pair has a protection system that monitors the voltage, current and speed. The reactive power is absorbed by induction generators and the rest of the reactive power required is provided by a 3 (Mvar) *STATCOM*.

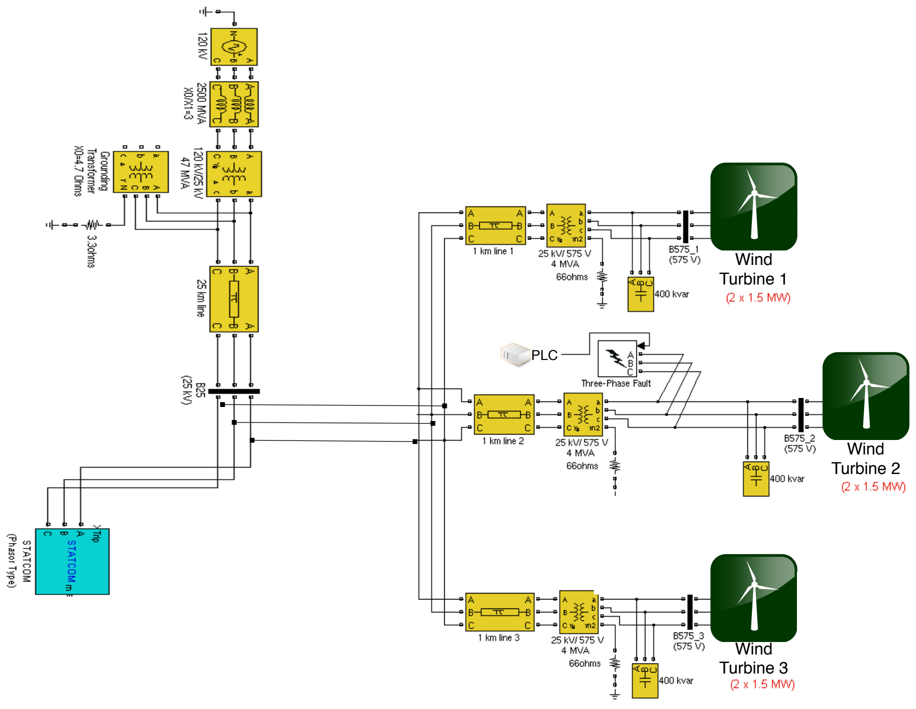


Figure 3.12: Wind farm power plant

3.4.2.1 Simulation environment

Figure 3.13 shows the SCADA network topology used in the simulation. The network consists of a field network connected with the utility provider corporate network through the Internet. The down left part of the figure shows part the smart grid generation and distribution simulation, more specifically, it shows the wind turbine pair number two, which will be attacked by a spoofing attack. The wind turbine has more PLCs monitoring its functioning, however, we chose to only show the PLC that will be targeted by the attack. The PLC is simulated by a MATLAB/Simulink *s-function*, a mechanism offered by MATLAB/Simulink where external code can be executed from the MATLAB/Simulink environment. In this case the function sends requests to the SCADASim PLC proxy informing the status of its monitored component, a three-phase fault block, and updates its status based on the response. The status value is passed along to the fault block. The communication between SCADASim and the PLC is made by the Modbus/TCP protocol (LibModbus library [Raimbault, 2010]).

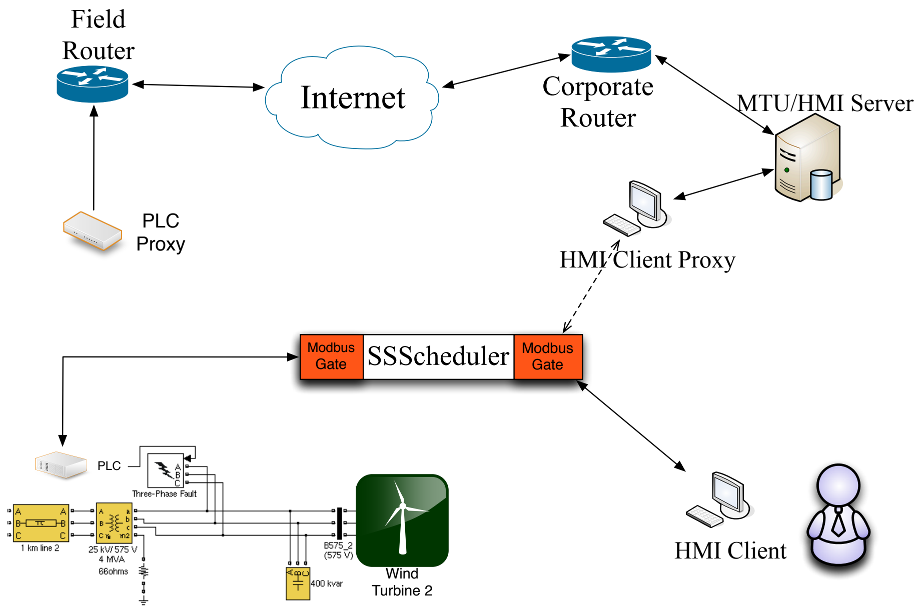


Figure 3.13: SCADA network topology

The attack involves connecting a computer to the communication link. So that, the attacker can read, modify and fabricate Modbus/TCP messages and/or network traffic easily. In this case the attacker will send a message to the field PLC, which monitors and controls the three-fault block, to generate a fault in the circuit. The fault will make the wind turbine pair number two unavailable, and as consequence the other turbine pairs (number one and three) will also become unavailable. Therefore making the entire energy generation and distribution to halt. Before the fault the turbine pairs are generating around 3 MW of power each (time < 26s). At simulation time of $26s$ the attacker sends a message to the PLC that makes the fault block to provoke a fault into the circuit. The fault cascades to the other turbines pairs as can be seen in Figure 3.14, the malicious attack breaks down the entire wind power plant as the turbines stop generating any power, even though only one pair of turbines were attacked.

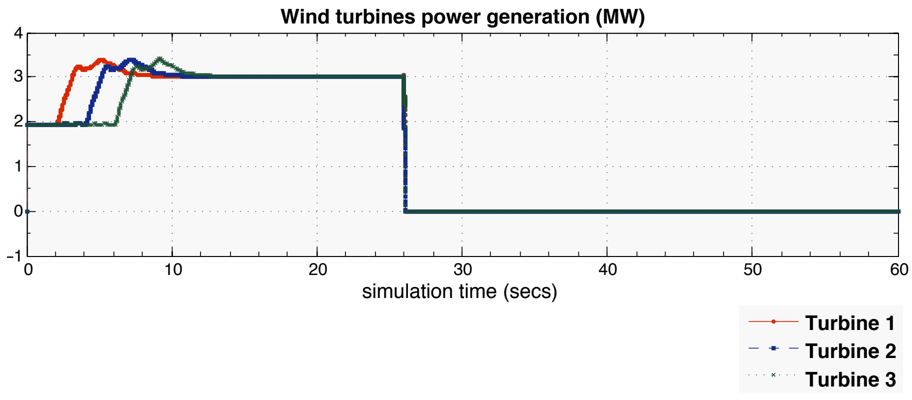


Figure 3.14: Spoofing attack scenario

3.5 Conclusion

SCADASim is a solution to create realistic simulations of SCADA systems based on a combination of network simulation and real devices connectivity. It solves a number of logical and implementation problems in the widely used OMNET++ simulation environment to allow real world devices (such as smart meters, RTUs, and so forth) to be attached to the simulator. The simulation framework, SCADASim, also allows us to study the effects of malicious attacks on the devices and on the simulated network. As security issues affecting SCADA systems become mainstream the need to have systems where one can try and test different scenarios in order to provide better solutions for current systems is immense. In addition to that, the lack of network traffic datasets on SCADA systems make security research on such systems harder. Therefore, a simulator that can provide a close to real simulations where data can be generated and consequently studied is very much appreciated.

Chapter 4

Survivability - A holistic approach

This chapter introduces a new approach for survivability. Holistic Survivability aims to fill the gap left by current survivability models that do not consider service interdependencies, are not flexible for supporting simulation based and data-driven models and do not provide an algorithmic approach for identifying essential services. In addition to provide a holistic view of the system. Holistic survivability focuses on four aspects: *joint service analysis*, *identification of essential and non-essential services*, *performance and security evaluation* and *empirical survivability analysis based on synthetic or acquired data*. The chapter also presents the formalism in which holistic survivability is built upon.

4.1 Introduction

Despite the fair amount of work on engineering reliability in the literature during the 1990s, researchers and practitioners realised that reliability was not sufficient for the analysis of computing systems [Voas et al., 1997]. Survivability was proposed as a new discipline, and in some ways as an answer for that insufficiency. In its seminal work [Ellison et al., 1997] introduced survivability analysis for computing systems with the idea of improving the security and fault-tolerance aspects of such systems, aspects that were lacking on current (at that time) analysis models based on reliability.

Part of this work appeared in IEEE Global Telecommunications Conference - GLOBECOM 2010

[Ellison et al., 1997] derived four key attributes for survivability: *Recognition*, *Resistance*, *Recovery* and *Adaptation*. The notion of these attributes is that the more a system is capable of recognising, resisting, recovering and adapting from undesired events the more survivable is the system. We believe that in terms of analysis, recovery and adaptation can be seen as only one attribute. Below we discuss these attributes in more details.

- **Recognition** - It is the capacity of recognising undesired events. The intuition is that if one can recognise undesired events, one can therefore be more effective in keeping the system survivable. In the case of malicious attacks, this is usually done via intrusion detection systems (IDS). Until recently there was not much tools on detecting intrusions for SCADA systems. Even though SCADA systems are now using the same type of COTS hardware and software already used by the IT industry, the protocols used by SCADA systems are different. Initiatives such as *QuickDraw SCADA IDS* [dig, 2012], which extends the existing IDS by adding signatures to protocols utilised by SCADA systems are changing this scenario. It is not the purpose of holistic survivability to recognise malicious attacks. However, it helps to find out how much damage in terms of disruption a malicious attack is able to carry out. By inferencing future scenarios holistic survivability can show the effects of malicious attacks on a SCADA system.

- **Resistance** - This relates to the capacity of resisting to undesired events. It is about the robustness of a system. If the system does not have vulnerabilities, its communication is secure, there are policies for authentication and authorisation in place and so forth, then it is harder for adversaries to perform malicious attacks. In the case of other undesired events, such as hardware and software failures, which are not related to malicious attacks, the fields of dependability, robustness, reliability and so forth, already provide a fair amount of research on such events [Al-Kuwaiti et al., 2009; Bologna et al., 2003; Nicol et al., 2004; Reussner and Firus, 2008; Sanders, 2004]. To the case of holistic survivability, it is not its goal to provide actions that makes the system more survivable. Rather it is a model where one can try different scenarios and consequently find out the ones that maximise the resistance of the system by looking at the survivability score.

- **Adaptation** - This is a key aspect of any system. Consider the example of biology organisms, they keep adapting to the environment they live in. Adaptation is the key for their survival. For intelligent creatures such as human beings, the adaptation becomes even more interesting. We learn from our past experience and use it to adapt to new situations and environments we are facing in a particular moment. The importance of adaptation should be no different in computing systems, and for the matter, in SCADA systems. However, systems are not intelligent beings (just yet), we cannot expect the same type of adaptation. Instead, we should expect adaptation that is designed by humans to make a system survive. For instance, consider a multi-core computer that is running on full speed with all cores. If the computer starts to heat up to a certain temperature, some cores will turn-off automatically to decrease the generation of heat and consequently the CPU temperature. Obviously, users are comfortable with system shutting off an overheating core as long as it can extend the operating life of their systems. However, one aspect is to design and implement adaptation to computer systems, another is how to measure it. There is some work on measuring adaptation of systems. [Reinecke and Wolter, 2008] introduced an adaptation metric to Web services, *adaptivity*, which is the capacity of the system to adapt itself. [Chen et al., 2010] uses the time between transitions to unacceptable and from acceptable states to measure the capacity of adaptation of a service. [Sheldon et al., 2004] proposed a model that takes clues from biological organisms to improve the survivability of systems. It is based on the ideas of autonomic computing [Kephart and Chess, 2003] to self-adapt in order to survive. However, they did not propose any metric to measure their models.

[Ellison et al., 1997] also introduced the concepts of essential and non-essential services, consequently proposing a discriminatory survivability analysis of computing systems. The definition of essential services is simple. *Essential services are services that a system relies most in order to provide its functionalities*. In other words, they are the important aspects to be looked at in a system. Everything else should be considered secondary. More importantly, and perhaps the intuition that differentiates survivability from similar concepts such as reliability, and dependability is that system survivability should be evaluated in terms of such services.

Since then a vast quantity of definitions and analysis models and tools for survivability has been introduced. Such mass adoption of survivability created a problem, the amount of different survivability definitions. [Westmark, 2004] alone catalogued more than 50 definitions. The lack of a common definition on survivability produced the erroneous idea that survivability was nothing new, in fact for many, after many years survivability remains a confused mixture of multiple domain ideas. Domains such as: dependability, availability, performability, fault-tolerance, resilience, and so forth, are constantly being compared with survivability [Al-Kuwaiti et al., 2009; Avizienis et al., 2004a]. Chapter 2 presents a more detailed view on survivability, its concepts and analysis models.

[Al-Kuwaiti et al., 2009] did a comparative analysis of *Security*, *Dependability*, *Fault-tolerance*, *Reliability* and *Survivability*, in an attempt to provide common and complementary characteristics of these concepts. As result, they proposed a conceptual framework that integrates those concepts on the basis of their definitions, attributes, relationships and performance evaluation measures. However, they did not present any mechanism to quantify survivability.

We believe that two things are necessary to evaluate the survivability of a system: a precise definition of survivability, and an accurate way of measuring survivability at any given time. In this chapter we address these two issues by introducing a new definition of survivability, which is then followed by a formalism that captures the problems we aim to solve through the concept of holistic survivability. Fundamentally, holistic survivability aims to provide a measure of strength against malicious attacks in a holistic way. We summarised the problems that holistic survivability aims to tackle into four main topics:

- **Single service analysis**. Currently the analysis of survivability is frequently restricted to single service analyses. This means if one wants to analyse the survivability of two services in a system, two models are applied, using two different metrics and two analyses are performed. Another issue is that even though the analysis is clearly related to a single service, usually it is referred as the survivability analysis of a system.

- **Essential service identification**. Currently essential services are often either manually defined by system experts or not identified at all. Such manual identification approach, when exists, is appropriate to evaluating either parts of a system or small systems. However, it is intractable for complex systems with hundreds of services. Therefore, we believe that an algorithmic approach to identify essential services is necessary.

- **Lack of security analysis**. Current models analyse the performance of services in order to perform a survivability analysis. Usually, such analysis are based on metrics such as processing time, throughput, blocking calls rates and so forth [Heegaard and Trivedi, 2009; Jha and Wing, 2001; Moitra and Konda, 2000; Wang and Liu, 2009]. Such metrics are adequate but they hide an essential issue, the lack of security aspects on the evaluation. After all, if survivability definitions identify security as an important attribute, the metrics used to evaluate survivability should incorporate such attribute. Chapter 2 presented some related work on survivability and security. Even though the work presented introduces security analysis on survivability evaluation they do not provide a holistic approach that combines service (inter)dependencies and service discriminatory aspects. In addition, they are not flexible enough to handle analytical and empirical analysis.

- **Analytical analysis**. As discussed in Chapter 2, most of survivability models are aimed at providing analytical analysis of systems' survivability. This could have been enough in the past, however currently we are in an era where data is being generated in a huge scale. We believe that is time to provide models that could use the data acquired in the field to perform more intelligent survivability analysis.

In order to highlight the intent of holistic survivability and how it fits in terms of previous taxonomies on survivability we provided a concept taxonomy that illustrates the proposed approach on survivability. The taxonomy is shown in Figure 4.1, its focus is on the identification and assessment of essential services and their (inter)dependencies regarding security and performance aspects.

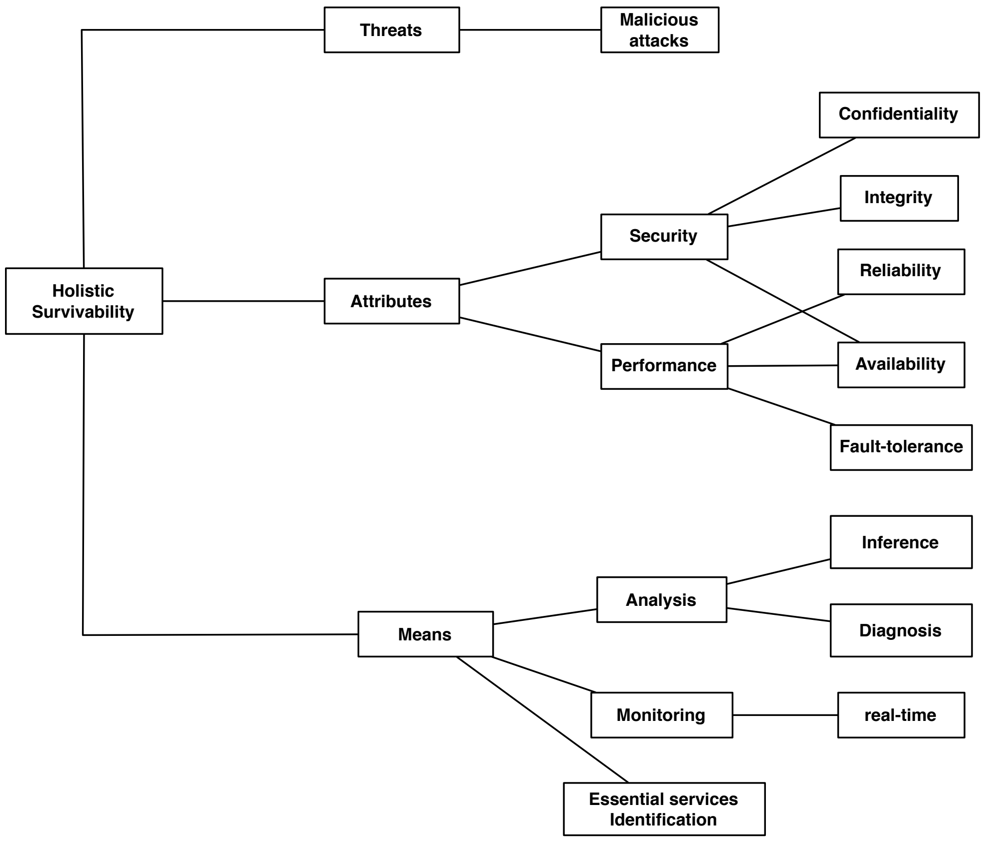


Figure 4.1: Holistic Survivability Conceptual Taxonomy

The proposed taxonomy is based on the three attributes previous discussed: Recognition, Resistance, and Adaptation. Those attributes are considered as the means to attain survivability. Translating them in terms of security, a survivable system should be able of recognising malicious attacks, resist to attacks, recover from damages caused by malicious attacks and finally, adapt itself to avoid more damage and new types of attacks. Clearly it seems a hard task to implement such attributes in the security context. The computer industry is trying for years to provide tools that would catch and prevent malicious attacks. Some advances has been made, but as mentioned in Chapter 1 we did not have such tools yet. On the other hand, such attributes have been widely studied on the context of failures and faults by the dependability community [Al-Kuwaiti et al., 2009; Avizienis et al., 2004a]. As discussed in Chapter 2, there is a fair amount of work on designing and implementing architectures in which survivable systems could be built upon. Such architectures provide mechanisms to the system to resist and to adapt to the changes in the environment. Also, there are some work on providing tools to help the identification (recognition) of undesired events. Unfortunately, nearly all of this work is not targeted to the recent and most challenging type of undesired events, **human made malicious attacks**. Such events are controlled and deployed by humans with the only intention of creating maximum damage as possible. Holistic survivability, in addition to support the current analysis of survivability, aims to bring a new approach to the fight against malicious attacks. It aims to analyse systems in terms of their security and performance.

Attributes can be seen as non-functional requirements of any computer system, however it is through attributes that survivability is accessed and quantified. Holistic survivability is based on two aspects: *Security* and *Performance*. Security includes attributes such as Availability, Confidentiality and Integrity, and performance includes Availability and Fault-tolerance. We believe that any measurement of holistic survivability should consider those attributes. The analysis models introduced in Chapters 6 and 7 use these attributes to quantify service states and consequently holistic survivability.

Services can be affected (compromised) in many different ways. [Avizienis et al., 2004a] described a taxonomy of threats that may affect systems during its lifetime. They characterised threats into three types: *Faults, Errors, and Failures*. Faults were classified into malicious and non-malicious. Even though all types of threats are important, we believe that the other types were exhaustively studied in previous work [Al-Kuwaiti et al., 2009; Avizienis et al., 2004a; Neuman, 2000]. Holistic survivability is interested in malicious faults, more specifically, malicious attacks intentionally caused. Most of work on malicious attacks has been on detection and prevention. As mentioned in Chapter 1 holistic survivability aims at keeping the essential services running even when the system is under attack. Therefore, it is not the goal of holistic survivability the prevention, identification or prediction of malicious attacks.

Holistic survivability follows the concepts devised by [Ellison et al., 1997], which defines that services differ in importance for a system. Some services are more important than others and they should be classified and handled accordingly. According to them, there two types of services: *essential* and *non-essential*. Essential services are the ones in which the measurement and analysis of survivability should care most. After all, they are the essence of survivability. The clear boundaries between survivability and dependability becomes blurry without essential services definition.

The identification of essential services is critical for a proper analysis of survivability. Currently, essential services are defined through a manual process. System experts choose which services are essential based on past experience. Usually such process is performed in preparation to survivability analysis. Chapter 5 proposes an algorithmic approach to find the essential services in a SCADA system.

4.2 Holistic survivability - A definition

To provide a formal definition of Holistic survivability we relate the motivations introduced earlier in this chapter in terms of a set of axioms that underpins the ideas of holistic survivability. The axioms are defined as follows.

1. *system as a whole* - survivability is an emergent attribute. Even though we can look at survivability of individual services to analyse survivability. To proper identify the survivability of the system, one should aggregate the individual scores taking into account the discriminatory factor of service importance in order to provide the system survivability.

2. *services discrimination* - services differ on their functionalities, such differentiation determines their criticality to the system. In the case of survivability either services are essential or not essential.

3. *dependencies* - in general services are not independent entities. Most of the time a service depends on other services to provide its functionality, therefore creating some form of dependency between them.

4. *uncertainty* - this axiom comes from the intuition the there is no absolute certainty in the evaluation process. From the data acquisition to the final score computation uncertainty is presented. The data acquired may be noisy, data can be missed during the collecting process, sensors may be faulty, and so forth. In addition to the uncertainty on data acquisition, most analytical models are built on uncertainty, they provide probabilistic results of their computations.

5. *malicious attacks are inevitable* - as pointed out in Chapter 1, and earlier in this chapter, malicious attacks are only increasing, with no sign of current solutions providing effective results. Rather than trying to stop every attack we assume that attacks are inevitable and we focus on ways of minimising their impact.

6. *survivability is not binary* - survivability is not a binary state (survivable or not survivable). Survivability can be measured in terms of degrees. Such degrees can be computed in terms of probabilistic quantifications with respect to the system. For instance, a system can have a probability of 0.8 of being survivable, or 0.75 and any other positive value between $[0.0, 1.0]$.

By considering the above axioms we provide the following definition of survivability.

Definition 5 *Holistic survivability is the capacity of essential services to provide their functionalities in cases of malicious attacks compromising parts of the system and that such services may rely on other services of the system, which are not necessarily essentials.*

To quantify the capacity of essential services providing their functionalities under undesired conditions, holistic survivability is represented by a five-tuple $< G, S, M, S_m, P >$ where:

- G - Represents the network graph that defines the computer system in question. Depending on how the system will be analysed the graph can be either directed or undirected. The network graph is defined in terms of services, with services being the nodes and the communication between such services represented as edges. Therefore, we call it **service graph**.
- S - Represents the services in the system. It is defined as a tuple $S =< E, I >$, where E are the essential services and $I = S - E$, are the non-essential ones. Usually those services are algorithmically computed. However, the definition is flexible enough to support manual selection.
- S_m - It describes the service state model utilised by holistic survivability. The model defines a set of possible in which services will be in during their operational time.
- M - It determines the set of metrics utilised to compute the service states. The metrics are used to compute in which state a service is in during the evaluation.
- P - It either describes the Conditional Probability Distributions (CPD) that represent service states during their operational time or the CPDs that represent potentials (see Section 2.1.2) during their runtime.

Over the next sections we detail the above components and we also show how the proposed model uses Probabilistic Graphical Models as a tool to quantify holistic survivability.

4.2.1 Service graph - G

The network structure of a computing system can be represented as a graph where computer hosts can be defined as graph nodes and the connection between these hosts can be represented by graph edges. However, in this model the graph nodes are represented by services and the communication between them as edges. Thus, a computer host that is hosting multiple services will have its services individually represented in the network graph.

The network graph is different from the ones that represent network topology of computer systems in general, it only includes services. Common network devices (such as routers, switches), which are commonly represented in a network topology are not part of the graph. In this thesis we call this network graph as **service graph**.

A service graph can be undirected or directed. It depends on how the interaction between the services is represented. However in both cases the graph nodes are represented by system services and and edge exists between two services if and only if there is at least one transaction between them (see Definition 7). A transaction is defined as follows.

Definition 6 *Given two services X and Y, there exists a transaction $T(X, Y)$ if a successful communication to exchange information is carried out either between X and Y or vice-versa.*

Definition 7 *A service graph $G = (S, L)$ is defined by a set of services represented as nodes $S = \{s_1, s_2, ...s_n\}$ and a set of links as edges $L = \{l_1, l_2, ..l_n\}$, where given two services x and y, $l_i(x, y) \in L \iff \exists\, T(x, y)$.*

The service graph is created by analysing the network traffic of the interactions between services. For the cases where no network traffic is provided the graph is generated by looking at the projected transactions between services. A projected transaction is a possible interaction between two services that was planned during the design phase of the system.

Figure 4.2 illustrates the difference between service graph and network topology. Figure 4.2 (a) shows a regular system network topology, with routers, switches, hosts and services. Figure 4.2 (b) shows the service network graph. Only services and links between them are included. For example, one can notice that service A does not communicate with service B because there is no link (edge) between them.

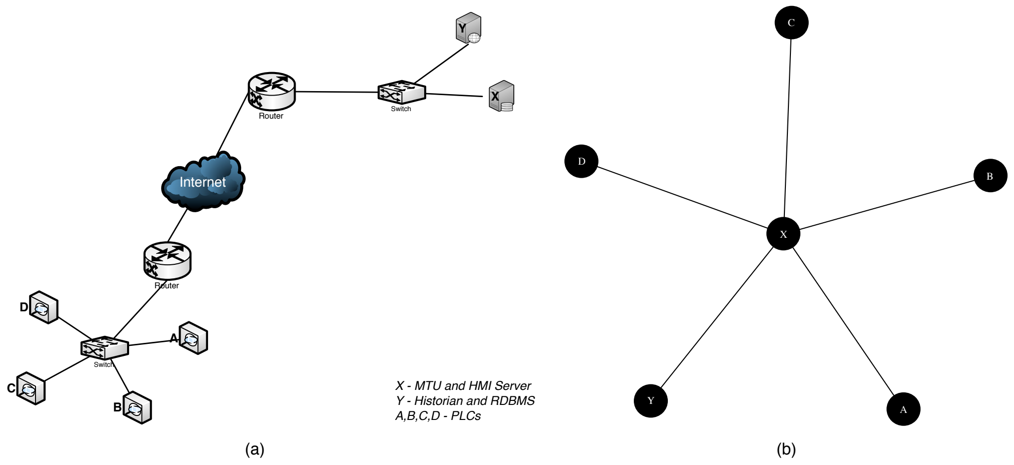


Figure 4.2: Comparison of real network structure (a) and service network structure (b). Note how the two structures may be different due to different services in different hosts communicating with each other at any time

Another type of service graph is called *dependency graph*. As it is the case with service graphs, dependency graphs are generated by looking at transactions between services. The main difference is that dependency graphs look at who initiated the transaction in order to create their directed edges. As a rule defined by this model, in a transaction between a service client and a service provider, the service client is dependent of the service provider. The intuition here is that a service that is dependent on another service needs this other services in order to provide its functionality. More formally,

Definition 8 *Dependency graph is a service graph whose edges are directed. The direction of the edges is based on a explicit dependency between two services. Given two services X and Y, X is dependent on Y ($X \leftarrow Y$) if there is a transaction $T(X,Y)$ initiated by X.*

Figure 4.3 (a) show a service graph, there is no direct links between services. Figure 4.3 (b) shows the same services now mapped by their dependencies. We can see that every other services is dependent of service X.

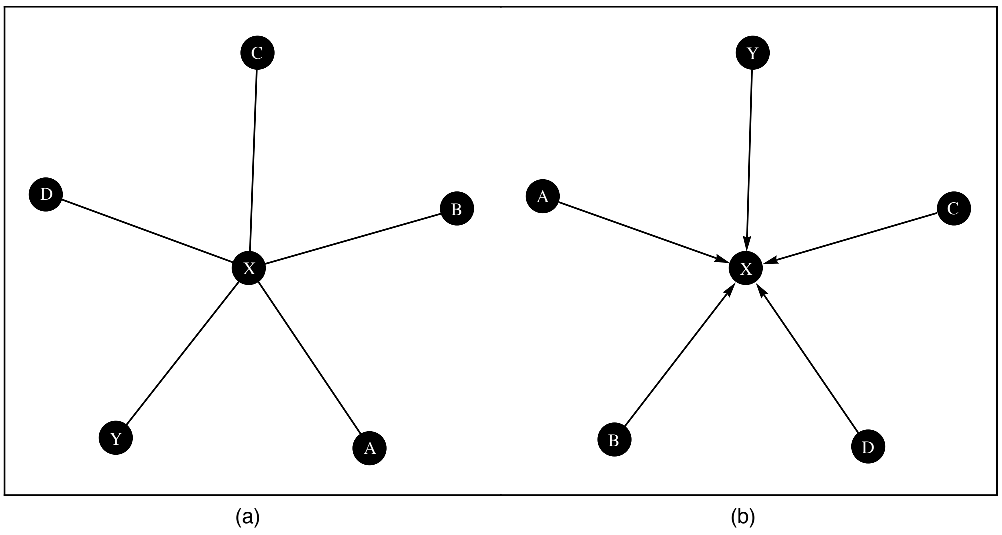


Figure 4.3: Dependency Graph

4.2.2 Services - S

Services are the basic entity for the holistic survivability evaluation of a SCADA system. To avoid misinterpretations about services and systems we provide two definitions that are important to a clear understanding of the relationship between them. We also show this relationship illustratively in Figure 4.4.

Definition 9 *A system $Sy = \{S_1, S_2, ..., S_m\}$ consists of a set of m services communicating with each other and providing functionalities to internal and external clients, where the internal clients are other services within the system.*

Definition 10 *A service is a software that provides a functionality consumed by other entities (services or clients) of the system. Services can have a variety of states that describe different levels of operability (e.g. Normal, Compromised, Acceptable, Unacceptable, and so forth).*

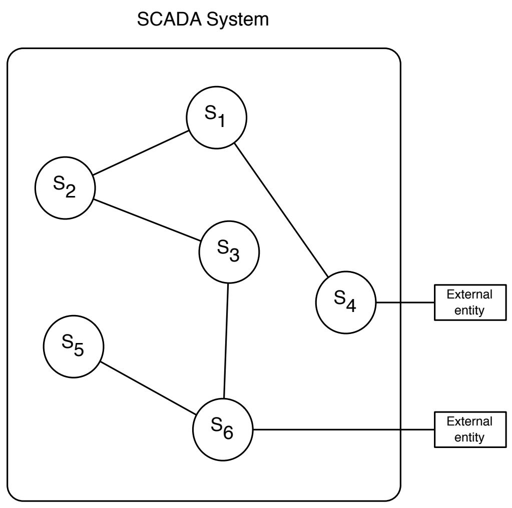


Figure 4.4: Service model

The classification of services in essential and non-essential is vital for holistic survivability. Even though the model is flexible enough to accept manual services classification, most of the time we rely on the algorithm proposed in Chapter 5 for defining essential and non-essential services. Therefore, we leave this discussion for Chapter 5.

4.2.3 Service state modelling - S_m

The definition of the possible states that a service can hold is necessary for the analysis of dependencies between services and consequently to survivability quantification. However such states are based on what it is being quantified. For instance, if one is interested about the performance of the system, the state-space required might be different from the one that is interested about security. A generic state-space definition can be possible as long as the same number and types of service states is shared between the types of quantification. For instance, if a state-space definition for performance defines three states to represent a service (e.g. Normal, Degraded and Unavailable) and if the state-space for security modelling also defines the same three states, both could use the same state-definition.

It is important to note that the way the state is computed is dependent of the metric being used to the state computation. Just because two different service modelling approaches use the same state-space definition it does not mean their states are computed in the same way and using the same metrics. In this work, we provide two types of quantification, namely performance, and security.

The service state model is represented by a set $S_m = \{s_1, s_2, ..., s_n\}$ where each item s_i defines a possible state of the service and n is the number of states. In Section 4.2.5 we show how the metrics are used in combination with state models to compute the state of a service in the system.

Depending on how holistic survivability will be used we rely on two types of techniques to compute service states:

1. *Analytical* - for analytical analysis we rely on analytical models that are used to compute the service distributions and based and consequently the service states.

2. *Empirical* - for empirical analysis we rely on machine learning techniques such as anomaly detection [Chandola et al., 2009]. Such techniques are based on the detection of *outliers*. An outlier is an observation that deviates from the other observations for a large margin, therefore raising suspicious about it [Hawkins, 1980]. In general there are three type of techniques to detect outliers: *supervised, semi-supervised* and *unsupervised*. The first two techniques depend on trained datasets. Sometimes it is not possible to have trained datasets, for such cases unsupervised techniques can be used. To compute the metrics in the proposed model the unsupervised type of detection is used. It is assumed there is no training data available. As it is the case for the the other types of detection, unsupervised anomaly detection also provides different techniques to detect outliers. To this proposed model statistical test is used. The statistical-based technique assumes that the normal data is generated by a known distribution, occur in a region with high probability and that the outliers deviates largely from this distribution. The technique works as follows.

   - Assume that the data follows a known distribution. Usually it is assumed that the data fits in a Gaussian distribution [Chandola et al., 2009]. When it is not the case, transformations to make the data more 'Guassian' can be performed.
   - Compute the mean (μ), the standard deviation (σ) and the variance (σ^2).
   - Find the outliers by looking for data points that deviate more than 3 times from the standard deviation (σ). Also known as 3σ rule. This rule represents the small left and right tails region in Figure 4.5.

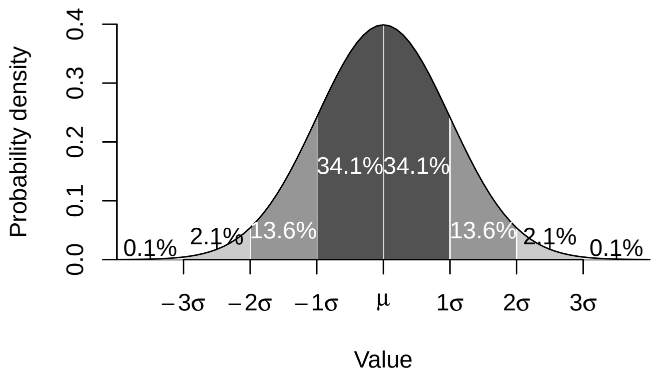


Figure 4.5: Gaussian distribution

4.2.4 Metrics - M

Metric is a unit of measurement and can be either qualitative or quantitative. As we are interested in survivability quantification, our focus is on quantitative metrics. The terms metrics and quantitative metrics will be used interchangeably throughout this thesis. Metrics are widely used by the scientific community where it is applied to provide understanding and meaning to theories and discoveries, and by industry that applies metrics to their business processes in order to get insights from the acquired data and help tune their models more accurately.

One of the challenges in defining metrics is to know what and how to measure. The more accurate and meaningful metrics give us better understanding into the processes of the system. In this thesis the metrics utilised to quantify the service states and consequently holistic survivability are based on the key attributes of survivability as originally defined by [Ellison et al., 1997]. However, the metrics will not measure recognition, resistance, recovery and adaptation directly. They will look at aspects of the system that are linked to these attributes. *Performance* will measure, as the name suggests, the performance of the services (see Chapter 6). By looking at the performance of services we can detect degradations that may be related with malicious attacks. *Security* will measure the probability of having the service confidentiality, availability and integrity compromised by looking at possible vulnerabilities affecting services and how easy they are to be exploited. Services without vulnerabilities are harder to attack as they are mode difficult to be break in, consequently they are more resistant to attacks.

Those metrics do not measure the attributes, for instance, malicious attacks that do not affect the performance of a service will likely not be detected by monitoring performance metrics. The same is valid for the security metric that monitors vulnerabilities, the security of a service may be degraded because the messages exchange between services are not encrypted, the proposed security metric (more detailed in Chapter 7) will not detect such compromise. Other properties were taken into account when devising metrics to quantify survivability.

4.2.5 Probability distribution - P

A probability distribution is associated with each service in the system. The distribution represents probabilities for each state of the service during its operational time. The number of states are defined by the state model as described in Section 4.2.3. For instance, lets say the state model defined three states $\{s_1, s_2, s_3\}$ for representing the possible states of a service, let M defines the metric to be measure and $t = 0.15$ a threshold used to define the rules for the service matching. The threshold t can be defined by a system expert or learnt from the data.

$$\begin{cases} s_1 & \text{if } M < t \\ s_2 & \text{if } M \geq t \text{ and } M < 2 \times t \\ s_3 & \text{if } M > 2 \times t \end{cases} \tag{4.2.1}$$

Then, applying the above to the measurements illustrated in Figure 4.6 (top) we have the states assigned as illustrated in Figure 4.6 (bottom).

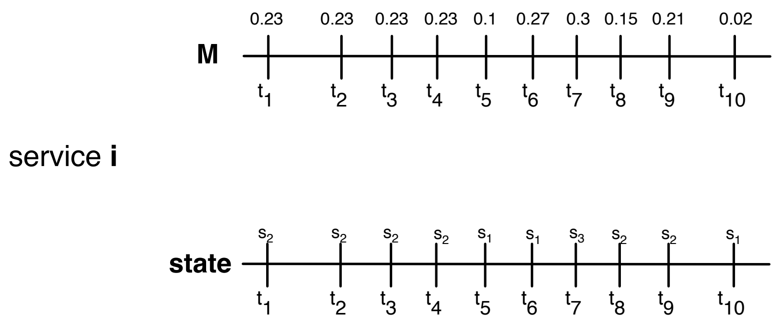


Figure 4.6: Metric M measurement (top). Service state based on its metric measurement (bottom)

We have the following rules been applied during a empirical experiment or though applying analytical models. Chapter 6 describes how to use analytical models and empirical data to compute the probabilities P.

It is also important to note that when services are dependent on other services P becomes a Conditional Probability Distribution CPD. The probability P can be assigned in two ways. If the system is evaluated in design time P is assigned manually, usually using known distributions as Exponential distribution, Uniform distribution and so forth. For the cases where an empirical evaluation is being performed, the probabilities P are learnt from the data acquired.

4.3 Quantifying holistic survivability

In order to provide a quantification tool that allows us to quantify holistic survivability we need to map its concepts to a mathematical formalism. As mentioned earlier, one of the features of holistic survivability is to consider all services in the system and their (inter)dependencies. We already have shown how a computing (SCADA) system is mapped as a service graph (see Section 4.2.1).

Now we extend this concept by mapping holistic survivability to probability theory. Services are defined as random variables X, with probability distributions to represent them $P(X)$. A joint probability distribution is used to represent all services in the system. Therefore, given a set of services $\{x_1, x_2, ..., x_n\}$, the joint distribution is represented by $P(x_1, x_2, ..., x_n)$, where n is the number of services in the system. This full joint probability distribution can answer all possible inference queries to the system. Essentially we will use this model to ask questions such as: (i) What is the survivability of the system if service X is on a state that is not defined as normal? (ii) What are the services that service X depends most?

Those questions are important to evaluate the survivability of the system. However they are not efficient to execute. Given n services with m discrete states, it would take m^n parameters to answer all queries to this model. Definitely, a not desired performance. Fortunately, there are more efficient ways of performing these operations. Probabilistic Graphical Models (PGM) formalism comes to help.

Probabilistic Graphical Models rely on Probability and Graph theory to its formalism. They offer more efficient mechanisms, which rely on conditional independence (described in Definition 1) to represent a joint distribution compactly. We use them as the base formalism for holistic survivability. We also rely on their formalism to perform inferences in our model. As mentioned in Chapter 1, among the capabilities of holistic survivability it is power of doing inference for prediction and diagnosis.

One of the key aspects of holistic survivability is the concept of essential services. Holistic survivability model allows the definition and use of more than one essential service. Chapter 5 describes an algorithmic approach to identify essential services. Once the services are identified they are used to compute the overall survivability of the system. More specifically, holistic survivability is quantified by computing the joint probability distribution through the essential services.

Definition 11 *Holistic survivability is quantified as the marginalisation of essential services over all services of the SCADA system, where the essential services are in an intended state.* $P(E = s) = \sum_I P(E, I)$*, where* E *represents the set of all essential services,* I *represents all non-essential services of the system and* s *is the operational service state that was designed for.*

Figure 4.7 illustrates a generic picture of the model and their main steps, which are based on services states, service interdependencies, and finally the overall system survivability inference. As mentioned earlier this thesis presents two different techniques to model and evaluate system survivability. One technique that relies on simulation models to define the service probability distributions and an empirical one where distributions are learnt from data.

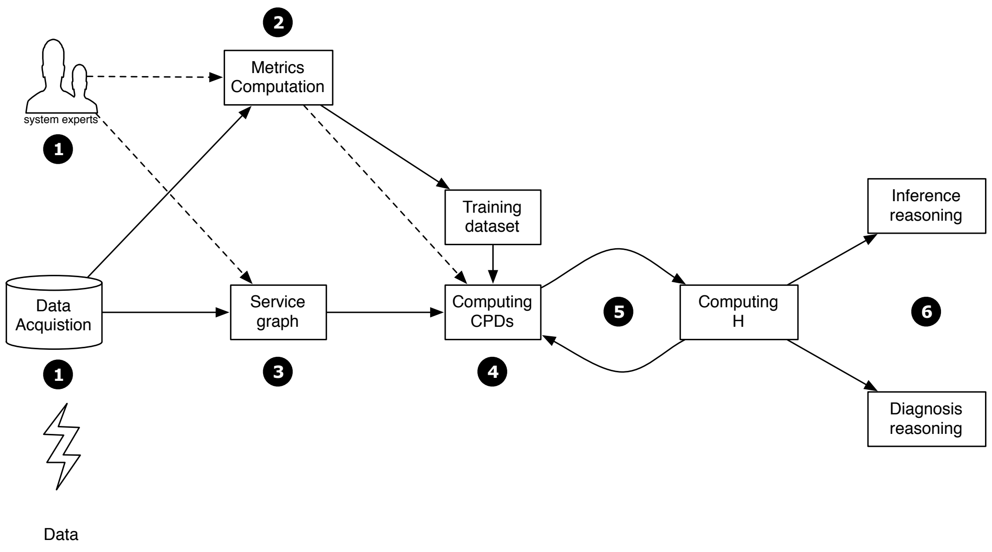


Figure 4.7: Holistic survivability model overall structure

The dashed arrows in Figure 4.7 represent the manual process of defining the graphical model, computing the metrics and then computing the inferences that define holistic survivability. This process is simple as it only contains three steps: System experts define the model to be simulated or computed numerically, the metrics used to define the state of the service are computed, and the service graph that represent the system is defined. Once the CPDs and the service graph are defined the system expert is ready to use the model for computing holistic survivability. Even though the process is simple it is labour intensive to make the system expert compute the metrics, the CPDs and define the service graph of the system. For systems with a high number of services this process become infeasible.

4.4 Learning from data

We recognise that is impractical to use a manual process to compute holistic survivability. In this section we describe how the learning from data helps on creating the model to compute holistic survivability. We present an algorithm that can be used to acquire and parse the data, generate the service graph, compute the metrics being analysed, generate the CPDs for all services (or potentials) in the system, and finally compute holistic survivability.

First, we describe a step-by-step following the steps illustrated in Figure 4.7, and then we present the algorithm that computes the entire process. Extra inferences and diagnosis are supported by the model, but they are not defined in the algorithm, as it would impossible to known what type of diagnosis and on what services a practitioner was interested.

4.4.1 Data acquisition (step 1)

The first step in the data acquisition process is to define what data should be recorded. In other words, it needs to define what information is important enough to describe the state of the service. As mentioned earlier, the state model (see Section 4.2.3) is defined in terms of metrics (see Section 4.2.4). Therefore, the data field can be a vector of multiple values, each measuring a metric of interest. This is the data that needs to be acquired or computed. Every time a transaction occurs its content should be parsed to acquire the following attributes: *source*, which represents the service that initiates the transaction, *target*, the service provider, and *data*, which as already mentioned represents the data of interest. Alternatively, the metrics in question could be acquired separately from the transactions. This is the case for the security model presented in Chapter 7. The data will eventually be merged as a single table.

The information stored in this table will also be used to build the service graph (see Section 4.2.1) of the system and consequently the network structure of the graphical model that will be used. Also it is important to notice that the data field is not mandatory. Depending of the analysis being executed the metrics used by the analysis may not come from this network traffic data. However, the source and target fields are always necessary, as they are used to build the service graph of the system.

Table 6.4 shows an example of the acquired information at the service side, where the two rows describe a transaction between two services, in this case services A and B.

Table 4.1: Transaction between two services

Source	Target	data
A	B	000200000006030100000001
B	A	00020000000403010100
...	...	...

In practical terms, there are multiple ways of acquiring this information without the need to rewrite service's code. One possibility is to add a machine that sniffs the traffic at the switch level. The machine acquires the information from one or more switch(es) to which all SCADA devices running the services are connected to. Techniques like ARP spoofing could also accomplish this easily. However, this may be considered hostile and network administrators may not allow such techniques. Another way of accomplishing data acquisition is to add 'proxy' devices between the SCADA services and the switch, so that the proxy can acquire and process the information being exchanged.

4.4.2 Metrics computation (step 2)

Once the raw data is acquired it needs to be transformed into a dataset so that the proposed model can use it to build the network structure and to learn the CPDs. The model is applied to historical data. Therefore, it is assumed that the acquired data is available on a central database. Information acquired by individual services are sent to a central repository that holds all the information in this central database. It contains information similar to Table 6.4. For example, the table contains three fields (Source, Target and Data), and numerous rows depicting different transactions between services.

The data transformation occurs in two steps. First, the data is transformed into a database where columns represent the services and rows represent the measurements at a specific time. Second, based on the metrics used a score representing the metrics is computed. Then, based on the state model defined a service state is assigned to the service. Figure 4.8 illustrates how the transformed data looks like after each step in the transformation.

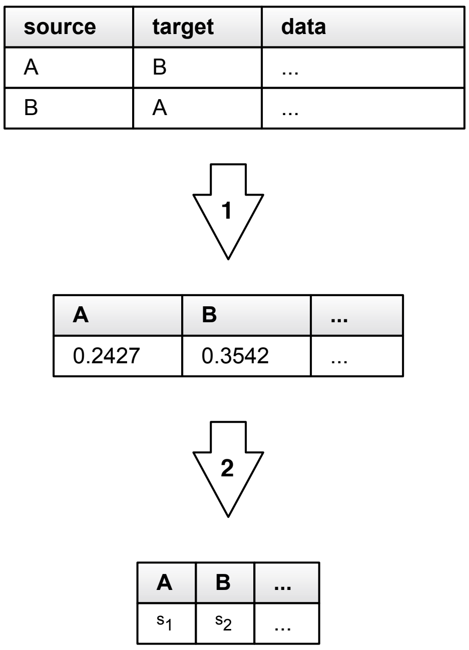


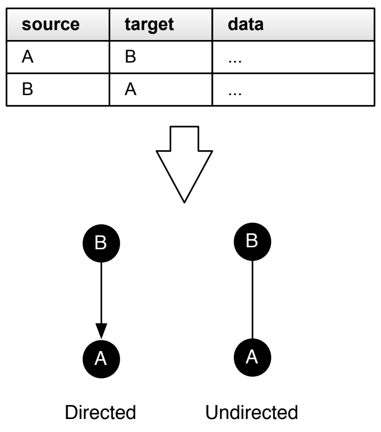
Figure 4.8: Computing metrics - 2 steps

In order to define the state behaviour for each service, score bound limits need to be defined for a service (as mentioned in Section 4.2.3). As discussed the proposed model uses statistical tests to find the limits for the metrics that define service states.

4.4.3 Generating the service graph (step 3)

The service graph is generated by processing the transactions recorded at the table defined in Table 6.4, the process aims to generate a graph structure similar to the one illustrated in Figure 4.3. On the top is the system network topology and on the bottom is the generated network structure.

The process to generate this structure starts by going through Table 6.4 and identifying transactions. Those transactions are then converted into nodes and edges in the graph. Figure 4.9 illustrates what happens when Algorithm 1 identifies a transaction in Table 6.4. Depending on the type of the graphical model used, the algorithm will run slightly different. For instance, for creating undirected graphs, the algorithm does not check for loops, as it is the case when creating directed acyclic graphs. The algorithm runs through all rows of Table 6.4, and at the end the service graph is generated.



source	target	data
A	B	...
B	A	...

Figure 4.9: Example of generating dependency graph or service graph from transaction table

Algorithm 1: Generate Service (Dependency) Graph

```
// Represents the Table of data logs
Input: DataLogs
// Type of graph to be generated: 1 - Directed, 2 -
Undirected
Input: T
Output: G
// Represents the generated graph
1  $G \leftarrow \emptyset$
2  foreach row in DataLogs do
3      if row == inverse(next(DataLogs)) then
4          $x, y \leftarrow extractServices(row)$
5          if T == 1 and NotLoop(G,edge(y,x)) then
6              $G \cup edge(y, x)$
7          else if T == 2 then
8              $G \cup edge(y, x)$
9          end
10     else
11         reverse(DataLogs)
12     end
13 end
14 return G
```

4.4.4 Computing the CPDs (Step 4)

To learn the CPDs it is assumed that the probability of each service state is independent of the probability of every other state in the dataset (i.i.d). In addition, a service (or dependency) graph is already known (Section 4.4.3). Depending if the dataset contains noisy data or not a different algorithm may be applied. In our model we will use two algorithms, one for when the dataset contains no missing data, and one for datasets with missing data.

We use Maximum-Likelihood Estimation (MLE) $\Theta_{ijk} = \frac{N_{ijk}}{\sum_k N_{ijk}}$ for learning CPDs from datasets with no missing data [Korb and Nicholson, 2004; Spiegelhalter and Lauritzen, 1990]. The process can be formally described as, given a service graph G that represents the system's network structure, where m represents the number of service (nodes) in the network structure, and $D = (\{< v_1^1, v_2^1, ..., v_m^1 >, ..., < v_1^n, v_2^n, ..., v_m^n > \})$, with n samples (rows), represents a dataset. The problem is to find the model M (CPDs) that maximises $Pr(D|M)$. More formally, the challenge is to find $\Theta_{ijk} = P(X_i = x_k | Pa(X_i) = pa_j)$, where X_i is a service, $Pa(X_i)$ are the parents of service X_i, x_k is a service state and pa_j is parent's service state.

On Markov Networks (Undirected Graphs), the graph has to be decomposable and no constraints are placed on the form of clique potentials [Barber, 2012]. For the cases where datasets contain missing data we resort to the Expectation Maximization (EM) algorithm [Barber, 2012]. This process is different from the previous one as now we must consider not only the observed data (as before), but also the method by which data is missing.

4.4.5 Calculating the survivability (step 5)

Once the graphical model is built, the final overall survivability score is computed. The computation is based on the factorisation derived from the service (dependency) graph itself, and the computation is based on it, however it uses Definition 11 to proper compute the holistic survivability. It is important to notice that even though we did not include the process of identifying essential services, which is the base of our holistic survivability definition, the process has to occur and it is described in Chapter 5.

4.5 Conclusion

This chapter introduced a new concept of survivability for SCADA systems. Holistic survivability evaluates the survivability of SCADA systems considering services interdependencies, essential services and it is flexible enough to support different metrics and models. The five components that defines holistic survivability were also introduced. Holistic survivability uses the formalism of probabilistic graphical models to compute survivability and it is suitable for design time analysis as well as analysis in runtime using the data generated by the system. A step-by-step guide was presented that shows how holistic survivability can be computed through an automatic process based on data. For the cases of dependency graphs this solution present a shortcoming, which we hope to fix on future work. Currently, only systems that do not create loops on their service transactions are supported. As dependency graphs are DAG (Direct Acyclic Graphs). A possible solution is to use another type of graphical model, one that supports temporal aspects and consequently remove cycles by having nodes defined with temporal tags such as Dynamic Bayesian Networks (DBN) [Koller and Friedman, 2009].

Chapter 5

Finding essential services

This chapter introduces an algorithm to identify the essential services of a SCADA system. The algorithm is based on **Service Rank**, a ranking measurement aimed at sorting the importance of services in a SCADA system. It uses network traffic and centrality techniques [Freeman, 1979; Newman, 2007] to rank the services. The chapter also provides an evaluation that demonstrates how to use the proposed algorithm to identify essential services in a SCADA system. The evaluation is created in SCADASim (see Chapter 3), a tool to create SCADA simulations. Further, we compare the proposed algorithm with other techniques used to find important nodes in a network such as network centrality metrics [Landherr et al., 2010].

5.1 Introduction

A key aspect of survivability is the importance of services. One of the first accounts of the concept of essential services was presented by [Ellison et al., 1997] in the definition of survivability for computer systems. However, most of survivability analysis over the years did not give importance to essential services. This is may be due to the fact that in general the analysis of survivability was focused to a specific service and not the system, which could have multiple services. By analysing one service their implicitly assumed that the service under analysis was an essential service.

We recognise that essential services are vital to the functioning of a system, and they

Part of this work to appear in IEEE Transactions on Industrial Informatics should have priority over the other services. However we do not assume that essential services are already identified by system experts in order to perform a survivability analysis. In addition we believe to the possibility that a system may contain multiple essential services. If this is the case such essential services should be analysed together in an holistic approach. In general survivability analysis is applied to a specific service.

The manual approach of identifying essential services is problematic. Even though a system expert could identify multiple essential services, such subjective approach could lead to different experts evaluating services differently. Also, the manual process for complex and large systems, where we could have hundreds of services interconnected could make the identification process erroneous at best.

There is a need for an algorithmic identification of essential services. The challenge is to provide a generic approach that does not take into account what system is under evaluation and what type of data is being exchanged. Looking at other research fields we found that the idea of identifying important actors on networks it is not novel. In fact, the domain of Social Network Analysis (SNA) has been studying it for many years. However, it is a challenging problem and SNA researchers have resorted to different approaches.

One of the methods to find important nodes in graphs is to remove nodes that will make the graph become disconnected. Unfortunately, such approach does not work on survivability and SCADA networks. Depending on who are the essential services, it is possible that the essential services and the other services needed by the essential ones are on the same sub-graph and therefore they still can provide their functionality. [Shetty and Adibi, 2005] recognised this problem and proposed a solution by applying graph entropy [Körner, 1986], however they did not remove graph cutting technique completely. Their approach was to look at the transactions between the nodes in the graph (e.g. email exchanges, phone calls) to compute the graph entropy, and then by applying cuts to the graph and computing the difference between before and after cuts they found the important nodes. The nodes that provide higher differences are the important ones.

[Borgatti, 2005] resorted to another approach. He described a typology of network flows based on two attributes: *types of trajectories* that traffic could follow in a graph (e.g. geodesics, walks, paths, or trails) and the *method of spread* (e.g. broadcast, serial replication, or transfer). His main argument is that in general people make implicit assumptions about the manner the traffic will flow in the network. This means that the degree centrality is not applicable for every type of network, as so as betweenness, eigenvector or any other type of centrality measurement. Such metrics should be applicable to the type of network that follows the traffic pattern they were designed for. [Ghosh and Lerman, 2010] agreed with [Borgatti, 2005] and they have shown empirically that the influence model has to match the dynamic process of the network. In other words, many topological properties of networks are defined by assuming that most of the information flows along shortest paths. Clearly, this is not the case for networked systems as they are not designed with this ‘feature’ in mind. In addition, data flows according to specific rules, which usually are not related with the shortest paths of the network.

With the Finding Essential Services (FES) algorithm we propose a different approach to identify importance and consequently essential services in a SCADA system. Our approach is based on the data exchanged between services. The network traffic is used to find transactions (see Definition 6) between services (nodes) and also to find out the diversity of such transactions. The FES algorithm has three main steps:

1. Computing *Service Rank*
2. Clustering the services based on their service ranks
3. Choosing the cluster that represents the essential services as the one that the highest ranked service is part of

Over the next sections we introduce *Service Rank*, a quantitative measurement devised specifically to rank the importance of services, and we present the FES algorithm, as well as we present an evaluation of the algorithm. We created a simulation of a SCADA system to evaluate our FES algorithm.

5.2 Service rank

Service Rank is a quantitative measurement that identifies the importance of a service within a system. The higher is the service rank the more essential is the service to the system. Service rank uses some ideas of centrality [Landherr et al., 2010], a metric that measures the importance/influence/power of a node in a network. It is widely used in the Social Network Analysis domain [Wasserman and Faust, 1994]. Centrality has been proved of great value to analyse and understand the roles of actors in social networks and measures different aspects of the network. [Freeman, 1979; Newman, 2007] describes the main metrics used to measure centrality: *Degree, Closeness, Betweenness, and Eigenvector*, among others.

An important aspect on the algorithmic process of identifying essential services is the choice of metrics. The metrics used to compute **Service Rank** should be generic enough to be used by any type of SCADA system. SCADA are information systems, their services exchange data from and to other services and external entities. It is reasonable to model the metric in terms of the data being exchanged. Following this approach we defined two primitives (attributes) as the core of the **Service Rank** algorithm: *information flow (I_F)*, and *information diversity (I_D)*.

To compute the service rank the information diversity and information flow are multiplied together as described in Equation 5.2.1.

$$S_R = I_F \times I_D \tag{5.2.1}$$

5.2.1 Information flow

Information flow measures the number of transactions a service performs during a period of time. A definition of transaction is described in Definition 6. The period of time is defined by who is performing the evaluation. However, we see two main approaches for this: Batch assessment, all the transactions are stored in a database and after the evaluation period finishes the batch data is processed. Still part of batch assessment is to use the data collected during the current assessment in addition to historical data collected by previous assessments. We prefer this option because the more data we have the better is for the algorithm computing the rankings. The other approach is real-time assessment; data is collected and computed as the system runs. Here, the best approach is to combine the historical data with the one being acquired. At first, because not much data is collected the initial rankings can be misleading. Therefore, it is recommended to use the algorithm when the system is on steady-state mode in terms of data exchange.

Equation 5.2.2 shows how the I_F score is computed. For each service the weights of its links (edges) is summed up and then it is normalised by the total sum of all services of the system.

$$I_F(s) = \frac{\sum_i t_{si}}{\sum_i \sum_j t_{ij}}, \text{ if } i \neq j \neq s \tag{5.2.2}$$

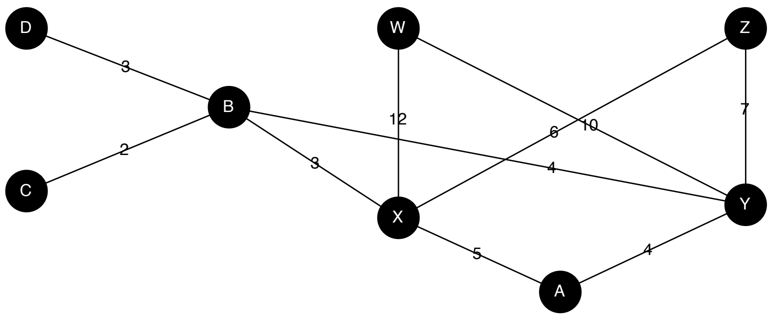
We use a weight undirected graph to represent the system. Each pair of services (i, j) that communicates with each other is represented by a graph edge between them, the edge weight t represents the number of transactions between the two services. Internally, each service has a data structure to store this information. The data structure contains two fields: *destination* and *number of transactions*. Table 5.1 shows the data structure of service X based on the example illustrated in Figure 5.1.

Algorithm 2 describes the process of computing the information flow of each service, and then normalise them based on the total information flow of the system. The *LocalInformationFlow* defined in Algorithm 2 sums up all the transactions processed by a service. For instance, taking as example the transactions for service X described in Table 5.1, the operation will return the value 28.

Algorithm 2: Information Flow

```
Input: ServicesList
Output: InformationFlowList
1  InformationFlowList ← $\emptyset$
2  totalInformationFlow ← 0
3  foreach service in ServicesList do
4      localIF ← $LocalInformationFlow(service)$
5      totalInformationFlow ← totalInformationFlow + localIF
6      put $< service, localIF >$ in InformationFlowList
7  end
8  foreach tuple in InformationFlowList do
9      inflow ← $tuple < localIF > /totalInformationFlow$
10     put $inflow$ to $tuple < localIF >$
11 end
12 return InformationFlowList
```

To provide a better understanding of the Information flow attribute we show an example of a system with 8 services and the service graph illustrated in Figure 5.1.



Service	I_F
A	0.1836
B	0.2448
C	0.0408
D	0.0612
W	0.4489
X	0.5306
Y	0.5102
Z	0.2653

Figure 5.1: Service graph with transactions as weights - An example of applying I_F

In this example graph, service X communicates quite often with service W (12 transactions), services C and D do not communicate (they are not linked), there are few interactions from services C and D to service B (2 and 3 respectively).

Table 5.1: Data structure for service X

Destination	Number of transactions
W	12
B	3
A	5
Z	8

The table in Figure 5.1 shows the I_F score for all services in this example. Service X has the highest Information flow score in this system as it concentrates most of the transactions in the network. For the sack of completeness we compare I_F with known centrality metrics such as: Closeness, Degree and Betweenness.

Closeness measures how rapidly information spreads in the network from one node to another. A central node can reach other nodes quicker (using fewer intermediaries nodes) than other nodes. Figure 5.2 illustrates the closeness of the nodes in the network. The closer to black they are the higher closeness they have. As we can see there are three services B, X, Y with highest closeness, they are in black in Figure 5.2. Using I_F, service B is not even between the three highest ranked services. Clearly, closeness does not capture a key aspect we are interested, the dynamism of the network regarding transactions.

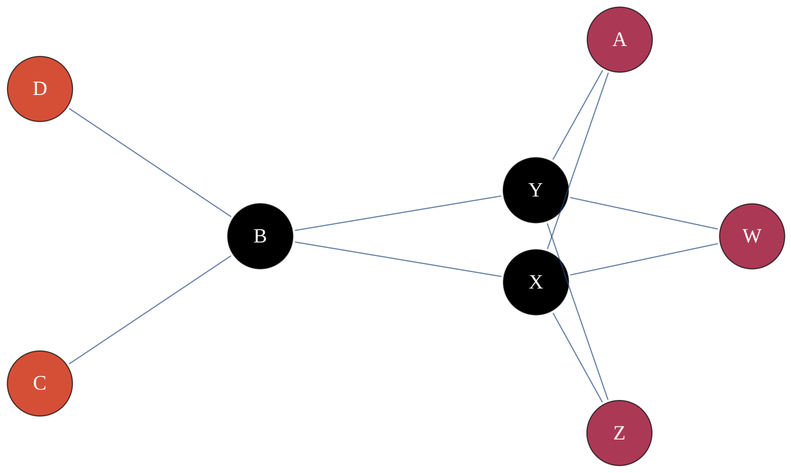


Figure 5.2: Closeness centrality

Degree centrality measures the nodes degrees in the graph. The intuition of using degree centrality is that higher degree nodes may be 'well' connected in the graph (system). With degree centrality we have the same behaviour as presented by closeness centrality. We have services defined as high rankings that are not even part of the top three ranked services in I_F, as illustrated in Figure 5.3.

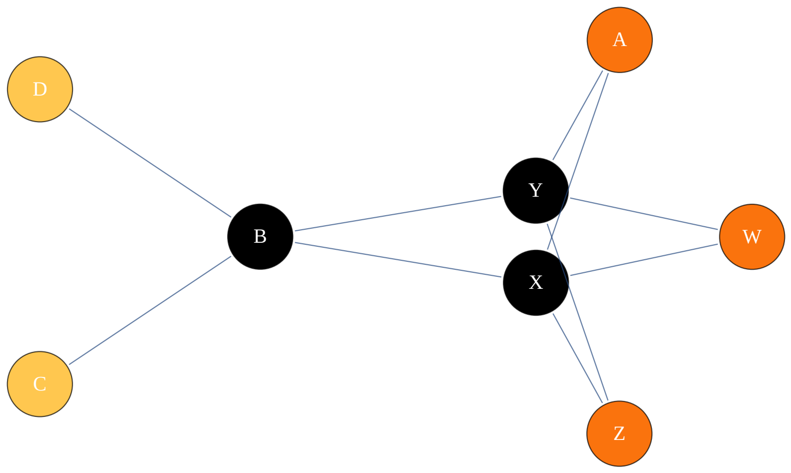


Figure 5.3: Degree centrality

And finally Betweenness, a centrality measure that bridges the communication between nodes in the graph. The intuition is that high ranked betweenness nodes may be important as they may ‘know’ what it is happening in terms of data in the network. Figure 5.4 illustrates the betweenness of all services in the graph. As we can see the results are even worse. Service B is the highest ranked service in the graph. Clearly, results are completely different from the ones provided by I_F.

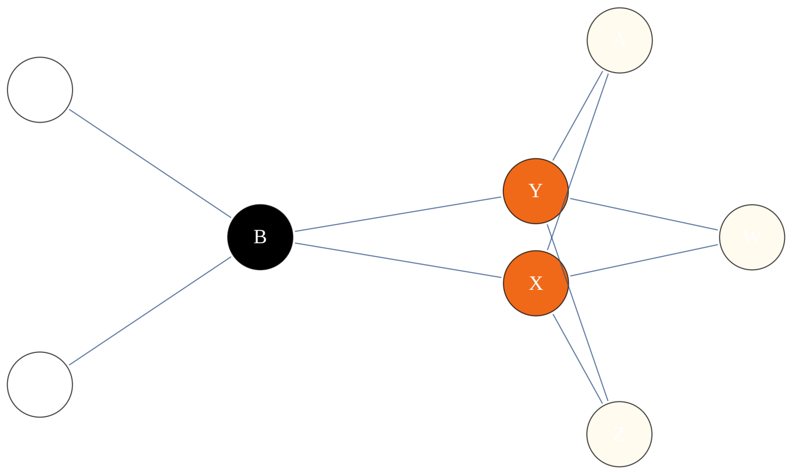


Figure 5.4: Betweenness centrality

5.2.2 Information diversity

Information diversity measures the diversity of information exchanged by a service. The intuition is that the more diverse is a set of messages the more information it contains. It works by tokenising data message in terms of content.

Essentially Information Diversity is the normalisation of Token Diversity score. Equation 5.2.3 shows how I_D is computed. The attribute is measured by computing the **Token Diversity** (T_D) of the messages exchanged by a service and then normalising it in order to compute the final I_D.

$$I_D(s) = \frac{T_D(s)}{\sum_i T_D(i)} \tag{5.2.3}$$

Figure 5.5 illustrates the overall process of computing *Token Diversity* and consequently, Information Diversity. The first step is to define a rule that will be used to tokenise the message. In other words, the rule will define what from a message is considered a token. Any rule is supported, as long as it can be computed. Once the rule is defined, the process will tokenise the messages and send the tokens to be processed by the main algorithm that will then compute the Token Diversity score.

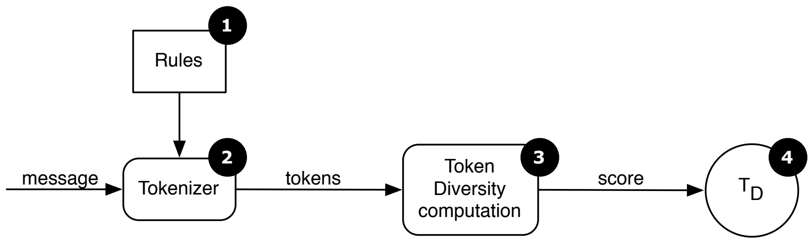


Figure 5.5: Overall process to compute individual (service) Token Diversity

The rule used in the algorithm to tokenise messages is defined as a function, therefore new function scan be defined to provide the desired functionality. Consider the message illustrated in Figure 5.6. The message is a generic message with two fields, header and payload. Usually it is how any protocol message looks like, the difference may be on having sub-fields in the header. A simple rule would be to use the entire message as a token, for the case of the generic message, it would mean to use *header* + *payload*. However, it may not be the best approach. Some protocols add counters to their message headers, if one adopted this simple approach of using *header* + *payload* as a token, it could end up having one message per token type. Another approach is to use only the *payload* of the message. It seems reasonable for us, as it is where the important content is located.

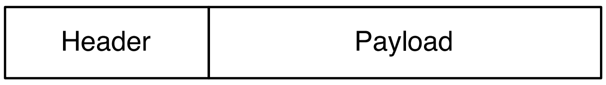


Figure 5.6: Generic message

The token diversity calculation is illustrated by Equation 5.2.4, where n is the number of token types handled by a service. The equation is a variation of Simpson's diversity index [Simpson, 1949]. In our case, rather than measuring diversity on nature, we are measuring information diversity on messages.

Algorithm 3 shows how token diversity T_D is computed. It counts the number of tokens per token type and then applies part of Equation 5.2.4 to get a temporary token diversity. And after it gets a summation of all temporary token diversity (one per token type) it applies Equation 5.2.4 to get the final token diversity score.

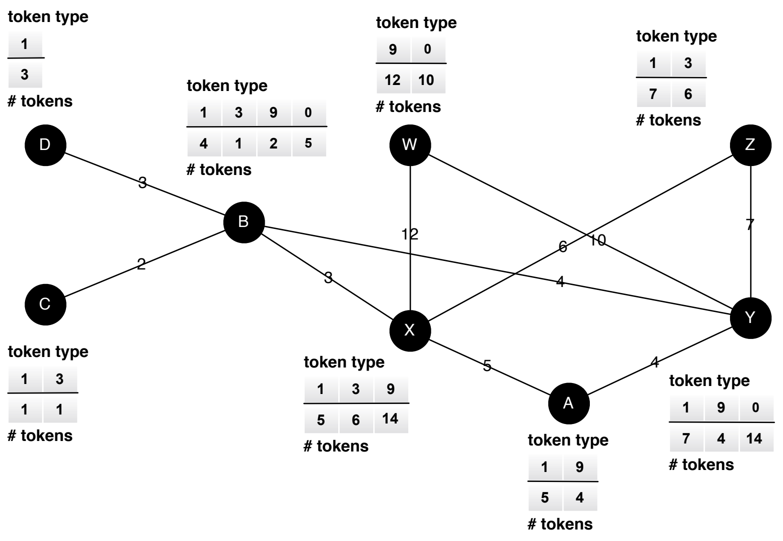
$$T_D = \frac{1}{\sum_{i=1}^{n} {p_i}^2} \times n \tag{5.2.4}$$

Algorithm 3: Token Diversity

```
  Input: TokenMap, totalTokensCounter
  Output: Token Diversity score $T_D$
1 $td, ttd \leftarrow 0$
2 $tokenTypes \leftarrow$ keys(TokenMap)
3 $numberTokenTypes \leftarrow$ length(tokenTypes)
4 foreach type in tokenTypes do
5     $counter \leftarrow$ value(TokenMap,type)
6     $ttd \leftarrow ttd + (\frac{counter}{totalTokensCounter})^2$
7 end
8 $td \leftarrow \frac{1}{ttd} \times numberTokenTypes$
9 return $td$
```

Once the token diversity is computed per service they need to be aggregated in a final step to compute a normalised version of the token score, which is the final I_D score. Equation 5.2.3 illustrates this approach. Each individual $T_D(i)$, where i is *i-th* service in the list, has its score divided by the sum of all T_D scores.

Figure 5.7 illustrates an example of an arbitrary networked system. Each service keeps track of the tokens they are exchanged with other services in a *HashMap* data structure as shown in Figure 5.7. The token type represents the key of the *HashMap* and the *# tokens* represents the value.



Service	I_D
A	0.0704
B	0.1487
C	0.0158
D	0.0118
W	0.1728
X	0.2407
Y	0.2370
Z	0.1023

Figure 5.7: Information diversity example

In the example illustrated in Figure 5.7 tokens are represented as numbers. A local service data structure that keeps track of the messages exchanged between service W and the other services would look like as Table 5.2, where each row represents a transaction.

Table 5.2: Information Diversity Data Structure for service W

Destination	token
X	9
X	9
Y	0
X	9
X	9
X	9
X	9
X	9
X	9
X	9
Y	0
X	9
X	9
X	9
Y	0
Y	0
Y	0
Y	0
Y	0
Y	0
Y	0
Y	0

5.3 Finding essential services algorithm

Algorithm 4 describes the entire FES computation process. As mentioned earlier, the FES algorithm has three main steps: Computing service rank, which is described in Algorithm 4, lines 2 to 9. After the service rank is computed the algorithm groups the services in two groups of services by applying the *k-means* clustering algorithm [Anderberg, 1973]. There is no special reason to use *k-means* apart of being an unsupervised machine learning technique and being widely used by practitioners, this process is represented in line 25 of Algorithm 4. Once the groups are created the algorithm labels the groups between essential and non-essential, which represent essential and non-essential services respectively. This process is represented by lines 27 to 33 of Algorithm 4.

For the case when a service rank value is sharing by all services of the system, some special check has to be made to avoid the clustering algorithm erroneously to group the services in two groups, even though it is not the case, as they clearly belong to the same group. The FES algorithm considers that by doing a check on the rank values of the services as it is described in lines 10 to 24 of Algorithm 4.

By applying the FES algorithm on examples illustrated by Figure 5.7 we provide an illustrative view of service rank results and how the clustering algorithm will group the services.

Figure 5.8 (a) illustrates a histogram with all services in the example. Service X has the highest service rank. Figure 5.8 (b) illustrates the two groups of services after the k-means algorithm is applied to the example described earlier. We can see that there are three services separated aside from the others and they are on the top right corner of the graph. Definitely, they are the three essential services of this system.

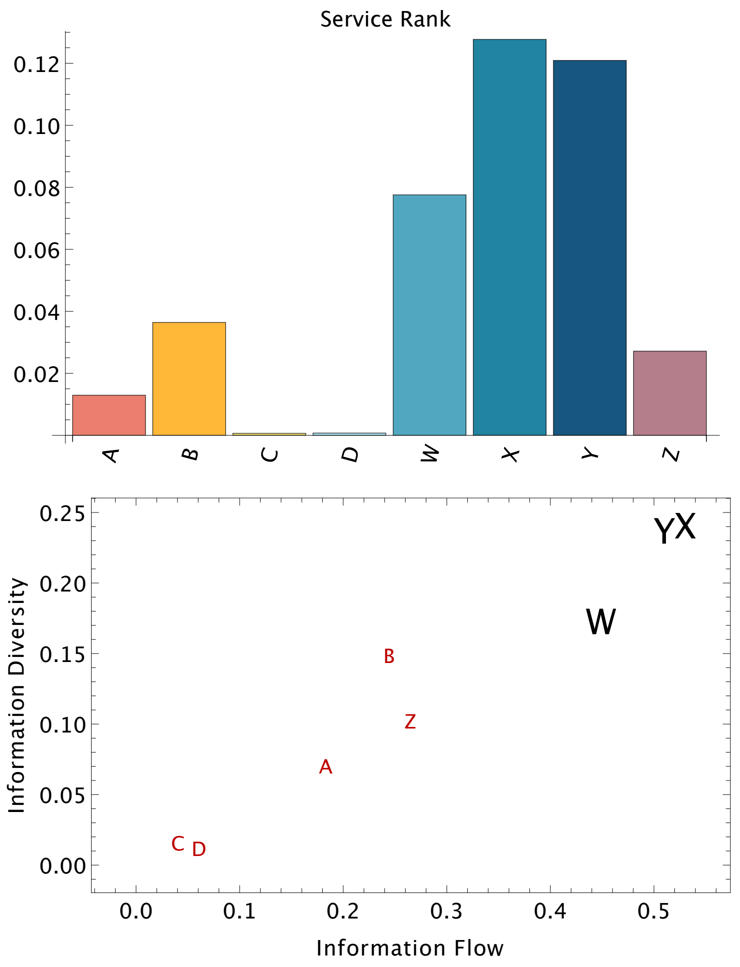


Figure 5.8: (a) histogram with all service ranks. (b) clustering of essential and non-essential services

Algorithm 4: Finding Essential Services

```
Input: ServicesList
Output: EssentialServicesList
// list of tuples $< service, rank >$ sorted by rank in descending order
ServiceRankList ← $\emptyset$
InformationFlow ← $ComputeInformationFlow(ServicesList)$
InformationDiversity ← $ComputeInformationDiv(ServicesList)$
foreach service in ServicesList do
    inflow ← $InformationFlow(service)$
    infdiv ← $InformationDiversity(service)$
    sr ← inflow $\times$ infdiv
    put $< service, sr >$ in ServiceRankList
end
// tally the number of times each service rank has occurred.
TallyList ← $\emptyset$
foreach tuple in ServiceRankList do
    sr ← get(tuple,sr)
    tallyTuple ← lookup(TallyList,sr)
    if tallyTuple !=NULL then
        $tallyTuple < sr, v + 1 >$
    else
        tallyTuple$< sr, 1 >$
        put tallyTuple in TallyList
    end
end
// check for repetition on service ranks between services. get any tuple
tallyTuple ← lookup(TallyList)
if get(tallyTuple,v) == Size(ServicesList) then
    return ServicesList
end
// FindClusters always returns two clusters
Clusters ← $FindClusters(ServiceRankList)$
EssentialServicesList ← $\emptyset$
// first element is ranked highest. list is sorted in descending order
highranked ← first tuple in ServiceRankList
foreach cluster in Clusters do
    if cluster contains highranked then
        EssentialServicesList ← cluster
        break
    end
end
return EssentialServicesList
```

The complexity of the FES algorithm depends on three others, algorithms 2, 3 and the *k-means*. Algorithm 2 complexity is defined in terms of the number of services in the system. It has two loops that iterate through each of the services. Let n be the number of services, by iterating two loops of size n, the algorithm complexity in the worst case scenario is defined as $O(n)$. Algorithm 3 complexity is defined in terms of the number of types of tokens exchanged in the system. It has a main loop that iterates through types of tokens.

The complexity of FES algorithm itself is defined by three loops in terms of number of services, in addition to the other three algorithms complexity. Therefore, let n be the number of services, then FES complexity can be defined as $O(n^3 \times logn)$.

5.4 Evaluation

To evaluate the essential services algorithm we devised a case study of a SCADA system very close to systems deployed in the real world. The SCADA system, as illustrated in Figure 5.9, consists of five networks: one corporate, three field and one remote network, comprising of 43 nodes. The corporate network is used by the corporation employees to monitor and control the field networks. The remote network is used by remote employees, so that they can access and manage the system remotely. The field networks are the ones that acquire monitoring data from the critical infrastructure system.

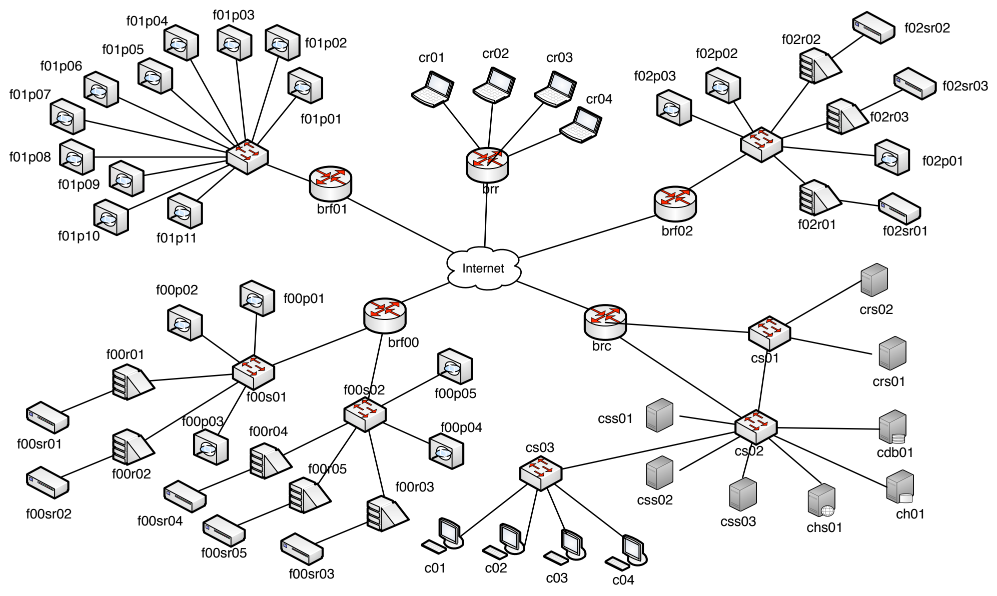


Figure 5.9: SCADA System network topology

In this SCADA system, the corporate network contains three MTUs ($cssXX$), one historian ($ch01$), one relational database ($cdb01$), one HMI server ($chsXX$), four workstations used by the employees (cXX) and two remote servers ($crsXX$) that provide access for remote users. The remote network consists of four clients using PC workstations ($rcXX$), and finally the field networks contain RTUs ($fXXr0X$), sensors ($fXXsrXX$) and PLCs ($fXXpXX$). Each network is connected to the internet through a router ($brXX$), and the networks also have internal switches ($fXXsXX$).

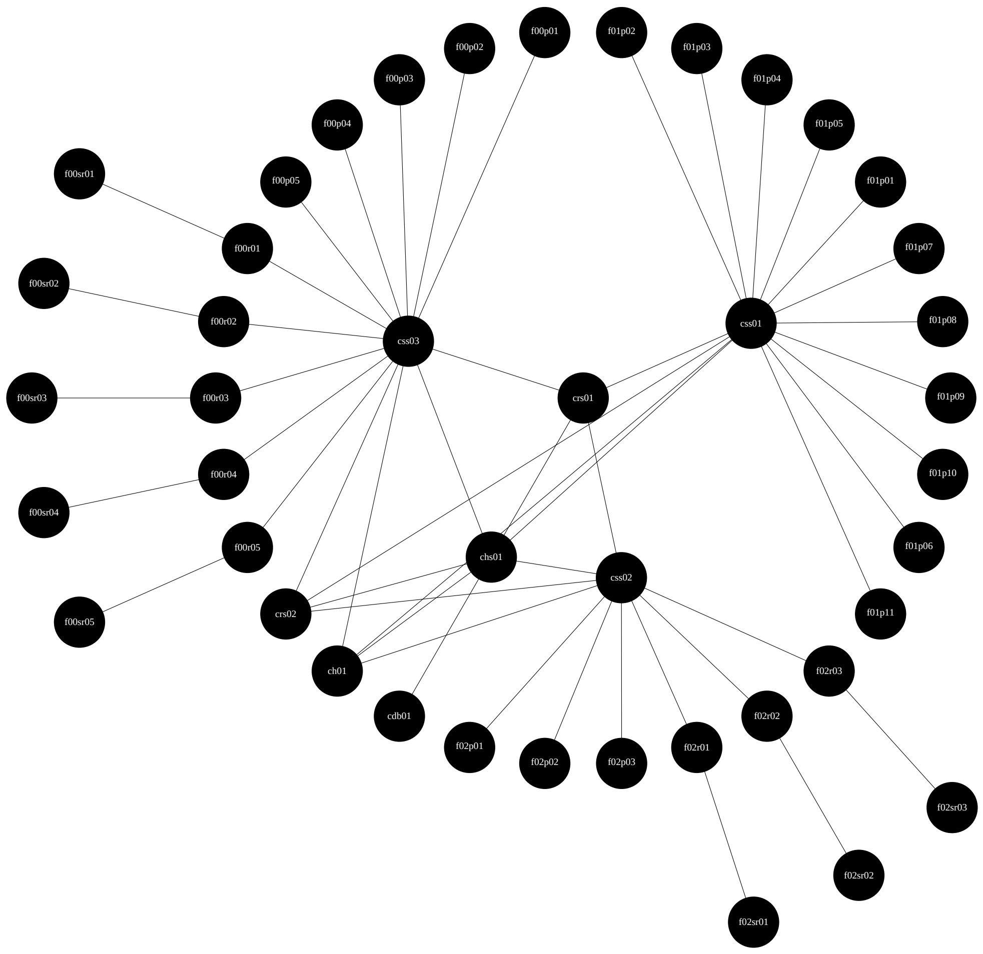


Figure 5.10: SCADA System *service graph*

The simulation is built in OMNET++ [Varga, 2001] and we use SCADASim (see Chapter 3) to build the simulation. From simulation network topology presented in Figure 5.9 we generate the service graph illustrated in Figure 5.10.

5.4.1 End system

The end system is the system being under control by the SCADA system. For this simulation we assume that this SCADA system is controlling a power grid plant. A power grid consists of four components: generation, transmission, distribution and load. The generation is responsible for generating electricity from sources such as: coal, hydro, nuclear, and so forth. Transmission is responsible for transmitting the generated power to the distribution site. And then distribution is responsible for providing the electricity we use everyday. The load is responsible for defining how much load each part of the grid will receive. For instance, Industrial areas usually have higher demand, and consequently higher load, than some other regular areas in the city.

A common analysis tool for this type of system is the power flow analysis. Its goal is to provide accurate information for each bus in the system due to specified load and generator real power. There are numerous tools that providing such analysis. We are interested on simulation tools that could be plugged into the SCADASim simulator. We decided to use Power Systems Analysis Toolkit (PSAT) [Milano, 2005], a tool used to design and analyse electrical power systems.

The end-system will be represented by power grid networks, each network being controlled by one SCADA field network. The goal of the SCADA system is to monitor the power grid networks, as we are interested on the network traffic generated by the services used to control and monitor the power grids. Field network *F01XX* is responsible for controlling power grid 1. Field network *F02XX* is responsible for controlling power grid 2 and field network *F00XX* is responsible for controlling power grid 3. Figure 5.11 illustrates the case for field network *F01XX*. The dashed lines represent the power grid network. For the nodes that are not attached to buses, same random data (uniform distribution) would be produced.

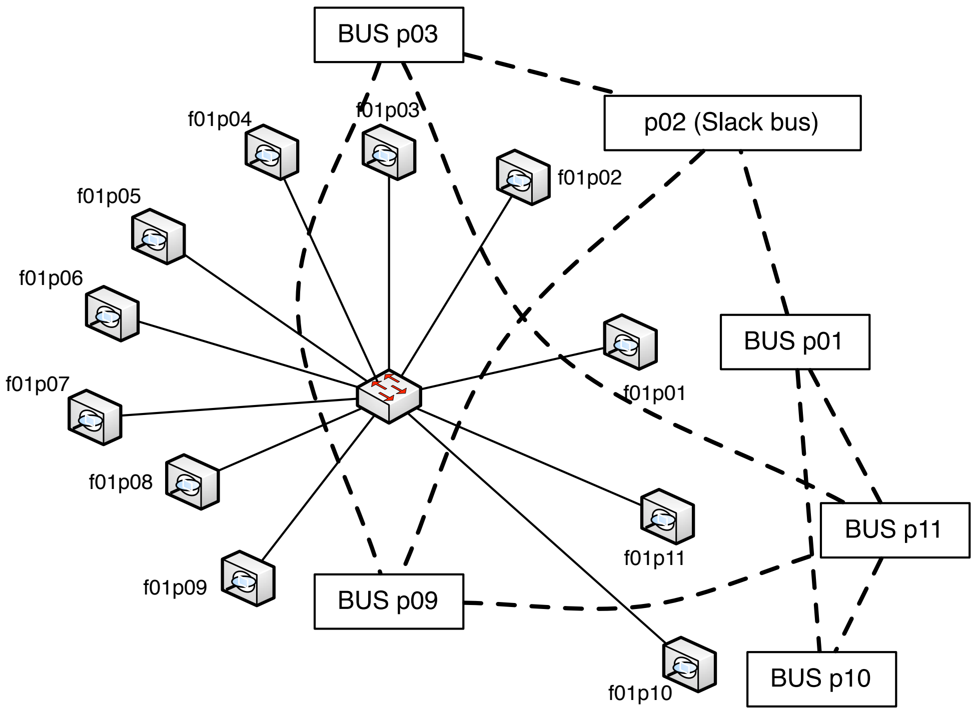


Figure 5.11: Field network monitoring power grid buses

5.4.2 Experiments

In this experiment we use the system described in Section 5.4, applying a network traffic with only one type of protocol (MODBUS). It means all services will exchange messages on the same protocol, with the only different being the content of the messages. We run the experiment for the equivalent to one day of runtime ($86400s$). The services in the system exchange data about the power grids they are monitoring and controlling. Therefore, a typical information exchanged is about the load produced, transmitted and distributed to final users. Figure 5.12 illustrates a colour-coded graph that shows the essential services of the system based on the data acquired from the simulation. Closer to black is the node colour the most essential it is. From Figure 5.12, we can see that nodes $css01, css02, css03, f01p03, f01p07, f01p08, f01p09, f01p10, f01p11$ have the highest service ranks. Comparing with Figure 5.13 that illustrates the betweenness of the same service graph, the nodes $chs01$ included in the betweenness score is not part of the FES algorithm result. This is due to the traffic generated by the nodes $f01XX$, which cannot be captured by the betweenness algorithm.

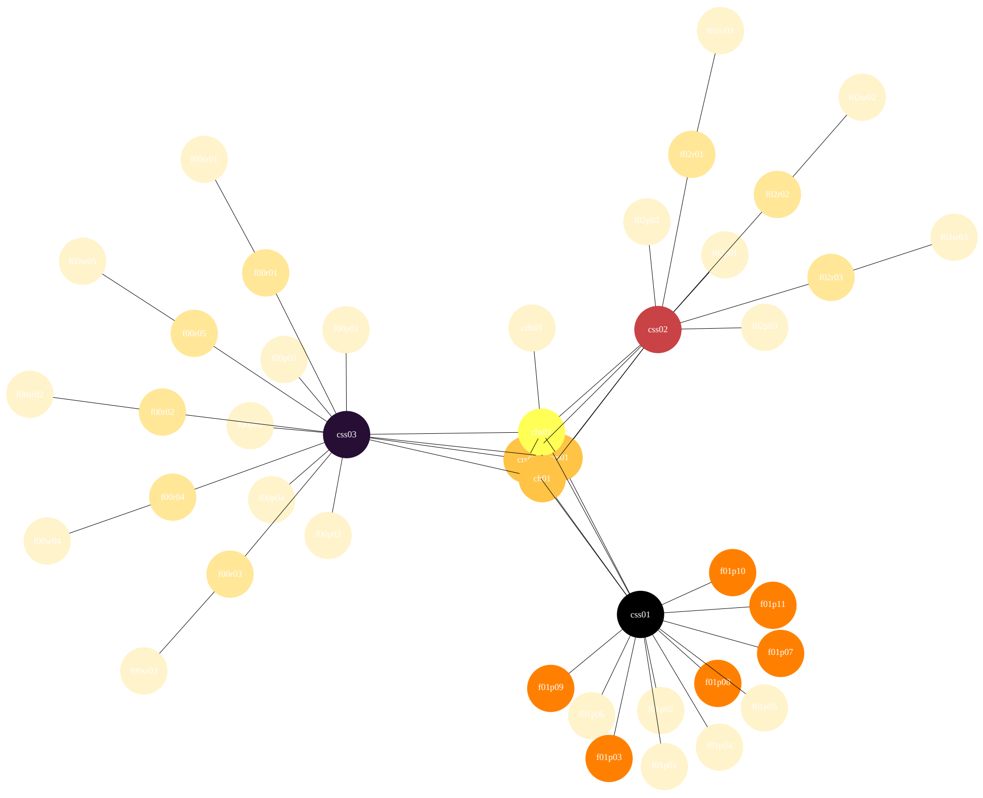


Figure 5.12: Colour-coded service rank results from simulation using MODBUS protocol to exchange data between services

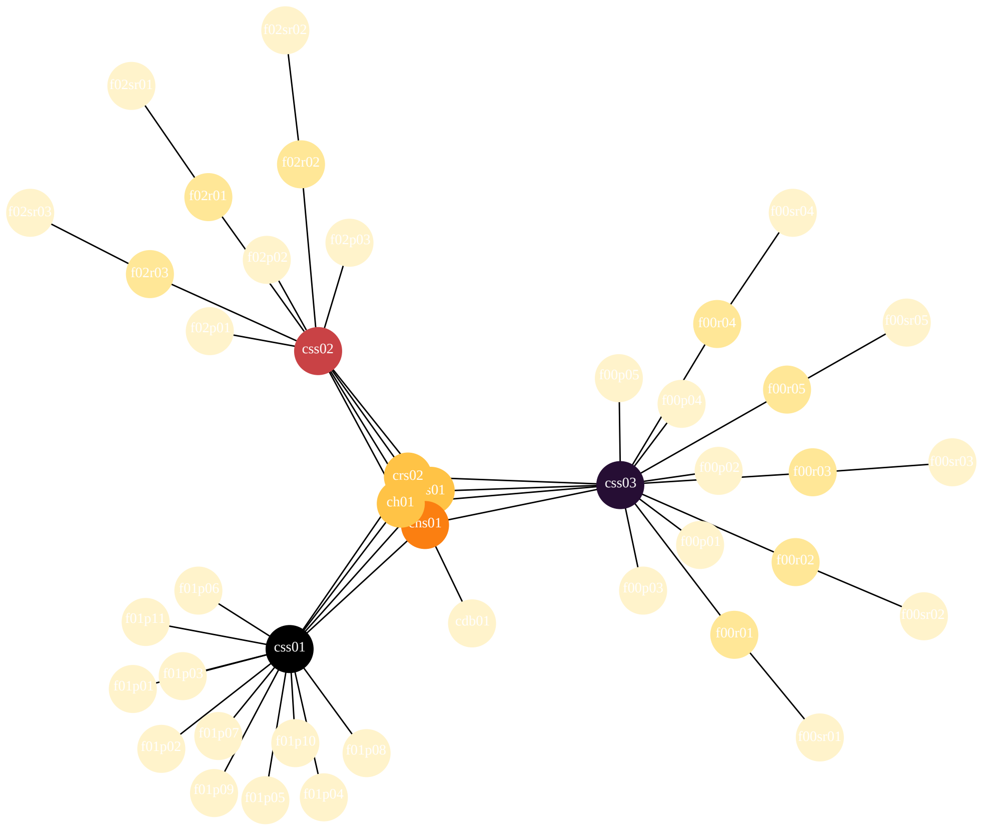


Figure 5.13: Colour-coded betweenness results from the simulated network graph

5.5 Conclusion

This chapter introduced an algorithmic approach to find essential services in SCADA systems. Such technique is relevant because is one of the underpinnings of survivability concept. By identifying essential services automatically and in a standard fashion practitioners will be free of the outdated error-prone approach of manual identification. The chapter also showed that existing techniques used to identify key actors in networks are not appropriate for networks where the dynamism is a key aspect of it. Such techniques work appropriately on networks where the information content is not relevant. The finding essential service algorithm is an essential part of the holistic survivability model as it is used by the proposed model to identify essential services.

Chapter 6

Holistic survivability using performance modelling

This chapter introduces a framework for evaluating the holistic survivability of SCADA systems based on performance modelling. The model uses analytical and simulation analysis to evaluate the status of services performance and the survivability of the overall system. The performance model is a model where services are evaluated individually and then they are put together in order to performance a joint evaluation. The model uses queuing theory to perform the individual analysis and a Bayesian network to perform the overall evaluation, and consequently compute the holistic survivability of the SCADA system.

6.1 Introduction

In this chapter we use holistic survivability combined with a performance model to quantify the survivability of a SCADA system. Performance is extensively applied to quantify the survivability of networked systems. [Liu et al., 2004] proposed a framework that combines performance and availability to quantify performability and survivability. The performance model is a homogeneous CTMC model based on the $M/M/m/m$ queuing model. It assumes memoryless inter-arrival and service times with m servers or processors and m jobs (requests). While the performance model

Part of this work appeared in IEEE 10th International Computer and Information Technology, 2010 handles performance measurements the availability model uses the notion of failures replacing arrival-time and service rates with failure and repair rates, respectively. The composite model evaluates the entire system as if the service model worked uniformly across all its services. Sadly, this assumption does not hold for most systems (SCADA and otherwise). Since services may behave differently based on their characteristics and roles in the system. Evaluating the system in terms of performance and availability without considering the systems heterogeneity gives an inaccurate system picture. Therefore, it is not flexible enough for use on different types of SCADA systems where there are diverse services. Our proposed model overcomes this limitation by considering both a service based model and service interrelationships. Furthermore, since the proposed work is focused on the implication of attacks on SCADA systems, it is assumed to be a fault-free system (i.e. does not consider hardware and software faults for performance degradation).

As presented in Chapter 2, the proposed performance based model is not the first attempt on quantifying the survivability of networked systems looking at performance aspects. However, we identified the following **issues** during the analysis of previous tools and frameworks.

- Existing models [Heegaard and Trivedi, 2009; Jha and Wing, 2001; Liu and Trivedi, 2006; Moitra and Konda, 2000] used to quantify survivability are **not sufficiently flexible** to be applied to SCADA systems since they do not consider the **heterogeneity** of such systems.
- Existing survivability measurement techniques [Jha and Wing, 2001; Liu and Trivedi, 2006; Moitra and Konda, 2000] do not take into account the **interdependencies between services**.

In addition to these issues, and perhaps more importantly, existing models do not focus on holistic approaches for survivability. Consequently, in addition to show how holistic survivability can be implemented we also address the issues mentioned above. We tackle them by:

- measuring the performance of each service according to metrics one is interested to investigate

- combining the services into a probabilistic graphic model that takes into account the aspects of holistic survivability
- computing the holistic survivability of the system by computing the dependencies and service states

It also is important to note that the proposed model does not attempt to produce new intrusion and attack detection systems. We aim to measure holistic survivability that is the ability of a system to cope with undesired events that may affect the performance of the system. In other words, the objective of the model is to respond questions such as (i) what is the overall system survivability in case Service X is unavailable? (ii) how important is service Y to the overall functioning of the system? We address the security aspects of measuring survivability in Chapter 7.

The proposed model defines an analytical and a simulation-based model that quantifies and predicts the survivability of SCADA systems based on specific attributes (e.g. service processing time and network traffic). The model combines queuing theory and Bayesian networks. First, it calculates survivability for each individual service and then aggregates the results into a final score. Over the next sections we will present the model and how it is built on top of holistic survivability concepts.

The first step on applying the proposed performance model is to map it to the context of holistic survivability. It means to describe each of the six-tuple elements that define holistic survivability.

6.2 A performance model based on holistic survivability

As introduced in Chapter 4, the service graph (G) represents the services as nodes and the communication between services as edges. Therefore, there is no change from what was introduced in Chapter 4. The state service model (S_m) is grouped into three states (*Normal, Degraded and Unavailable*). As mentioned in Chapter 4 the states and the numbers of states utilised in a model are not constrained. The modeller should define the number of states and their behaviour. In this proposed model we use three states that defines normal, degraded and unavailable behaviours. We believe that these three states cover the possible behaviours when looking at performance of services. Often we expect a service behaving on a normal manner. It means its performance is according to the limits defined to it. Degraded behaviour may happen when a service is hit with a Denial of Service (DoS) attack on any other undesired event that affects the expected performance of the service. Unavailable services are the ones that do not respond requests. Many factors may be responsible for such a behaviour, including malicious attacks (e.g DDoS, DoS). Even though one may argues that more states and consequently behaviours should be possible, we believe that these three states are enough to quantify survivability in terms of performance. And as discussed earlier in this thesis, the state model is flexible enough to accept the inclusion of other states if necessary.

Even though we introduced an algorithmic approach to find the essential (E) and non-essential (I) services of a SCADA system in Chapter 5, in this chapter we use the manual approach for defining such services, as the initial goal of introducing the performance model and show how it is computed. Then, we show how the model can be learnt from data acquired from network traffic.

6.2.1 Performance metrics - M

In this section we introduce our performance metrics used to measure the behaviour of SCADA systems regarding its performance. The metric goals are to evaluating the performance of a SCADA system by looking at its services.

The basis for computer system performance can be derived from the ISO 9126 standard. It measures the quality of computer systems. According to them performance is characterised by efficiency, and it is further divided into two categories: time behaviour and resource utilisation. Time behaviour metrics reflect the speed efficiency of the system/service being measured. Usually lower times mean better performance and higher speed (of processing). On the other hand, resource utilisation measures the use of resources within a period of time. CPU utilisation, memory usage, I/O usage are examples where can apply resource utilisation metrics for evaluation. Resource utilisation is defined by the ratio of busy time of the resource versus the total elapsed time of the time period.

Even though resource utilisation metrics are important to evaluate system performance we are more interested on the time behaviour category because it is related with time that is a quite important aspect of real-time systems and systems with real-time and non real-time aspects such as SCADA. The time behaviour category provides various metrics for performance measurement.

Metrics related to time are very important to capture the performance of a service or system regarding speed and time to service. Some services such as real-time ones have pre-assigned times to service. They have deadlines to delivery a service and if it cannot delivery by the deadline the request has to be dropped. It means the client will not be serviced and drastic consequences may affect the users of the system. For instance, a water plant may have to open a valve at a specific time of day, a failure on that may compromise people lives that depend on that water supply. We see four metrics as the most important ones for the type of systems we are dealing with:

- *Response time* the time a service takes to respond a request. It includes the waiting time and the actual processing time.
- *Network latency* the time it takes to either a request travel from the origin to the destination or vice-versa.
- *Waiting time* the time a request has to wait in the queue to be processed.
- *Processing time* the actual time taken by the service to process the request.

We believe that the evaluation of the services is the best approach to have an accurate picture of the system and consequently to detect possible failure states. The proposed performance model uses time behaviour metrics as attributes: *waiting time, processing time and response time*.

In addition to these three attributes we also use a network traffic attribute that measures the traffic of the systems network. However we did not include network latency as part of the metric attributes. It is manly because we are more interested on the service behaviour. Any delay on the network will affect the service performance, and consequently will be captured by the metrics.

Performance metrics are widely used to evaluate the performance of servers, networks, and services. Latency, throughput, response time, service processing time, completion time, and speed up are examples of widely known metrics utilised in performance measurement [Menasce et al., 2004]. All metrics aimed to evaluate the performance of a service are under the Performance category. Those metrics are used to evaluate the system in terms of availability, reliability, and performability. Performance metrics are the favourite metrics to analyse survivability, and dependability. In this thesis we do not define any new performance metric, we show how the already used metrics could be used in our model. Although the ones described here should be relevant for SCADA systems.

A SCADA system is a chain of services where a degraded service may compromise the entire system. A regular SCADA deployment may contain from dozens to hundreds of field devices located on remote sites and servers that are responsible for managing these remote devices and to provide monitoring and controlling tools to system operators. These servers are COTS machines widely used in all type of industries and business. Therefore, they should be evaluated as such.

The field devices are extremely vital to the functioning of the system. Because they provide the field information needed to system operators take decisions. Such decisions affect the quality of the service provided, nevertheless to say any wrong decision may have disastrous consequences. More specifically attacks on such devices may lead to disastrous consequences.

SCADA are, in essence, systems based on the client and server architecture. There are clients requesting functionalities from services, which can be considered servers. Service time, response time, are common metrics used to evaluate the performance of servers in systems based on the client/server model [Menasce et al., 2004].

Service time is used to evaluate the time a service takes to process a request from a client. Response time evaluates the time a request waits to be processed plus the actual processing time. Here we define **Service Processing Time (SPT)** as the response time, and it also includes the time to response arrive at the client. Different from the individual metrics SPT explicitly trace all steps taken by a request. It measures the request round trip to and from the service plus the time the request took to be processed. For the service client, the measurement includes the effects of network latency, request waiting time and the actual processing time by the service. Figure 6.1 illustrates a typical user request using a service time-line. Service client requests a service at time t_0, the request arrives at the service provider at time t_1 and goes into its waiting queue, at t_2 the request is removed from the waiting queue for processing. The request is processed at time t_3 and at time t_4 the response arrives at the user. From the point of view of the client, processing time represents the time that takes to the request being delivered, processed, responded and delivered back to it.

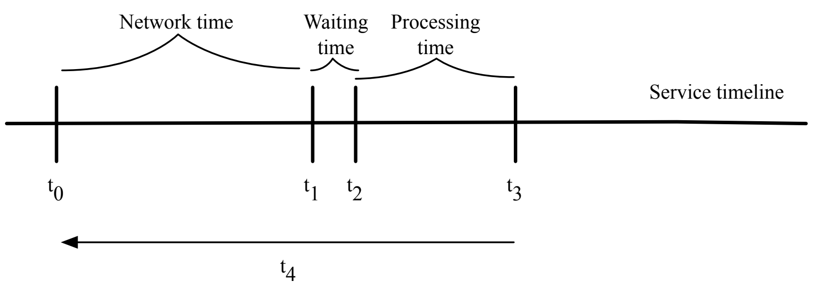


Figure 6.1: Service time-line

Figure 6.1 illustrates these two perceived service processing times. For the service provider, processing time is represented by $st = (t_2 - t_1) + (t_3 - t_2)$, and for the service client is defined as $(spt = t_1 - t_0) + (t_2 - t_1) + (t_3 - t_2) + t_4$, where t_4 is the time to the response being delivered to the client. To refer to service time we use the notation SPT and for service processing time from client's point of view SPT_c. Therefore, Equation 6.2.1 shows the metrics (M) applied to the proposed performance model.

$$M = \{SPT, SPT_c\} \tag{6.2.1}$$

6.2.2 Probability distributions - P

Queueing theory is commonly used to analyse the behaviour of computer systems via their performance [Bolch et al., 1998]. The analysis is made through a queueing model that represent appropriately the system. It takes into consideration some aspects of the system such as inter-arrival rates, service times, number of servers and the size of the queues in those servers. Such approach is widely used in the survivability literature, as described in Chapter 2. Queuing theory models such as $/M/M/n$ and $/M/M/n/k$ are usually applied to analyse survivability. Those models provide closed formulas making easier to analyse systems analytically. For instance, [Liu, 2010] has shown diverse case studies applying different metrics and using queueing theory models to analyse the survivability of computer and networked systems.

The holistic survivability model proposed here deviates from such models by making it flexible to be used with multiple metrics, using analytical tools, supporting datacentric approaches, where simulated and real data could be fed into the model, and also supporting the input of system experts, as the model uses Bayesian networks and therefore is flexible to accepts external inputs (priors) (see Chapter 2).

We also understand that sometimes one metric could be more important than other, or even not be relevant at all, therefore making the use of different metrics and giving the flexibility of using them accordingly makes the model more powerful and consequently more useful. In this section we show how holistic survivability can be used with performance metrics to measure and analyse the survivability of a SCADA system.

To execute the analytical analysis with system experts inputs and the simulation-based analysis we rely on queuing theory, as mentioned before. However we apply different queuing theory models. SCADA services have different requirements and run on different types of hardware. To represent such differences on the model we use two different queueing models. We defined two type of services: *Enterprise and Field* to represent services in SCADA systems.

Field services are services that run on field devices with very limited resources (e.g. RTU, PLC). Enterprise services run the ones that run on more powerful hardware, usually server machines, which are designed to be scalable (e.g. MTU, HMI Server). Therefore, enterprise services are represented by $M/M/n$ model and field services by $M/M/1/K$ model.

6.3 Quantifying holistic survivability using a performance model

As introduced in Chapter 4, holistic survivability is quantified by the joint distribution that marginalises over all services of the system where the essential services are on an intended state. In this performance model the intended state is the *normal* state. Therefore, Equation 6.3.1 describes the quantification of holistic survivability using a performance model. E represents the essential services and I the non-essential ones.

$$P(E = normal) = \sum_{I} P(E, I) \tag{6.3.1}$$

The proposed model combines queuing theory and Bayesian networks to quantify the holistic survivability. Bayesian networks is not the only probabilistic graphical model available, as described in Chapter 2, in addition to Bayesian networks there is Markov Random Fields (MRF), also known as Markov networks. Bayesian networks are based on directionality, they are Direct Acyclic Graphs (DAG), where the nodes correspond intuitively to direct influence of one to another. This is exactly what we want to show with performance metrics in holistic survivability. A service that needs someone's else functionality is dependent of that service in order to provide its own service. A client that needs a service provider is dependent of that service or set of services. This dependency is represented by a direct edge between the two nodes.

To compute holistic survivability first at service level the state of each service is measured. Then, they are aggregated into a Bayesian network that is used to compute the final score. In other words, the Bayesian network is used to compute Equation 6.3.1, which represents holistic survivability.

The methodology to compute holistic survivability based on a performance model is divided into three parts:

1. Individual service performance analysis - Computes survivability for each service.

2. Bayesian network construction - Builds the Bayesian network. It can be created either manually, by a system expert, or automatically through the learning from data process. This process is introduced in Section 7.6.

3. Computing survivability - Perform queries to quantify the overall survivability of the SCADA system.

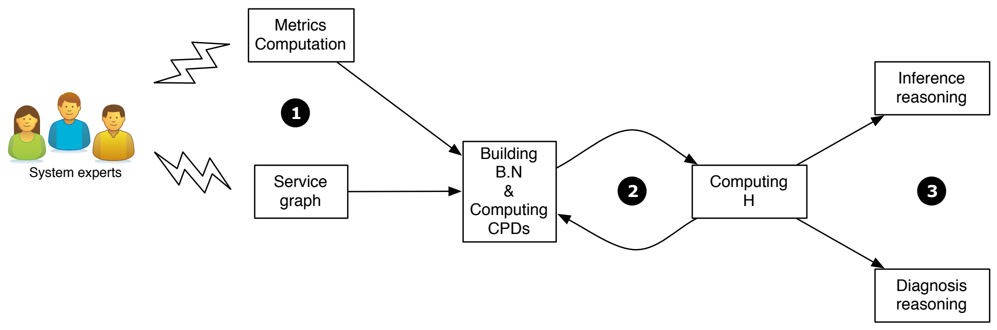


Figure 6.2: Three steps to compute holistic survivability

Figure 6.2 illustrates the three steps needed to compute holistic survivability. It is an adaptation from Figure 4.7 that provides a generic view of the steps to quantify holistic survivability.

6.3.1 Individual service performance analysis

As mentioned earlier, holistic survivability is represented by a probabilistic graphical model, where the nodes are represented by probability distributions. This makes the states of a service be necessarily defined in terms of a probability distribution, where the distribution represents the probability of a service being in each of possible states, in this case three possible states ($S_m = \{Normal, Degraded, Unavailable\}$). In the next sections we show how we apply queuing theory to generate the service state distributions for both Field and Enterprise services.

6.3.1.1 Computing SPT for field services

Field services are modelled as Continuous Time Markov Chains (CTMC). The number of requests in the service represents a state of the Markov chain. Field services have limited capacity for handling requests. This characteristic can be represented analytically by the $M/M/1/K$ queuing model [Bolch et al., 1998; Pardoux, 2008]. It is assumed that request arrival rates are Poisson processes with λ rates, service times are exponentially distributed with rate μ and buffer size K (including one being processed) and runs on a FCFS (First Come First Serve) basis.

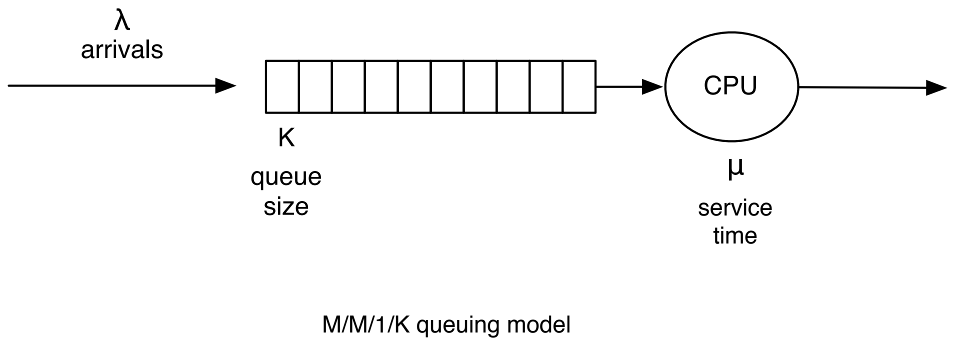


Figure 6.3: M/M/1/K Queuing model

The $M/M/1/K$ model has been chosen because it has limited buffer K and it is used to model servers with one CPU only. It is reasonable to assume that field devices have only one CPU and limited capacity to handle requests.

Analytically, the probability of a request being rejected is proportional to the time the queue is full [Pardoux, 2008]. The probability P_K defines the **unavailable** status of the service. Equation 6.3.2 calculates the probability of a service being in state n. The Equation is derived by solving the balance equations that show the limiting probabilities of each state in a CTMC [Bolch et al., 1998; Hillier et al., 1990; Pardoux, 2008].

$$\begin{cases} (\frac{\lambda}{\mu})^K \frac{1-(\lambda/\mu)}{1-(\lambda/\mu)^{K+1}}, & \text{if } \lambda \neq \mu, \\ \frac{1}{K+1}, & \text{if } \lambda = \mu. \end{cases} \tag{6.3.2}$$

Given λ, μ and K, $P\{Unavailable\}$ is calculated by Equation 6.3.4, $P\{Degraded\}$ by Equation 6.3.3 and $P\{Normal\}$ that represents the probability of **normal** status by Equation 6.3.5.

A service is considered as **degraded** when its service time is over a threshold value t. The threshold is defined by a system expert. W_q is the mean service waiting time in the queue, the **degraded** status probability is defined by the summation of probability states for cases where $n \times W_q$ is greater than t for $1 < n < K$. More formally,

$$P\{Degraded\} = \begin{cases} \sum_{i=2}^{n-1} P_i, & \text{if } i \times W_q > t \end{cases} \tag{6.3.3}$$

$$P\{Unavailable\} = P_n, \text{ where n = K} \tag{6.3.4}$$

$$P\{Normal\} = 1 - P\{Unavailable\} - P\{Degraded\} \tag{6.3.5}$$

It is important to notice that even though the survivability of a service could be computed analytically, the numerical analysis grows in complexity as the number of services and their interconnections are added to the system. For such cases it is more appropriate to use simulations to compute the states of services.

In a simulation the service states are computed differently. If a field service is dropping connections it is considered *unavailable*, if the service time is over a threshold t it is *degraded*, otherwise it is *normal*. Equation 6.3.6 illustrates such cases.

$$\begin{cases} \text{Unavailable}, & \text{if Dropped connections} > 0 \\ \text{Degraded}, & \text{if SPT} > t \\ \text{Normal}, & \text{otherwise} \end{cases} \tag{6.3.6}$$

6.3.1.2 Computing SPT for enterprise services

Enterprise services are modelled as CTMCs where the state of the Markov chain is defined by the number of requests being processed. The model assumes a Poisson arrival rate λ, an exponential service time μ and requests are handled in a FCFS (First Come First Serve) basis. The $M/M/s$ assumes an infinite buffer and can process s requests at once [Chan, 2000]. Figure 6.4 illustrates the model.

Enterprise services represent common application servers such as HTTP and database servers. It is reasonable to assume that such services can either be deployed in cluster environments with s servers or in multiprocessor machines with s CPUs, consequently making them able to process s requests at once. The equations described here assume that $\lambda < s \times \mu$, so that $\frac{\lambda}{s \times \mu} < 1$. Otherwise, the summations will not converge [Hillier et al., 1990].

Analytically, Equation 6.3.8 is used to compute the status probabilities. It computes the probability of the service being in state n, which represents the number of requests in the service.

$$P_0 = (\sum_{n=0}^{s-1} \frac{(\frac{\lambda}{\mu})^n}{n!} + \frac{(\frac{\lambda}{\mu})^s}{s!} \frac{1}{1 - \frac{\lambda}{s \times \mu}})^{-1} \tag{6.3.7}$$

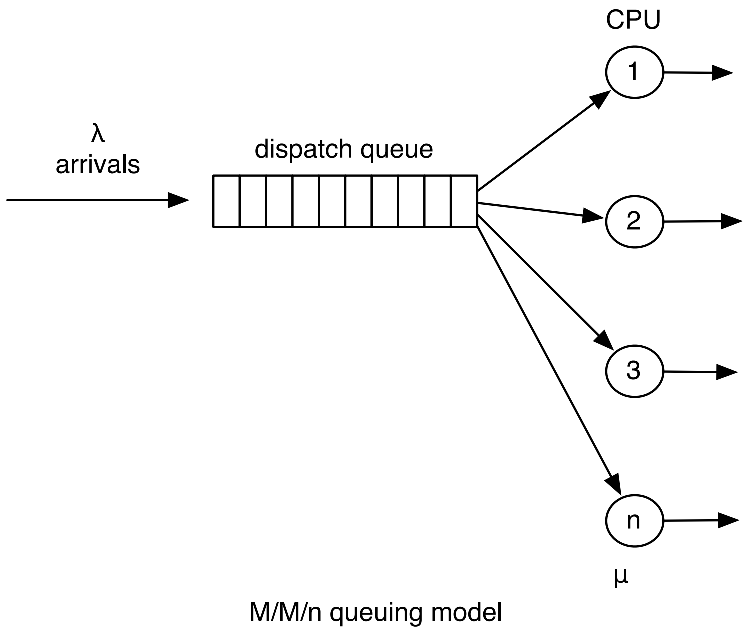


Figure 6.4: M/M/n Queuing model

$$P_n = \begin{cases} \frac{(\frac{\lambda}{\mu})^n}{n!} P_0, & \text{if } 0 \leq n \leq s, \\ \frac{(\frac{\lambda}{\mu})^n}{s! s^{n-s}} P_0, & \text{if } n \geq s \end{cases} \tag{6.3.8}$$

We define a service as being **normal** when there is no delay to process requests, i.e. no waiting time ($W_q = 0$). The probability of being normal is defined by Equation 6.3.9. The probability of being **degraded** is defined by Equation 6.3.10. Because the model assumes every request will be processed (sooner or later) the **unavailable** status probability is calculated by Equation 6.3.11. The threshold t can either be defined by a system expert or learnt from data.

$$P\{Normal\} = \sum_{n=0}^{s-1} P_n \tag{6.3.9}$$

$$P\{Degraded\} = \sum_{n=s}^{\infty} P_n, \text{ if } W_q \times n \leq t \tag{6.3.10}$$

$$P\{Unavailable\} = 1 - P\{Degraded\} - P\{Normal\} \tag{6.3.11}$$

For simulations, the states of enterprise services are computed differently. If the service time is over a threshold e the service we consider the service as *unavailable*, if there is latency time (LT) and the latency time plus the service time (ST) is not over threshold e we consider the service as *degraded* and if there is no latency time and the service time is not over threshold e we consider the service as *normal*. Equation 6.3.12 illustrates the cases.

$$\begin{cases} \text{Unavailable,} & \text{if ST} > \text{e} \\ \text{Degraded,} & \text{if } LT > 0 \text{ and } LT + ST \leq e \\ \text{Normal,} & \text{otherwise} \end{cases} \tag{6.3.12}$$

6.3.2 Bayesian network construction

As introduced earlier, holistic survivability is quantified by computing the joint probability distribution of the services in the system, where the marginalisation over essential services is of interest. To compute this joint distribution we will use a Bayesian network, whose computation is represented by the chain rule introduced in Chapter 2.

There are two ways of building the Bayesian network. One way is to leave for the experts to build it manually. The other way is building the network automatically. We describe the automatic process (also called learning) in Section 7.6. To build the Bayesian network and consequently to use the holistic survivability model we need three things:

- Bayesian network structure - Because we are using a manual approach the structure is defined by system experts based on how the system as designed to communicate when functional. This network structure uses the dependency graph devised from the system. Nodes as services and links between the nodes as edges.
- Conditional Probability Distributions - They represent the probabilities of service states. The most used forms of CPDs are the Conditional Probability Tables (CPT), which are widely used when networks are built by system experts. In

theory a system expert could define the probabilities they wish to represent the states of the service. However, this is not a clever approach as the system should be modelled as close as the one deployed (or to be deployed). To have behaviour that approximates to the reality we model the system using Queueing theory, as already discussed in Section 6.3.1.

- Essential services - One important aspect of survivability and consequently of holistic survivability is the essential services. In order to quantify the survivability of a system essential services needed to be defined. In this section essential services will be defined manually. However, in Section 7.6 we apply the algorithm devised in Chapter 5 to find essential services automatically.

To facilitate the understanding we provide a small and simple client server system that is use to demonstrate the feature of this performance model on quantifying holistic survivability. A more detailed and thoroughly evaluation of the model is presented in Section 7.7.

Figure 6.5 illustrates a simple system with two variables (nodes) $Var = \{A, B\}$, where A is a service client and B a service provider. There is also a client but it is not part of the evaluation. Figure 6.5 also shows that A performs requests and B responses to such requests. By applying Definition 8, we have the dependency graph illustrated in Figure 6.5 (right). And consequently, by applying Definition 11, the holistic survivability for this system is defined as $P(A = normal) = \sum_B (A, B)$, where A is the essential service and B the non-essential one. The fact that A is an essential service is not based on the algorithmic approach introduced in Chapter 5, but to the system expert choice. Chapter 5 shows how essential services can be identified through an algorithmic approach.

The dependency graph is the network structure that is used to build the Bayesian network. Then by using Definition 3 the factorisation that represents the joint probability distribution is defined as $P(A, B) = P(A|B) \times P(B)$.

Once that the network structure is built the next step is to define the CPDs that represent the probabilities of the services (A, B) being in one of the three states *Normal, Degraded and Unavailable*. There are two type of CPDs on a Bayesian network: conditional CPDs, and non-conditional CPDs. The conditional CPDs are the ones whose its values are dependent of its parents, whose parents are other services in the system.

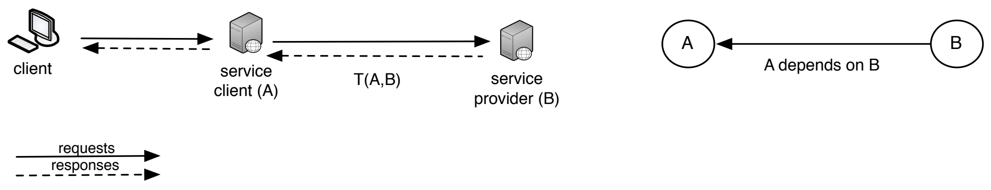


Figure 6.5: Simple system example

In this example. B is a parent of A, consequently, the CPDs assigned to A is conditional on B, and B has a non-conditional CPD. For each node X_i there is a CPD $P(X_i|Par_G(X_i))$.

For this small case one could have a system expert setting up the CPDs for the two services. This is a reasonable approach when the system is relatively small and more importantly, when the expert has knowledge about the system. As first step the expert should define the CPD for service B. As we can see from the dependency graph (Figure 6.5), service B is a parent node, it means it does not have any dependency, therefore the process of creating its CPD is straightforward. The expert can just define three percentage scores, one for each possible state ($S_m = \{Normal, Degraded, Unavailable\}$) or by using the equations defined in Section 6.3.1 and defining inter-arrival rates λ and service rates μ the expert could come up with the CPD for service B. Lets say the expert computed the following values for the states: $N = 0.88, D = 0.106, U = 0.014$. It means, the expert expects that service B keeps most part of its time on a normal state, (88%) of that time, and it is also expected a very low probability for the service being under degraded or unavailable states. For service A the process of defining the CPD is the same, however more information is necessary as there is a conditional dependence between service A and B, $P(A|B)$. Therefore, the CPD that represent service A is defined by Table 6.1. The state names in Table 6.1 are abbreviated due to space restrictions.

Here, the expert could resort for some simulations to help on creating the CPD for service A. Applying analytical equations defined in Section 6.3.1 will not work properly because the conditional dependence could not be modelled. To make easier to create the CPDs a simulation using field services (see Section 6.3.1) was performed. The simulation provides information about number of dropped connections and the SPT for each service. Figure 6.7 shows the number of jobs processed and dropped by the services. For instance, service A processed 10000 but 2732 of them were dropped, around 27% of the jobs. Service B performed better, only 104 of 7268 jobs were dropped, around 1.4% of the jobs. The dropped jobs represent the unavailable state in the performance model (see Equation 6.3.6). Figure 6.6 illustrates the SPT for both services in the system. From this information the CPDs for each service can be derived.

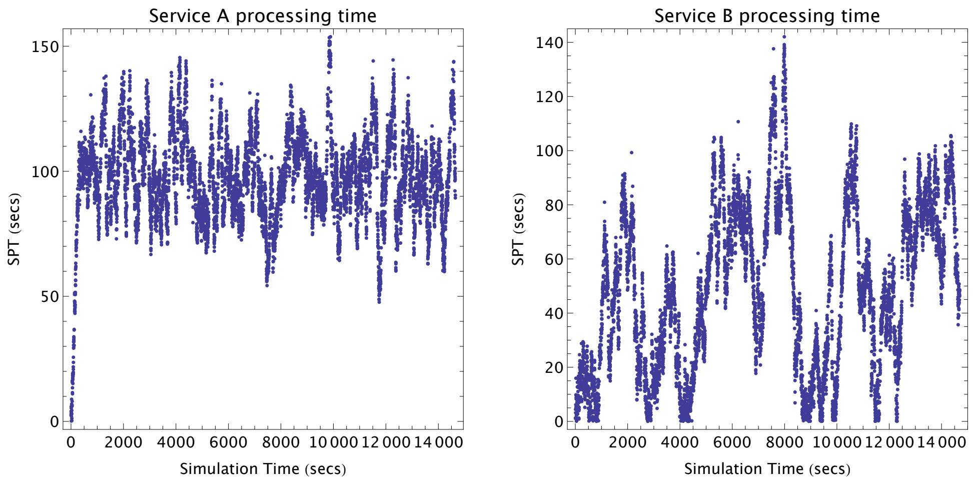


Figure 6.6: Services A and B - Service Processing Time

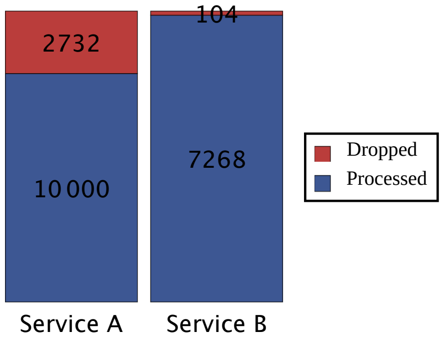


Figure 6.7: Dropped jobs by services A and B

Table 6.1: Service A conditional states

$P(A = N\|B = N) = 0.95$	$P(A = N\|B = D) = 0.50$	$P(A = N\|B = U) = 0.0$
$P(A = D\|B = N) = 0.04$	$P(A = D\|B = D) = 0.30$	$P(A = D\|B = U) = 0.0$
$P(A = U\|B = N) = 0.01$	$P(A = U\|B = D) = 0.20$	$P(A = U\|B = U) = 1.0$

Once that the CPDs are defined the joint probability distribution based on the factorisation that represents the network structure can be computed. Table 6.2 shows how the joint distribution is computed. By using this table a system expert can many different type of questions regarding the system.

Table 6.2: Joint distribution - $P(A, B) = P(A|B) \times P(B)$

$0.95 \times 0.88 = \mathbf{0.8360}$	$0.50 \times 0.106 = \mathbf{0.053}$	$0.0 \times 0.014 = \mathbf{0.0}$
$0.04 \times 0.88 = 0.0352$	$0.30 \times 0.106 = 0.0318$	$0.0 \times 0.014 = 0.0$
$0.01 \times 0.88 = 0.0088$	$0.20 \times 0.106 = 0.0212$	$1.0 \times 0.014 = 0.014$

As introduced earlier, the holistic probability of this system is the probability of service A being normal, $P(A = normal) = \sum_B (A, B)$. Therefore, we are interested on the first line of Table 6.2, which represents the marginal of service $A = normal$. More specifically, by summing up all values of line 1 in Table 6.2 we compute the holistic survivability of the system, as the summation represents $P(A = normal) = \sum_B (A, B)$. Therefore, the holistic survivability for the system illustrated in Figure 6.5 (a) is: $0.8360 + 0.053 + 0.0 = 0.889$.

In the next section we provide a more thoroughly evaluation by demonstrating holistic survivability using performance metrics a close to reality SCADA system.

6.4 Learning from data

Even though, analytical models used to assess survivability are demonstrated useful they are not suitable for survivability evaluation of functional deployed systems. A typical SCADA system may contain hundreds of services, which makes the process of manually building the network and the CPTs impractical even by system experts. Holistic survivability provides ways of evaluating the survivability of systems currently deployed. The data-driven survivability evaluation uses data acquired in the field.

There are existing bodies of research [Cooper and Herskovits, 1992; Jensen, 1996; Murphy, 2002] to generate Bayesian networks from empirical observations. Usually their problem is to find the best network structure between all possible ones. Where the 'best' means the one the fits to what they want to achieve. The proposed model uses a different approach. We use the interaction between the services to create the network structure. So that, we only have one network structure. Usually, the network structure dictates the conditional independence of the variables of the Bayesian network. Otherwise, we would have a highly connected graph representing the entire joint distribution between all variables of the network. Clearly, this is not ideal, as the number of possible network structures grows more than exponentially in the number of services. In the proposed model, the network structure represents the dependencies among the services of a SCADA system. However, there are three important issues to consider before we can apply the proposed technique to build the network structure that represents the services of a SCADA system.

1. The nodes in the Bayesian network must correspond to the services in the SCADA system.
2. There is no straightforward way to use the network logs to learn the Bayesian network values since a log entry is just a description of the connections between any two nodes (services) at any point in time and does not describe the performance issues needed to analyse the dependencies among services.
3. Once the network structure has been created, the CPTs must be generated. To generate the CPTs we need a dataset that shows the variables, in this case services, and its relationships, which can be used to build the CPTs.

Much of the process of learning from data was already described in Section 7.6. The only difference here is that the performance model is interested on the SPT metric, therefore the values recorded utilising the data structure introduced in Section 4.4.1, are the SPT measured at running time (simulations or otherwise). It is important to notice that the SPTs will be converted based on the state model defined in Section 4.2.3.

6.5 Evaluation

To evaluate the Holistic survivability model we use the simulation scenario described in Section 5.4. Services are mapped as nodes and the interconnection between the nodes is based on the dependency graph illustrated in Figure 6.8. SCADASim supports multiple network traffic simulation, as used it to generate traffic from different protocols to provide a more realistic view of the SCADA system. Table6.3 presents the mapping between the three type of traffic profiles and the services used by the simulation

Table 6.3: Traffic profiles

Connection	**traffic profile**
css01 - field devices (fXX)	1
css02 - field devices (fXX)	1
css03 - field devices (fXX)	1
crsXX - chs01	2
crsXX - cdb01	2
crsXX - ch01	2
crsXX - cssXX	2
chs01 - ch01	2
chs01 - cdb01	2
chs01 - cssXX	2
css01 - ch01	3
css02 - ch01	3
css03 - ch01	3

The first step to run the performance model is define the dependency graph that will be used by the model. As described in Chapter 4, the dependency graph is generated with information from the network traffic. Figure 6.8 shows the graph that is used in this simulation.

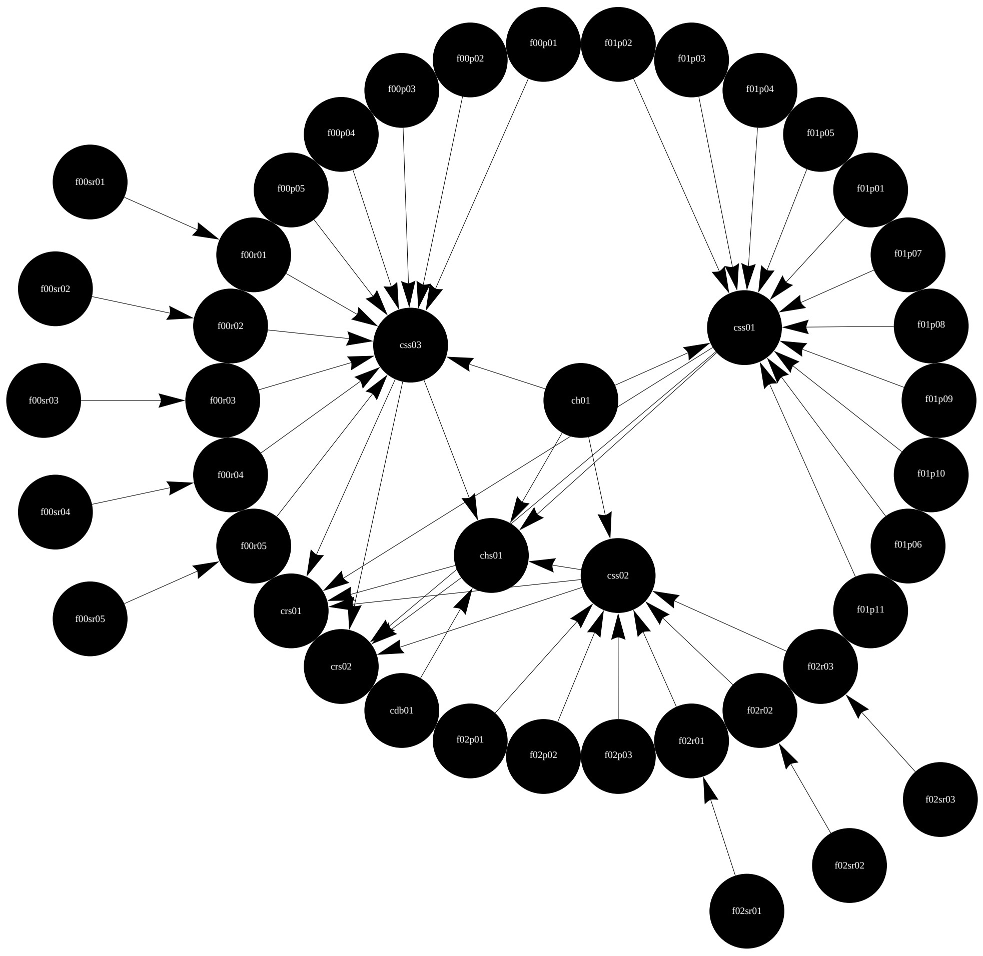


Figure 6.8: SCADA System *dependency graph*

Once the dependency graph is generated the next step is to execute the finding essential services algorithm (see Chapter 5), to then create the factorisation equation that describes holistic survivability. As one can see in Figure 6.9, four essential services were identified. The ones on black. Based on this information the holistic survivability will be quantified by computing the equation 6.5.1, where S is the set of all services in the system.

$$P(css03 = n, css02 = n, css01 = n, ch01 = n) = \sum_{\{css03, css02, css01, ch01\}^c \cap S} P(S) \tag{6.5.1}$$

Table 6.4: Dataset D

Source	Target	SPT	SPT_c
$css03$	$f00p01$	0.35	0.53
$f00p01$	$css03$	0.48	0.76
...	...	...	...

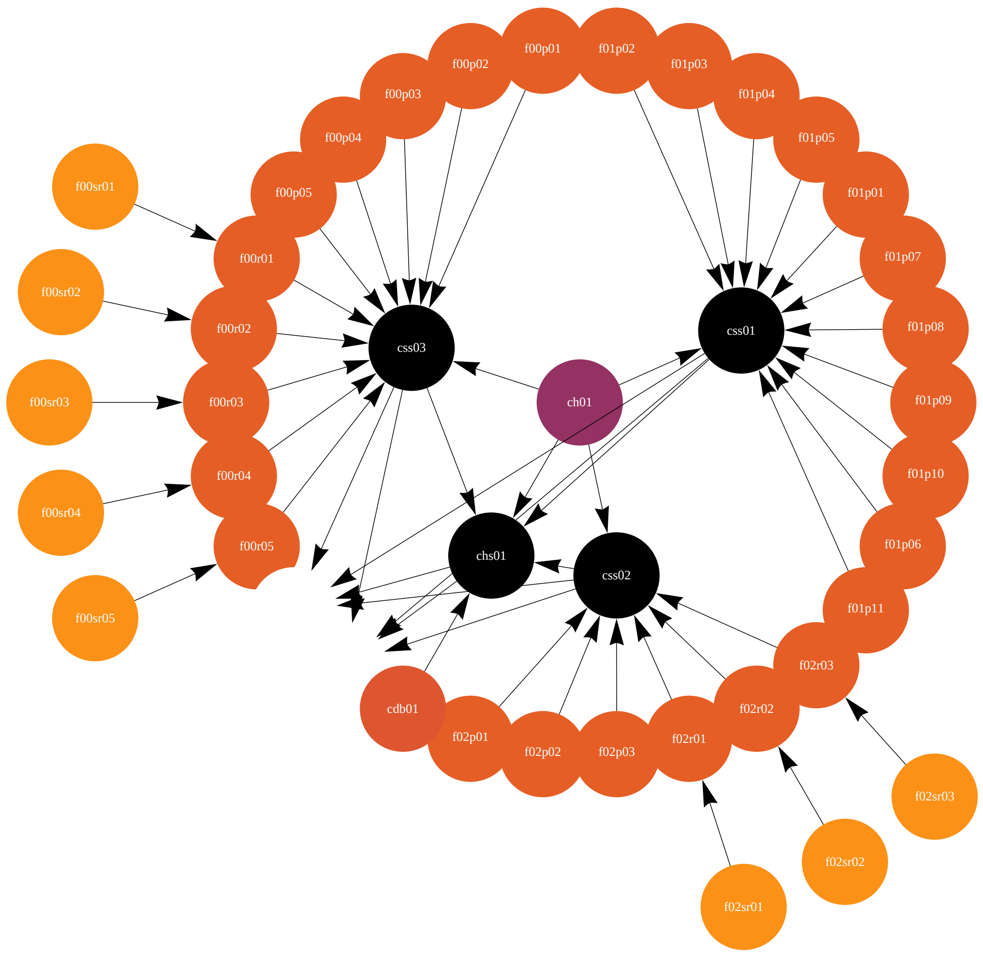


Figure 6.9: Essential services. Colour-coded from white to black (essential services)

At this phase the data is acquired from the deployed services. The first step is to acquire the information being generated by the service. As discussed earlier, every time a transaction occurs the service should parse the content of the transaction to acquire the following attributes; the *source*, which represents the initiator of the transaction, the *target*, the receiver, and *SPT* for both components (service client and service provider).

The factors values were omitted in this part due to space constraints, however their generation is similar to the information provided in Table 1. In this evaluation we compute the overall survivability of the system, which is defined as follows:

$$P(css03 = n, css02 = n, css01 = n, ch01 = n) \approx 0.78$$

6.6 Conclusion

This chapter proposed a performance modelling applied to holistic survivability. The performance model proposed was mapped to use the holistic survivability components and consequently to infer measurements about the overall survivability of a simulated SCADA system. Two techniques (analytical model and learning model) were demonstrated. The learning aspect is the most interesting one as it shows that such models can be used to evaluate systems deployed in the field. Even though only one performance metric was introduced (SPT). Because holistic survivability is flexible enough other metrics could be added to the proposed performance model easily. It only requires an adaptation on the service state modelling and on the metrics component.

Chapter 7

Holistic survivability using a security model

This chapter introduces a security model for holistic survivability and also a security metric tailored for survivability. The security model follows the requirements defined by the holistic survivability definition (see Chapter 4), however it uses another graphical model to representation. The proposed security model uses Markov networks (see Chapter 2 for an overview) for its representation and formalism. The Contextual Vulnerability Index (CVI), a security metric aimed at computer systems, and more specifically, at SCADA systems. CVI extends current CVSS score system by adding attributes pertinent to the concepts of survivability. We conclude the chapter by providing an evaluation of the model using a SCADA system simulated with SCADASim.

7.1 Introduction

When survivability was introduced to computer systems security was one key motivation. [Ellison et al., 1997] clearly mentioned it on his seminal paper that presents survivability to the world of computer systems. Unfortunately, most of work on analysing the survivability of computer systems is based on performance aspects. Chapter 2 provides a more detailed discussion about the current survivability models. We believe that performance models are important as such analysis provide meaningful information about the behaviour of the system, and this is one of the reasons we presented a performance model using the holistic survivability formalism in Chapter 6.

However, in more recent years, there is a tendency on associating survivability with security. [Nicol et al., 2004] presented a survey on model-based evaluation techniques to quantify dependability and security; [Al-Kuwaiti et al., 2009] have shown that security is related to survivability; and [Trivedi et al., 2008] described techniques that makes a clear relationship between survivability and security. Such direction comes with no surprise, as it seems clear that absolute security is impossible to achieve. From the few models addressing security and survivability on computer systems, nearly everyone follows the pattern of attack graphs (trees) and the models are not really aimed at survivability. For instance, there is not much about essential services on such models.

On the other hand, recent attacks targeting SCADA systems (see Chapter 1) demonstrate the fact that such systems are no longer immune to attacks [Falliere et al., 2011; Igure et al., 2006; Munro, 2008; Patel and Sanyal, 2008; Slay and Miller, 2007]. And the current state of affairs in security systems that provide mechanisms based on prevention and interdiction [Coutinho et al., 2008; Goseva-Popstojanova et al., 2001; Gula, 2007; Kolesnikov and Lee, 2006; Krugel and Toth, 2001; Long et al., 2005] are not working. The inability to stop every attack and the complex interactions in SCADA systems demands a different solution. One that focuses on providing critical services in the presence of undesired events, either malicious attacks or not.

In this chapter we introduce a model to evaluate survivability in terms of security. As part of the modelling process we present a new metric, which is focused on the security aspects of SCADA systems. The Contextual Vulnerability Index (CVI) extends the Common Vulnerability System Score (CVSS) [Mell et al., 2007] by adding contextual aspects. Two new attributes that capture the context of the system in terms of survivability are added. The first attribute is *Reachability*, which measures how close a service is to another service in the network. The second one is *Essentiality*, which is based on the service ranking function introduced in Chapter 4. Essential services should be more relevant to the security aspects of the system.

The proposed security model aims to tackle the following issues:

- Lack of survivability models considering a holistic approach on security and survivability. Current models are aimed at vulnerability aspects, how a vulnerability can lead to another, and so forth. The models do not focus on the service aspect. How the vulnerability of a service can degrade other services and consequently the entire system, for instance.

- The security metrics utilised are not tailored or devised with survivability in mind. Usually, they adopt known security metrics to survivability modelling [Wang et al., 2008].
- The models are not flexible enough to support the addition of new metrics or the composition of multiple metrics.

The main goal of this chapter is to introduce a security model applied to holistic survivability. Therefore, over the next sections we map the components of holistic survivability $< G, S_m, S, M, P >$ to the proposed security model.

7.2 Service graph - G

The construction of the service graph follows the rules defined in Section 4.2.1, if there is a transaction between two services in the system, a edge (link) between such services is created. Manual and automatic approaches are supported for the graph construction. A system expert can generate its service graph based on previous experience about the system. However, a better approach is to generate the graph automatically. Such process was introduced in Section 4.4.3. Nevertheless to say, each service represents a node in the service graph. The mapping process is the same already defined in Section 4.2.2.

7.3 State model - S_m

As introduced in Section 4.2.3, the state model describes the possible states a service may have in a system. In this model we defined two services to represent all possible states of a service. *Acceptable* is a state in which the service either has no known vulnerability or the vulnerability has a very low risk. *Unacceptable* refers to a state where the service has a higher probability of being attacked, usually because there are known vulnerabilities in the wind with a high change of being exploited. More formally,

$$\begin{cases} Acceptable & = 1 - CVI \\ Unacceptable & = CVI \end{cases} \tag{7.3.1}$$

7.4 Security metrics - M

Security is an important aspect of survivability, however it has not been recognised as such. We introduce a new security metric aimed specifically at survivability aspects. The Contextual Vulnerability Index (CVI) extends current CVSS score system [Mell et al., 2007] by adding attributes pertinent to the concepts of survivability.

This security metric may draw some similarity with attack graphs. In fact, there is some work in the literature on attack graphs that propose probabilistic metrics to analyse the security of systems [Wang et al., 2008]. The vulnerability score metric differentiates from previous work on attack graphs, as probabilistic metrics in many ways. In attack graphs, the security is related to system state. An exploit is a walk through services vulnerabilities, and consequently, system states. In our case, we map services to nodes, and not vulnerabilities to nodes. An attack graph is a graph with nodes representing vulnerabilities, and the edges representing the probabilities of a vulnerability being exploited in order to chain an exploitation of vulnerabilities. There are some work on attack graphs using probabilistic graphical models. [Liu and Man, 2005] proposed a model that uses a Bayesian network that maps nodes as services (vulnerabilities) and edges as paths to attacks. We think this approach is limited because it based on Bayesian networks that uses directional dependencies. By using such approach we are limiting the number of possibilities on chaining attacks based on the vulnerabilities. In our model we use Markov networks to avoid such a limitation.

We have defined one solely metric to quantify the resistance of systems upon attacks. The metric is defined by a combination of graphs, vulnerabilities scoring system and services interdependency trees. The novelty is on using these three techniques together. Attack graphs, tree and scenarios [Mauw and Oostdijk, 2006; Schneier, 1999] are widely used to describe the potential of attacks and attackers targeting a system. The use of attack trees to assessing vulnerabilities in SCADA systems is not new [Byres et al., 2004], however as far as we know there is no work focusing on these three aspects together.

We have adopted the CVSS system as the base for our security metric because it gives more flexibility for scoring the vulnerabilities. The context-scoring feature gives the flexibility of scoring vulnerabilities based on the context. For instance, vulnerability can be highly scored for Internet based systems and it can be irrelevant for SCADA systems as it affects resources not utilised by them. By adopting the CVSS scoring system we may have different vulnerability scoring based on the context of the vulnerability.

As CVI extends CVSS before we introduce it we give a brief overview on CVSS, and then we introduce CVI and show how it can be incorporated in terms of holistic survivability. For a more detailed description about CVSS we recommend its specification [Mell et al., 2007], which introduces CVSS in further details. CVSS gives scores based on three groups of metrics: *base*, *temporal* and *environmental*. Figure 7.1 illustrates how the three metrics are related. Temporal and Environmental groups are optional and if used they can not provide a score higher than one produced by the base metric group. As shown in Figure 7.1, the temporal metric group can only be used if the Temporal group is also used. CVSS is computed per vulnerability and produces a score ranging from 0.0 to 10.0.

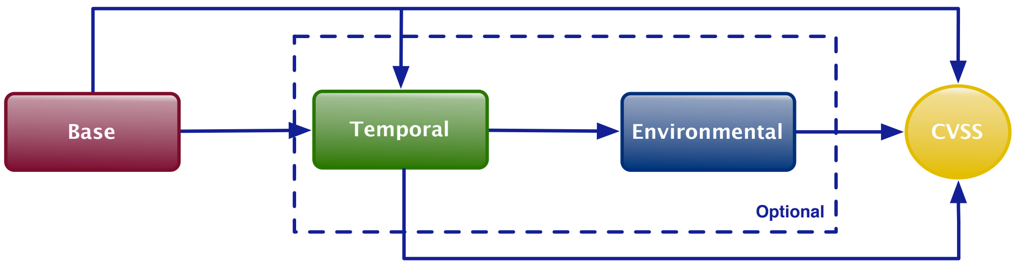


Figure 7.1: CVSS computation flow

Base metric The base metric describes the characteristics of the vulnerabilities. Usually, they are constant with time and across different environments. Table 7.1 describes all the attributes that are part of the base metric.

Table 7.1: Base metric attributes

Abbr.	Name	Description
AV	Access Vector	The attribute describes how the vulnerability could be exploited. If remotely, locally, and so forth. Three possible values: Local(L), Adjacent Network(A) and Network (N).
AC	Access Complexity	Describes the complexity of the attack. If an attacker needs to gain further privileges, how much technical skills are needed, and so on. Also has three possible values: High (H), Medium (M) and Low (L).
Au	Authentication	Measures the number of authentications an attacker needs to exploit a vulnerability. Three possible values: Multiple (M), Single (S) and None (N).
C	Confidentiality Impact	Measures how the vulnerability exploitation impacts the confidentiality of the system. The idea of system is related to the machine in which the vulnerable software is running. For instance, once exploited the vulnerability gives the attacker access to the file system, memory, and so on. It applies for Integrity and Availability as well. Three possible values: None (N), Partial (P) and Complete (C).
I	Integrity Impact	Measures how the vulnerability exploitation impacts the integrity of the system. Three possible values: None (N), Partial (P) and Complete (C).
A	Availability Impact	Measures how the system is affected in terms of availability if the vulnerability is exploited. Three possible values: None (N), Partial (P) and Complete (C).

The base metric is computed as follows:

$$
\begin{aligned}
Ba = {} & RoundToOneDecimal[(((0.6 \times Impact) + (0.4 \times Exploitability) - 1.5) \\
& \times f(Impact)] \\
& Impact = 10.41 \times (1 - (1 - C) * (1 - I) * (1 - A)) \\
& Exploitability = 20 \times AV \times AC \times Au \\
& f(Impact) = \begin{cases} 1.176 & when \text{ Impact} > 0 \\ 0 & when \text{ Impact} = 0 \end{cases} \\
& AV = \begin{cases} 0.395 & when \text{ Local} \\ 0.646 & when \text{ Adjacent Network} \\ 1.0 & when \text{ Network} \end{cases} \\
& AC = \begin{cases} 0.35 & when \text{ High} \\ 0.61 & when \text{ Medium} \\ 0.71 & when \text{ Low} \end{cases} \\
& Au = \begin{cases} 0.45 & when \text{ Multiple} \\ 0.56 & when \text{ Single} \\ 0.704 & when \text{ No Authentication} \end{cases} \\
& C, I, A = \begin{cases} 0.0 & when \text{ None} \\ 0.275 & when \text{ Partial} \\ 0.660 & when \text{ Complete} \end{cases}
\end{aligned}
$$

The CVSS metrics are usually distributed as a vector, in addition to the overall numeric score. Below is the base score metric as a vector.

Base *AV:[L,N,A]/AC:[H,M,L]/Au:[M,S,N]/C:[N,P,C]/I:[N,P,C]/A:[N,P,C]*

Each attribute is defined in its abbreviate form follow by a colon and then the attribute value. For instance, the following vector **AV:L/AC:M/Au:N/C:P/I:N/A:C** is interpreted as: Access Vector requires local access (**AV:L**), Access Complexity is Medium (**AC:M**), There is no Authentication (**Au:N**), Confidentiality Impact is partial (**C:P**), Integrity Impact is none (**I:N**) and Availability Impact is complete (**A:C**).

Temporal metric The temporal metric represents vulnerabilities properties that are mutable over time, it modifies the base score lowering it as much as $\frac{1}{3}$.

Table 7.2: Temporal metric attributes

Abbr.	Name	Description
E	Exploitability	Measures the current status of exploit techniques and code availability. For instance, if there is some code available that exploits the vulnerability the exploitability score should be higher if compared when there is no code available. Five possible values: Unproven (U), Proof-of-Concept (POC), Functional (F), High (H) and Not Defined (ND).
RL	Remediation Level	Describes if there are fixes around. Users could use to prioritise updates. Five possible values: Official Fix (OF), Temporary Fix (TF), Workaround (W), Unavailable (U) and Not Defined (ND).
RC	Report Confidence	Measures the credibility of known details about the vulnerability and the existence of it. For instance, the vulnerability may be just a rumour. Four possible values: Unconfirmed (UC), Uncorroborated (UR), Confirmed (C) and Not Defined (ND).

The temporal metric is computed as follows:

$$Te = RoundToOneDecimal[Ba \times E \times RL \times RC]$$

$$E = \begin{cases} 0.85 & when \text{ Unproven} \\ 0.90 & when \text{ Proof-of-Concept} \\ 0.95 & when \text{ Functional} \\ 1.00 & when \text{ High} \\ 1.00 & when \text{ Not Defined} \end{cases}$$

$$RL = \begin{cases} 0.87 & when \text{ Official Fix} \\ 0.90 & when \text{ Temporal Fix} \\ 0.95 & when \text{ Workaround} \\ 1.00 & when \text{ Unavailable} \\ 1.00 & when \text{ Not Defined} \end{cases}$$

$$RC = \begin{cases} 0.90 & when \text{ Unconfirmed} \\ 0.95 & when \text{ Uncorroborated} \\ 1.00 & when \text{ Confirmed} \\ 1.00 & when \text{ Not Defined} \end{cases}$$

The temporal metric represented as a vector.

Temporal *E:[U,POC,F,H,ND]/RL:[OF,TF,W,U,ND]/RC:[UC,UR,C,ND]*

Environmental metric The environmental metric provides the context of the vulnerability inside the organisation the system is located.

Table 7.3: Environmental metric attributes

Abbr.	**Name**	**Description**
CDP	Collateral Damage Potential	Measures the potential for loss of life and physical assets damage. It can also measure productivity loss or revenue loss. It has six possible values: None (N), Low (L), Low-Medium (LM), Medium-High (MH), High (H) and Not Defined (ND).
TD	Target Distribution	Measures how much of the environment could be target by the vulnerability. It has five possible values: None (N), Low (L), Medium (M), High (H), and Not Defined (ND).
CR, IR, and AR	Security Requirements	Provides the analyst with the option to weight the importance of the affected assets, measured in terms of Confidentiality (CR), Integrity (IR) and Availability (AR). Four possible values: Low (L), Medium (M), High (H) and Not Defined (ND).

The environmental metrics is computed as follows:

$$En = RoundToOneDecimal[(ATe + (10 - ATe) \times CDP) \times TD]$$

$$ATe = \text{Te recomputed with Ba's } Impact \text{ replaced by } AI$$

$$AI = \min(10, 10.41 \times (1 - C \times CR) \times (1 - I \times IR) \times (1 - A \times AR))$$

$$CDP = \begin{cases} 0.0 & when \text{ None} \\ 0.1 & when \text{ Low} \\ 0.3 & when \text{ Low-Medium} \\ 0.4 & when \text{ Medium-High} \\ 0.5 & when \text{ High} \\ 0.0 & when \text{ Not Defined} \end{cases}$$

$$TD = \begin{cases} 0.00 & when \text{ None} \\ 0.25 & when \text{ Low} \\ 0.75 & when \text{ Medium} \\ 1.00 & when \text{ High} \\ 1.00 & when \text{ Not Defined} \end{cases}$$

$$CR, IR, AR = \begin{cases} 0.50 & when \text{ Low} \\ 1.00 & when \text{ Medium} \\ 1.51 & when \text{ High} \\ 1.00 & when \text{ Not Defined} \end{cases}$$

The environmental metric represented as a vector.

Environmental *CDP:[N,L,LM,MH,H,ND]/TD:[N,L,M,H,ND]/CR:[L,M,H,ND] /IR[L,M,H,ND]/AR:[L,M,H,ND]*

7.4.1 Contextual vulnerability index - CVI

The Contextual Vulnerability Index extends CVSS [Mell et al., 2007] by adding two attributes that provides a context in terms of survivability for SCADA systems. The changes are on the Base and Environmental metrics. CVI does not change the overall computation process as shown in Figure 7.1. The contextual base score can still be used alone, temporal score depends on contextual score, the environmental score depends on the temporal score and temporal and environmental scores are kept optional.

The way the base metric is computed has been changed, and it is now renamed to *Contextual Base Score*. Another change is that the overall CVI score ranges from 0.0 to 1.0, rather than 0.0 to 10.0 as before. The environmental metric has been changed as well, however the changes are minor. All changes are described below.

Contextual base score The Contextual Base Score introduces two new attributes (*reachability* and *essentiality*) and a constant (ρ), which defines how much of the contextual score should be used by the *reachability* attribute.

Table 7.4: Contextual Base metric attributes

Abbr.	Name	Description
R	Reachability	It is a variation of the closeness centrality network metric [Newman, 2007], which measures the mean geodesic distance to the other nodes of the network. The metric attribute measures how close the other services of the system are from the original service in terms of link connections. It aims to measure how central the service is to the system when compared to the other services. The intuition behind reachability is that services with good reachability are more central to the system, and consequently any malicious attack to those services may have catastrophic consequences. In other words, reachability amplifies the vulnerability attribute in the sense that a high vulnerable service with a high reachability is provides a higher risk to the system. Services that are able to reach other services at shorter path lengths have better reachability.
Es	Essentiality	Measures the type of the service in which the vulnerability has been found. Two possible values: Essential (E) and Non-Essential (NE). The concept of essential and non-essential services are described in Chapters 4 and 5.

The Contextual base score computation is described in Equation 7.4.1 with the *essen-*

tiality computation described in Chapter 5. In the *reachability* (R) equation, s is the current service, j another service in the graph, and d_{sj} the distance between them.

$$\begin{aligned}
CBa &= \max\left(\left(\left(0.6 \times Impact\right) + \left(0.4 \times Exploitability\right) - \rho\right), 0\right) + \left(R \times \rho\right) \\
Impact &= Es \times \left(1 - (C) * (1 - I) * (1 - A)\right) \\
Es &= \begin{cases} 1.041 & when \text{ Essential} \\ 0.520 & when \text{ Non-Essential} \end{cases} \\
Exploitability &= 2 \times AV \times AC \times Au \\
R &= \frac{1}{\sum_j d_{sj}}
\end{aligned} \tag{7.4.1}$$

The constant ρ is used to define how much of the metric should be allowed to the reachability attribute. The $f(Impact)$ function, which was previously used as a constant to cases when $Impact > 0$, has been removed. Instead the reachability attribute is added. The $Impact$ and $Exploitability$ attributes are compared to zero, and are only used if they are greater than zero.

Environmental score The changes to the environmental metric are in the Adjusted Impact (AI) equation. Before, the analyst could decide which weight to choose for CR, IR and AR, now AR is prefixed to the value $High = 0.151$. The idea is to reflect one characteristic of SCADA systems that prioritises Availability in detriment to Integrity and Confidentiality (see Chapter 2). Equation 7.4.2 shows the changes.

$$AI = \min\left(1, 1.041 \times (1 - C \times CR) \times (1 - I \times IR) \times (1 - A \times 0.151)\right) \tag{7.4.2}$$

Another change is related to the scaling of values defined by the CVSS score. The CVI score is scaled down to between 0.0 and 1.0. Scaling down the values is necessary to have the same range of values between $Reachability$, $Essentiality$ and the other attributes.

7.4.2 Computing the contextual vulnerability index

Algorithms 5, and 6 describe how individual CVI scores (per vulnerability) and for all services of the system are computed. Algorithms 5 goes through all services of the system. For each service CVI scores are computed and then summed up (see Algorithm 5), line 9. As CVI is based on CVSS scores, Algorithm 6 is responsible for parsing a vector that represents a vulnerability and computes the CVI score for it. As mentioned earlier CVSS score are expressed as a vector. Once all CVI service scores are computed Algorithm 5 normalises the scores. This process is defined in Equation 7.4.3 and it is described in line 17 of Algorithm 5. The function *computeReach* described in line 5 of Algorithm 5. The *findVul* function (line 6) searches for all vulnerabilities found for a particular service. The search is on the OSVDB database [OSV, 2012], an open source database of vulnerabilities. Function *findEssential* executes Algorithm 4 that finds essential services. And function *computeCVI* represents Algorithm 6.

$$CVI(s) = \frac{CVI(s)}{\sum_{i=1}^{m} CVI(i)} \text{ If } i \neq s \tag{7.4.3}$$

To the context of security, CVI measures the probability of potential risk a service brings to the system. It is also important to highlight that vulnerability is inversely proportional with the resistance of the system, $Resistance = \frac{1}{CVI}$. Systems with services having low ranked CVI scores are more resistant than the ones with services with having high ranked CVI scores.

7.5 Quantifying holistic survivability using a security model

As introduced in Chapter 4, holistic survivability evaluates the capacity of systems to provide their functionalities in cases of undesired events compromising parts of the system. It is quantified by the joint distribution that marginalises over all services of the system where the essential services are on an intended state. In the proposed security model two possible service states are defined: (*Acceptable*) and *Unacceptable*. The intended state is the *acceptable* state. Therefore, Equation 7.5.1 describes the quantification of holistic survivability using this security model, where E represents the essential

Algorithm 5: Computing CVI for services

```
   Input: $G, \rho, \theta$
   Output: $servicesMap$
1  $CVI \leftarrow 0$
2  $services \leftarrow extractNodes(G)$
3  $totalCVI \leftarrow 0$
4  foreach $s$ in $services$ do
5      $R \leftarrow computeReach(s, G)$
6      $slist \leftarrow findVul(s)$
7      $isEssential \leftarrow findEssential(s)$
8      foreach $v$ of $slist$ do
9          $CVI \leftarrow CVI + computeCVI(v, isEssential, R, \rho)$
10     end
11     if $CVI == 0$ then
12         $CVI \leftarrow \theta$
13     end
14     $put(servicesMap, s, CVI)$
15     $totalCVI \leftarrow totalCVI + CVI$
16 end
   // Normalise all CVI scores.  See Equation 7.4.3]
17 foreach $s$ in $services$ do
18     $CVI \leftarrow get(servicesMap, s)$
19     $newCVI \leftarrow \frac{CVI}{totalCVI}$
20     $put(servicesMap, s, newCVI)$
21 end
22 return $servicesMap$
```

services and I the non-essential ones.

$$P(E = acceptable) = \sum_{I} P(E, I) \tag{7.5.1}$$

As in the other models, service states are computed at the service level, and then they are aggregated into a graphical model, which provides a compact representation of the system, to then compute holistic survivability. Markov networks is the graphical model used for representation of the proposed model. A Markov network [Koller and Friedman, 2009] is an undirected graph, where nodes are random variables and edges define the dependencies and relationships between the nodes. Chapter 2 provides a brief overview on Markov networks, for a more complete introduction on Markov networks see [Koller and Friedman, 2009] and [Barber, 2012].

To use Markov networks, services are mapped as nodes, and the relationship between the nodes is defined by transactions carried out between services. As one can notice, the proposed model is not based on Bayesian networks as the model formalism. The reason for that is that services are not directly dependent of their neighbours. The vulnerability of a service is not dependent of the vulnerability of other service, even though by exploiting the vulnerability of a service one may exploit other services vulnerabilities. There is some relationship but not a direct dependency as presented in the performance model introduced in Chapter 6.

As discussed in Chapter 2, a Markov network is defined in terms of *potentials*. In this thesis a potential is a function from $Val(X)$ to $\mathbb{R}^+$. This model does not consider negative potentials. Potentials are used to parametrise the distribution that is represented by the network. The overall joint distribution considers all potentials defined in the network graph. By adapting the definition of Markov networks and factors to our model, we have the following.

A Markov network is defined by Equation 7.5.2, where X is a set of variables, and Z is a normalisation constant, also called *partition function* [Koller and Friedman, 2009]. Then holistic survivability is represented by Equation 7.5.3.

$$P(X) = \frac{1}{Z} \times \prod_i \phi_i(x_i) \tag{7.5.2}$$

$$Z = \sum_{s \in S} \prod_i \phi_i(s_i)$$

$$P(E = acceptable) = \sum_I P(E, I) = \frac{1}{Z} \times \prod_i^m \phi(x_i) \tag{7.5.3}$$

The methodology to quantify holistic survivability using a security model is divided into three parts:

1. Individual service CVI score computation and service state assignments. It is achieved by executing Algorithms 5 and 6.

2. Markov network construction. The network is based on the service graph discussed in Chapter 4. It is generated by looking at the transactions between services in the system. In addition the factors are also generated by looking for maximal cliques in the graph.

3. Computing holistic survivability. Once the model is built one can perform queries to find out the overall survivability (marginal over the essential services), and can also perform some types of diagnosis such as what service is responsible for the degradation on the overall holistic survivability, for instance.

The process of computing CVI scores for services in a system is already described in Section 7.4.2, therefore it will not be described here. Instead the construction of the Markov network and the services CPDs is described. To facilitate that understanding a small exampled is provided.

7.5.1 Markov network construction

To build the Markov network three things are necessary. First, the network structure. The structure is based on the service graph presented in Chapter 4. A system expert has the option to build the service graph manually, without any help from data. This option is reasonable when the system has not been deployed, therefore there is no data or simulation to help. However, in most cases the service graph is built automatically from the data acquired from the system. The network traffic can be processed to build the graph. Section 4.4.3 and Algorithm 1 describe how a service graph can be built from data. Second, the essential services of the system need to be identified. As they will be used to compute individual CVI scores and also to compute the overall system holistic survivability. And third, each potential needs a CPD associated with it that describes its probability of the services being under states *Acceptable* and *Unacceptable*.

Figure 7.2 illustrates a service graph with four nodes (services). This service graph contains two maximal cliques (potentials), which will be used to compute the holistic survivability of this simple system. Based on the service graph the potentials are $\phi_1(A, B, C)$ and $\phi_2(B, C, X)$, and the problem is to compute $\phi_1(A, B, C)\times\phi_2(B, C, X)$, therefore assuming the C is the essential service, holistic survivability is defined as: $P(C = a) = \frac{1}{Z} \times \phi_1(A, B, C) \times \phi_2(B, C, X)$, where $Z = \sum_{A,B,C,X} \phi_1(A, B, C) \times \phi_2(B, C, X)$.

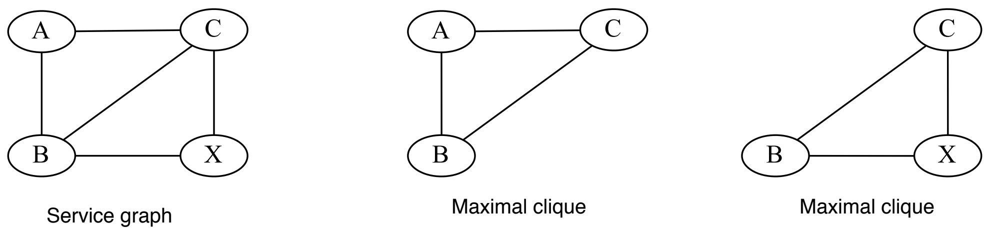


Figure 7.2: Service graph - Two maximal cliques

Lets say that each service has the following CVI as described in Table 7.5. Once that the services CVI are provided the next step is to transform these values into the possible service states and then generate the CPDs for each potential. Based on the service state model presented in Section 7.3, and the marginal for service C illustrated in Table 7.6, the holistic survivability is defined as: $0.011738 + 0.066515 + 0.063906 + 0.362136 + 0.010835 + 0.061398 + 0.058991 + 0.334280 =$ **0.9698**.

Table 7.5: Services CVI

$A = 0.85$	$B = 0.70$
$C = 0.15$	$X = 0.48$

Table 7.6: Marginal for service $C = a$

$A = a$	$B = a$	x=$x = a$	**0.011738**
$A = u$	$B = a$	x=$x = a$	**0.066515**
$A = a$	$B = u$	x=$x = a$	**0.063906**
$A = u$	$B = u$	x=$x = a$	**0.362136**
$A = a$	$B = a$	x=$x = u$	**0.010835**
$A = u$	$B = a$	x=$x = u$	**0.061398**
$A = a$	$B = u$	x=$x = u$	**0.058991**
$A = u$	$B = u$	x=$x = u$	**0.334280**

Figure 7.3 illustrates the holistic survivability of the system when there is evidences that A, B or X are *unacceptable*. As one can see from Figure 7.3, the holistic survivability for this example can vary substantially, and even though X is not an essential service its degradation affects the holistic survivability immensely. As illustrated in Figure 7.3,

the overall survivability goes from 0.97 to 0.56, if it is revealed that service X is an *unacceptable* state.

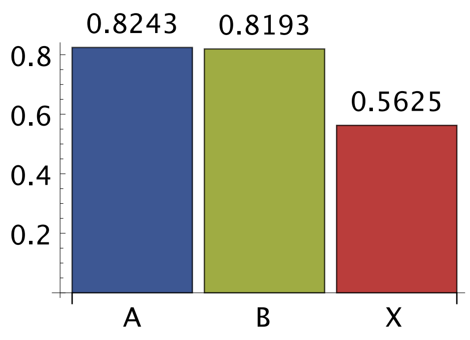


Figure 7.3: Holistic survivability for different evidences

7.6 Learning from data

The process of learning from data in this model only applies for the service graph. In other words, the only element learnt from data is the network structure of the graphical model. This is due to the fact the the values for the CVI metric are obtained by searching for vulnerabilities on public domain vulnerability databases. Once those values are gathered they are processed and converted into probability distributions following the rules defined in Sections 7.3 and 7.5.1. This process is much simpler than the one described in Chapter 6, for instance. Therefore the steps provided by Chapter 4 will not be provided here.

7.7 Evaluation

The evaluation consists of using the SCADA system simulation introduced in Chapter 5 to show how the proposed model applies for close to real live systems. Because the system being used is not a real system, there is no much information about vendors, and vulnerabilities associated with products. The experiments demonstrated here are based on three scenarios:

Unfortunately, there is no known distributions describing vulnerabilities on deployed SCADA systems. Figure 1.1 illustrates vulnerabilities per category affecting SCADA systems, however it is not a distribution for particular systems. Therefore, this evaluation will use the scenarios described in Table 7.7, where **Es** represents the number of essential services, CVI represents the number of CVI simulated and **Default CVI** represents the number of services using the default CVI score. The default score is $\theta = 0.15$. The simulated system utilised here was introduced in Chapter 5.

Table 7.7: Scenarios

Scenario	# Es	# CVI	Essential services
1	1	1	**css02**
2	3	3	**css01, css02, css03**
3	5	5	**css02, ch01, css03, chs01, crs01**
4	10	2	**f00p01, f01p02, f00p02, f01p03, css01, crs01, crs02, ch01, chs01, f02p01**

For each of the scenarios described below a holistic survivability quantification was computed. Figure 7.4 shows the results for these four simulations. The goal of this simulation is to show the the model works and how it can be generated automatically.

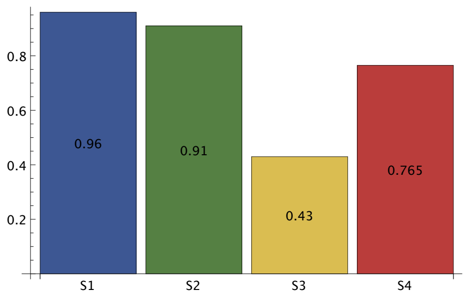


Figure 7.4: Holistic survivability based on scenarios

7.8 Conclusion

The security model presented here confirms that holistic survivability is flexible enough to be used as a framework for models based on different metrics and computational aspects. The presented model follows the requirements of holistic survivability, however it introduces a new security metric and also uses another graphical model for representation and quantification. The security metric (CVI) extends CVSS by adding contextual information pertinent to survivability such as essential services and availability importance. Even though the graphical model used is different from Bayesian networks it is based on the same foundations such as potentials and conditional dependences. And finally, as the restrictions with holistic survivability the presented model assume that service graphs are connected. To a further analyse of the model more discussion on the CVI scores computation is necessary. And if they were generated for the essential services or not, is required.

Algorithm 6: CVI algorithm

```
Input: $V$,$isEssential$,$R$,$\rho$
Output: $CVI$
1  $values \leftarrow stringSplit(stringSplit(V, "/"), " : ")$
2  $C, I, A, AV, AC, Au, E, RL, RC \leftarrow computeAttributes(values)$
3  if $isEssential$ then
4      $Es \leftarrow 1.041$
5  else
6      $Es \leftarrow 0.5205$
7  end
8  $Impact \leftarrow Es \times (1 - (1 - C) \times (1 - I) \times (1 - A))$
9  $Exploitability \leftarrow 2 \times AV \times AC \times Au$
   // CBa is the contextual base, Te is the temporal and
   En the environment score.
10 $CBa \leftarrow round(max(((.6 \times Impact) + (0.4 \times Exploitability) - \rho) + (R \times \rho), 0)$
11 if $E > 0, RL > 0, RC > 0$ then
12     $Te \leftarrow CBa \times E \times RC \times RL$
13 else
14     $Te \leftarrow CBa$
15 end
16 if $Te <> CBa$ then
17     $CR, IR, AR, CDP, TD \leftarrow computeAttributes(values)$
18     if $CR < 0, IR < 0, AR < 0, CDP < 0, TD < 0$ then
19         $En \leftarrow -1$
20     else
21         $AI \leftarrow min(1, Es \times (1 - (1 - C * CR) \times (1 - I \times IR) \times (1 - A \times 1.51)))$
22         $Exploitability \leftarrow 2 \times AV \times AC \times Au$
23         $CBa \leftarrow max(((.6 \times AI) + (0.4 \times Exploitability) - \rho) + (R \times rho), 0)$
24         $Te \leftarrow CBa \times E \times Rc \times RL$
25         $En \leftarrow Te + (1 - Te) \times CDP) \times TD$
26         $CVI \leftarrow En$
27     end
28 else
29     $CVI \leftarrow CBa$
30 end
31 return $CVI$
```

Chapter 8

Conclusion

This thesis has introduced a new concept of survivability. Holistic survivability provides a different approach on evaluating the survivability of SCADA systems. It differentiates from current survivability models in four ways: First, when evaluating systems all services are considered. Second, because essential services are a integral part of the system, holistic survivability supports automatic recognition of essential services and the concept of multiple essential services, for systems that have multiple. Third, data is an important part of holistic survivability. It recognises that due to increasing amount of data being produced, and acquired, it is important to have a model that supports data in order to provide quantifications. In other words, analytical models are important, but they should not be the only way of quantifying survivability. Data support is on holistic survivability core. Both models presented in this thesis have shown how data can help on the process on model generation and evaluation.

And Fourth, security. Security is a core component of holistic survivability. Since its conceptual definitions to its implementation. There are security models targeting the survivability of systems, however they are not tailored for survivability, they use metrics that not consider survivability unique characteristics such as essential services.

In the other hand, holistic survivability supports and promotes security as a key aspect since its inception. After all the main motivation for holistic survivability is to provide a mechanism that can cope with malicious attacks. More specifically, rather than trying to stop and detect every attack such effort could be better used on assessing the impact of attacks and therefore trying to minimise them.

Another key contribution of this thesis is the simulator of SCADA systems. As security issues affecting SCADA systems become mainstream the need to have systems where one can try and test different scenarios in order to provide better solutions for current systems is immense. In addition to that, the lack of network traffic datasets on SCADA systems make security research on such systems harder. Therefore, a simulator that can provide a close to real simulations where data can be generated and consequently studied is very much appreciated.

Even though, throughout the thesis SCADA systems were used as examples and main targets of holistic survivability we believe that the model can be easily adapted to other information systems. Holistic survivability is a step on the direction of having information systems under automatic assessment providing predictions to systems users. However, holistic survivability is not without its shortcomings. The model contains some limitations that should be fixed on future work. Holistic survivability was demonstrated to work with two types of models, one based on performance, and another one based one security. As one may known, survivability is underpinning by four key attributes: Recognition, Resistance, Recovery and Adaptation. We believe that recovery and adaptation could be merged into one attribute, however adaptation should be supported. In Chapter 4, a brief discussion about adaptation was introduced, but no concrete models that consider adaptation were presented. Evaluation of adaptation is not an easy task. For instance, how one can measure adaptation, Is measuring a change on a metric a valid reasonable of adaptation? How good is this in terms of adaptation? Lets say a particular metric moved from state one to two, Is this a good adaptation? Or lets say there is a hypothetical system that has its services working under normal conditions all the time, nothing changes and the system seems to be perfect. Is this system performing bad on adaptation? What is the ground zero for adaptation? From where one can starting measuring it in order to have something that really represents improvements and downgrades? All of these are legitimate questions that this thesis does not provide an answer, and we see them as part of our future work.

A aspect related to data acquisition that was not addressed in this thesis is real-time data acquisition, also known as stream computing. Such technology allows systems to acquire and process data on demand. Data is only stored after it has been processed and only if it is relevant for future analysis. During the investigations on processing acquired data in this thesis was assumed that the data was already stored on some type of database, and then the models only has to read and process it. With stream processing the model would be built in real-time and on demand. In principle holistic survivability would support such mechanism, but only if all nodes that are part of the system were known before hand, and if the inference process only started after connected graphs were created. This is due to the fact that holistic survivability does not support disconnected graphs. It is a limitation of the model that we aim to solve on future work.

We believe that holistic survivability is perfect for the current scenario. Increasing amount of data being produced and acquired, new and improved algorithms to mining data, large sensors adoption, and a problem that has not an easy fix - security. All these conditions make holistic survivability ideal for taking on current challenges to provide a better answer when comparing with the state of the art in the field.

It is really hard to compare holistic survivability with other models, as it uses unique techniques that were not implemented in this domain. For example, there is fair amount of work in the literature on algorithms, systems, and applications using machine learning techniques to try to fix issues such as security, and so forth. Definitely, holistic survivability is not the first. However, holistic survivability is the first to bring these techniques to the domain of survivability on networked systems. By providing a framework in which other models can be built on top of it makes holistic survivability relevant.

Another aspect that should be investigated is the composition of different holistic survivability analyses. For instance, how the composite of security and performance could be created to analyse survivability? It is not a trivial problem as service state model may be different and the service dependencies also could have different characteristics. For instance, as presented in this thesis the performance model defines three service states and a directional dependence between services. The security model defines two service states and not directional dependencies. How to merge them? A possible solution is to merge the states to therefore create an unique service state shared across the models. And for the service graph, a hybrid model has to be used. There is some work in the literature of hybrid graphical models that contain direct and non-direct edges. This is definitely a good point of investigation for future work.

And finally, a scenario that was not investigated because it was not part of the main goal of this thesis is to incorporate holistic survivability as an architectural component of SCADA and other computer systems. The idea was to create a holistic survivability layer responsible for monitoring and inferring scenarios and possible issues with the system in real-time. In addition, combined with an orchestration layer, it would be possible to have really adaptive computing systems.

Bibliography

Open source vulnerability database, June 2012. URL http://osvdb.org.

The repository of security incidents, June 2012. URL http://www.securityincidents.net/index.php/products/indepth/risi_annual_report/.

Digital bond - securing industrial control systems, June 2012. URL http://www.digitalbond.com.

National SCADA Test Bed, Aug 2012. URL http://energy.sandia.gov/?page_id=859.

M. Al-Kuwaiti, N. Kyriakopoulos, and S. Hussein. A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability. *IEEE Communications Surveys & Tutorials*, 11(2):106 – 124, 2009.

M. Anderberg. *Cluster analysis for applications*. Probability and mathematical statistics. Academic Press, New York, NY, USA, 1973.

R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Publishing, Inc., Cambridge, UK, 2nd edition, Jan 2010.

Z. Anwar and R. Campbell. Automated assessment of compliance with security best practices. In M. Papa and S. Shenoi, editors, *Critical Infrastructure Protection II*, volume 290 of *IFIP International Federation for Information Processing*, chapter 13, pages 173–187. Springer Boston, 2009.

A. Avizienis, J. Laprie, and B. Randell. Fundamental Concepts of Dependability. Technical report, University of Newcastle, Jan. 2001.

A. Avizienis, J. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11 – 33, Jan 2004a.

A. Avizienis, J.-C. Laprie, and B. Randell. Dependability and Its Threats: A Taxonomy. In R. Jacquart, editor, *Proc. IFIP 18th World Computer Congress: Building the Information Society*, pages 91–120, Toulouse, France, 2004b. Kluwer Academic Publishers.

D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, Cambridge, UK, 2012.

K. Barnes and B. Johnson. Introduction to SCADA Protection and Vulnerabilities. Technical Report INEEL/EXT-04-01710, Idaho National Engineering and Environmental Laboratory, January 2004.

C. Bector and S. Chandra. *Fuzzy Mathematical Programming and Fuzzy Matrix Games*. 2005.

D. Bergman, D. Jin, D. Nicol, and T. Yardley. The virtual power system testbed and inter-testbed integration. In *Proceedings of the 2nd Workshop on Cyber Security Experimentation and Test*, page 5, Aug 2009.

D. Bing-Lin, W. Xue-Guang, and Z. Shi-Yong. Research on Survivability of Networked Information System. In *2009 International Conference on Signal Processing Systems*, pages 56–60, 2009.

G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2nd edition, 1998.

S. Bologna, C. Balducelli, G. Dipoppa, and G. Vicoli. Dependability and survivability of large complex critical infrastructures. In V. Palade, R. Howlett, and L. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 2773 of *Lecture Notes in Computer Science*, pages 342–353. Springer Berlin / Heidelberg, Jan 2003.

S. P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, Jan. 2005.

T. Bowen, M. Segal, and R. Sekar. On preventing intrusions by process behavior monitoring. In *8th USENIX Security Symposium*, Washington, DC, 1999.

R. Browne, J. Valente, and S. Hariri. An advanced middleware architecture for secure and survivable mobile C4I systems. In *Proceedings of IEEE Military Communications Conference*, pages 506–513, 1999.

E. Byres and D. Leversage. Security incidents and trends in SCADA and process industries. *The Industrial Ethernet Book*, 2007.

E. Byres, M. Franz, and D. Miller. The use of attack trees in assessing vulnerabilities in scada systems. In *International Infrastructure Survivability Workshop*, Jan 2004.

R. Carlson. Sandia scada program - high-security scada ldrd final report. Technical Report SAND2002-0729, Sandia National Laboratories, April 2002.

W. Chan. *Performance analysis of telecommunications and local area networks*. Kluwer Academic Publishers, Norwell, MA, USA, 2000.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, July 2009.

M. Cheminod, I. C. Bertolotti, L. Durante, P. Maggi, D. Pozza, R. Sisto, and A. Valenzano. Detecting Chains of Vulnerabilities in Industrial Networks. *IEEE Transactions on Industrial Informatics*, 5(2):181–193, 2009.

L. Chen, Q. Wang, W. Xu, and L. Zhang. Evaluating the Survivability of SOA Systems Based on HMM. In *Proceedings of the 2010 IEEE International Conference on Web Services*, pages 673–675. IEEE Computer Society, July 2010.

R. Cheung, M. Demokan, and K. Chow. Modeling and simulation using network software to evaluate scada system configuration. In *Advances in Power System Control, Operation and Management. APSCOM-97. Fourth International Conference on (Conf. Publ. No. 450)*, volume 2, pages 609–613, Nov 1997. doi: 10.1049/cp:19971904.

S. Cheung, B. Dutertre, M. Fong, and U. Lindqvist. Using model-based intrusion detection for scada networks. In D. Peterson, editor, *Proceedings of SCADA Security Scientific Symposium*, number 1, page 12. Digital Bond Press, Jan 2007.

H. Christiansson and E. Luiijf. Creating a european scada security testbed. In E. Goetz and S. Shenoi, editors, *Critical Infrastructure Protection*, volume 253 of *IFIP International Federation for Information Processing*, pages 237–247. Springer Boston, 2007.

G. Clarke and D. Reynders. *Practical Modern SCADA Protocols- DNP3, 60870.5 and Related Systems*. Elsevier, Burlington, MA, USA, 2006.

G. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

M. Coutinho, G. Lambert-Torres, and L. B. da Silva. Improving a methodology to extract rules to identify attacks in power system critical infrastructure: New results. *Transmission and Distribution Conference and Exposition. IEEE PES*, 1:1–6, Jan 2008.

J. Crain. DNP3 implementation, May 2011. URL http://code.google.com/p/dnp3/.

Y. Dai, Y. Pan, and R. Raje. *Advanced parallel and distributed computing: evaluation, improvement and practice*. Nova Science Publishers, New York, USA, 2007.

C. Davis, J. Tate, H. Okhravi, C. Grier, T. Overbye, and D. Nicol. Scada cyber security testbed development. In *Power Symposium. NAPS 2006. 38th North American*, pages 483–488, Sep 2006.

D. Dittrich. The dos project's 'trinoo' distributed denial of service attack tool. Technical report, University of Washington, Jan 1999.

S. East, J. Butts, M. Papa, and S. Shenoi. A taxonomy of attacks on the dnp3 protocol. In C. Palmer and S. Shenoi, editors, *Critical Infrastructure Protection III*, volume 311 of *IFIP Advances in Information and Communication Technology*, pages 67–81. Springer Boston, 2009.

M. El-Sharkawi. Vulnerability assessment and control of power system. In *IEEE/PES Transmission and Distribution Conference and Exhibition*, pages 656–660. IEEE Power Engineering Society, Jan 2002.

R. Ellison and A. Moore. Architectural refinement for the design of survivable systems. Technical Report CMU/SEI-2001-TN-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Jan 2001.

R. Ellison and C. Woody. Survivability Analysis Framework. Technical Report CMU/SEI-2010-TN-013, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, June 2010.

R. Ellison, D. Fisher, R. Linger, and H. Lipson. Survivable network systems: An emerging discipline. Technical report, Software Engineering Institute - Carnegie Mellon University, Jan 1997.

R. Ellison, D. Fisher, R. Linger, and H. Lipson. Survivability: Protecting your critical systems. *IEEE Internet Computing*, 3(6):55–63, Jan 1999.

N. Falliere, L. O. Murchu, and E. Chien. W32.stuxnet dossier. Technical Report 1.4, Symantec, February 2011.

J. P. Farwell and R. Rohozinski. Stuxnet and the Future of Cyber War. *Survival*, 53(1): 23–40, Feb. 2011.

J. Fernandez and A. Fernandez. Scada systems: vulnerabilities and remediation. *Journal of Computing Sciences in Colleges*, 20(4):160–168, Jan 2005.

D. A. Fisher and H. F. Lipson. Emergent Algorithms - A New Method for Enhancing Survivability in Unbounded Systems. In *HICSS '99: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, Jan. 1999.

T. Fleury, H. Khurana, and V. Welch. Towards a taxonomy of attacks against energy control systems. *Critical Infrastructure Protection II*, pages 71–85, 2009.

J. C. Foster. *Writing security tools and exploits*. Syngress Media Inc., New Jersey, USA, 2006.

I. Fovino and Carcano. A Secure and Survivable Architecture for SCADA Systems. In *Second International Conference on Dependability*, pages 34–39, 2009.

I. Fovino, M. Masera, and R. Leszczyna. Ict security assessment of a power plant, a case study. *proceedings of the Second Annual IFIP Working Group*, 11.

I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta. An experimental investigation of malware attacks on scada systems. *International Journal of Critical Infrastructure Protection*, 2(4):139–145, 2009.

H. Frank. Survivability analysis of command and control communications networks – part i, ii. *IEEE Transactions on Communications*, 22(5):589–605, Jan 1974.

H. Frank and I. Frisch. Analysis and design of survivable networks. *IEEE Transactions on Communications*, 18(5):501–519, 1970.

L. Freeman. Centrality in social networks conceptual clarification. *Social Networks*, 1 (3):215–239, Jan. 1979.

S. L. Garfinkel. The cybersecurity risk. *Commun. ACM*, 55(6):29–32, June 2012.

D. Gaushell and H. Darlington. Supervisory control and data acquisition. *Proceedings of the IEEE*, 75(12):1645–1658, Jan 1987.

B. Genge, I. Nai Fovino, C. Siaterlis, and M. Masera. Analyzing Cyber-Physical Attacks on Networked Industrial Control Systems. *Critical Infrastructure Protection V*, pages 167–183, 2011.

R. Ghosh and K. Lerman. Predicting influential users in online social networks. In *Proceedings of The Fourth KDD Workshop on Social Network Analysis*, page 10. ACM SIGKDD, ACM, 2010.

H. Ginzburg and E. Reis. Consequences of the nuclear power plant accident at chernobyl. *Public Health Reports*, Jan 1991.

S. Gokhale. Survivability evaluation and modelling. In Y. Dai, Y. Pan, and R. Raje, editors, *Advanced Parallel And Distributed Computing: Evaluation, Improvement And Practice*, volume 2, chapter 6, pages 75–97. Nova Science Publishers, Inc., New York, USA, Jan 2006.

K. Goseva-Popstojanova, F. Wang, R. Wang, F. Gong, K. Vaidyanathan, K. Trivedi, and B. Muthusamy. Characterizing intrusion tolerant systems using a state transition model. *Proceedings of DARPA Information Survivability Conference & Exposition II*, 2:211 – 221, Jun 2001.

D. U. Group. DNP3 Specification. Technical report, DNP Users Group, May 2011.

R. Gula. Correlating ids alerts with vulnerability information. Technical report, Tenable Network Security, Jan 2007.

A. Hahn, B. Kregel, M. Govindarasu, J. Fitzpatrick, R. Adnan, S. Sridhar, and M. Higdon. Development of the PowerCyber SCADA security testbed. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, pages 1–4. ACM, 2010.

M. Hatch, E. Ron, A. Bouville, and L. Zablotska. The chernobyl disaster: cancer following the accident at the chernobyl nuclear power plant. *Epidemiologic Reviews*, 27(1):56–66, Jan 2005.

M. Hawkins. *Identification of Outliers*. Chapman, Hall, London, UK, 1980.

P. Heegaard and K. Trivedi. Survivability quantification of communication services. In *IEEE International Conference on Dependable Systems and Networks*, pages 462 – 471, May 2008. doi: 10.1109/DSN.2008.4630117.

P. E. Heegaard and K. S. Trivedi. Network survivability modeling. *Computer Networks*, 53(8):1215–1234, June 2009.

D. Heimbigner, A. Wolf, A. Carzaniga, J. Knight, J. Hill, P. Devanbu, and M. Gertz. The willow architecture: Comprehensive survivability for large-scale distributed applications. In *Intrusion Tolerance Workshop - The International Conference on Dependable Systems and Networks*, Washington DC, USA, 2002.

F. Hillier, G. Lieberman, and G. Liberman. *Introduction to operations research*. McGraw-Hill New York, 7th edition, 1990.

M. Hiltunen, R. Schlichting, C. Ugarte, and G. Wong. Survivability through customization and adaptability: The cactus approach. *Proceedings of DARPA Information Survivability Conference and Exposition*, 1:294 – 307, Dec 1999.

M. A. Hiltunen, R. D. Schlichting, and C. A. Ugarte. Enhancing survivability of security services using redundancy. In *in Proceedings of The International Conference on Dependable Systems and Networks*, pages 173–182, 2001.

Å. Holmgren. Vulnerability analysis of electric power delivery networks. Master's thesis, KTH, Superseded Departments, Land and Water Resources Engineering, Jan 2004.

P. Huitsing, R. Chandia, M. Papa, and S. Shenoi. Attack taxonomies for the modbus protocols. *International Journal of Critical Infrastructure Protection*, 1:37–44, Dec 2008.

M. Hypponen. Why antivirus companies like mine failed to catch flame and stuxnet, June 2012. URL http://www.wired.com/threatlevel/2012/06/internet-security-fail/.

V. Igure, S. Laughter, and R. Williams. Security issues in scada networks. *Computers & Security*, 25(7):498–506, 2006.

INET. INET Framework, Sep 2012. URL http://inet.omnetpp.org.

S. Jajodia, P. Liu, and P. Ammann. A fault tolerance approach to survivability. *Ammann P E. Computer Security*, Jan. 1999.

F. V. Jensen. *An Introduction to Bayesian networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

S. Jha and J. Wing. Survivability analysis of networked systems. *Proceedings of the 23rd international conference on software engineering*, 1:307–317, Jan 2001.

R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere. The Architecture Tradeoff Analysis Method. In *4th IEEE International Conference on Engineering of Complex Computer Systems*, pages 68–78. IEEE Comput. Soc, 1998.

J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1): 41–50, Jan. 2003.

K. Kihlstrom and P. Narasimhan. The starfish system: Providing intrusion detection and intrusion tolerance for middleware systems. *IEEE Workshop on Object-oriented Realtime Dependable Systems*, Jan 2003. URL http://doi.ieeecomputersociety.org/10.1109/WORDS.2003.1218083.

R. Kinderman and K. L. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, Providence, Rhode Island, USA, 1980.

E. D. Knapp. *Industrial Network Security*. Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems. Syngress, Waltham, MA, USA, Aug. 2011.

J. Knight and E. Strunk. Achieving critical system survivability through software architectures. In R. Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems II*, volume 3069 of *Lecture Notes in Computer Science*, pages 51—78. Springer-Verlag Berlin Heidelberg, 2004.

J. Knight and K. Sullivan. On the definition of survivability. Technical Report CS-TR-33-00, University of Virginia, Department of Computer Science, 2000.

J. C. Knight, R. W. Lubinsky, J. McHugh, and K. J. Sullivan. Architectural approaches to information survivability. Technical report, Charlottesville, VA, USA, 1997.

T. Kobayashi, A. B. Jr, J. Medeiros, J. F. Filho, A. B. Jr, and P. Pires. Analysis of malicious traffic in modbus/tcp communications. *Critical Information Infrastructure Security*, 5508:200–210, 2009.

O. Kolesnikov and W. Lee. Advanced polymorphic worms: Evading ids by blending in with normal traffic. Technical report, College of Computing, Georgia Tech., Jan 2006.

D. Koller and N. Friedman. *Probabilistic Graphical Models Principles and Techniques*. MIT Press, Cambridge, Massachusetts, USA, 2009.

K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. CRC Press LLC, London, UK, Dec. 2004.

J. Körner. Fredman-kolmo's bounds and information theory. *SIAM J. Algebraic Discrete Methods*, 7(4):560–570, Oct. 1986.

A. Krings and A. Azadmanesh. A graph based model for survivability applications. *European Journal of Operational Research*, 164(3):680–689, Jan 2005.

C. Krugel and T. Toth. Applying mobile agent technology to intrusion detection. In *ICSE Workshop on Software Engineering and Mobility*, pages 1841–2002, Jan 2001.

R. Krutz. *Securing SCADA systems*. Wiley Publishing, Inc., Indianapolis, IN, USA, 2006.

A. Landherr, B. Friedl, and J. Heidemann. A critical review of centrality measures in social networks. *Business & Information Systems Engineering*, 2(6):371–385, 2010.

S. Liew and K. Lu. A framework for network survivability characterization. In *IEEE International Conference on Communications, 1992*, pages 405–410. IEEE, 1992.

R. Linger and A. Moore. Foundations for survivable system development: service traces, intrusion traces, and evaluation models. Technical Report CMU/SEI-2001-TR-029, Software Engineering Institute, Hanscom, MA, USA, Oct 2001.

Y. Liu. *Survivability of networked systems*. PhD thesis, Duke University, 2010.

Y. Liu and H. Man. Network vulnerability assessment using Bayesian networks. *Proceedings of SPIE*, 5812:61–71, 2005.

Y. Liu and K. Trivedi. A general framework for network survivability quantification. *MMB & PGTS 2004: 12th GI/ITG Conference on Measuring*, Jan 2004.

Y. Liu and K. Trivedi. Survivability quantification: The analytical modeling approach. *International Journal of Performability Engineering*, 2(1):29–44, Jan 2006.

Y. Liu, V. Mendiratta, and K. Trivedi. Survivability analysis of telephone access network. *15th International Symposium on Software Reliability Engineering*, 1:367–377, 2004.

M. Long, C. Wu, and J. Hung. Denial of service attacks on network-based control systems: impact and mitigation. *IEEE Transactions on Industrial Informatics*, 1(2): 85–96, 2005.

S. Lüders. Control systems under attack? Technical Report CERN-OPEN-2005-025, CERN, Geneva, Jan 2005.

A. Mahmood, C. Leckie, J. Hu, Z. Tari, and M. Atiquzzaman. Network Traffic Analysis and SCADA Security. *Handbook of Information and Communication Security*, 1: 383–405, 2009.

M. Majdalawieh, F. Parisi-Presicce, and D. Wijesekera. Dnpsec: Distributed network protocol version 3 (dnp3) security framework. *Advances in Computer, Information, and Systems Sciences, and Engineering*, 1:227–234, Jan 2006.

M. Masera and I. Fovino. A service-oriented approach for assessing infrastructure security. In *IFIP International Federation for Information Processing*, volume 253, 2010.

M. Masera, I. Fovino, and R. Leszczyna. Security Assessment Of A Turbo-Gas Power Plant. *Critical Infrastructure Protection II*, 290:31–40, Jan 2008.

S. Mauw and M. Oostdijk. Foundations of attack trees. In *International Conference on Information Security and Cryptology*, pages 186–198. Springer, Jan 2006.

C. P. Mayer and T. Gamer. Integrating real world applications into OMNeT++. Technical Report TM-2008-2, Institute of Telematics, Universität Karlsruhe (TH), Feb. 2008.

N. Mead, R. Ellison, R. Linger, T. Longstaff, and J. McHugh. Survivable network analysis method. *Technical Report*, (2):35, 2000.

N. R. Mead, R. C. Linger, J. McHugh, and H. F. Lipson. Annals of Software Engineering, Volume 11, Number 1 - SpringerLink. *Annals of Software Engineering*, 11(1): 45–78, 2001.

P. Mell, K. Kent, and S. Romanosky. The common vulnerability scoring system (cvss) and its applicability to federal agency systems. Technical report, NIST, Jan 2007.

D. Menasce, L. Dowdy, and V. Almeida. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA, Jan 2004.

M. Merideth and P. Narasimhan. Metrics for the evaluation of proactive and reactive survivability. In *International Conference on Dependable Systems and Networks*, page 2, San Francisco, CA, USA., June 2003.

J. Meyer. On evaluating the performability of degradable computing systems. *IEEE Transactions on computers*, Jan 1980.

F. Milano. An open source power system analysis toolbox. *IEEE Transactions on Power Systems*, 20(3):1199–1206, 2005.

K. Mitnick. *Ghost in the Wires: My Adventures as the World's Most Wanted Hacker*. Little, Brown and Company, New York, USA, first edition, August 2011.

S. Moitra and S. Konda. A simulation model for managing survivability of networked information systems. Technical Report CMU/SEI-2000-TR-020, Carnegie Mellon - Software Engineering Institute, Jan 2000.

S. Moitra and S. Konda. An empirical investigation of network attacks on computer systems. *Computers & Security*, 23:43–51, Jan 2004.

A. Moore, R. Ellison, and R. Linger. Attack modeling for information security and survivability. Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Jan 2001.

K. Munro. Scada – a critical situation. *Network Security*, 2008(1):4–6, Jan 2008. doi: 10.1016/S1353-4858(08)70005-9.

K. Murphy. *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, UC Berkeley, Jan 2002.

P. Narasimhan, K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. Providing support for survivable corba applications with the immune system. In *International Conference on Distributed Computing Systems*, pages 5–7, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

P. G. Neuman. Practical architectures for survivable systems and networks. Technical report, SRI International, Menlo Park, CA, Jun 2000.

M. Newman. The mathematics of networks. *The New Palgrave Encyclopedia of Economics*, pages 1–12, 2007.

D. Nicol, W. Sanders, and K. Trivedi. Model-based evaluation: From dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1(1):48 – 65, Jan 2004.

L. O. M. Nicolas Falliere and E. Chien. W32.stuxnet dossier. Whitepaper, Symantec, Jan 2011. URL http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

ns 3 maintainers. NS3 Simulator, May 2011. URL http://www.nsnam.org/.

P. Pal, R. Schantz, M. Atighetchi, J. Loyall, and F. Webber. Whats next in intrusion tolerance? In *3rd Recent Advances in Intrusion Tolerance Workshop*. IEEE/IFIP Distributed Systems and Networks Conference, 2009.

E. Pardoux. *Markov Process and Applications: Algorithms, Networks, Genome and Finance*. John Wiley & Sons Ltd, Chichester, SXW, UK, 2008.

R. Parks and E. Rogers. Vulnerability assessment for critical infrastructure control systems. *IEEE Security & Privacy*, 6(6):37–43, Nov 2008.

S. Patel and P. Sanyal. Securing scada systems. *Information Management & Computer Security*, 16(4):398–414, Jan 2008.

J. Perl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, USA., 1988.

Y. Qian, D. Tipper, P. Krishnamurthy, and J. Joshi. Information assurance: Dependability and security in networked systems. *Book*, pages 1–535, Jul 2008.

C. Queiroz. Scada vulnerabilities according to osvdb, June 2012. URL http://github.com/caxqueiroz/.

C. Queiroz, A. Mahmood, J. Hu, Z. Tari, and X. Yu. Building a scada security testbed. In *Third International Conference on Network and System Security*, pages 357–364, Oct 2009.

S. Raimbault. Libmodbus, Aug 2010. URL http://www.libmodbus.org/.

P. Reinecke and K. Wolter. Adaptivity metric and performance for restart strategies in web services reliable messaging. In *Proceedings of the 7th international workshop on Software and performance*. ACM Request Permissions, June 2008.

R. Reussner and V. Firus. Basic and dependent metrics. In I. Eusgeld, F. Freiling, and R. Reussner, editors, *Dependability Metrics*, volume 4909 of *Lecture Notes in Computer Science*, pages 37–38. Springer Berlin / Heidelberg, 2008.

S. M. Ross. *Introduction to Probability Models*. Academic Press, Inc., Orlando, FL, USA, 2007.

W. H. Sanders. Stochastic methods for dependability, performability, and security evaluation. In J. Cortadella and W. Reisig, editors, *Applications and Theory of Petri Nets 2004*, volume 3099 of *Lecture Notes in Computer Science*, pages 97–97. Springer Berlin Heidelberg, 2004.

D. E. Sanger. *Confront and Conceal - Obama's Secret Wars and Surprising Use of American Power*. Random House Inc., New York, USA, June 2012.

B. Schneier. Attack trees. *Dr. Dobb's Journal*, Jan 1999. URL http://www.cs.utk.edu/~dunigan/cs594-cns00/attacktrees.pdf.

F. Sheldon, T. Potok, M. Langston, A. Krings, and P. Oman. Autonomic Approach to Survivable Cyber-Secure Infrastructures. *IEEE Int. Conf. on Web Services (ICWS 2004), California, USA*, 2004.

J. Shetty and J. Adibi. Discovering important nodes through graph entropy the case of enron email database. In *Proceedings of the 3rd international workshop on Link discovery*, LinkKDD '05, pages 74–81, New York, NY, USA, 2005. ACM.

E. H. Simpson. Measurement of Diversity. *Nature*, 163(4148):688–688, Apr. 1949.

J. Slay and M. Miller. Lessons learned from the maroochy water breach. In E. Goetz and S. Shenoi, editors, *Critical Infrastructure Protection*, volume 253 of *IFIP International Federation for Information Processing*, pages 73–82. Springer Boston, 2007.

D. J. Spiegelhalter and S. L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579–605, Aug. 1990.

J. Stamp, P. Campbell, J. DePoy, J. Dillinger, and W. Young. Sustainable security for infrastructure scada. Technical Report SAND2003-4670C, Sandia National Laboratories, Jan 2003.

T. Stephanou. Assessing and exploiting the internal security of an organization. *SANS Institute*, 2001.

K. Sullivan, J. Knight, X. Du, and S. Geist. Information survivability control systems. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 184–192, 1999.

Sun Microsystems. Secure enterprise computing with the solaris 8 operating environment. Technical report, Sun Microsystems, 2000. URL http://www.sun.com/software/whitepapers/wp-s8security/wp-s8security.pdf.

N. Svendsen and S. Wolthusen. Modeling and detecting anomalies in scada systems. In M. Papa and S. Shenoi, editors, *Critical Infrastructure Protection II*, volume 290 of *IFIP International Federation for Information Processing*, pages 101–113. Springer Boston, 2009.

A. Swales. Open modbus/tcp specification, Jan 1999. URL http://www.rtaautomation.com/modbustcp/files/Open_ModbusTCP_Standard.pdf.

A. T1A1.2. Technical report on enhanced network survivability performance. Technical Report 68, Working Group on Network Survivability Performance, Feb 2001.

I. C. S. C. E. R. Team. Ics-cert incident summary report. Technical report, Industrial Control Systems Cyber Emergency Response Team - ICS-CERT, 2012.

C. Ten, C. Liu, and M. Govindarasu. Vulnerability assessment of cybersecurity for scada systems using attack trees. *Power Engineering Society General Meeting*, Jan 2007.

C.-W. Ten, C.-C. Liu, and G. Manimaran. Vulnerability Assessment of Cybersecurity for SCADA Systems. *IEEE Transactions on Power Systems*, 23(4):1836–1846, 2008.

K. Trivedi, V. Jindal, and S. Dharmaraja. Stochastic modeling techniques for secure and survivable systems. In Y. Qian, D. Tipper, P. Krishnamurthy, and J. Joshi, editors, *Information Assurance: Dependability and Security in Networked Systems*, chapter 7, pages 171 –207. Morgan Kaufmann, Burlington, MA, USA, Jul 2008.

A. Varga. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference*, 2001.

A. Varga and R. Hornig. An overview of the omnet++ simulation environment. In *First international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 10, Mar, 2008.

J. Voas, G. McGraw, and A. Ghosh. Reducing uncertainty about survivability. In M. R. Barbacci, editor, *Information Survivability Workshop*, pages 12–13, Pittsburgh, PA, USA, 1997. Software Engineering Institute - Carnegie Mellon University, IEEE Computer Society.

D. Wang, B. Madan, and K. Trivedi. Security analysis of sitar intrusion tolerance system. *Proceedings of the 2003 ACM workshop on Survivable and self-healing*, Jan 2003.

H. Wang and P. Liu. Survivability Evaluation Modeling Techniques and Measures. In J. N. D. Gupta and S. Sharma, editors, *Handbook of Research on Information Security and Assurance*, pages 504–517. IGI Global, Hershey, Pennsylvania, USA, 2009.

L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia. An attack graph-based probabilistic security metric. In V. Atluri, editor, *Data and Applications Security XXII*, volume 5094 of *Lecture Notes in Computer Science*, pages 283–296. Springer Berlin / Heidelberg, Jan 2008.

S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, UK, 1994.

J. Weiss. *Protecting Industrial Control Systems from Electronic Threats*. Momentum Press, New York, USA, 1st edition, May 2010.

D. Wells, S. Ford, D. Langworthy, and N. Wells. Software survivability. In *Proceedings of DARPA Information Survivability Conference and Exposition*, volume 2, pages 241 –255, 2000.

V. Westmark. A definition for information system survivability. *System Sciences, Proceedings of the 37th Annual Hawaii International Conference on*, 9(9):10, Jan 2004.

W. Yurcik. A Survivability-Over-Security (SOS) Approach to Holistic Cyber-Ecosystem Assurance. In *IEEE Workshop on Information Assurance*, West Point, NY, USA, 2002.

L.-J. Zhang, W. Wang, L. Guo, W. Yang, and Y.-T. Yang;. A survivability quantitative analysis model for network system based on attack graph. In *International Conference on Machine Learning and Cybernetics*, volume 6, pages 3211 – 3216, Jul 2007.

B. Zhu and S. Sastry. SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems*, volume 1, page 16, Stockholm, Sweden, 2010. TRUST - Team for Research in Ubiquitous System Technology.