Investigating Common SCADA Security Vulnerabilities Using Penetration Testing

Sello Glenton Ralethe

A research report submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering

Johannesburg, September 2014

DECLARATION

I declare that this research report was composed by myself, that the work herein is my own except where explicitly stated otherwise in the text. It is being submitted to the Degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other University.

……..
(Signature of Candidate)

…… day of …….. year …….

Acknowledgement

I would like to thank my supervisor, Prof. Ken Nixon, for all his invaluable input towards this research; it was a pleasure working with him. I give thanks to my family and friends, for their emotional support and words of encouragement during my journey. Lastly, I would like to thank the National Research Foundation for providing me with financial support.

Abstract

Supervisory Control and Data Acquisition (SCADA) systems were developed to assist in the management, control and monitor of critical infrastructure functions such as gas, water, waste, railway, electricity and traffic. In the past, these systems had little connectivity to the Internet because they ran on dedicated networks with proprietary control protocols and used hardware and software specific to the vendor. As a result, SCADA systems were secure, and did not face challenging vulnerabilities associated with the Internet. The need for remote connectedness, in order to collect and analyse data from remote locations, resulted in SCADA systems being increasingly getting connected to the Internet and corporate networks. Therefore, SCADA systems are no longer immune to cyber-attacks. There are reported cases on cyber-attacks targeted at SCADA systems. This research utilises penetration testing to investigate common SCADA security vulnerabilities. The investigation is conducted through experiments, under two different scenarios. Experiments were conducted using virtual plant environment. The results revealed vulnerabilities which are considered as common by the Idaho National Laboratory and others which are not common. Recommendations are provided on how to mitigate the vulnerabilities discovered in this research.

Table of Contents

List of Figures

List of Tables

1 Introduction

Industrial control systems can be considered as part of the backbone of a large number of industries that are affecting almost every basic service required by the modern society. These control systems are usually large networked computer systems and are used to assist in the management, control and monitoring of critical infrastructure functions such as gas, water, waste, railway, electricity production and distribution, and traffic. Due to their prominent and increasing importance, these systems can be considered important asset whose safety and security must be protected. However, it is a difficult task to secure these systems.

With the recent growth in cyber-attacks targeted at computer networks and systems in the infrastructure of major nations, it is important to investigate and understand the vulnerabilities of these control systems in order to develop appropriate and effective mitigation techniques that will ensure the systems' security and protection.

Supervisory Control and Data Acquisition (SCADA) systems are the common computer systems used to monitor and control major infrastructure. Among the basic functions of a SCADA system is its ability to provide data related to the operating state of the system and allow operators to remotely control the distributed system. According to Daniels and Salter (1999), one of the benefits of using a SCADA system is that it allows the high-level management of industrial process by merging data from the many distributed portions of the process. This can help enhance the robustness and reliability of the system.

Chikuni and Dondo (2009) mentioned that, in the past, SCADA systems had little connectivity to the Internet because they ran on dedicated networks with proprietary control protocols and used hardware and software specific to the vendor. As a result, SCADA systems were secure, and did not face challenging vulnerabilities associated with the Internet. Intrusion could only have been accomplished with physical access, so physical security measures were sufficient to repel computer-based attacks.

Queiroz et al. (2009) mentioned that the need for remote connectedness, in order to collect and analyse data from remote locations, resulted in SCADA systems being increasingly getting connected to the Internet and corporate networks. Therefore, SCADA systems are no longer immune to cyber-attacks, and such increased connectivity exposes SCADA systems to an enlarged attack surface (Queiroz, et al., 2009).

According to Queiroz et al. (2009), when designing the networks for SCADA systems, little attention was paid to security aspects in order to maximise functionality. These design decisions implied that the security of the system was traded for performance, reliability and flexibility, and as a result penetrating the control network can be accomplished without physical access to the SCADA system by remote attackers exploiting vulnerabilities in the gateway between the corporate and SCADA systems (Queiroz, et al., 2009).

Queiroz, et al. (2009) stipulated that the threat posed to the SCADA systems has a greater impact and scale of attack than common computer vulnerabilities. The impairment of SCADA networks could cause interruption of critical services, process redirection, or manipulation of operational data that could have serious consequences for the population. As a result, research into the security of SCADA systems is on the rise. The rapidly evolving landscape of SCADA systems warrants an increased understanding of and focus on their protection from cyber-attacks.

According to Byres and Lowe (2004), the Industrial Security Incident reported an increase in cyber-attacks during the period it collected incident reports. A popular attack on SCADA systems that further demonstrate the need for investing in the security of SCADA systems is the Stuxnet worm. This attack provides a good example of why protecting critical infrastructure is an important part of national security. Falliere et al. (2011) mentioned that the Stuxnet worm was first identified in June 2010 and was likely targeted at SCADA systems in Iran.

Stuxnet worm infected field devices controlling centrifuges in a Natanz nuclear facility. According to Farwell and Rohozinski (2011), the Stuxnet worm was able to maliciously vary centrifuge speeds to force them outside normal operating conditions and sabotage the system. The attack inflicted by the Stuxnet worm is significant because it demonstrated the capabilities of cyber-attacks on critical infrastructure and showcased the potential they have in cyber warfare.

Stuxnet was not the only worm to target SCADA systems in Iran. During the month of November 2011, Iranian officials said that they have detected a computer virus called Duqu that experts say is based on Stuxnet. According to Kaspersky Lab (2011), Duqu acts differently from Stuxnet, although they use the same code. Instead of destroying the systems it infects, Duqu secretly penetrates them and, according to some experts, creates “back door” vulnerabilities that can be exploited to destroy the networks at any time its creators may choose (Kaspersky Lab, 2011). Kaspersky Lab (2011) experts mentioned that Duqu embeds itself in computer systems for 36 days and “analyses and profiles” the system's workings before sending its findings out to a secure server and self-destructing.

Events such as the Stuxnet worm demonstrate the possibility to conduct espionage attacks on foreign states remotely. According to InfoSec Institute (2013), this has increased awareness of cyber threats, and the need to implement proper countermeasures to mitigate risk.

It is often the case that SCADA system components are under government of local authorities. In most cases these authorities do not deal with adequately trained personnel who operate with limited budgets (InfoSec Institute, 2013). As a results, SCADA systems are installed everywhere without being qualified in the installation phase. InfoSec Institute (2013) reported that there are many systems deployed with factory settings, pre-set standard configurations, and they’re common to entire classes of devices.

The increase in cyber-attacks on SCADA systems has led to a rapid increase in research programs in the security analysis of SCADA systems. SCADA security is different from the traditional network security and as a result research on SCADA security cannot be approached from the perspective of currently available network security research. Therefore, it must be investigated as its own research area.

It was after the discovery of the Stuxnet worm that governments and intelligence agencies all over the world requested assessment of security for critical infrastructure of their countries (InfoSec Institute, 2013). However, the focus here was on the evaluation of efficiency offered by defensive measures adopted to protect SCADA systems from cyber-attacks (InfoSec Institute, 2013).

Despite greater awareness of cyber threats, critical infrastructures of countries are still too vulnerable. In the Internet Security Threat Report published by Symantec (2012), there were 85 public SCADA vulnerabilities. This was a huge decrease over the 129 vulnerabilities in 2011. Since the emergence of the Stuxnet worm in 2010, SCADA systems have attracted more attention from security researchers (Symantec, 2012).

Research programs in the security analysis of SCADA systems include vulnerability analysis, penetration testing, security assessment, etc. (Wang Chunlei, et al., 2010). Penetration testing is utilised in this research to explore common SCADA security vulnerabilities, and is discussed in section 2.

The investigation of common SCADA vulnerabilities in this research was conducted through experiments. The experiments were simulated using 3 different computers under two different scenarios. The experiments yielded vulnerabilities which are classified by the Idaho National Laboratory (2011) as common to all SCADA systems and others which were not classified. From the list of the vulnerabilities discovered during the experiments, recommendations are presented in this research report towards how to mitigate the discovered vulnerabilities.

The structure of this paper is as follows: Section 2 provides background information for SCADA systems and Penetration testing and Penetration testing tools used in this research. Section 3 discusses the hypothesis addressed by this research report. Section 4 provides implementation of the tools used to facilitate SCADA simulation. In section 5, experiments are conducted and results of the experiments are collected. Section 6 provides a discussion of the results collected in section 5. Section 7 discusses recommendations on how to mitigate the vulnerabilities discovered during this research. Section 8 provides conclusion and possible future work.

Appendix A provides an overview of Nmap and how it was utilised in this research. Screenshots showing the commands used and the results obtained are presented here. Appendix B provides an overview of Nessus and how it was utilised in this research. Screenshots showing the commands used during the run of Nessus and the results obtained are presented in this Appendix. Lastly, Appendix C provides an overview of Metasploit and

how it was used in this research. Screenshots showing the commands used during the run of Metasploit are provided in Appendix C.

2 Background

This section provides background information related to the architecture and operation of SCADA systems, vulnerabilities SCADA systems can be exposure to, and Penetration testing and tools used in this research.

2.1 SCADA Systems

2.1.1 Overview

Computerised control systems are used to manage a large number of industrial processes. The diversity of purposes of industrial processes implies that the implementation of industrial control systems is also diverse. A system whose assets are highly distributed geographically is frequently referred to as a SCADA system. These systems can be relatively simple, such as one that monitors environmental conditions of a small office building, or very complex, such as a system that monitors all the activity in a nuclear power plant.

A SCADA system consist of a number of remote terminal units (RTUs) and/or Programmable Logic Controllers (PLCs), and the central host and the operator terminals. The RTUs collect field data and are connected back to a master station via a communication system (Bently Systems, Inc, 2004). For example, a SCADA can collect information regarding a leak on a pipe at the plant site, transfer the data about the leak back to a central site, then alert the master station that a leak has occurred. The collected data is normally real-time and it allows for the optimization of the operation of the plant and process. The master station displays the acquired data and also allows an operator to perform remote control tasks (Bently Systems, Inc, 2004).

In their early days, SCADA systems made use of Public Switched Network (PSN) for monitoring purposes. Today many systems are monitored using the infrastructure of the corporate Local Area Network/Wide Area Network (WAN). Wireless technologies are now being widely deployed for purposes of monitoring (Daneels & Salter, 1999).

2.1.2 Architecture

Even though SCADA systems are used in a varied of scale and purpose, they usually are designed based on a similar architecture. It is important to recognise the fundamental

similarities in SCADA systems because that makes it possible for researchers to make use of general models of the class of all SCADA systems. The general model is composed of four major parts: the process to be controlled, the field devices (RTUs, PLCs) physically connected to it, the centralised control centre, and the network that connects the controller and field devices (Communication Technologies, Inc, 2004). The relationship between these components is shown in figure 1.

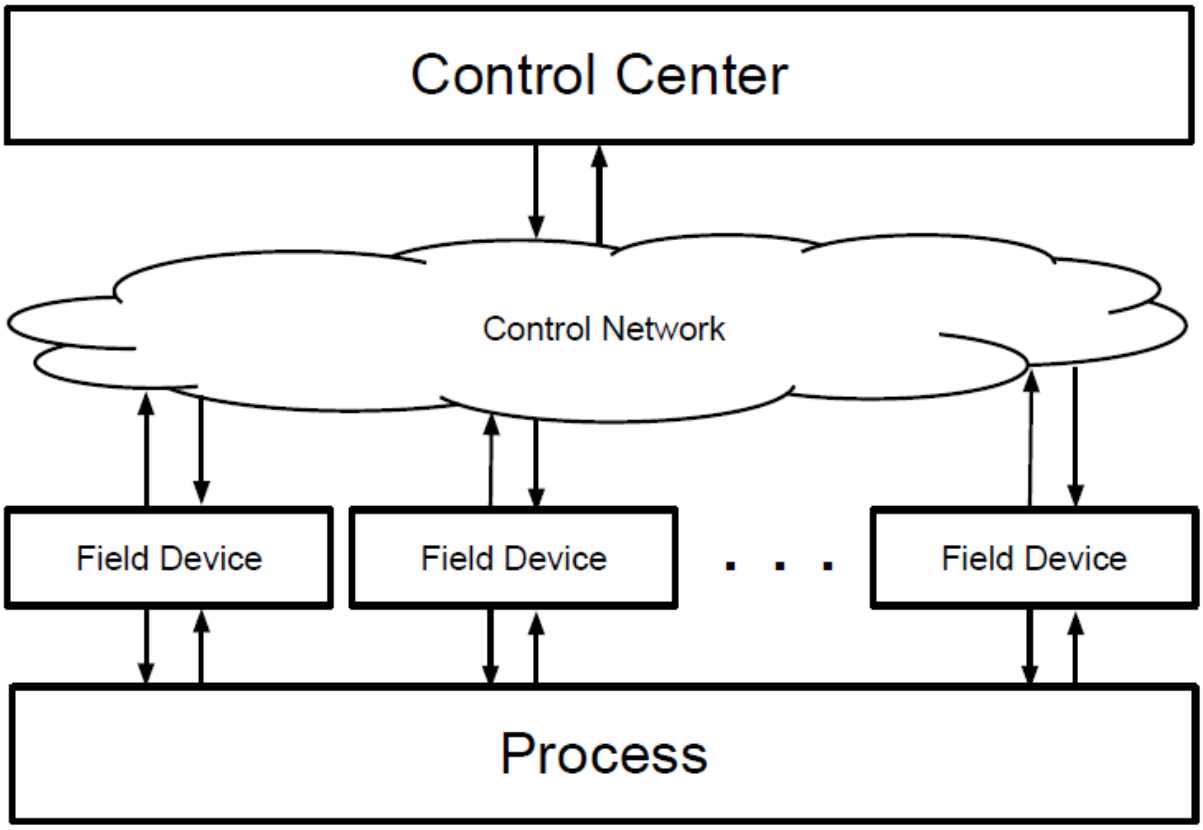


Figure 1. Typical SCADA Architecture

According to Communication Technologies Inc. (2004), a typical SCADA system consist of the following:

- One or more field data interface devices, usually RTUs, or PLCs, which interface to field sensing devices and local control switchboxes and valve actuators
- A communications system used to transfer data between field data interface devices and control units and the computers in the SCADA central host.
- A central host computer server or servers
- A collection of standard and/or custom software systems used to provide the SCADA central host and operator terminal application. The software is usually called Human Machine Interface (HMI).

Field devices can be considered as the "eyes and ears" of an SCADA system. For example, devices such as reservoir level meters, water flow meters, etc., can provide information that can inform an operator about the performance of a water distribution system.

According to Communication Technologies Inc. (2004), before an automation or remote monitoring can be achieved, the information coming from and to the field devices must first be converted to a form that is compatible with the language of the SCADA system. RTUs provide this interface. The primary function of RTUs is to convert electronic signals received from field interface devices into the language used to transmit the data over a communication channel. This language is regarded as the communication protocol (Communication Technologies, Inc, 2004).

The limited bandwidth of communication links between the SCADA central computer and the field devices requires the instructions for the automation or remote monitoring to be stored locally. Traditionally, such instructions are placed within the PLCs. In the past, the PLCs were physically separate from RTUs. Modern PLCs are directly connected to field devices and they have programmed intelligence in the form of logical procedures that will be executed in the event of certain field conditions (Communication Technologies, Inc, 2004).

Daneels and Salter (1999) mentioned that the origin of PLCs is in the automation industry and they were often used in manufacturing and process plant applications. In these applications, there was no great need to connect PLCs to communication channels. This was because PLCs were only required to replace traditional relay logic systems or pneumatic controllers (Daneels & Salter, 1999).

Communication Technologies (2004) stated that as PLCs became used more often to replace relay switching logic control systems, telemetry was used a lot with PLCs at the remote sites. According to Communication Technologies (200), it then became desirable to influence the program within the PLC through the use of a remote signal. Communication Technologies (2004) mentioned that this is in effect the "Supervisory Control" part of the SCADA acronym.

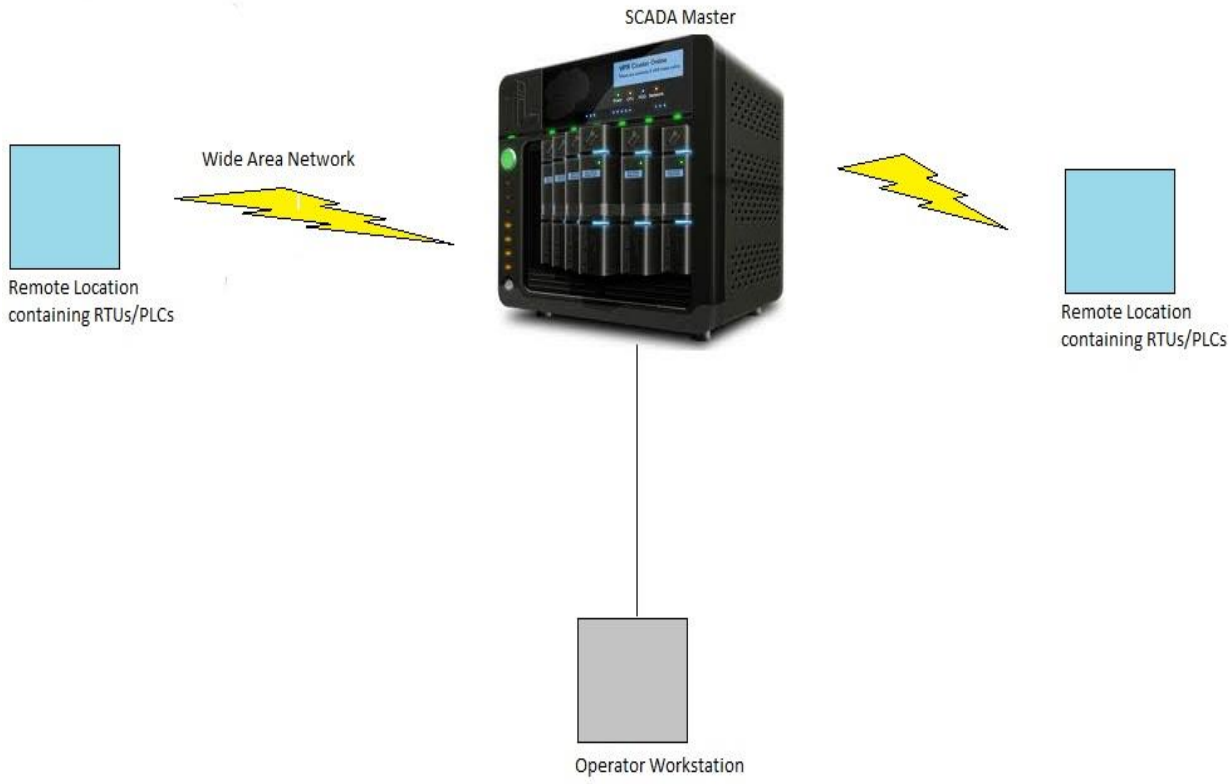


Figure 2. Typical SCADA system (Communication Technologies, Inc, 2004)

The Operator Workstation provides the interface to the human operators of the system. This software component is called the Human Machine Interface (HMI) and allows the operators to see an aggregated view of the state of the process and provides the means to send control commands to the field devices (Figure 1) in order to maintain correct operation. However, software of some form pervades all levels of a SCADA system (Communication Technologies, Inc, 2004). Depending on the size and nature of the SCADA application, software can be a significant cost item when developing, maintaining, expanding a SCADA system. According to Communication Technologies Inc. (2004), when software is well defined, designed, written, checked, and tested, a successful SCADA system will likely be produced. If there are poor performances in any of these project phases, a SCADA project can very easily fail.

A lot of SCADA systems make use of commercial proprietary software upon which the SCADA system is developed. The proprietary software is often configured for a specific hardware platform and may not interface with the software or hardware produced by competing vendors (Communication Technologies, Inc, 2004). A wide range of commercial off-the-shelf (COTS) software products are also available, some of which may suit the required application. Communication Technologies Inc. (2004) mentioned that COTS software is more flexible, and will interface with different types of hardware and software. For this research, a free version of COTS software is used.

The role of the communication network in figure 2 is to provide the means by which data can be transferred between the central host computer servers and the field devices. The communication network refers to the equipment needed to transfer data to and from different sites. The medium used can be either telephone or radio (Communication Technologies, Inc, 2004).

In a factory setting, the use of cable is usually implemented. However, this will not be practical for systems that cover large geographical areas. This is because of the cost of cables, conduits and the extensive labour in installing them. As a result, an economic solution for systems with large coverage is the use of telephone lines (i.e., leased or dial-up). According to Communication Technologies Inc. (2004), systems that require on-line connection with the remote stations can be covered by leased line. This is costly because one telephone line will be required per site. Dial-up can be used on systems that require updates at regular intervals. This can be achieved by using ordinary telephone lines. The host can dial a particular number of a remote site to get the readings and send commands (Communication Technologies, Inc, 2004).

However, remote sites are usually not accessible by telephone lines. Therefore, the use of radio offers an economical solution. Daneels and Salter (1999) mentioned that radio modems are used to connect the remote sites to the host. An on-line operation can also be implemented on the radio systems. A radio repeater can be used for locations where a direct radio link cannot be established.

Communication Technologies Inc. (2004) stated that historically, SCADA networks have been dedicated networks. However, according to Communication Technologies Inc. (2004), the increase in deployment of office LANs and WANs as a solution for interoffice computer networking, introduced the possibility to integrate SCADA LANs into everyday office computer networks.

One advantage of this arrangement is that there is no need to invest in a separate computer network for SCADA operator terminals. In addition, it also provides an easy path to integrate SCADA data with existing office applications such as spreadsheets, work management systems, etc. (Communication Technologies, Inc, 2004).

The master station shown in figure 2 is most often a single computer or a network of computer servers that provide a man-machine operator to the SCADA system. The role of the computers is to process the information received from and sent to the RTU/PLC sites and present it to human operators in a form that the operators can work with. Operator terminals are connected to the central host computer by LAN/WAN so that the viewing screens and associated data can be displayed for the operators (Communication Technologies, Inc, 2004).

Historically, SCADA vendors offered proprietary hardware, operating systems, and software that was largely incompatible with other vendors' SCADA systems. Daneels and Salter

(1999) mentioned that expanding the system required further contract with the original SCADA vendor.

However, the increased use of the personal computer imply that computer networking is commonplace in the office and therefore SCADA systems are now available that can network with office-based personal computers. Many modern SCADA systems can reside on computer servers that are identical to those servers and computers used for traditional office applications (Communication Technologies, Inc, 2004).

2.1.3 Security Vulnerabilities

The use of open protocols and standard devices and the connection to public network have made SCADA systems a major target of cyber-attacks. However, compared to common IT systems, attacks on the SCADA systems could have serious consequences, such as countrywide electricity blackout, crashing of stock markets (which will lead to significant financial impacts) and environmental damages (Wang Chunlei, et al., 2010).

As discussed in the previous subsection, a SCADA system is composed of multiple components. According to InfoSec Institute (2013), attackers could target each of these components in order to compromise a controlled process.

There are reported cases of attacks and potential threats on SCADA systems worldwide. For example, the "Aurora Generator Test" (Meserve, 2007) conducted in March 2007 simulated a remote cyber-attack on a generator control station which resulted in the partial destruction of a diesel-electric generator. Another example is the attempted distributed denial of service (DDoS) attack on an Israeli power plant (Wang Chunlei, et al., 2010).

According to Chikuni and Dondo (2007), these attacks are not unique to SCADA systems as they can affect all networked computers regardless of where they are deployed. What makes the SCADA systems unique is that they are highly customised and most of them do not have the same configuration and functionality. As a result, it requires specific and detailed knowledge in order to attack SCADA systems (Chikuni & Dondo, 2007).

Similarly, the vulnerabilities faced by regular computer systems hardware can also affect SCADA systems. These vulnerabilities include interruption, interception, and eavesdropping. Also, the communication links between SCADA hardware devices are susceptible to vulnerabilities faced by regular computers (Chikuni & Dondo, 2007).

Wang Chunlei, et al. (2010) reported that the communication links through which SCADA information travels is exposed and in most cases the messages are not encrypted. This vulnerability may result in data interception, regardless of the use of proprietary protocols. According to Wang Chunlei et al (2010), this kind of hardware vulnerability can be stopped by putting a strong security policy in place. The use of encryption on the communication links can help to reduce the risk of interception and modification of SCADA data (Wang Chunlei, et al., 2010).

Besides hardware vulnerabilities, SCADA system are also susceptible to software vulnerabilities. Chikuni and Dondo (2007) mentioned that out of all possible attacks on software, the ones that would cause the most damage to the SCADA system are those related to software modification. These kind of attacks modify software of the attacked host to either make the host fail or perform unintended tasks. Bugs are also very serious form of software vulnerability because if not fixed in time, hackers with very little skills can take advantage of publicised vulnerabilities to launch attacks on the SCADA system (Chikuni & Dondo, 2007).

According to Chikuni and Dondo (2007), these attacks can be prevented in a similar manner as on regular computer systems. However, in the cases of SCADA systems stricter controls such as configuration management systems to control software access can reduce the risk of software modification, deletion, and theft.

Regardless of the risks associated with software and hardware vulnerabilities, SCADA data has more value to an attacker than the software and hardware (Chikuni & Dondo, 2007). As mentioned by Chikuni and Dondo (2007), the control data from SCADA can be used by an attacker to the detriment of the utility if not handled with caution. SCADA data can also be stolen for sabotage reasons or for use by competitors. One method of safeguarding data integrity is to encrypt vital data (Chikuni & Dondo, 2007).

Although the vulnerabilities discussed above are associated with cyber-attacks via IP networks, there are reported cases of attacks via serial communication products. According to Zetter (2013), researchers have mentioned that breaching a power system through serial communication devices can actually be easier than attacking though the IP network since it does not require bypassing layers of firewalls.

Zetter (2013) mentioned that researchers, Chris Sistrunk and Adam Crain, found vulnerabilities in the products of more than 20 vendors. The vulnerabilities include some that would allow an attacker to crash or send a master server into an infinite loop, preventing operators from monitoring or controlling operations (Zetter, 2013). According to Zetter (2013), the vulnerabilities were found in devices that are used for serial and network communications between servers and substations.

According to Zetter (2013), an intruder could exploit the vulnerabilities by gaining physical access to a substation or by breaching the wireless radio network over which the communication passes to the server. Once an intruder has gained access to the network, they can send malformed message to the server to exploit the weakness.

As mentioned by Zetter (2013), in light of these developments, the ICS-CERT published a number of advisories about the discovered vulnerabilities, and some vendors have distributed patches for nine of the new vulnerabilities, but the rest remain unpatched so far. Zetter (2013) quoted Chris Sistrunk saying that many utilities have not applied the patches because they are not aware of the serious nature of the vulnerabilities.

According to Zetter (2013), the exploited systems make use of a protocol for serial communications that is used in almost all electrical utilities in the United States and Canada to transport communication between servers which are located in data centres and field devices such as PLCs and RTUs. This protocol is called DNP3 (Zetter, 2013). Generally, electric utilities comprise of a data centre with two or three servers that can each monitor and communicate with a lot of substations, depending on the size of the utility (Zetter, 2013).

As mentioned earlier about the purpose of the PLCs and RTUs, intruders can obscure operators to the conditions in the field by causing the server to crash or enter an infinite look. Operators might not initially realise this because a crashed server in the data centre does not always register to operators who work in other locations (Zetter, 2013). Because a lot of utilities make use of master servers for security purposes to control alarm systems, crashing the servers would potentially disable alarms as well (Zetter, 2013).

Out of the 25 vulnerabilities discovered by Chris Sistrunk and Adam Crain, Zetter (2013) mentioned that the most serious one was the buffer overrun. This vulnerability can allow an intruder to inject arbitrary code into the system and own the server (Zetter, 2013).

Zetter (2013) mentioned that the problem does not lie with the standard for DNP3, but that the vulnerabilities are introduced in the insecure ways that vendors have implemented it. According to Zetter (2013), separate security standards set by the North American Electric Reliability Corporation for how to secure power systems focus only on IP communications. This worsen the problem because the standards overlook the real vulnerabilities that serial communications also present (Zetter, 2013).

It was after the occurrence of some of the events discussed above that many security firms have started designing solutions to address security problems of SCADA systems. InfoSec Institute (2013) mentioned that the major challenge for governments is the inclusion of protection for these critical components in their cyber strategies. According to InfoSec Institute (2013), audits that were carried out by several governments illustrated the lack of security mechanisms for the many SCADA systems located all over the world.

According to InfoSec Institute (2013), the most cost-effective approach to securing SCADA systems is to establish and follow a risk management framework. The North American Electric Reliability Corporation (NERC) published Cyber Security Standards that highlight cyber security lifecycle. The NERC process has three stages: identifying risks, implementing controls/mitigating risks; and maintaining acceptable levels through evaluation and monitoring (NERC, 2005).

In addition to the vulnerabilities mentioned above, there are other sources of security vulnerabilities associated with the web services of SCADA systems. These can be uncovered by performing penetration testing on the SCADA system. Penetration testing is discussed in the next subsection.

2.2 Penetration Testing

2.2.1 Overview

Modern SCADA systems provide better access to plant information by means of web services. Web services provide a strategic mean for data exchange because of their simple interface. However, Antunes and Vieira (2009) noted that since web services are widely

exposed, the existence of a security vulnerability in these services can be uncovered and exploited by hackers.

Most web services make use of relational database for persistent storage. Thus, security vulnerabilities such as SQL Injection are particularly relevant to web services of SCADA systems. According to Antunes and Vieira (2009), SQL Injection vulnerabilities are related directly to structure of the code of web services. SQL Injection attacks change SQL commands that are sent to the database. Improperly validated input parameters make it possible for SQL Injection (Antunes & Vieira, 2009).

Penetration testing is one of the well-known techniques used by web service developers to discover security vulnerabilities in their code. Antunes and Vieira (2009) mentioned that penetration testing takes a "black-box" approach in the sense that it stresses the application from an attacker's point of view. In other words, penetration testing discovers vulnerabilities by simulating attacks from hackers on a target application (Antunes & Vieira, 2009).

Antunes and Vieira (2009) mentioned that unlike other techniques, penetration testing tools provide an automatic way to search for vulnerabilities. Therefore, these tools avoid the repetitive and tedious task of carrying out tests manually for each vulnerability type (Antunes & Vieira, 2009). According to Antunes and Vieira (2009), these tools do not require access to the source code of the target application. Another reason that make penetration testing more famous for testing the security of web application is that it tests applications in context. This allows for the discovery of vulnerabilities that arise from specific configuration and environment issues (Halfond, et al., 2009).

According to Halfond et al. (2009), penetration testing can be divided into three phases: information gathering, attack generation, and response analysis. During information gathering, testers aim to gain information about the target application. This can be achieved by using techniques such as automated scanning, web crawlers, and social engineering. The information gathered is then used, together with domain knowledge about possible vulnerabilities, to generate an attack during the attack generation phase. The generation of attacks can be automated by using commercial or open-source tools (Halfond, et al., 2009).

Halfond et al. (2009) mentioned that under the response analysis phase, testers determine if an attack has succeeded, if so, testers then log information about the attack. The discovered vulnerabilities are then detailed in a report at the end of the penetration testing process. Developers can use this information to eliminate the vulnerabilities and improve the security of their software (Halfond, et al., 2009).

2.2.2 Identifying Security Vulnerabilities

Steps taken to identify vulnerability under penetration testing are similar to the steps that an unauthorized attacker may take. However, the attacker may choose to proceed more slowly to avoid detection. On the other hand, penetration testing can be slowed down so that the target company can learn where their detection threshold is and make improvements (Northcutt, et al., 2006).

The first step in a penetration test is scouting of the target network. During this step, the tester attempts to learn as much as possible about the target network. According to Northcutt et al. (2006), this normally starts with identifying publicly accessible services such as mail and web servers from their service banners. This is based on the knowledge that a lot of servers will report the Operating System they are running on, the version of software they are running, patches and modules that have been enabled, the current time, and perhaps even some internal information like an internal server name or IP address (Northcutt, et al., 2006).

After scouting the target network, the penetration tester will then proceed to verify the gathered information. Although the tester does not have solid knowledge on what is running, he may have a pretty good idea. The collected information can be combined and then compared with known vulnerabilities, and then those vulnerabilities can be tested to see if the results support or contradict the prior information (Northcutt, et al., 2006).

2.2.3 Why Perform Penetration Testing?

There are different reasons for carrying out a penetration test. One of the main reasons is to find vulnerabilities and fix them before an attacker does. Another reason for performing a penetration test is to give the IT department at the target company a chance to respond to an attack (Northcutt, et al., 2006).

Northcutt et al. (2006) mentioned that attackers are employing a variety of automated tools and launching network attacks looking for ways to penetrate systems. Penetration testing allows the IT department of a company to find holes in the system before somebody else does. In a sense, penetration testing provides IT management with a view of their network from a malicious point of view.

2.2.4 Penetration Testing Tools

There are a wide variety of tools that are used in penetration testing. According to Northcutt et al. (2006), there are two main types of penetration testing tools; reconnaissance or vulnerability testing tools and exploitation tools. However, Northcutt et al. (2006) noted that penetration testing is more directly tied to the exploitation tools and the initial scanning and reconnaissance is often done using less intrusive tools. The difference between these two types is not clear cut. For this research, 3 different tools were used. The tools were chosen in line with the objectives of this study as outlined in section 3. The tools used for this research are: Metasploit, Nessus, and Nmap.

Metasploit is a “framework” for cyber exploitation. As a framework, it eases the effort to exploit known vulnerabilities in networks, operating systems and applications, and to develop new exploits for new or unknown vulnerabilities (RAPID7, 2013). Metasploit provides attack libraries attack payloads that can be put together in a modular manner. The main purpose of Metasploit is to get to a command prompt on the target computer (RAPID7, 2013). Once a

security tester has gotten to a command-line, it is quite possible that the target computer will be under his total in a short time. The version of Metasploit used for this research is the trial version of the Metasploit Pro.

Nessus is a tool designed to automate the testing and discovery of known security problems. According to Tenable Network Security (2013), Nessus has a library of vulnerabilities together with tests to identify those vulnerabilities. In a lot of cases, Nessus depends on the responses from the target computer instead of actually trying to exploit the system. The version of Nessus used for this research is the trial version of Nessus 5.2.

Similar to Nessus, Nmap is a port scanning tool. Port scanning is usually a part of the reconnaissance phase of a penetration test or an attack (Northcutt, et al., 2006). Sometimes attackers will limit their testing to a few ports while other times they will scan all available ports. According to Northcutt et al. (2006), in order for a vulnerability scanner to do a thorough job, it should scan all ports. An actual attacker may choose to not scan all ports if he finds a vulnerability that can be exploited because of the "noise" (excess traffic) a port scanner creates (NMAP.ORG, 2009).

Another capability of Nmap is its ability to determine the operating system of the target computer. Northcutt et al. (2006) mentioned that different networking implementations will respond differently to different network packets. Nmap maintains a type of database and will match the responses to make a guess at what type of operating system the target computer is running (NMAP.ORG, 2009). According to Northcutt et al. (2006), this OS detection is not perfectly accurate but it can help the attacker tailor his attack strategy, especially when coupled with other pieces of information. The free version of Nmap was used for this research.

3 Research Hypothesis

This section presents the research hypothesis addressed in this research report.

3.1 Hypothesis

The Vulnerability Analysis of Energy Delivery Control Systems report, prepared by Idaho National Laboratory (2011), describes the common vulnerabilities on energy sector control systems, and provides recommendations for vendors and owners of those systems to identify and reduce those risks.

The report presents vulnerabilities at a high level to provide awareness of the common SCADA security vulnerability areas without divulging product-specific information. Vulnerabilities that could be used as part of an attack against an SCADA are consolidated into generic common SCADA vulnerabilities. The vulnerabilities described in the report by Idaho National Laboratory (2011) were routinely discovered in NSTB (National SCADA Test Bed) assessments using a variety of typical attack methods to manipulate or disrupt system operations (Idaho National Laboratory, 2011).

The 10 most significant cyber security risks identified by Idaho National Laboratory (2011) during NSTB software and production SCADA assessments are:

1. Unpatched published known vulnerabilities
2. Web Human-Machine Interface (HMI) vulnerabilities
3. Use of vulnerable remote display protocols
4. Improper access control (authorization)
5. Improper authentication
6. Buffer overflows in SCADA services
7. SCADA data and command message manipulation and injection
8. SQL injection

9. Use of standard IT protocols with clear-text authentication
10. Unprotected transport of application credentials

The common vulnerabilities in the report were found on two or more unique SCADA configurations. Idaho National Laboratory (2011) mentioned that even though SCADA functions, designs, and configurations vary among vendors, versions, and installations, their high-level vulnerabilities and defensive recommendations are similar. Therefore, the following hypothesis arises.

Hypothesis: Penetration testing on a given SCADA, system which was not part of the research covered by Idaho National Laboratory (2011), will reveal vulnerabilities which are classified as common by the Idaho National Laboratory (2011) report, as well as other uncommon vulnerabilities.

4 Implementation

This section describes the implementation of a virtual plant for experimental purpose. The design tools used in the implementation are described here.

4.1 Overview

Design of the virtual plant environment consist of three aspects: SCADA software, OPC Server, and Simulated Process Model. The SCADA software and the Simulated Process model are linked together using an OPC interface. The conceptual design of the plant environment is illustrated in figure 3. This setup and implementation were adapted from Eltayeb (2009). The only difference is in the choice of the HMI software.

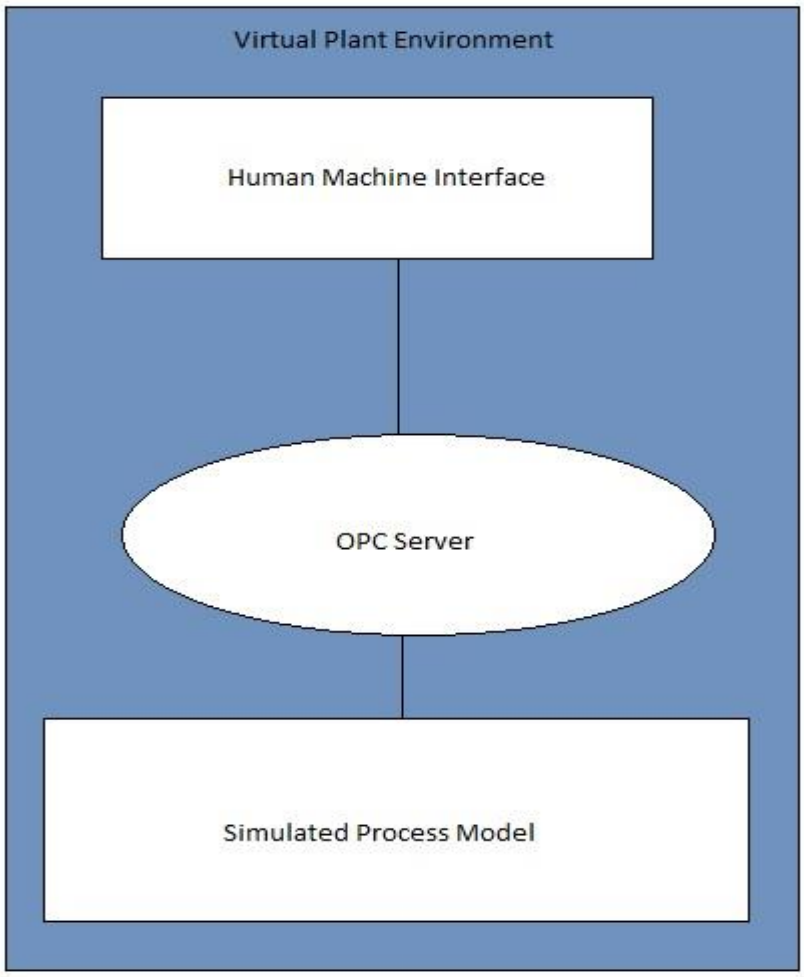


Figure 3. Conceptual Design of the virtual plant environment (Eltayeb, 2009).

An OLE (Object Linking and Embedding) for Process Control (OPC) is a standard mechanism that enables the communication and data exchanging between various types of devices and control applications. An OPC consist of an OPC Server and an OPC Client (Eltayeb, 2009).

An OPC Server is a software application that acts as an API (Application Programming Interface) or protocol converter. It allows Windows programs to communicate with industrial hardware devices such as PLC, or any data source such as database or User Interface, and translate the data into the OPC Client (Eltayeb, 2009).

An OPC Client is a software application used to access (for reading and/or writing) information provided by the OPC Server through the OPC standard. For this research, the OPC interface was implemented using KEPServerEX software. The other tools used in the design are Simulink, OPC Toolbox and SCADA software (Eltayeb, 2009).

4.2 Simulink

Simulink is a commercial tool for modelling, simulating and analysing multi-domain dynamic systems. Simulink is developed by the MathWorks, and comes as a package of MATLAB. Simulink offers integration with the rest of the MATLAB environment and can either drive MATLAB or be scripted from it.

4.3 OPC Toolbox

OPC Toolbox software is a collection of functions that extend the capability of the MATLAB environment, and blocks that extend the Simulink simulation environment. Using OPC Toolbox functions and blocks, it is possible to acquire live OPC data directly into MATLAB and Simulink, and write data directly to the OPC Server from MATLAB and Simulink. The OPC Toolbox block library includes the capability of running Simulink models in pseudo real time, by slowing the simulation to match the system clock. Using Simulink and the OPC Toolbox block library, it is possible to prototype control systems, provide plant simulators, and perform optimization and tuning tasks (Eltayeb, 2009).

4.4 KepServerEx

KEPServerEx is a 32-bit Windows application that provides a means of bringing data and information from a wide range of industrial devices and systems into client applications on Windows PC. In the industrial market, it has usually come to mean the sharing of manufacturing or production data between a variety of applications ranging from human machine interface software and data historians, to large MEX and ERP applications (Eltayeb, 2009). Figure 4 illustrate the tools used in the design.

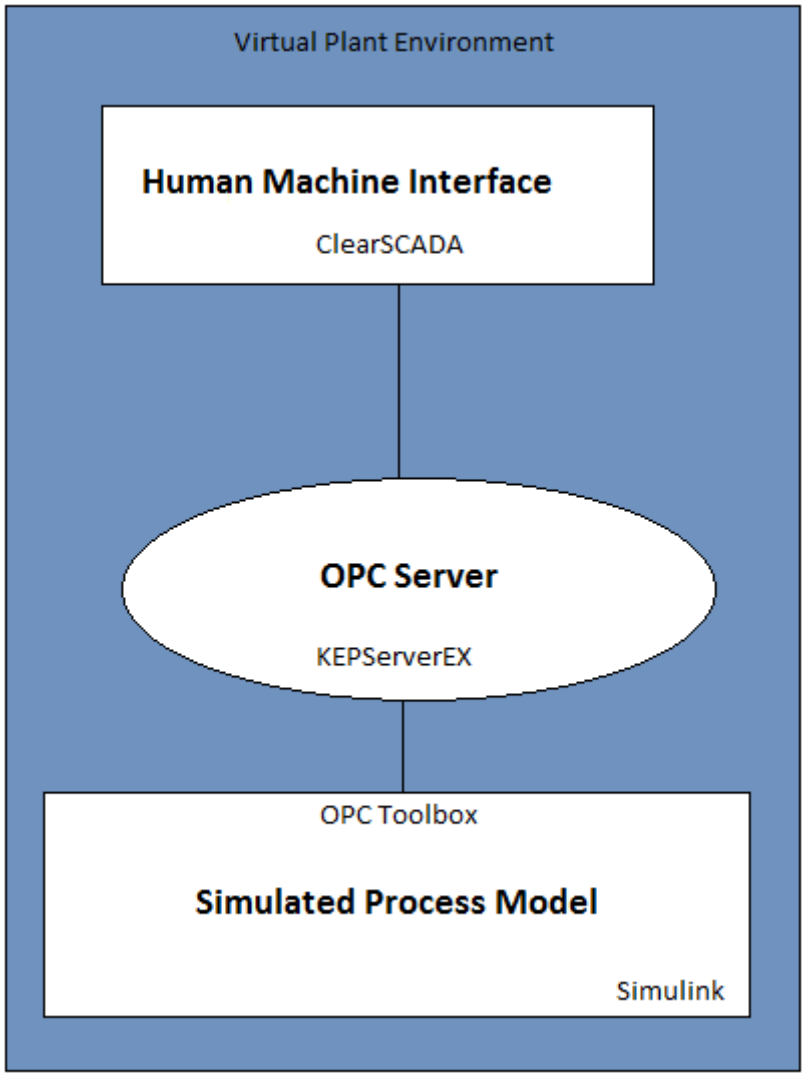


Figure 4. Tools used in the design (Eltayeb, 2009).

This is the design used by Eltayeb (2009). For this research, the HMI used was the free version of ClearSCADA developed by Schneider Electric. According to Schneider Electric (2012), this free version has the same capabilities as the paid version, the difference being that the free version is only limited to 50 objects in a project.

5 Experimental Setup and Results

This section presents the experimental setup designed to test the hypothesis, and the results collected from the experiments.

5.1 Overview

This research was carried out in terms of experiments that involve simulations. Three computers were: one running a SCADA system, one simulating a plant, and the other was used to run penetration tests targeted at the SCADA system. The SCADA system and the virtual plant were on the same network. In one experiment the attacker PC was on the same network and in the other experiment it was on a different network. This setup demonstrate the validity of the tools under varying attack vectors. Figures 5 and 6 illustrate the setup.

5.2 Scenario One

This scenario is depicted in figure 5. Under this scenario, the attacker launches the attack on the SCADA system from within the company network.

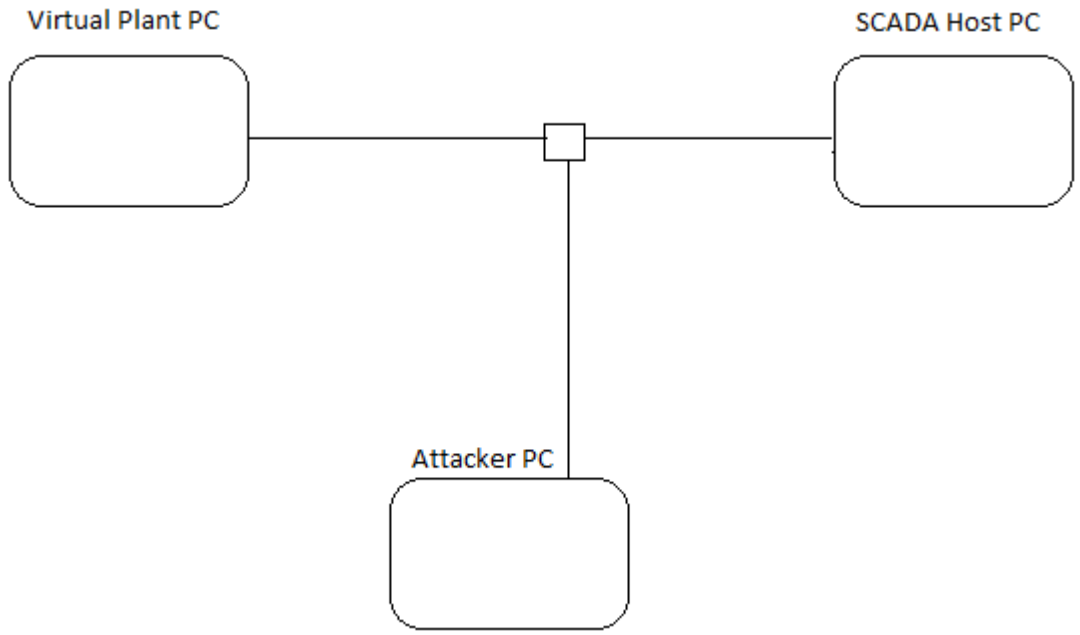


Figure 5. Scenario One: Attack from within the company network.

Table 1 presents the results collected after running Nmap targeting the computer running the SCADA system. Appendix A describes the commands used on Nmap to obtain these results, and screenshots highlighting the results outlined there.

Table 1. Nmap Results

PORT	STATE	SERVICE
23/tcp	open	telnet
513/tcp	open	tcpwrapped
514/tcp	open	tcpwrapped

Even though Nmap found three open ports, it was not able to identify the operating system of the computer running the SCADA system after three unsuccessful tries.

Table 2 presents results collected after running Nessus targeting the computer running the SCADA system. The commands and screenshots associated with these results can be found in Appendix B.

Table 2. Nessus Results

Vulnerability	Risk Factor
Distributed Denial of Service	High
Information Disclosure	Medium
SQL Injection	High
SQL pg_dump	Medium
SMB Signing Disabled	Medium
SSL Certificate cannot be Trusted	Medium
SSL Self-Signed Certificate	Medium
SSL Certificate with Wrong Hostname	Medium
SSL RC4 Cipher Suites Supported	Low

Table 3 presents results collected after running Metasploit targeting the computer running the SCADA system. The commands used and screenshots associated with these results are presented in Appendix C.

Table 3. Metasploit Results

Vulnerability	Risk Factor
Man-in-The-Middle attack	High
Weak Cryptography	Medium
Insecure renegotiation of TLS/SSL	Medium
IP Forwarding	Medium

5.3 Scenario Two

This scenario is depicted in figure 5. Under this scenario, the attacker launches the attack on the SCADA system from outside the company network.

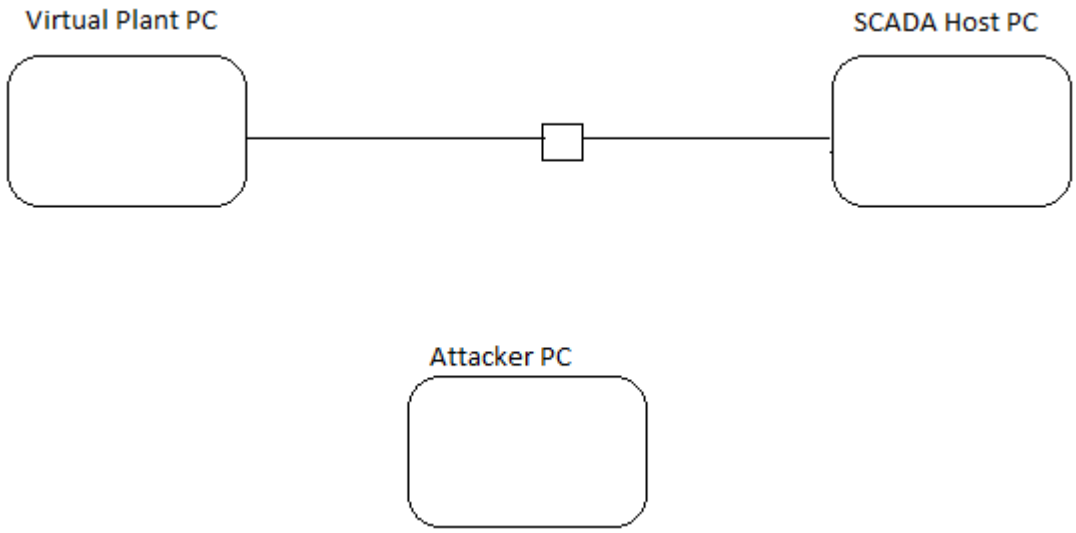


Figure 5. Scenario Two: Attack from outside the company network.

To model this scenario, three computers were used. The computers were all on different networks. This was done in order to closely model real world environment where, for example, a company has a plant in a remote location and controls and monitors that plant using wide area network. Also, since the attacker will launch attacks from a remote location, this scenario depicts the real world cases in which attackers can launch attacks on SCADA systems in foreign countries.

Table 3 presents the results collected after running Nmap targeting the computer running the SCADA system. Appendix A describes the commands used on Nmap to obtain these results, and screenshots highlighting the results outlined there.

Table 4. Nmap Results for Scenario 2

PORT	**STATE**	**SERVICE**
23/tcp	open	telnet
513/tcp	open	tcpwrapped
514/tcp	open	tcpwrapped

Unlike in scenario 1, here Nmap was able to identify the operating system of the computer running the SCADA system.

Table 5 presents results collected after running Nessus targeting the computer running the SCADA system. The commands and screenshots associated with these results can be found in Appendix B.

Table 5. Nessus Results for Scenario 2

Vulnerability	**Risk Factor**
DNS Server Zone Transfer Information Disclosure	High
mDNS Detection	Medium
DNS Server Cache Snooping Remote Information Disclosure	High
Open SSL Vulnerabilities	Medium

Table 3 presents results collected after running Metasploit targeting the computer running the SCADA system. The commands used and screenshots associated with these results are presented in Appendix C.

Table 6. Metasploit Results

Vulnerability	**Risk Factor**
Man-in-The-Middle attack	High
Microsoft Windows SMB LanMan Pipe Server Listing Disclosure	Medium
Microsoft Windows NTLMSSP Authentication Request Remote Network Name Disclosure	Medium
Network Time Protocol (NTP) Server Detection	Low

6 Discussions

Results collected from the previous sections are discussed here.

6.1 Scenario One

In the first run of experiments, Nmap was ran, targeting the computer running the SCADA system. Nmap is a port scanning tool and cannot exploit the vulnerabilities of the ports. During the run, Nmap identified open ports on the host which could be used by an attacker to gain improper access to the SCADA system and be able to manipulate SCADA data and command messages.

One of the open port was the one associated with telnet service. Telnet is a protocol that allows one computer to execute a text terminal on another. According to Falliere et al. (2011), Telnet is not encrypted; passwords and all other data will be transmitted as clear text.

However, Nmap failed to identify the operating system of the host running the SCADA system. A possible explanation can be that, the host is running Microsoft's Windows 8 and the version of Nmap used did not have updated functionalities that can identify if the operating system is Microsoft's Windows 8.

The second run of experiments was conducted by using Nessus to target the computer running the SCADA system. The Nessus tool gives a detailed report on the vulnerabilities discovered and the effect of those vulnerabilities on the system. The first vulnerability reported by Nessus was the denial of service vulnerability. Nessus report revealed that the version of Apache HTTP Server running on the target computer was affected by a denial of service (DoS) vulnerability. By making a series of HTTP request with overlapping ranges in the *Range* or *Request-Range* request headers can result in memory and CPU exhaustion (Tenable Network Security, 2011). A remote, unauthenticated attacker could exploit this to make the system unresponsive. Tenable Network Security (2011) mentioned that exploit code is publicly available and attacks have reportedly been observed.

The other vulnerability associated with the Apache HTTP server was the information disclosure vulnerability. Nessus report revealed that sending a request with HTTP headers long enough to exceed the server limit caused the web server to respond with an HTTP 400. By default, the offending HTTP header and value are displayed on the *400 error page*. When used in conjunction with other attacks (e.g., cross-site scripting), this could result in the compromise of *httpOnly* cookies (Tenable Network Security, 2011).

These vulnerabilities demonstrate the vulnerabilities associated with web services. Since modern SCADA systems make use of web services, they are susceptible to these vulnerabilities. The other vulnerability discovered by Nessus is associated with SQL injection. The scan discovered that *PostgreSQL* incorrectly checked permissions on functions called by a trigger. An attacker could attach a trigger to a table they owned and possibility

escalate privileges (Tenable Network Security, 2013). The scan also revealed that *PostgreSQL* incorrectly truncated SSL certificate name checks to 32 characters. According to Tenable Network Security (2013), if a host name was exactly 32 characters, this issue could be exploited by an attacker to spoof the SSL certificate.

Another vulnerability associated with the SSL certificate is that the certificate did not have a signature from a known public certificate authority. According to Tenable Network Security (2013), this situation can occur in three different ways, each of which results in a break in the chain below which certificates cannot be trusted.

First, the top of the certificate chain sent by the server might not be descended from a known public certificate authority. This can occur either when the top of the chain is an unrecognized, self-signed certificate, or when intermediate certificates are missing that would connect the top of the certificate chain to a known public certificate authority (Tenable Network Security, 2013).

Second, the certificate chain may contain a certificate that is not valid at the time of the scan. This can occur either when the scan occurs before one of the certificate's *notBefore* dates, or after one of the certificate's *notAfter* dates (Tenable Network Security, 2013).

Third, the certificate chain may contain a signature that either didn't match the certificate's information, or could not be verified. As mentioned by Tenable Network Security (2013), bad signatures can be fixed by getting the certificate with the bad signature to be reassigned by its issuer. Furthermore, Tenable Network Security (2013) stated that signatures that could not be verified are the result of the certificate's issuer using a signing algorithm that Nessus either does not support or does not recognise.

Tenable Network Security (2013) mentioned that if the remote host is a public host in production, any break in the chain nullifies the use of SSL as anyone could establish a man-in-the-middle attack against the remote host.

Still on SSL; the Nessus scan revealed that the host computer supports the use of RC4 in one or more cipher suites. According to Tenable Network Security (2013), the RC4 cipher is flawed in its generation of a pseudo-random stream of bytes so that a wide variety of small biases are introduced into the stream, decreasing its randomness. Furthermore, Tenable Network Security (2013) noted that if plaintext is repeatedly encrypted (e.g. HTTP cookies), and an attacker is able to obtain many (i.e. tens of millions) cipher texts, the attacker may be able to derive the plaintext.

Another vulnerabilities associated with SSL certificate was the certificate chain was not signed by a recognized certificate authority. Tenable Network Security (2013) reported that this nullifies the use of SSL as anyone could establish a man-in-the-middle attack against the host. However, Tenable Network Security (2013) noted that Nessus does not check for certificate chains that end in a certificate that is not self-signed, but is signed by an unrecognised certificate authority.

Lastly, the scan discovered that *PostgreSQL pg_dump* utility incorrectly filtered line breaks in object names. An attacker could create object names that execute arbitrary SQL commands when a dump script is reloaded (Tenable Network Security, 2013).

The third run of experiments was conducted by using Metasploit to target the computer running the SCADA system. The first vulnerability discovered was the Microsoft Windows Remote Desktop Protocol Server man-in-the-middle attack. The report from Metasploit penetration testing stated that the RDP client makes no effort to validate the identity of the server when setting up encryption. An attacker with the ability to intercept traffic from the RDP server can establish encryption with the client and server without being detected. A MiTM attack of this nature would allow the attacker to obtain any sensitive information transmitted, including authentication credentials.

According to Rouse (2007), this flaw exists because the RDP server stores a hardcoded RSA private key in the *mstlsapi.dll* library. Any local user with access to this file (on any Windows system) can retrieve the key and use it for this attack (Rouse, 2007). This vulnerability can easily be solved by forcing the use of SSL as a transport layer for the RDP service, and allowing connections only from computers running Remote Desktop with Network Level Authentication.

Metasploit also revealed that the computer running the SCADA system uses weak cryptography. Using weak cryptography may allow an attacker to eavesdrop on the communications more easily and obtain screenshots and/or keystrokes. Another vulnerability discovered by Metasploit was that the host computer allows insecure renegotiation of TLS/SSL connections. Although the host computer encrypts traffic using TLS/SSL but it allows a client to insecurely renegotiate the connection after the initial handshake. An unauthenticated, remote attacker may be able to leverage this issue to inject an arbitrary amount of plaintext into the beginning on the application protocol stream, which could facilitate man-in-the-middle attacks if the host computer assumes that the sessions before and after renegotiation are from the same ‘client’ and merges them at the application layer (Microsoft, 2010).

Furthermore, it was discovered that the host computer does not limit the number of renegotiations for a single TLS/SSL connection. This vulnerability permits another computer to open several simultaneous connections with the host computer and repeatedly renegotiate them, possibly leading to a denial of service condition (IBM, 2012).

Lastly, Metasploit discovered that the host computer had IP forwarding enabled. An attacker may use this vulnerability to route packets through the host and potentially bypass some firewalls/routers/NAC filtering. The simple solution for this vulnerability is to disable forwarding on the host computer.

6.2 Scenario Two

As in scenario one, the results for Nmap were similar. This is because the same computer running the SCADA system was used. Although in this case Nmap successfully identified the Operating system of the host computer. The explanation here is that the Nmap version used under scenario one is different to the version used under scenario two. The latest version of Nmap was used for scenario two.

The second run of simulations under scenario two involved Nessus. The vulnerabilities discovered differ from those in scenario one. The differences can be attributed to the differences in IT policies. Under scenario one, the experiments were conducted on the University network, while scenario two experiments involved home network and the University network.

The first vulnerability reported by Nessus was the DNS Server Zone Transfer Information Disclosure. The synopsis of this vulnerability is that the remote name server allows DNS zone transfers to be performed. According to Tenable Network Security (2011), a zone transfer allows a remote attacker to instantly populate a list of potential targets. This can be a problem because most companies usually use a naming convention that can give hints as to a server's primary application. For instance, a company can name their servers proxy.example.com, payroll.example.com, etc.

The next vulnerability discovered was the mDNS (Multicast Domain Name Server) Detection. This vulnerability allows an attacker to determine the mDNS protocol, by which the attacker can be able to uncover information from the remote host such as its hostname, and the list of services it is running (Tenable Network Security, 2011).

Nessus also discovered that the remote DNS Server is vulnerable to cache snooping attacks. According to Tenable Network Security (20111), this vulnerability can allow a remote attacker to determine which domains have recently been resolved via this name server, and therefore which hosts have been recently visited. For instance, if an attacker was interested in whether a company uses any online services, they would be able to use this attack to build a statistical model regarding company usage of those online services.

Attackers can use DNS cache snooping for other different reasons. One of those is to hijack a user's session on a website that requires authentication (Grangeia, 2004). Websites that require authentication maintain a session between HTTP requests. Grangeia (2004) mentioned that a session ID is generated when the user first accesses the site, and then that ID is passed to the user via a HTTP cookie, which is sent on every HTTP request. According to Grangeia (2004), some poorly written session ID generators rely solely on the actual time of day to generate its "pseudo-random" part. Usually seconds to microseconds are used for part of the entropy (Grangeia, 2004).

Thus, if an attacker knows exactly (by the second) when a user first accesses a given site can be helpful in predicting session ID's in an attempt to hijack a user's session. Cached DNS information can help in this matter (Grangeia, 2004). According to Grangeia (2004), an attacker can easily calculate the exact time when a user first accessed the site by subtracting the difference between the initial TTL set by the authoritative server and the cached TTL to

the local time. This can be achieved by snooping the DNS cache that the user is using for name resolutions (Grangeia, 2004).

DNS cache snooping can also be used to locate users on the Internet through their IP, and track Email conversations, in order to know if two companies are exchanging emails (Grangeia, 2004).

Lastly, the vulnerability discovered by Nessus was the OpenSSL vulnerability. It was discovered that the elliptic curve cryptography subsystem in OpenSSL did not properly implement curves over binary fields. Tenable Network Security (2011) mentioned that this could allow an attacker to determine private keys of a TLS server via multiple handshake attempts.

The third run of experiments was conducted by using Metasploit to target the computer running the SCADA system. The first vulnerability discovered was the Microsoft Windows Remote Desktop Protocol Server man-in-the-middle attack. This vulnerability was also discovered in scenario one.

The next vulnerability discovered by Metasploit was the Microsoft Windows SMB LanMan Pipe Server Listing Disclosure. This vulnerability makes it possible to obtain the browse list of the remote Windows system by sending a request to the LANMAN pipe. According to Microsoft (2010), the browse list is the list of the nearest Windows systems of the remote host.

The next vulnerability was the Microsoft Windows NTLMSSP Authentication Request Remote Network Name Disclosure. This vulnerability makes it possible for the attacker to obtain the network name of the remote host. By sending an NTLMSSP authentication request, it is possible to obtain the name of the remote system and the name of its domain.

The last vulnerability discovered by Metasploit was the Network Time Protocol (NTP) Server Detection. NTP server provides information about the current date and time of the remote system and may provide system information that could be used in an attack.

7 Recommendations

This section brings forward recommendations for mitigating the vulnerabilities discovered by Nessus and Metasploit. The vulnerabilities covered here are those with high risk factor. The other vulnerabilities are not covered here because the strategies to mitigate are trivial, and in most cases can be fixed with clicking of buttons.

7.1 Nessus Vulnerabilities

7.1.1 Distributed denial-of-service

Rouse (2013) defines a distributed denial-of-service (DDoS) attack as an attack in which a multitude of compromised systems attack a single target, thereby causing denial of service for users of the target system. Furthermore, the incoming flood of messages to the target system essentially forces it to shut down, thereby denying service to the system to legitimate users (Rouse, 2013). The aim here is to make the target system or network resource unavailable to its intended users by sending thousands of packets to the target system.

During a typical DDoS attack, the attacker starts by exploiting a vulnerability in one computer system and making it the DDoS master. The attack master, also known as the *botmaster*, identifies and infects other vulnerable systems with malware. A computer under the control of an attacker is known as a zombie or bot. As mentioned by Rouse (2013), the attacker eventually instructs the controlled machines (botnet or a zombie army) to launch an attack against a specified target.

According to Lambert (2013), DDoS can take multiple forms. There is a type of DDoS attack called *Syn Attack*. Under this attack, the attacker opens a TCP connection, the way one would normally connect to a website, but never finishes the initial handshake (Lambert, 2012). Basically, this leaves the server hanging. Similar to this attack is one Rouse (2013) calls network-centric attack. This attack overloads a service by using up bandwidth.

Another form of DDoS is through the use of DNS. Lambert (2013) mentioned that a lot of network providers have their DNS servers configured to allow anyone to launch queries, even people that are not their customers. The fact that DNS uses UDP, which is a stateless protocol, plus the configuration of some DNS servers make a potent way to create a denial of service. According to Lambert (2013), the attacker need only find open DNS resolvers, craft a fake UDP packet that has a spoofed address, the one of the target site, and send it to the DNS server.

While the request would be from the attacker, the server would think that the request came from the server, and as a result the server will send the reply to that location. So instead of having the actual *botmaster* conduct the attack, the only thing the target system will see is a bunch of DNS replies coming from many open resolvers, all around the internet (Lambert,

2012). This is a very scalable type of attack, according to Lambert (2013), because the attacker can send a single UDP packet to a DNS server asking for a full dump of certain domain, and receive a very large reply.

Recently, Google Ideas (2013) announced a project called Digital Attack Map, which is a fascinating, interactive map that monitors DDoS attacks around the world. According to Groll (2013), Google hope that this effort will raise awareness about the problems associated with DDoS. The Digital Attack Map draws on data collected by the network security firm Arbor Networks.

The result of this is a visualisation of what cyber-war looks like in real time. For example, Google Ideas (2013) provides a snapshot of the Digital Attack Map on the 27th of August 2013, when a portion of Chinese *.cn* domains were knocked offline.

According to Groll (2013), the event on the 27th of August was described by the Chinese authorities as the largest cyber-attack in the country's history. Although the Chinese authorities did not point any fingers at any particular party, a snapshot of the Digital Attack Map shows where the attacks originated.

On the Digital Attack Map, the attacks whose origin and destination are both known are depicted as an arc between the two countries, with the data travelling from source to victim (Groll, 2013). Attacks whose origins are unknown but whose victims are clear are depicted on the map as downward flow into the victim country (Groll, 2013).

From the Digital Attack Map, the attack that took out the *.cn* domain came from both the United States and the Netherlands. However, Groll (2013) mentioned that there are several ways for attackers to obscure their location and make it appear as if attacks are originating in different countries.

Another interesting event was depicted on the Digital Attack Map on the 25th of June 2013, which marked the 63rd anniversary of the start of the Korean War. On this day, South Korea was struck by a cyber-attack by the DarkSeoul gang (Groll, 2013). According to Groll (2013), the DarkSeoul gang has been linked to North Korea and is believed to work on its behalf. The attack by the DarkSeoul took down major media and government websites. This attack represented a high-profile flare-up in ongoing tensions in the Korean Peninsula (Groll, 2013).

The most striking thing about the attack that targeted South Korea is that it was was able to take down a series of prominent websites while using relatively little bandwidth (Groll, 2013).

The Digital Attack Map also captured part of a massive six-day attack on the United States (Groll, 2013). During this attack hackers targeted US banks, among other things. This attack is notable for the incredible bandwidth used, which was far larger than that in a typical attack (Groll, 2013).

There are different forms of DDoS attacks. Therefore, it is important to consider these different forms when building a defence against DDoS attacks. According to Lambert (2012), the easiest, although costly, way of defence is to buy more bandwidth. DDoS is a game of capacity. For example if there are 10,000 systems sending 1 Mbps to one target system, that means that the target system is getting 10 Gb of data every second. Clearly this is a lot of

traffic. In this case, the same rules for normal redundancy apply. To achieve a good load balancing, multiple servers must be spread around various datacentres. This will help with the load and hopefully the pipes will be large enough to handle all the traffic (Lambert, 2012). However, modern DDoS attacks are getting large, and quite often can be much bigger than what finances will allow in terms of bandwidth. Therefore, the method of increasing bandwidth might not be an effective one (Lambert, 2012).

One critical piece of network that administrators can look to in order to mitigate DDoS attacks is the DNS server. According to Lambert (2012), it is not a good idea to leave the DNS server as an open resolver, and it should be locked down in order to save the organisation from being used as part of an attack. The question that arises, then, is that what if the DNS server came under attack? Lambert (2012) mentioned that if no one can connect to the DNS server of the organisation, then that is just as bad as a DDoS attack. Even though most domain registrations are done with two DNS servers, quite often that may still not be enough (Lambert, 2012).

According to Lambert (2012), the DNS should be protected behind the same type of load balancing as for web and other resources. Also, there are companies that provide redundant DNS that other companies can use. For example, many companies use content delivery networks to serve files to customers in a distributed way, which is a great way to also protect them against DDoS attacks.

For companies that serve their own data and manage their own networks, Lambert (2012) mentioned that there are a lot of things those companies may do to protect themselves at the network layer. Firstly, such companies should ensure that their routers drop junk packets, and they should set up good firewalls. For example, in cases where the services running on company systems are not going to be asking random DNS servers for queries, there is no reason to allow UDP port 53 packets heading to the company servers. Companies should block everything they can at their network border, where they have the largest pipe, or they should get their upstream providers to block them for them (Lambert, 2012).

Lambert (2012) mentioned that there is a lot of Internet providers that gives their customers the ability to have unwanted traffic blocked. In a similar way, there are many ways to protect company network from *Syn* attacks, by increasing TCP backlog, reducing the *Syn-Received* timer, or using *Syn* caches.

The other case to consider when building defence against DDoS attacks is the case in which the attack does reach the target system. For example, most modern services use many dynamic resources. Even though the actual bandwidth from an attack may be manageable, Lambert (2012) mentioned that too often the database end up failing, or the custom scripts running. A company can make use of caching servers in order to provide as much static content as possible. There should be a plan in place to quickly replace dynamic resources with static ones, in the event that a company comes under attack (Lambert, 2012).

Companies should also have detection systems in place. The worst thing for any company is for its network to go down. Companies should be able to be alerted as soon as an attack starts, and be ready to deal with it. According to Lambert (2012), the manner in which a DDoS is carried out, it would be incredibly difficult to halt a DDoS attack at the source. However,

setting up an infrastructure that is distributed, hardened, and secure is possible, and that is something companies should think about when setting up their networks (Lambert, 2012).

According to Gaffan (2012), regardless of the type of solution best suited to be effective against DDoS attacks, there are five "must-haves" for any DDoS mitigation checklist:

- Transparent mitigation. Gaffan (2012) mentioned that attackers count on users losing access to company services during an attack. Since users don not need to know and do not care that a company network is under attack, any mitigation technology must continue to let people access the service provided by the company without delay (Gaffan, 2012).
- Bots cannot talk, humans can. Attackers carry out DDoS attacks to cause a nuisance by inconveniencing users. According to Gaffan (2012), companies should give users a legitimate fail-safe outlet for complaining or addressing automated lockouts as this will make users appreciative company's thinking ahead of the attackers plot, and giving them outlet to report their experience. Furthermore, Gaffan (2012) mentioned that this outlet would provide companies with further insight into the performance of anti-DDoS system.
- Make sure you whack all the bots. According to Gaffin (2012), most sites have very little headroom to such an extent that even 50 excess page views per second can slow down or take down a site. Companies should make sure their screening is airtight, blocking all application layer bot requests (Gaffan, 2012). Furthermore, Gaffin (2012) mentioned that this should not come at the expense of blocking the good bots such as Google, Bing and all other benevolent Internet bots that should be granted access all times.
- Expect the biggest tidal wave. Gaffin (2012) mentioned that network attacks are getting bigger and amplification techniques are getting more widely used. According to Gaffin (2012), network DDoS is less about brute force and more about preparing a database of open DNS servers, or SNMP servers with open "public" communities.
- Without accurate detection, it will be too late. According to Gaffin (2012), there are two parts to DDoS protection: the first is detecting a site under attack and the second is applying an effective defence. Gaffin (2012) mentioned that detection often gets overlooked due to its tricky nature. Companies should ensure that their solutions are capable to accurately detect attacks but remain inactive when they are not under attack (Gaffan, 2012). According to Gaffin (2012), defensive measures are just as bad as no defence measures at all.

Understanding the tactics of attackers can help network administrators to gauge how to economise and optimise their forces against an attacker's efforts (Gaffan, 2012).

7.1.2 SQL Injection

Mackay (2005) defines SQL injection attack as a form of attack that comes from user input that has not been checked to see that it is valid. According to Mackay (2005), the objective of such an attack is to fool the database system into running malicious code that will reveal sensitive information or otherwise compromise the server. The specially crafted user data tricks the application into executing unintended commands or changing data.

By launching an SQL injection attack, an attacker can create, read, update, alter, or delete data stored in the back-end database. In its most common form, an SQL injection attack gives access to sensitive information such as identity numbers, credit card numbers or other financial data.

Mackay (2005) mentioned that there are two main types of SQL injection attacks. First-order attacks are when the attacker receives the desired result immediately, either by direct response from the application they are interacting with or some other response mechanism, such as email (Mackay, 2005). Second-order attacks occurs when the attacker injects some data that will reside in the database, but the payload will not be immediately activated.

Regardless of the order of the attack, Glynn (2013) mentioned that the key concepts of an SQL injection attack are:

- SQL injection is a software vulnerability that occurs when data entered by users is sent to the SQL interpreter as a part of an SQL query
- Attackers provide specially crafted input data to the SQL interpreter and trick the interpreter to execute unintended commands
- Attackers utilise this vulnerability by providing specially crafted input data to the SQL interpreter in such a manner that the interpreter is not able to distinguish between the intended commands and the attacker's specially crafted data. The interpreter is tricked into executing unintended commands
- An SQL injection attack exploits security vulnerabilities at the database layer. By exploiting the SQL injection flaw, attackers can create, read, modify, or delete sensitive data

According to Imperva (2013), the most common way of detecting SQL injection attacks is by looking for SQL signatures in the incoming HTTP stream. For example, looking for SQL commands such as UNION, SELECT or *xp_*. Imperva (2013) mentioned that the problem with this approach is the very high rate of false positives. Because most SQL commands are legitimate words that could normally appear in the incoming HTTP stream, this might cause the user to either disable or ignore any SQL alert reported. According to Imperva (2013), in order to overcome this problem to some extent, the product must learn where it should and should not expect SQL signatures to appear. Imperva (20130 mentioned that the ability to discern parameter values from the entire HTTP request and the ability to handle various encoding scenarios are a must in this case.

SQL Injection can be effective mitigated through the following (Glynn, 2013):

- Administrators should adopt an input validation technique in which user input is authenticated against a set of defined rules for length, type, and syntax and also against business rules
- Administrators used ensure that users with the permission to access the database have the least privileges. According to Glynn (2013), the use of system administrator accounts such as *sa* for Web applications should be avoided. Furthermore, administrators should always make sure that a database user is created only for a specific application and this user is not able to access other applications. In addition to this, Glynn (2013) mentioned that all stored procedures that are not in use should be removed
- Administrators should use strongly typed parameterised query APIs with placeholder substitution markers, even when calling stored procedures
- Administrators should show care when using stored procedures since they are generally safe from injection. According to Glynn (2013), special care is needed because stored procedures can be injectable (such as via the use of *exec()* or concatenating arguments within the stored procedure).

The basic principles to prevent an SQL injection are similar, even though the exact code differs depending on the programming language used.

7.1.3 DNS Server Cache Snooping Remote Information Disclosure

The Domain Name System (DNS) can be regarded as a distributed and fault-tolerant database that contains, among other things, information on domains and hostnames and how they relate to the IP addresses that are assigned to the various computer systems that compose the Internet (Grangeia, 2004).

The DNS works like a directory-based service. According to Grangeia (2004), there are two main user approaches to the DNS system: the Publisher, who wants to make their information available to others to look up; and the Browser who queries the system for information for their personal use. This relationship is transposed to the DNS architecture in orderto separate functionality. As a result, sometimes a DNS system will act as a cache and store recently queried information to optimize further local queries, and on other occasions the system will serve information about hostnames and/or IP's (Grangeia, 2004).

Grangeia (2004) mentioned that the need to separate "DNS servers" and "DNS caches" has been the source of controversy and discussion in the literature. There is a camp that argues that caches and servers should be run as independent services on different IP addresses, while the other camp maintain the opposite can also be secure. This confusion is escalated by the fact that most programs that implement DNS functionality bundle both functions in one software package.

Grangeia (2004) argued that a logical separation is needed between DNS servers and DNS caches. This, according to Grangeia (2004), can greatly improve the security of the DNS infrastructure and prevent vulnerabilities such as DNS cache snooping.

According to JH Software (2013), DNS cache snooping is when someone queries a DNS server in order to find out (snoop) if the DNS server has a specific DNS record cached. By so doing, the person can be abled to deduce if the DNS server's owner or its users have recently visited a specific website. This process is possible even if the DNS server is not configured to resolve recursively for queries originating from other networks (JH Software, 2013).

There are different ways to snoop a DNS cache. The most effective way is using iterative queries (Grangeia, 2004). Using this way, an attacker asks the cache for a given resource record of any type. If the response is cached the response will be valid, else the cache will reply with information of another server that can better answer our query, or most commonly, send back the *root.hints* file contents (Grangeia, 2004).

The other way to snoop a DNS cache is through recursive queries. The major disadvantage of this method is that using recursive queries will pollute the cache (Grangeia, 2004). In other words, if a given record is not present in the cache, it will be after the first query is made (Grangeia, 2004).

There are several guidelines that are available that can significantly reduce the exposure to DNS cache snooping. Microsoft (2012) mentioned that there is no code fix for this vulnerability because this is a configuration choice. One of the available options is to leave recursion enabled if the DNS Server resides on a corporate network that cannot be reached by untrusted clients. The other option is to restrict public access to the DNS servers that perform recursion. By default, Microsoft DNS Servers are configured to allow recursion (Microsoft TechNet, 2005).

Alternatively, network administrators can alleviate DNS cache snooping by configuring the cache to only allow access by local users or child caches (Grangeia, 2004). According to Grangeia (2004), the principle of caching is that of locality, so it makes no sense to allow a user from a totally different network to access caches in other networks. To minimise the chances of DNS snooping, any DNS system (server/cache) should respond non-authoritatively to known clients only. The other step that administrators can take is to disallow non-authoritative requests to DNS caches (Grangeia, 2004). This option will eliminate the possibility to do non-recursive snooping.

Mitigating against DNS cache snooping is vital to administrators of SCADA systems that run on web services. As explained above, an attacker can use DNS cache snooping to find out which web site is visited frequently, and with this information an attacker can then launch a DDoS attack targeting that website, and in turn disrupt the functioning of the SCADA system.

7.2 Metasploit Vulnerabilities

7.2.1 Man-in-the-Middle Attack

Coates (2010) defines man-in-the-middle (MitM) attack as an attack where the communication exchange between two users is surreptitiously monitored and possibly modified by a third, unauthorised, party. In addition, this third party will be performing this attack in real time. This is to say, stealing logs or reviewing captured traffic at a later time would not qualify as a MitM (Coates, 2010).

According to Hargrave (2012), MitM attack makes use of a technique called Address Resolution Protocol (ARP) spoofing to trick the computer of the first user into thinking that it is communicating with the computer of the second user. This technique allows the network traffic between the two computers to flow through the attacker's system, which enables the attacker to inspect all the data that is sent between the victims (Hargrave , 2012). This kind of cyber-attack can be particularly effective at cafes and libraries that offer their patrons Wi-Fi access to the Internet. Hargrave (2012) mentioned that in such open networking environments, network traffic can be readily snatched due to the unencrypted networks.

A MitM attack can be performed in two different ways, according to Coates (2010):

- The attacker is in control of a router along the normal point of traffic communication between the victim and the server the victim is communicating with.
- The attacker is located on the same broadcast domain as the victim, or the attacker is located on the same broadcast domain as an of the routing devices used by the victim to route traffic

Coates (2010) mentioned that a MitM attack will exploit the weaknesses found in network communication protocols in order to convince a host that traffic should be routed through the attacker instead of through the normal router. What is happening here is that the attacker is advertising that they are the router and the client should update their routing records appropriately. This is, in essence, ARP spoofing. According to Coates (2010), the greatly simplified purpose of ARP is to enable IP address to MAC address translations for hosts. This is a requirement for facilitating the movement of packets from one host to another (Coates, 2010).

The design of ARP is such that there is no authentication. As a result, any host can reply to an ARP request or send an unsolicited ARP response to a specific host. Coates (2010) mentioned that these ARP messages are used by the attacker to instruct the victim's machine that the appropriate MAC address for a given IP address is now the MAC address of the attacker's machine. To be more specific, the attacker is instructing the victim to overwrite their ARP cache for the IP to MAC entry for the router (Coates, 2010). As a result, the IP address for the router will correspond to the MAC address for the attacker's machine.

By overwriting the IP to MAC entry for the router, all of the victim's traffic will be routed through the attacker's system. Now, in order to allow the traffic to reach the Internet, the

attacker then configures his system to also forward this traffic to the original router. Coates (2010) mentioned that in addition to this, the attacker performs a similar ARP spoofing attack against the router. This is how the attacker can know to reroute traffic that was destined for the victim, to the attacker's system instead. The attacker then forwards on the traffic to the victim. This completes the whole process and it places the attacker "in the middle" of the communication (Coates, 2010).

According to Coates (2010), it is at this point that the attacker has the ability to view and modify any TCP traffic sent to or from the victim machine. Because HTTP traffic is unencrypted and contains no authentication, it can be trivially monitored/modified by the attacker (Coates, 2010).

Even though it is trivial to deal with HTTP, more devious means are needed to perform a MitM against SSL/TLS. Coates (2010) mentioned that the attacker can attempt to intercept HTTPS traffic by using a custom certificate. However, this would present a certificate warning message in the user's browser and likely alert the user to the attack. According to Coates (2010), most users would ignore the warning and continue, thus exposing all of their data.

According to Hildayatullah (2010), it can be very difficult to detect a MitM attack. In such cases, it is better to prevent MitM since there are few methods to detect these attacks.

There are reported ways to prevent MitM attack. Hargrave (2012) mentioned that in practice, ARP spoofing is difficult to prevent with the conventional security tools that come with standard computers. However, users can make it difficult for people to view network traffic by using encrypted network connections provided by HTTPS or VPN technology. The use of VPN creates additional secure layers when users access their company's confidential networks over links like Wi-Fi (Hildayatullah, 2010)

Hildayatullah (2010) mentioned that internal MitM attack can be avoided by setting up an intrusion detection system (IDS). According to Hildayatullah (2010), the purpose of the IDS will be to monitor the company network. In cases where someone tries to hijack traffic flow, the IDS will give immediate alerts (Hildayatullah, 2010). However, the downside of IDS is that it may raise false attack alerts many a times. And as a result, users end up disabling it.

According to Hildayatullah (2010), tools which use the advanced address resolution protocol and measures such as implementing dynamic host configuration protocol (DHCP) snooping on switches can limit or prevent ARP spoofing. This will in turn help prevent MitM attacks. Additionally, companies should have proper auditing and monitoring in place so that they can be aware of their staff's activities (Hildayatullah, 2010).

8 Conclusion and Future Work

8.1 Conclusion

SCADA system security is an area of growing interest due to the security threats faced by SCADA systems. A research conducted by the Idaho National Laboratory (2011) on the security of SCADA systems revealed common SCADA vulnerabilities that are faced by all SCADA even though functions, designs, and configurations vary among vendors, versions, and installations. The research presented in this paper utilised penetration testing to investigate common SCADA vulnerabilities.

The experiments were conducted under two different scenarios. In one scenario, the attacker launched the attack at the host computer from within the company network, and in the other scenario, the attacker was outside the company network. There were no results from the second scenario due to the security of the University network. The results from the experiments under the first scenario show that there are common vulnerabilities among different SCADA system.

Even though there were no new vulnerabilities discovered during the experiments, this study demonstrated that the statement put forward by the Idaho National Laboratory can be verified. Also, the study demonstrated the ability of penetration testing tools to discover vulnerabilities. The tools used in the research found different vulnerabilities. This highlighted the differences between the tools and further supported the idea of utilising multiple tools for this research. This can imply that the use of more tools will reveal new vulnerabilities and some of those might be among the ones classified by the Idaho National Laboratory.

8.2 Future Work

This research sought to investigate common SCADA vulnerabilities under different network topologies. However, due to the security of the campus network, it was not possible to launch an attack from outside the university network. As part of future work, a network simulating tool, such as OMNeT++, can be used to simulate different network topologies.

Also as part of future research, more penetration testing tools can be used in order to give more credibility to the results of this research.

References

Antunes, N. & Vieira, M., 2009. Comparing the Effectiveness of Penetration Testing and Static Code Analysis on the Detection of SQL Injection Vulnerabilities in Web Services. In: *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2009, Shanghai, China, 16-18 November 2009.* Shanghai: IEEE Computer Society, pp. 301-306.

Bently Systems, Inc, 2004. *The Fundamentals of SCADA.* [Online]
Available at: ftp://ftp2.bentley.com/dist/collateral/whitepaper/fundscada_whitepaper.pdf
[Accessed 15 August 2013].

Byres, E. & Lowe, J., 2004. *The myths and facts behind cyber security risks for industrial control systems. Technical Report,* London: PA Consulting Group.

Chikuni, E. & Dondo, M., 2007. *Investigating the security of electrical power systems SCADA.* AFRICON 2007: Windhoek, Institute of Electrical and Electronics Engineers, pp. 1-7.

Chunlei, W., Lan, F. & Yiqi, D., 2010. A Simulation Environment for SCADA Security Analysis and Assessment. *Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on,* Volume 1, pp. 342-347.

Coates, M., 2010. *Man In The Middle Attack - Explained.* [Online]
Available at: http://michael-coates.blogspot.com/2010/03/man-in-middle-attack-explained.html
[Accessed 2 October 2013].

Communication Technologies, Inc, 2004. *Supervisory Control and Data Acquisition (SCADA) Systems.* [Online]
Available at: www.comtechnologies.com
[Accessed 12 August 2013].

Daneels, A. & Salter, W., 1999. What is SCADA?. *International Conference on Accelerator and Large Experimental Physics Physics Control Systems*.

Eltayeb, M. K. M., 2009. *Implementation of a Virtual Plant Using SCADA/HMI Technologies,* Khartoum: University of Khartoum.

Falliere, N., Murchu, L. O. & Chien, E., 2011. *W32.stuxnet dossier. Technical report,* Mountain View: Symantec Corporation.

Farwell, J. P. & Rohozinski, R., 2011. Stuxnet and the future of cyber war. *Survival,* 53(1), pp. 23-40.

Gaffan, M., 2012. *The 5 Essentials of DDoS Mitigation.* [Online]
Available at: http://www.wired.com/insights/2012/12/the-5-essentials-of-ddos-mitigation/
[Accessed 1 October 2013].

Glynn, F., 2013. *SQL Injection Tutorial: Learn About Injection Attacks, Vulnerabilities and How to Prevent SQL Injections.* [Online]
Available at: http://www.veracode.com/security/sql-injection
[Accessed 2 October 2013].

Google Ideas, 2013. *Digital Attack Map.* [Online]
Available at: http://www.digitalattackmap.com/
[Accessed 14 OCtober 2013].

Groll, E., 2013. *Mapped: What Global Cyberwar Looks Like in Real Time.* [Online]
Available at:
http://blog.foreignpolicy.com/posts/2013/10/22/map_what_global_cyberwar_looks_like_in_real_time_google_ddos_attacks
[Accessed 21 October 2013].

Halfond, W., Choudhary, S. & Orso, A., 2009. Penetration Testing with Improved Input Vector Identification. In: *International Conference on Software Testing Verification and Validation, 2009. ICST '09..* Denver, CO: Institute of Electrical and Electronics Engineers, pp. 346-355.

Hargrave , V., 2012. *What Are Man-in-the-Middle Attacks and How Can I Protect Myself From Them?.* [Online]
Available at: http://fearlessweb.trendmicro.com/2012/tips-and-tricks/what-are-man-in-the-middle-attacks-and-how-can-i-protect-myself-from-them/
[Accessed 2 October 2013].

Hildayatullah, S., 2010. *Man in the middle attack prevention strategies.* [Online]
Available at: http://searchsecurity.techtarget.in/tip/Man-in-the-middle-attack-prevention-strategies
[Accessed 2 October 2013].

IBM, 2012. *Transport Layer Security (TLS) handshake renegotiation weak security (CVE-2009-3555) in relation to WebSphere Application Server products.* [Online]
Available at: http://www-01.ibm.com/support/docview.wss?uid=swg21413714
[Accessed 09 September 2013].

Idaho National Laboratory, 2011. *Vulnerability Analysis of Energy Delivery Control Systems.* [Online]
Available at: http://energy.gov/oe/downloads/vulnerability-analysis-energy-delivery-control-systems
[Accessed 12 August 2013].

Imperva, 2013. *SQL Injection.* [Online]
Available at: http://www.imperva.com/resources/glossary/sql_injection.html
[Accessed 2 October 2013].

Lambert, P., 2012. *DDoS attack methods and how to prevent or mitigate them.* [Online]
Available at: http://www.techrepublic.com/blog/it-security/ddos-attack-methods-and-how-to-prevent-or-mitigate-them/
[Accessed 30 September 2013].

Mackay, C. A., 2005. *SQL Injection Attacks and Some Tips on How to Prevent Them.* [Online]
Available at: http://www.codeproject.com/Articles/9378/SQL-Injection-Attacks-and-Some-Tips-on-How-to-Prev
[Accessed 1 October 2013].

Meserve, J., 2007. *Staged cyber attack revels vulnerability in power grid.* [Online]
Available at: http://edition.cnn.com/2007/US/09/26/power.at.risk/index.html
[Accessed 13 April 2013].

Microsoft, 2010. *Microsoft Security Advisory: Vulnerability in TLS/SSL could allow spoofing.* [Online]
Available at: http://support.microsoft.com/kb/977377
[Accessed 09 September 2013].

NMAP.ORG, 2009. *Nmap Security Scanner.* [Online]
Available at: http://nmap.org/
[Accessed 25 August 2013].

Northcutt, S. et al., 2006. Penetration Testing: Assessing Your Overall Security Before Attackers Do. *SANS Analyst Program*, June.

Pillai, B., Vishal, M. & Patel, N., 2012. Development of Supervisory Control and Data Acquisition system for Laboratory Based Mini Thermal Power Plant using LabView. *International Journal of Emerging Technology and Advanced Engineering,* 2(5).

Queiroz, C. et al., 2009. Building a SCADA Security Testbed. In: *Third International Conference on Network and System Security, 2009. NSS '09..* Gold Coast, QLD: Institute of Electrical and Electronics Engineers, pp. 357-364.

RAPID7, 2013. *Penetration Testing Software: Metasploit.* [Online]
Available at: http://www.rapid7.com/products/metasploit/editions-and-features.jsp
[Accessed 25 August 2013].

Rouse, M., 2007. *Man in the Middle Attack (Fire Brigade Attack).* [Online]
Available at: http://searchsecurity.techtarget.com/definition/man-in-the-middle-attack
[Accessed 09 September 2013].

Rouse, M., 2013. *distributed denial-of-service attack (DDoS).* [Online]
Available at: http://searchsecurity.techtarget.com/definition/distributed-denial-of-service-attack
[Accessed 30 September 2013].

Scheider Electric, 2012. *IGSS FREE50.* [Online]
Available at: http://igss.schneider-electric.com/products/igss/download/free-scada.aspx
[Accessed 24 August 2013].

Tenable Network Security, 2011. *Apache HTTP Server Byte Range DoS.* [Online]
Available at: http://www.tenable.com/plugins/index.php?view=single&id=55976
[Accessed 9 September 2013].

Tenable Network Security, 2013. *Nessus Report.* Columbia: Tenable Network Security.

Tenable Network Security, 2013. *Nessus Vulnerability Scanner.* [Online]
Available at: http://www.tenable.com/products/nessus
[Accessed 25 August 2013].

Wang Chunlei, Fang Lan & Dai Yiqi, 2010. A Simulation Environment for SCADA Security Analysis and Assessment. In: *International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 2010.* Washington, DC: IEEE Computer Society, pp. 342-347.

Zetter, K., 2013. *Researchers Uncover Holes That Open Power Stations to Hacking.* [Online]
Available at: http://www.wired.com/threatlevel/2013/10/ics/
[Accessed 17 October 2013].

Appendix A

This appendix provides an overview of Nmap and how it was utilised in this research. Screenshots showing the commands used and the results obtained are presented here.

Overview

Nmap stands for “Network Mapper” and it is a command line tool for network discovery and security auditing. Nmap is a free, open source and multiplatform tool. It can run on Windows, Mac OS X and Linux. For this research, Nmap was used together with another program called Zenmap – a special client which provides a visual interface for Nmap.

How Nmap works is that the user specifies the IP address of the computer or device they want to scan. If a user knows the Computer name, they can use that instead of the IP. Nmap provides a number of scans that users can select. These range from intense scan to slow comprehensive scan.

The results from a scan are displayed in the Nmap output tab. Figure 7 shows the results tab.

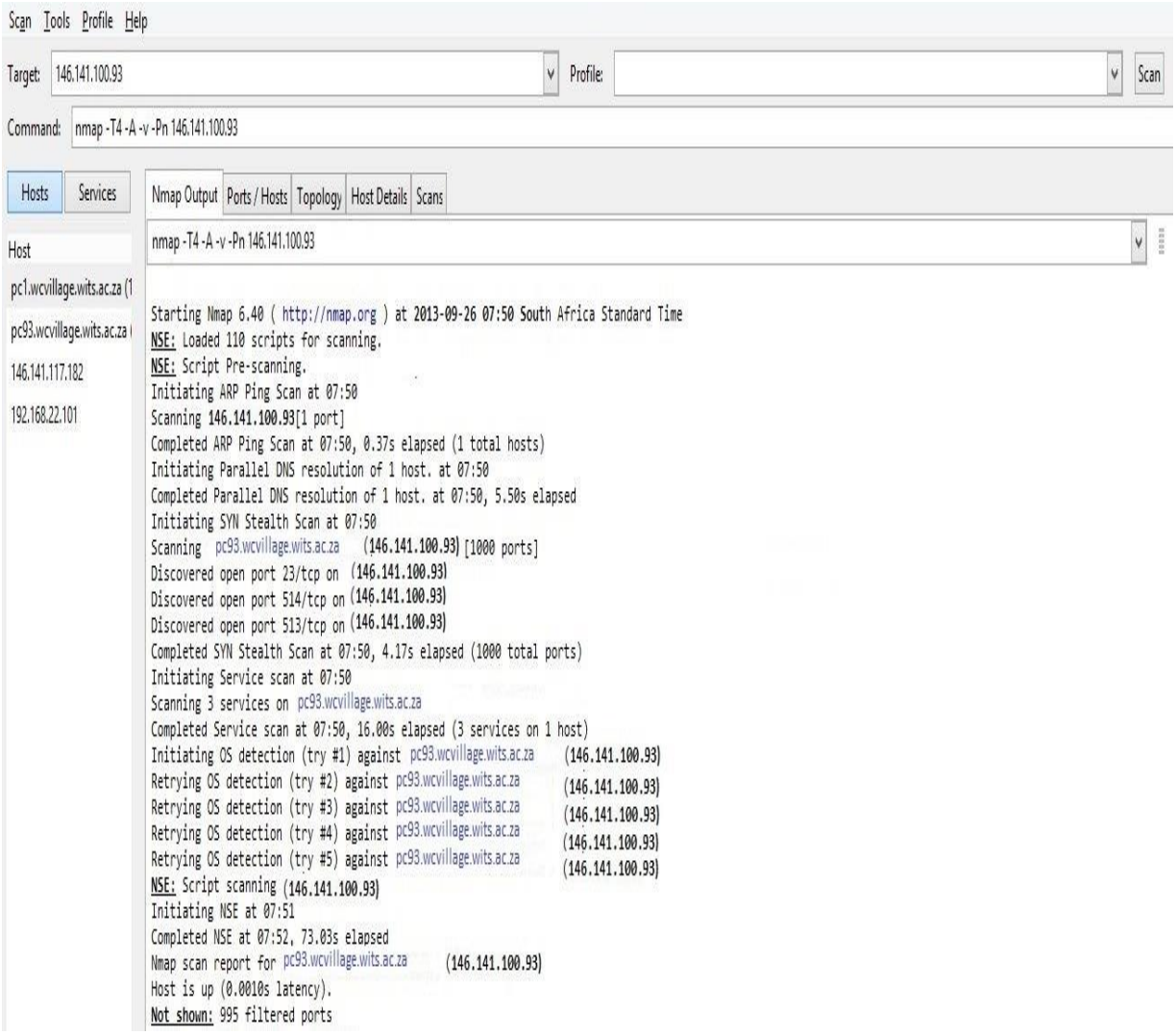


Figure 7. Nmap Output

The Ports/Hosts tab lists all the ports that were found open, as shown in figure 8

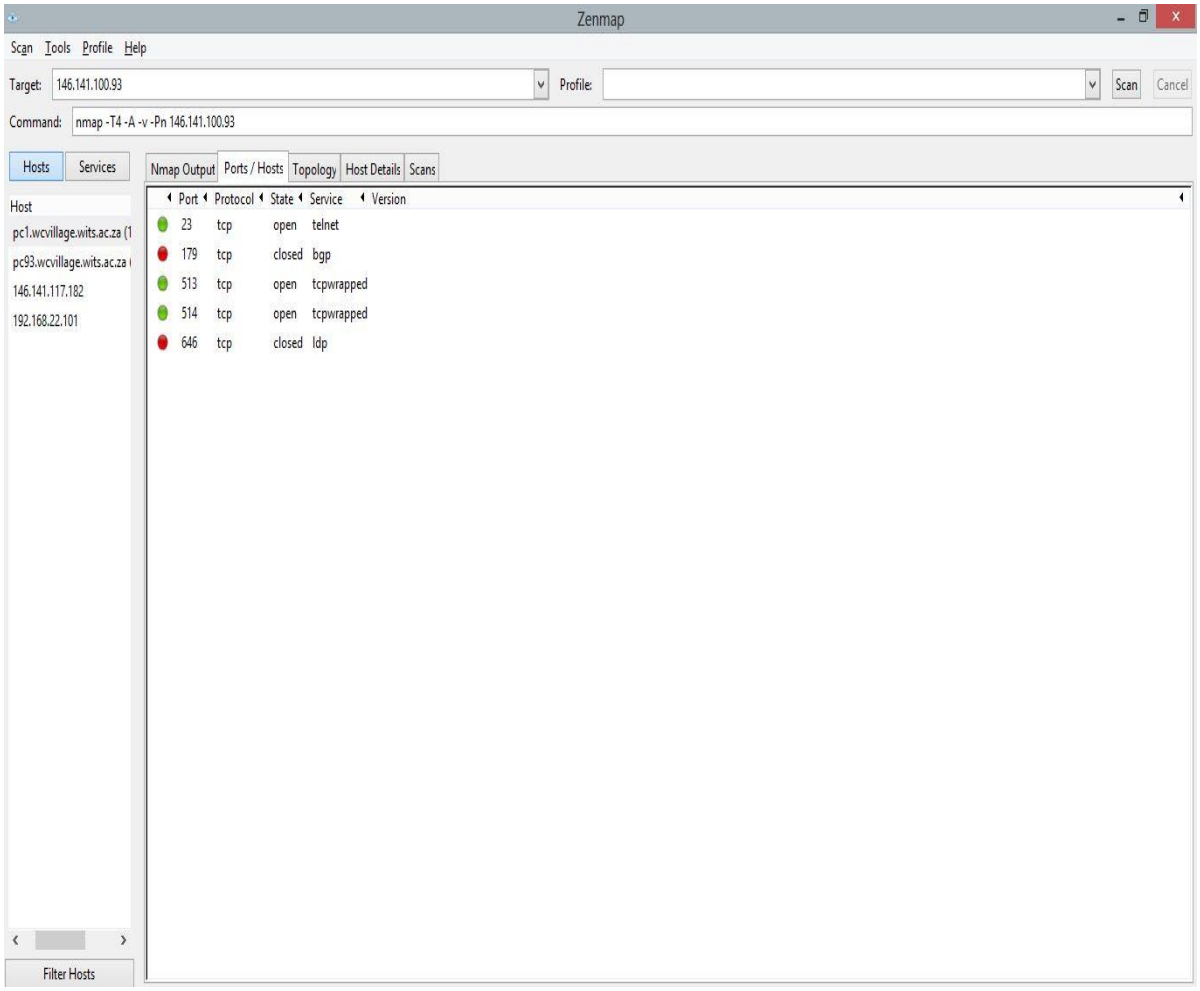


Figure 8. Open Ports/Hosts

The topology tab shares a visual overview of how the scanned computer/device is positioned in the network, relative to the computer from where the scan was made. This is shown in figure 9.

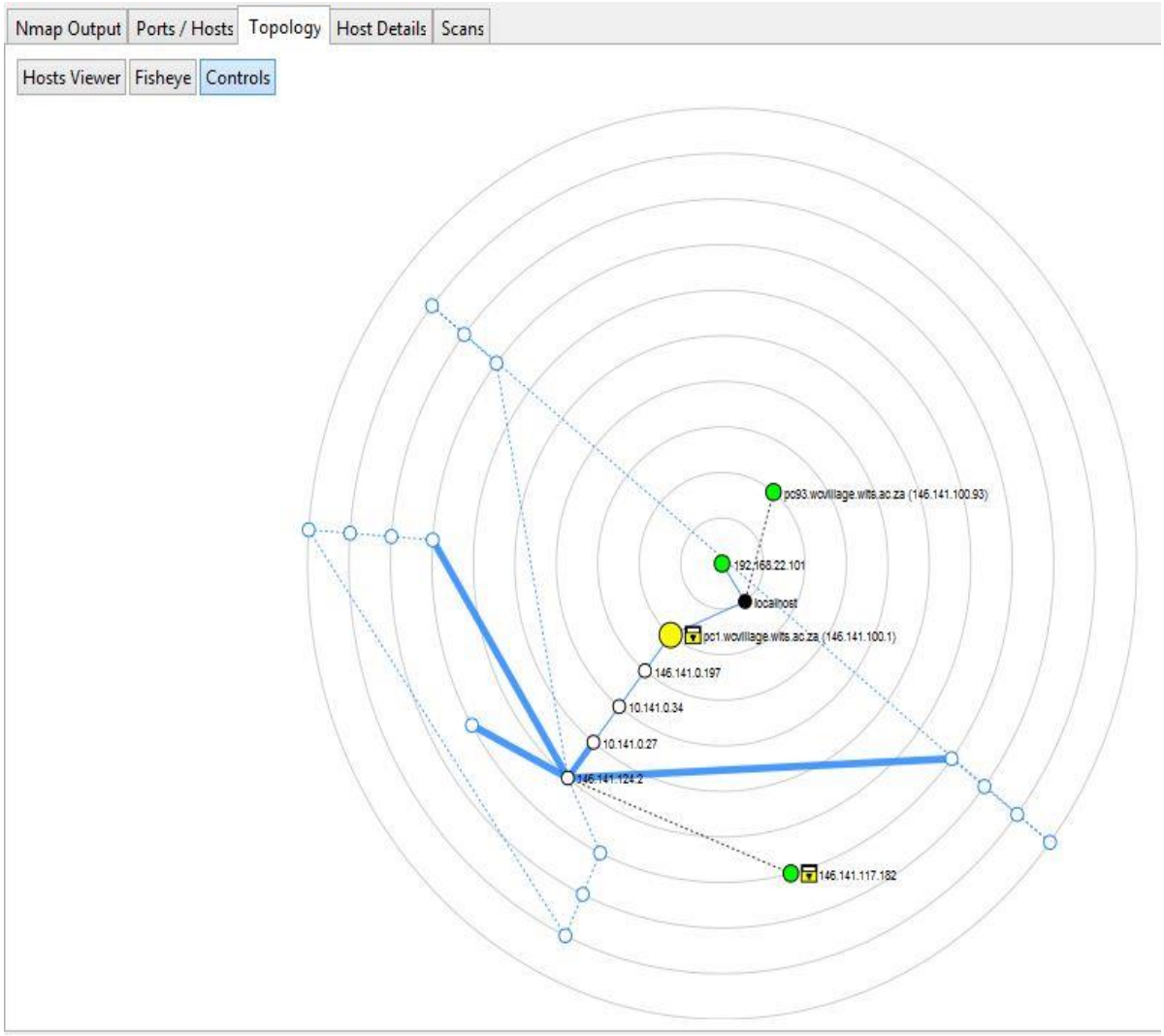


Figure 9. Network Topology

The above figures were obtained during the experiments of this research. The IP address of the computer running the SCADA system was (146.141.100.93). In figure 9, “local host” depicts the computer from which Nmap was ran.

Appendix B

This appendix provides an overview of Nessus and how it was utilised in this research. Screenshots showing the commands used and the results obtained are presented here.

Overview

Nessus uses a Web Interface to set up, scan and view reports. It uses different policies to scan target computers. The user can select a policy based on the objectives set by the user. Nessus comes with 4 predefined policies, and it offers user the ability to define and add their own policies.

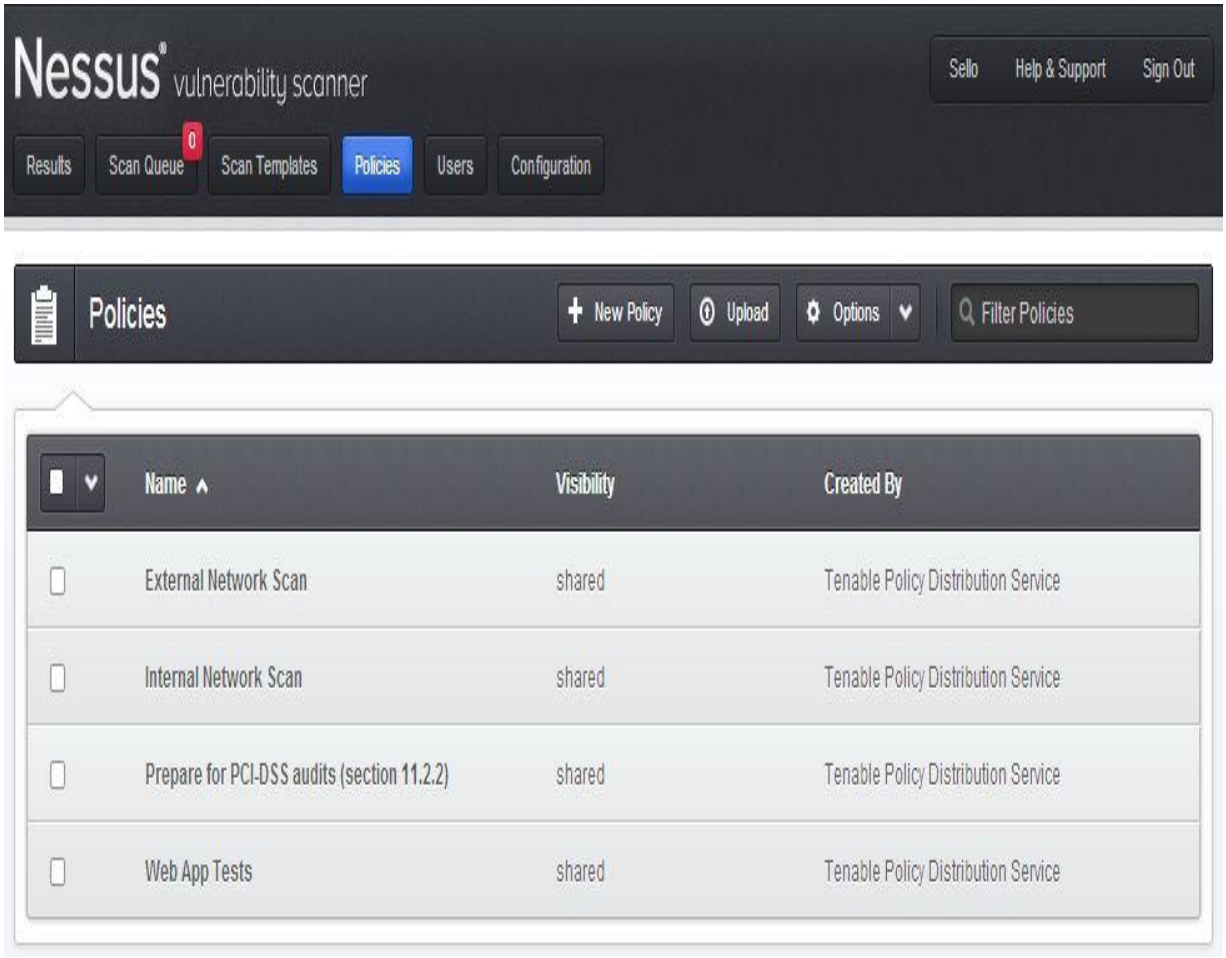


Figure 10. Nessus Policies

The External Network Scan policy is tuned to scan externally facing hosts, which typically present fewer services to the network (Tenable Network Security, 2013). The Internal Network Scan policy is tuned for better performance, taking into account that it may be used to scan large internal networks with many hosts, several exposed services, and embedded systems such as printers (Tenable Network Security, 2013). According to Tenable Network

Security (2013), the Prepare for PCI DSS audits policy enables the built-in PCI standards and produces a report on the user's compliance posture.

However, Tenable Network Security noted that a successful compliance scan does not guarantee compliance or a secure infrastructure. Users preparing for a PCI DSS assessment can use this policy to prepare their network and systems for PCI DSS compliance. The Web App Tests policy is used if a user want to scan their systems and have Nessus detect both known and unknown vulnerabilities with their web applications. According to Tenable Network Security (2013), the fuzzing capabilities in Nessus are enabled in this policy, which will cause Nessus to spider all discovered web sites and then look for vulnerabilities present in each of the parameters, including XSS, SQL, command injection and several more.

For this research, two policies were utilised; the External Network Scan and the Internal Network Scan. Figures 11 and 12 shows the setup for the two scans under the two policies.

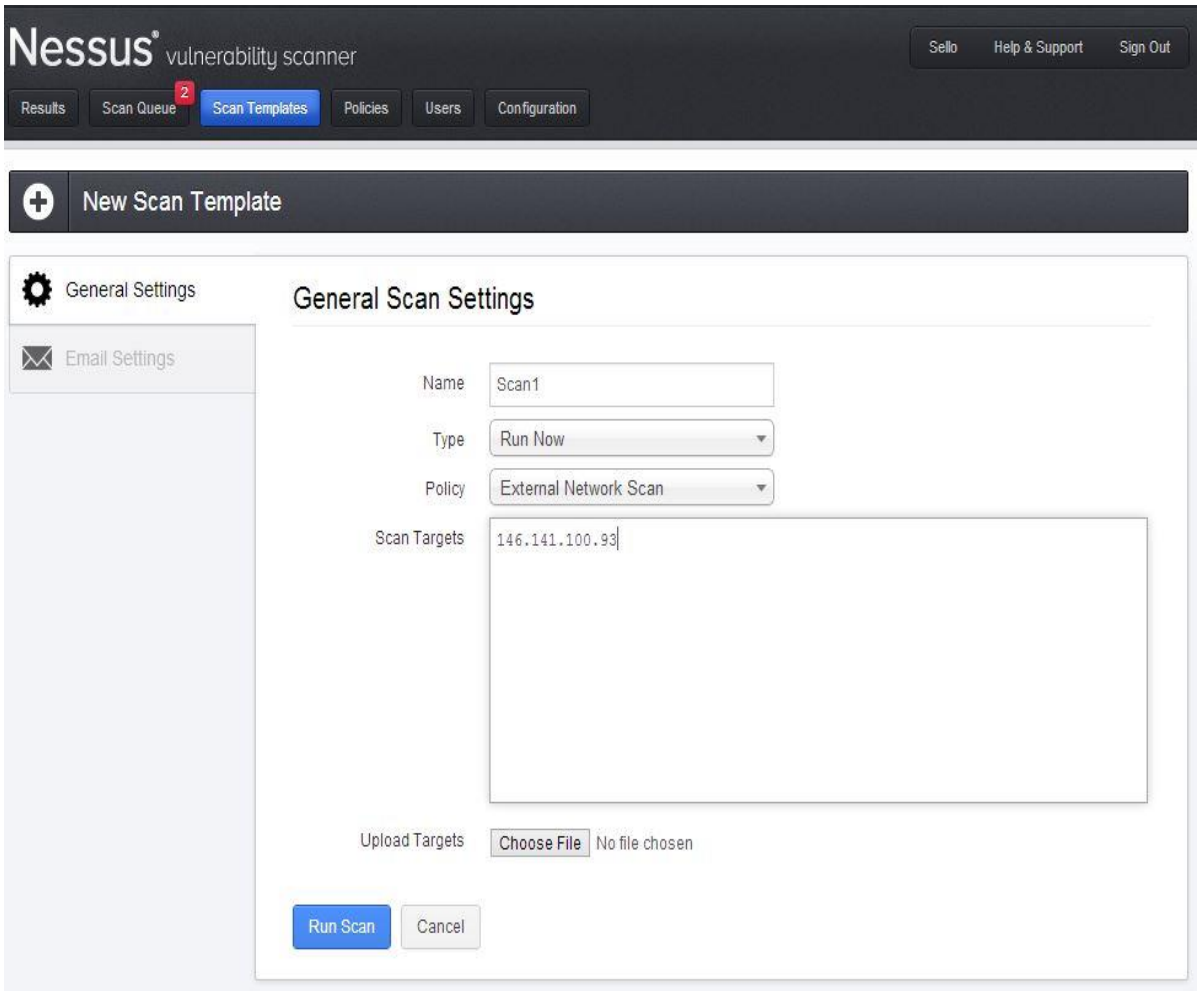


Figure 11. External Network Scan Policy setting

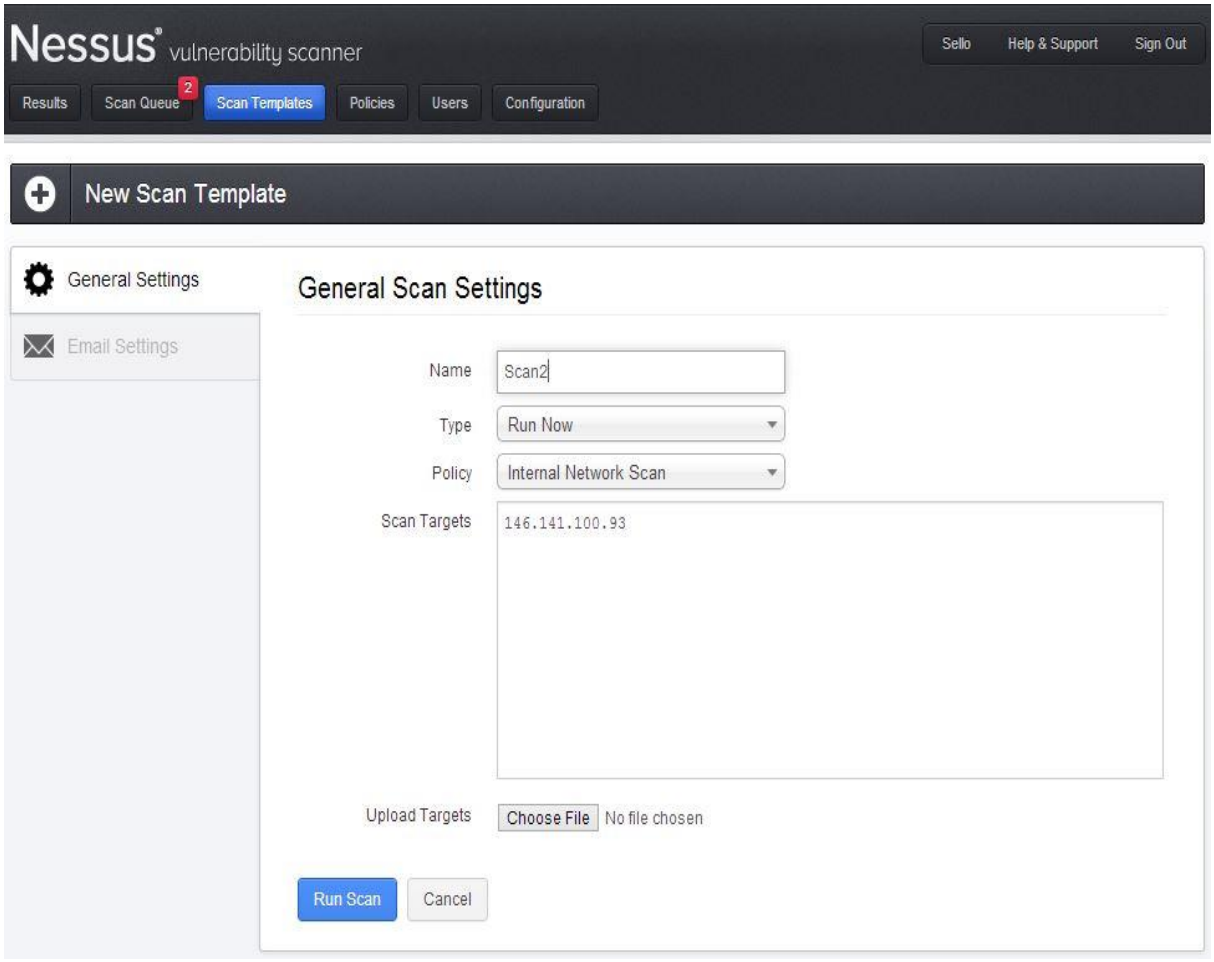


Figure 12. Internal Network Scan Policy Setting

The two scans revealed the same vulnerabilities. Figure 13 shows the summary from one of the scans.

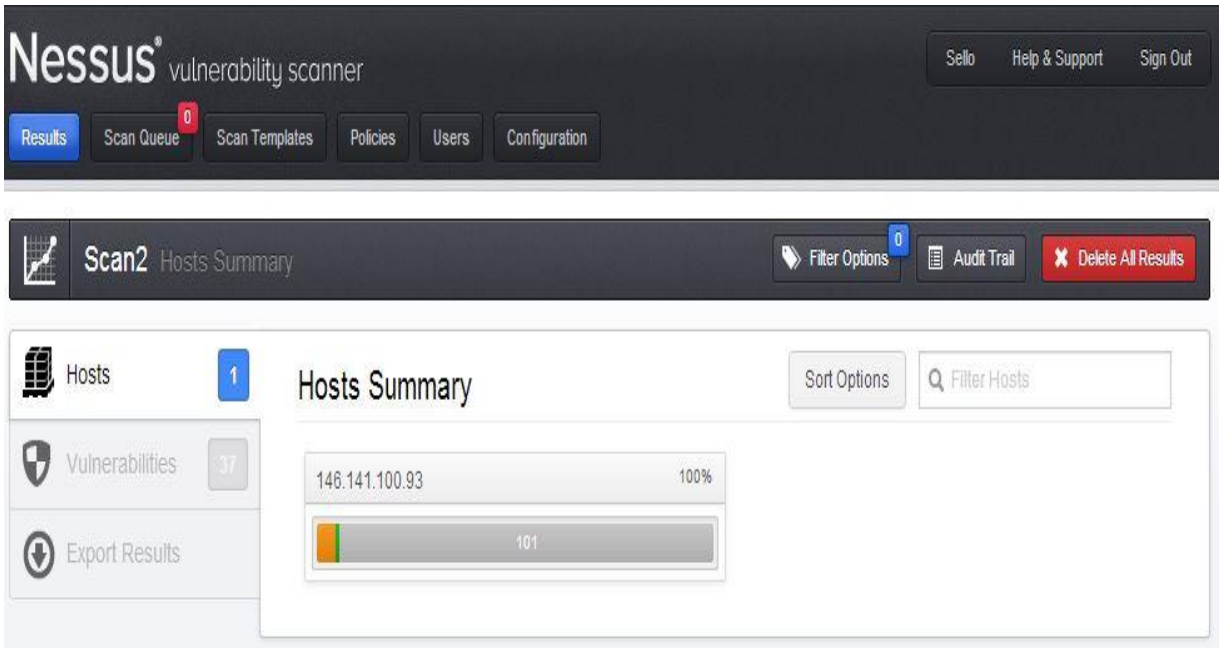


Figure 13. Scan Summary

Appendix C

This appendix provides an overview of Metasploit and how it was utilised in this research. Screenshots showing the commands used and the results obtained are presented here.

Overview

Metasploit, like Nessus, utilises a Web Interface to set up, scan and view reports. Metasploit provides users with 3 default functions: Quick PenTest, Phishing Campaign and Web App Test. As shown in figure 14, users can create new projects from scratch and define their own functions.

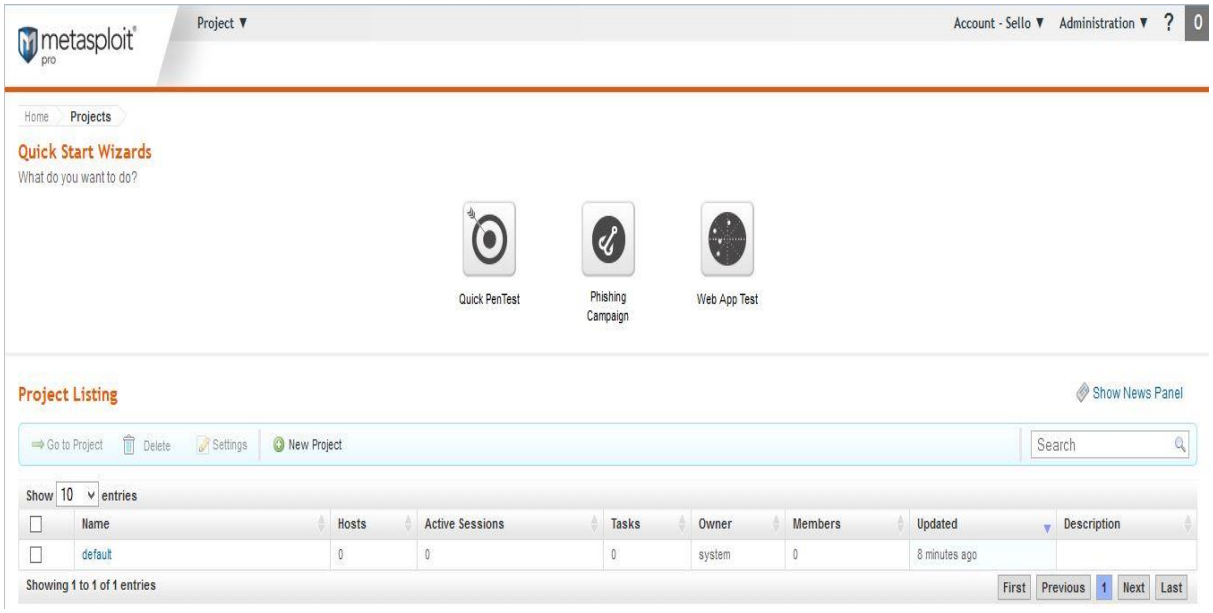


Figure 14. Metasploit Main Window

The Quick PenTest wizard is a guided interface that helps users configure the most common tasks associated with penetration test, such as scanning, exploiting and reporting (RAPID7, 2013).The goal of the Quick PenTest Wizard is to provide users with an easy way to create and launch a penetration test with very little configuration. According to RAPID7 (2013), users can launch the test immediately after they have provided a project name and the target addresses. With the Quick PenTest wizard, users are able to configure target settings, scan settings, auto-exploitation, and report generation options. Figure 15 shows the settings used for a Quick PenTest wizard for this research.

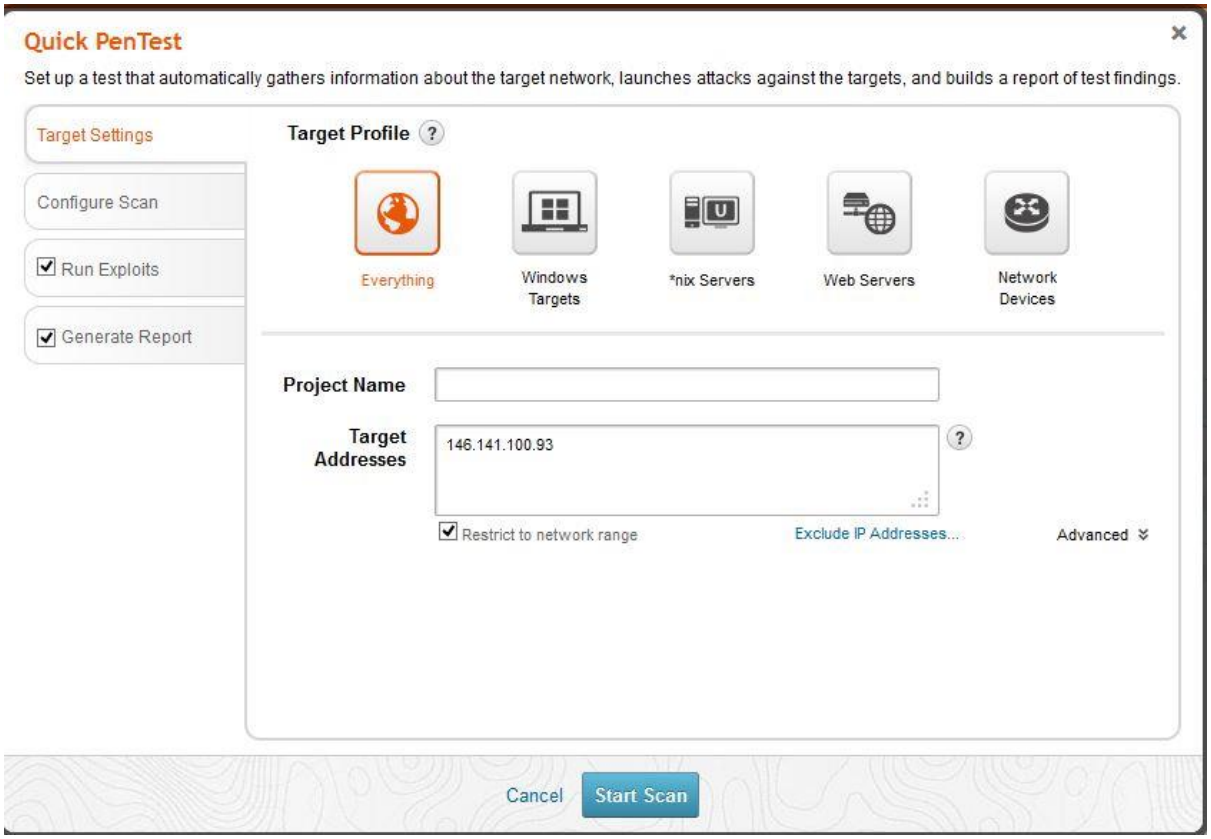


Figure 15. Quick PenTest Settings

As stated above, Metasploit Pro provides a canned phishing campaign that can be used to create a phishing attack. The phishing campaign contains all the components needed to set up a phishing attack as well as many default, canned settings that can be used to quickly get up and running (RAPID7, 2013). However, phishing is outside the scope of this research. Also, Web App Test was not explored because such test would be outside the scope of this research.

One other thing to note about the initial steps of a Quick PenTest Wizard is that Nmap is called first, to perform reconnaissance on the target computer. Metasploit has Nmap as one of its internal functions. Figure 16 shows the initial stages of a Quick PenTest

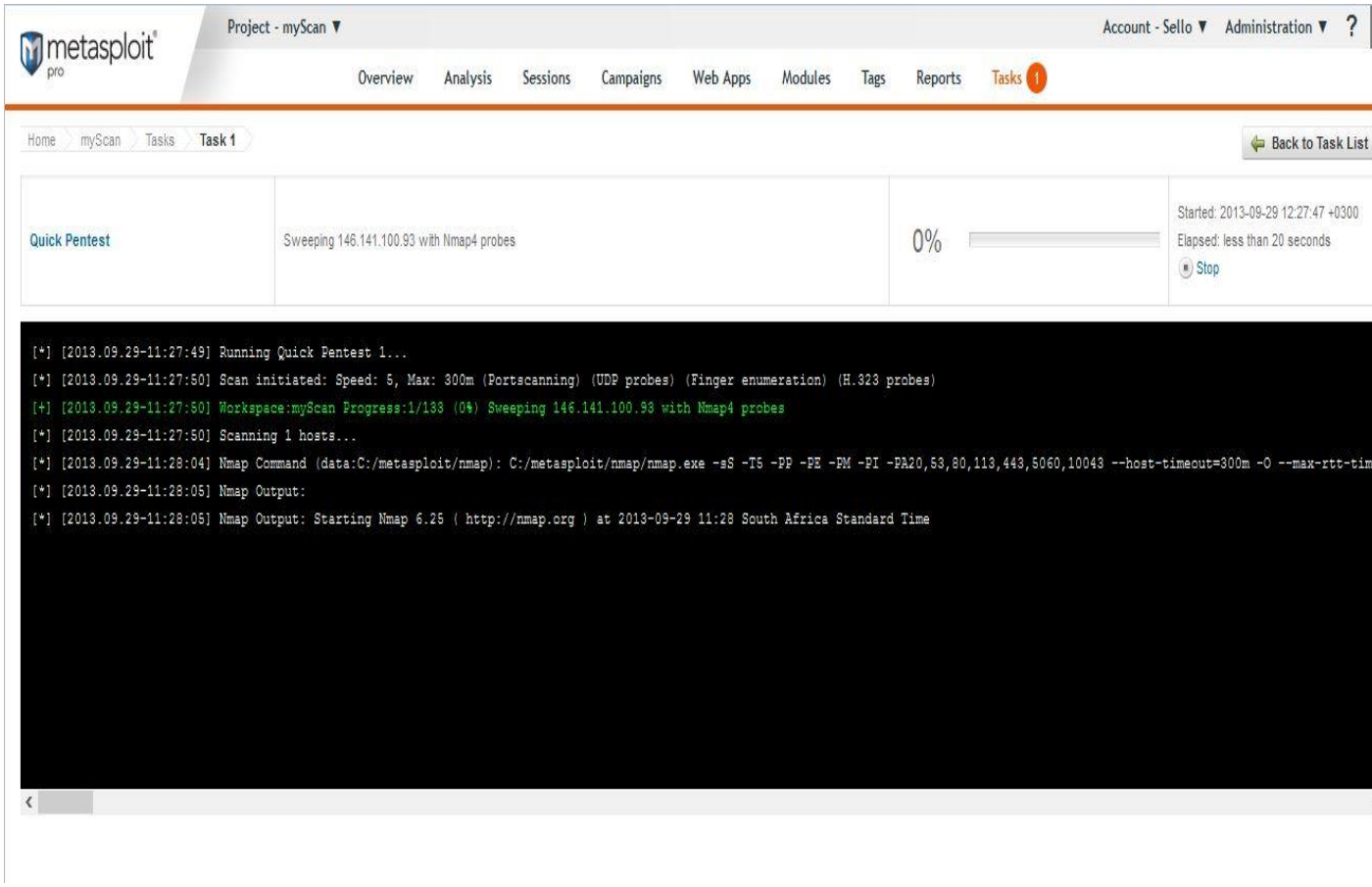
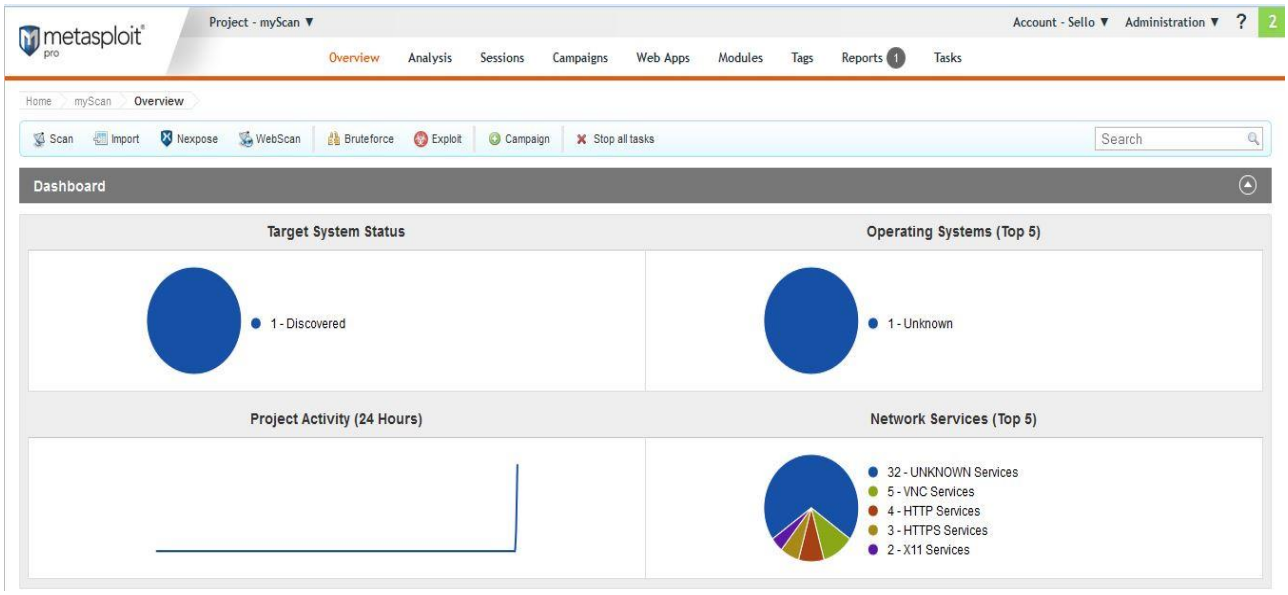


Figure 16. Nmap Output within Metasploit

Upon completion of a scan, Metasploit provides users with a dashboard that displays the summary of the scan. Figure 17 shows the dashboard after a Quick PenTest scan



The results show that the operating system could not be determined. This is in line with the earlier observations under Nmap and the same reasoning applies here.