University of BRISTOL

DEPARTMENT OF COMPUTER SCIENCE

Investigating the Viability of Fingerprinting the Toolset Used To Probe an Operational Technology Network, Providing Another Indicative Vector to Use in Intrusion Detection Systems.

Zack Dove

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Bachelor of Science in the Faculty of Engineering.

Friday 18th December, 2020

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of BSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Zack Dove, Friday 18th December, 2020

Abstract

In this work we investigated the viability of fingerprinting asset discovery tools used to probe operational technology networks, in order to provide an additional vector to use in intrusion detection systems. We collected packet captures from several scanning tools used on a real PLC (Programmable Logic Controller), and then extracted different groups of features from each packet capture, using this to build several different classifiers. We evaluated these classifiers on test data with additional noise added, achieving an exact match accuracy of 92% - 100%, showing that it is indeed possible to fingerprint the toolset used to a high degree of accuracy. We also discussed the viability of this technique being used within an IDS (intrusion detection system).

This project did not require ethical review, as determined by my supervisor Dr. Barnaby Craggs.

Impact of the COVID-19 Pandemic

Event / Issue	Impact on Project	Action Taken to Mitigate Impact	Remaining Impact
No physical access to test environment	Could not perform necessary experimentation	Remote solution developed, explained in Chapter 3	Some traffic cannot be captured this way, even with remote connection (Profibus/Ethernet only). Missing some features that would be found with a physical connection. Previous scans had to be redone in order to be consistent. Each scan also took much longer (up to two hours for a single scan in some cases).
No access to fast MVB lab machines	Feature extraction from the pcap files was very slow (over two hours total), some of the packet captures are 100s of MB large	Code was optimised	Still slow. This in combination with the extra time needed for the previous row, meant that time spent on experimentation was inflated greatly beyond previous planning, reducing time for tweaking the experiment and writing the paper.

Contents

Chapter 1

Introduction

The societal services that form critical national infrastructure (CNI) such as water and waste treatment, electricity generation, and public transport all depend upon industrial control systems (ICS). The frequency and impact of CNI attacks has been increasing quickly in recent time, with recent major examples of Stuxnet, Triton and the 2015 Ukrainian Power Outage attack reaching news headlines across the world [1, 2, 3].

CNI is traditionally composed of three parts, information technology (IT), operational technology (OT) and physical processes. IT relates to technology involved in the processing and communication of information, whilst the OT relates to the technology used in systems that handle psychical processes. ICS (industrial control systems) are a subset of OT and refer to the systems that manage and control industrial processes.

	IT	**OT**
Users	Companywide, from HR, to CEO	Engineers
Priority	Confidentiality, Integrity, Availability	Safety, Availability, Integrity, Confidentiality
Devices	Laptops, Servers, Databases	PLCs, HMIs, Industrial Equipment, SCADA Control
Scope	General	Specialised
Protocols	HTTP, HTTPS, RDP	Modbus, Profinet, EtherNet/IP
Device Lifespan	3-5 years	15+ years

Table 1.1: Typical differentiating features between IT and OT.

As networks grow larger and ever more complex, the process of discovering and collecting data of the devices on these networks - asset discovery - is more important than ever. Asset management has numerous business benefits common to both IT and OT, such as optimisation and auditing, as well as forming a vital part of the cyber security of a system. Keeping all devices patched with security updates is an efficacious measure in preventing malicious activity, and visibility of a network can be used in both preventing an attack and supporting an investigation into an attack post-hoc. On the adversarial side of security, asset discovery is also an essential part of the attack process, where attackers use asset discovery to perform reconnaissance and enumerate a network. There are a vast number of tools used for asset discovery, and there is some commonality between the tools used for legitimate and illegitimate purposes.

Interconnected networks of IT devices have been around for a long time, meaning that both the attack and defence of IT networks are highly sophisticated and developed. Unfortunately, it is not a matter of using all of the same techniques and technologies used to secure IT for OT. The protocols and architectures in OT are entirely different, and many legacy ICS devices are very sensitive and can be impaired by something as simple as a port scan [4]. As a result, we must come up with novel security solutions.

Intrusion detection systems (IDS) are used to detect when an attack is occurring, allowing security teams to intervene and either stop the attack or control the damage. Many IDS such as SiLK and Snort are able to detect that an asset discovery scan is occurring, but are not able to detect which tool-set is being used [5, 6]. Since administrators and security systems may also use these asset discovery tools, IDS will flag these as malicious, generating a huge number of false positives. If an IDS could detect which asset discovery toolset is being used, the number of false positives could be reduced and the information given to security analysts improved.

1.1 Aims

The aim of this thesis is to investigate whether we can detect which asset discovery tool is being used to probe an OT network. The use of this technique in IDS is as follows: if an organisation regularly uses asset discovery tools from the set X, but some new tool y is detected, such that $y \notin X$, then we can say that the use of tool y is more likely to be malicious[1]. This would reduce the number of false positives from benign asset discovery tools, and provide more information to security analysts, strengthening OT security. In order to build our technique of detection, we performed an analysis on the traffic generated by some common asset discovery tools, and then built a system capable of detecting a scanner from some given traffic. We found that we are indeed able to fingerprint each scanner, with a combination of different features, achieving an exact match accuracy of 92%. As such, we define our research aims for this dissertation as follows:

1. Can we identify which asset discovery tool is being used to probe an OT network?
2. Can we use intelligence about which asset discovery tool being used to probe an OT network as an additional vector to use for intrusion detection?

1.2 The Report

The remainder of this paper is structured as follows. First we discuss background information, and related work pertaining to operatioal technology, asset discovery and intrusion detection(Chapter 2). Next we lay out our methodology for the experiment (Chapter 3), following that we analyse the results (Chapter 4), provide a discussion (Chapter 5) and finally a conclusion (Chapter 6).

[1] The use of tools where $y \in X$ may still be malicious, therefore in Chapter 5 we propose that this technique merely raises the relative suspicion level rather than decide absolutely where traffic is malicious or not.

1.3 Key Terms

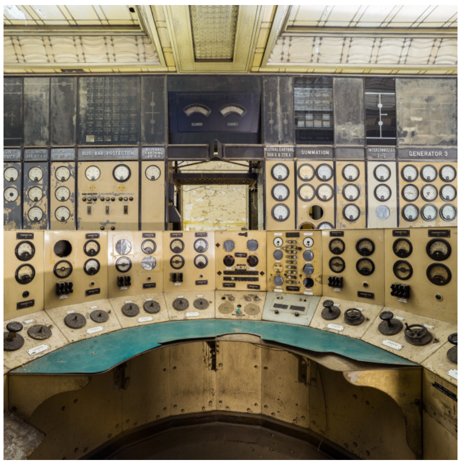
PLC - Programmable Logic Controller - ruggedised computers used to control industrial equipment
SCADA - Supervisory Control and Data Acquisition - control system architecture
DCS - Distributed Cotrol System - another control system architecture
IDS - Intrusion Detection System
Scanning/Enumeration/Discovery - The process of gaining information about a device or network of devices
OT - Operational Technology
ICS - Industrial Control Systems
IT - Information Technology
Packet - A unit of data carried used by some communication protocols

Chapter 2

Background

2.1 Industrial Control Systems

Historically, OT appliances such as pumps and valves were controlled manually on each machine - a slow and inefficient system. We began to centralise control by transmitting individual analogue signals from a control room to each of the many machines in a plant. Whilst this was a substantial improvement, the new analogue connections and interfaces were inflexible, short ranged, prone to malfunction and needed to be manually rewired each time a device was added or changed. As a result of these downfalls and some timely advancements in technology, the process was digitsed. Distributed Control Systems (DCS) and Supervisory Control and Data Acquisition (SCADA) systems are two common digital control system architectures used in industrial control systems. DCS and SCADA systems are very similar, their purpose is to gather data and present it to the system operators, as well as being able to make decisions and control the plant without the need for human interaction.



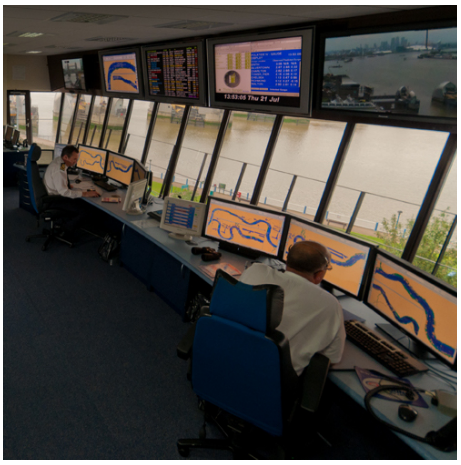


Figure 2.1: Left: A pre-DCS era control system in Battersea Power Station. The analogue controls are centralised but not integrated into one system[7]. Right: A modern SCADA control room for the Thames Barrier[8].

Figure 2.2: A Rockwell CompactLogix 5380 (Left), and PLC controlled pumps in a plant (Right)[9][10].

	DCS	**SCADA**
Priority	Process oriented	Data acquisition oriented
Area	Local to a plant	Distributed across a large area
Device support	Proprietary and support only devices from one manufacturer	Open, can support many different device manufacturers
Flexibility	Rigid	Flexible
Protocols	Profibus, Profinet, Ethernet/IP	GSM, Modbus, S7Comm, GPRS, Ethernet

Table 2.1: A comparison of the typical features and constraints belonging to DCS and SCADA systems.

DCS are process and state oriented, and are suited to control the entirety of a singular plant. Programmable Logic Controllers (PLCs) are industrial computers that are used to control a field device like a valve or pump. PLCs are designed to be extremely rugged, lasting for 20 to 30 years and operating under the extreme vibration, temperature and humidity conditions that are found in industrial environments. SCADA systems are data and event driven, and use PLCs to form distributed networks suitable for large scale management of multiple plants. SCADA systems are open and flexible in nature, in contrast to DCS which are rigid, inflexible and often proprietary and vendor locked. In return for the lack of flexibility, DCS typically gain easier management, which as a result often improves security.

Common OT equipment vendors include:

- Siemens
- Rockwell Automation
- Allen Bradley
- General Electric
- Delta
- Schneider Electric
- Honeywell
- ABB
- Mitsubishi

Recently, we have seen the introduction of the Internet of Things (IoT) - a system of interconnected devices, machines and objects that communicate together without human interaction. Smart thermometers, lights and even toasters can be found in a growing number households. Households are not the only place where IoT devices are useful, OT environments can benefit greatly from the enhanced levels of visibility and automation provided by IoT devices and systems, such as smart sensors and cloud based machine learning. This has been termed the Industrial Internet of Things (IIoT). Figure 2.3 shows the convergence of IoT and OT forming IIoT, where systems use a mixture of traditional ICS devices and modern IoT devices. Rashid et al. outline an attack against an IoT/OT environment using an expansive Gunt CE581 water treatment plant test environment.

One factor which we must take into account when considering the security of OT networks is the attitudes and priorities of OT management towards security. IT systems focus on confidentiality and integrity first, whereas OT systems focus on safety and availability above all else. This has many repercussions. For one, during the development of much of the OT software and hardware in use today, security was not a strong consideration, resulting in a vast number of vulnerabilities. Beresford, Hui & McLaughlin, and Lim et al. show just a small sample of these [11, 12, 13]. Vulnerabilities in these systems are typically fixed through security patches, however since operators wish to keep the system running at all times, devices are often intentionally unpatched instead of taking them offline for updating or risking compatibility issues with the other legacy equipment [14, 15]. Whilst fixing vulnerabilities is a short term fix, there is no guarantee against novel attacks and it is not nearly as effective a measure as security through design.

Decisions around ICS are highly influenced by availability over security. Sometimes, the most simple way to stop a cyber attack is to turn everything off, but if the systems are running an oil production plant, turning everything off for a couple of hours would result in a considerable loss of revenue - unacceptable to the producers. Similar decisions are made around adding in new security, if it results in a loss of availability, it is unlikely to be implemented.

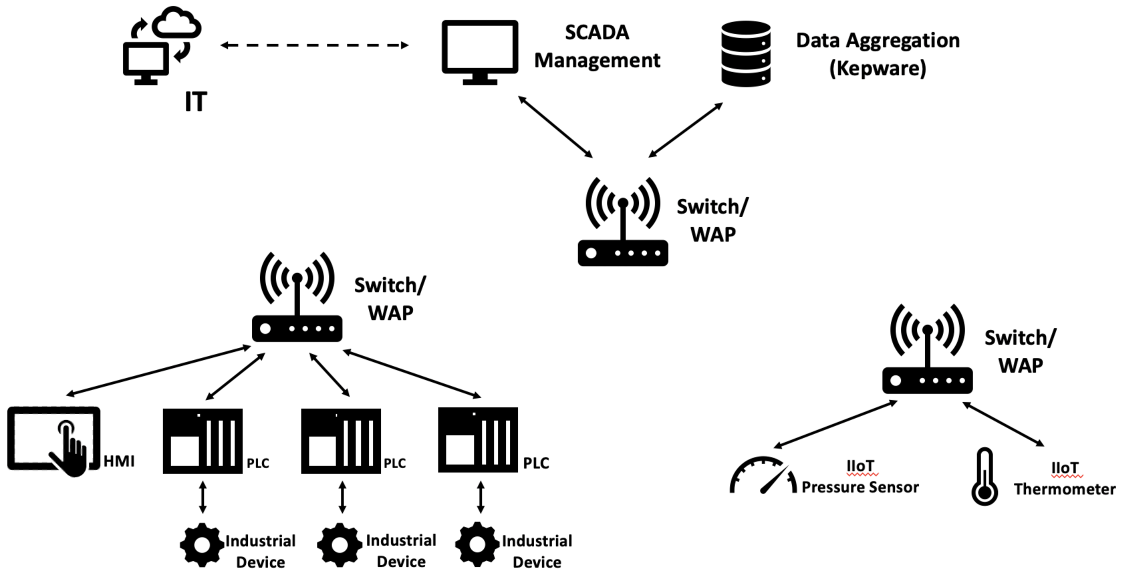


Figure 2.3: A simplified diagram showing a typical SCADA IIoT environment with a mix of devices and protocols interacting with one another.

Many OT components were simply never designed to be connected to a network at all, let alone the Internet, but a combination of IIoT convergence, and a lack of training in security means that this occurs more than one would think. At the time of writing this paper, there are more than 25,000 individual internet-connected Rockwell, Siemens and Schneider Electric OT devices on Shodan, a IoT search engine [16]. To compromise a number of these devices, an adversary would not even need to plan a sophisticated attack - many of these devices will use default login credentials which can be found in user manuals online. Since OT devices in general are designed to be very robust and last for decades, there are a large number of legacy devices still in use. These devices were designed with old, pre-internet networks in mind, and as a result are fragile when it comes to connectivity. Care must be taken when working with devices such as these, Coffey et al. discuss the negative impacts that operations as simple as a port scan can cause on ICS devices [4].

2.2 Kill Chain

Hutchins et al. describe the traditional cyber kill chain, the typical phases of a cyber attack: Reconnaissance, Weaponisation, Delivery, Exploitation, Installation, Command and Control, Action on Objective [17]. Assante & Lee propose an alternate cyber kill chain for OT [18]. They suggest that an adversary will go through the same steps as the typical cyber kill chain, but upon reaching the final stage – 'action on objective' - a further set of phases begins. Once the attackers have penetrated the network, they then perform further reconnaissance inside the network. It is here that asset discovery scanning tools are used to gain information about the assets on the network. Here, an adversary would first enumerate the entirety of a network to find live devices, and then probe a smaller number of individual devices in order to gain information about the hardware and running applications. The information discovered in this stage would then be used by the adversary to develop another attack.

With knowledge of the system and an attack developed, the adversary must either test their attack on their own testbed, or fire blindly. Firing blindly may result in nothing more than a PLC powering off - which whilst would have some detrimental effect on the target system, the capacity for damage is much smaller than a covert, sophisticated attack like Stuxet. OT devices are prohibitively expensive, meaning that testing an attack is limited to a well funded threat actor such as a nation state or large criminal group.

The malware discovered in 2013 known as Havex can be used to exemplify the steps of the cyber kill chain [19]. Firstly, reconnaissance was used to discover suitable targets for a spearphishing campaign, and to learn information about the target systems. Then, malicious emails were used as delivery, which in turn exploited the system and installed the malware. Havex then scanned the environment for ICS devices using the OPC communication standard, before reporting back to command and control with all the information gained. If the malicious scanning was noticed at this stage, the attack could have been mitigated or stopped entirely. From here, the ICS vendor websites were compromised through a separate kill chain, and targeted malware was downloaded directly onto the device, bypassing perimeter protections such as firewalls. No subsequent actions from

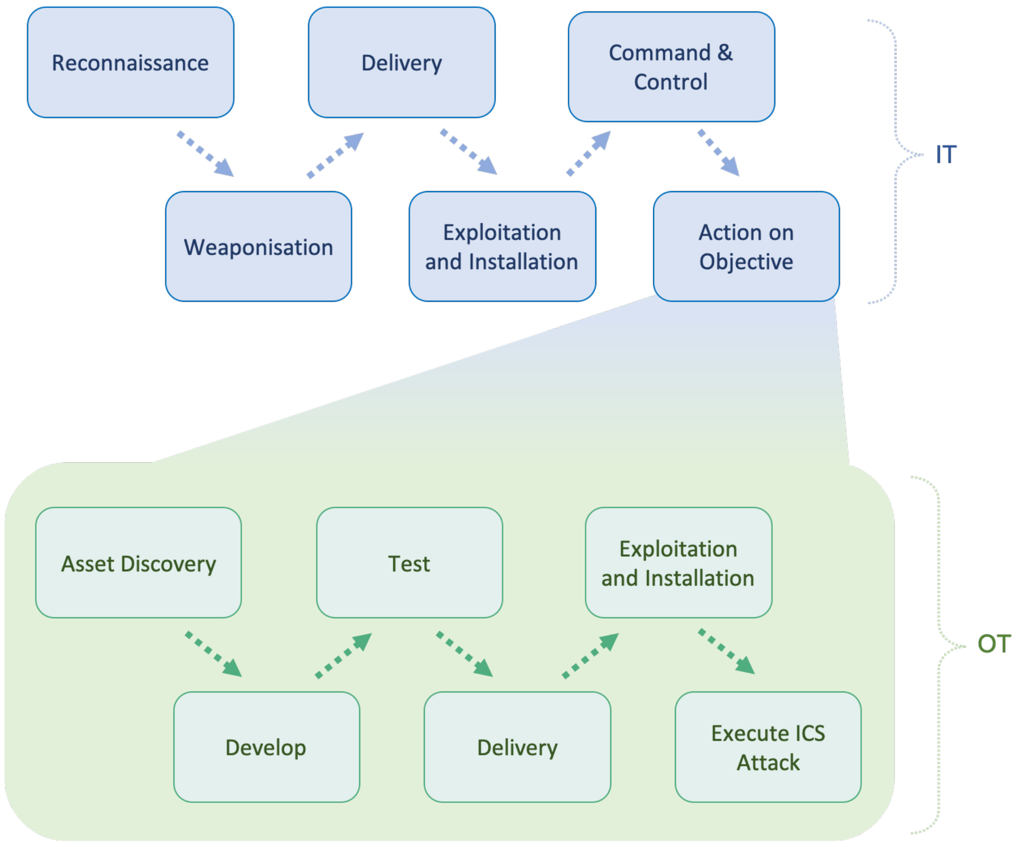


Figure 2.4: The industrial control system cyber kill chain [18].

the Havex malware have been observed by the cyber security research community, but certainly knowledge of the system topology could have been used for an attack.

2.3 Attacks

In 2019 Kapersky reported that of a sample of 359 companies working in industrial sectors, 49% experienced an ICS security incident in the previous year [20]. The attacks we typically see in IT often aim to steal information, but attacks on ICS are typically more destructive in nature - making PLCs misbehave such as spinning a motor too fast or better yet, spinning a motor but reporting to SCADA as if the motor is stationary. Attacks on ICS can be motivated by financial gain, where an adversary attempts to gain directly such as the 2019 ransomware attack that cost Norsk Hydro £45 million, or indirectly where a competitor gains from a loss of availability [21]. Attacks on ICS can also be political or military in nature, such as Stuxnet or Operation Cleaver [1, 22]. Attacks on CNI can not only be devastating in terms of a company's revenue, but since CNI is by definition critical - a major attack on CNI could prove lethal [23].

2.4 Techniques To Prevent Attacks

Fortunately, there are a number of techniques used to mitigate and prevent against attacks. It has been shown throughout the relatively short history of cyber security that the most effective measures are those that completely remove the risk, rather than those that attempt to reduce the likelihood or control the damage.

2.4.1 Separation

Separation is perhaps the most simple method of security. If a computer or local network has no connection to any other network, the attack space is virtually zero. In many cases, separation is simply not possible - organisations need to access the internet and other machines on their local network. It is common practice that the devices on corporate networks have their connection to the internet managed through a DMZ (demilitarized zone). Separation can also be used as an immediate defence against an attack in progress - if the connection between a compromised system and target system is breached or the target system turned off, then the attack cannot proceed to the target system. In some cases, turning off a system or breaching a connection in the face of an attack is simply not viable if the fiscal impact is too high.

Separation is not a perfect defence though. There have been attacks on separated networks that make use of communal USBs, or using social manipulation to bridge the gaps. Virtual machines also employ a form of separation. Cloud computing providers such as Amazon Web Services and IBM Cloud allow users to save money by hosting a large number of virtual machines on a single system, where each acts in isolation as if it is the only machine. Ristenpart et al. showed that it is possible to break this isolation by locating some target VM, placing their own VM adjacent to it, and then use a cross-VM side channel attack to extract information [24].

2.4.2 Patching

Patch management is another method of preventing attacks. Many software and hardware manufacturers release regular security updates (patches), in which they fix bugs in the code which may be used to exploit a device. These bugs can be found by internal developers or consultants, but more often that not, these patches are for bugs where there has already been some public attack using the vulnerability. We call these vulnerabilities 'zero days'. Even if a device is patched regularly, they are unable to prevent attacks using zero days. IBM's BigFix is a commercial tool with the capacity to automatically patch devices in a large organisation [25].

Some older devices are no longer supported by their developers, meaning that there is no official support against new vulnerabilities. OT devices often fall into this category, since the devices are extremely old. Patches can also have detrimental effects. Spectre and Meltdown are two vulnerabilities that rely on flawed optimisations [26, 27]. These optimisations are necessary in keeping many processors up to speed, and the first patches for these vulnerabilities simply disabled these optimisations. There have been many subsequent attempts to patch these vulnerabilities whilst retaining the optimisation benefits, but many issues have been reported with these. System administrators may argue that patching these devices is simply not viable, and that it is better to run the risk of exploitation until the devices reach the end of their cycle, in which case they will be replaced with a device that is (hopefully) more secure.

2.4.3 Firewalls

Firewalls provide security by allowing or blocking traffic according to some set of rules. These rules can be based on IP address, protocol, time, packet data or some combination of them all. Blacklisting is where a system allows everything except things that match the rules, whereas whitelisting is where a system blocks everything by default, except traffic that matches the rules. Whitelisting is often more secure than blacklisting, but greatly reduces ease of use. Firewalls are often placed at many different intersections throughout the network, since different rules suit different areas. Firewall systems have been developed specifically for ICS such as Nivethan et al. and Kim et al., as well as being supported in commercial solutions such as Cisco's ISA 3000 and CheckPoint's Next Generation Firewalls [28, 29].

2.4.4 Training and Culture

As noted by Boyce et al., in 50-80% of cases, human error is the leading cause of information system security failure [30]. Even with the strictest technical security, human error can lead to an attack. In 2008 an attack on a USA military base in the Middle East occurred which resulted in one of the worst breaches of U.S. military computers in history [31]. One of the military personnel found a seemingly harmless USB drive in the car park of the military base, which they then plugged into their computer - little did they know that the USB drive contained malware which would compromise a large portion of the military computer system. Nissim et al. provide a comprehensive survey of different USB peripheral based attacks [32].

This problem extends into the world of OT, where many engineers are not provided with the same level of security training as in IT environments. One example of this might be an operator innocently connecting some ICS equipment to the internet so that they can work on it from home. Whilst this may seem extreme, one only needs consider the 25,000 internet connected PLC devices that we found on Shodan to gain some amount of persuasion about this. The lack of a security culture means that the potency of social engineering methods in OT environments is rather high. Green et al. asses the impact of social engineering in ICS, with one worrying highlight being "[where] an attacker sends out ten e-mails, there is a 90% chance at least one person will fall victim" [33]. Training and building a culture of security is essential in defending an organisation, but the security techniques we mention here are essential to cover the gaps where training cannot or currently does not.

2.4.5 Design

Security should be one of the highest priorities from the start when developing any kind of system. Whilst all the other techniques we discuss are effective, simply making sure that the vulnerability is never there is the most successful. IIoT is one of the worst industries in this aspect. Many devices such as smart light bulbs and children's toys are manufactured very cheaply, in countries with lax laws concerning security. But developed, western markets are not innocent in this aspect either. Many developers will develop products and release them with security as an afterthought - to them, releasing first means maximising profit. There have been many high profile cases of security flaws that have come from bad design, such as the 2005 attack on TalkTalk where they fell victim to SQL injection [34]. When Chapter 5 "*Human Error? No, Bad Design*" of Don Norman's renowned *Design of Everday Things*, is combined with the finding that 52% of cyber security incidents in the "State of Industrial Cybersecurity 2019" Kapersky report, it seems obvious that major reform is needed in the design of ICS [35, 20].

2.4.6 Honeypots

Honeypots are devices put on a network that have no real value or contain no data of value, and often have weak security to make them more enticing. Since they are not used on the network, any activity on a honeypot is inherently suspicious. Organisations can use this activity as a trip-wire, indicating that an attack is occurring, or instead use and analyse the activity in order to understand the attacker's methods, and improve their defences accordingly. Honeypots have been deployed for research purposes in IDS such as that by Buffey and Piggins [36].

2.4.7 Penetration Testing

Penetration tests are a method of finding weaknesses and vulnerabilities in a system's security by authorising an intentional cyber attack on the system. Vulnerabilities can then be fixed, improving the overall security of the system. Often, the exact same tools are used that an adversary might use, and the penetration testers may be given private system information, or instead are only given the systems name. The most widely used security standard, ISO 27001 does not mandate that penetration tests are carried out, but it is often recommended.

When penetration testing is carried out on ICS, care must be taken. Testing usually stops short of accessing/penetration testing the ICS devices due to the safety risk, which contrasts penetration testing in IT where they may well attack all devices.

2.4.8 Intrusion Detection Systems

In this thesis we focus on intrusion detection systems (IDS), which are physical devices or software that monitor and analyse network traffic, and present possible intrusions to analysts via security information and event managemet (SIEM) software. Analysts triage the alerts generated by IDS and take action upon these intrusions. IDS can either be placed separately around a network, or can be placed centrally with data feeds coming in from many different parts of the network. IDS can be broken down into two categories:

- Signature Based – Packets on the network are matched against a database of ‘bad’ signatures of known threats such as malware. Signature based attacks have a high accuracy for known attacks but are ineffective against novel attacks. New signatures are created when novel attacks are discovered, and can be downloaded from community based or proprietary sources. Examples include IDIOT and STAT [37, 38].

- Anomaly Based – Network traffic is compared to ‘normal’ traffic using statistics or machine learning in order to provide a likelihood that malicious activity is occurring. Anomaly based methods are effective against high intensity attacks such as brute forcing a password, but less effective against ‘low and slow’ attacks, and generate a large number of false positives. IDES is an example of an anomaly based IDS [39].

Intrusion Prevention Systems go hand in hand with IDS - where once an intrusion is detected, the system autonomously takes action to prevent the attack.

Issues with IDS

The amount of traffic passing through a sizeable network can be considerably large. IDS solutions must be able to analyse traffic at least as fast as the network, which is problematic in terms of computation. Fortunately analysing traffic is suitable for concurrent computation across several cores or several machines. Even with this, the amount of traffic is still very large, so some form of filtering of benign traffic is necessary, but therein lies a difficult balance between retaining necessary information for intrusion detection and efficiency.

Since IDS are so commonly used, most adversaries will assume that a target network uses some form of IDS and take action to evade detection. The literature around IDS evasion is vast; common techniques include spreading the attack across a long period of time as with NMaps slow scan, obfuscation by encryption, exhausting the IDS CPU or attacking the machine learning algorithm [40, 41, 42, 43].

In a large network with thousands of devices and users it becomes very difficult to accurately detect attacks, but also even more necessary. Common IDS solutions such as SNORT and DarkTrace generate an extortionate number of false positives, where benign and non-malicious network activity is flagged as an intrusion. Operators must be trained to decide which alarms are real and are false, resulting in a prolongation in the amount of time before operators start to take action against an attack [44].

SENAMI

A number of IDS used in OT environments use are just modified IDS that are typically used in IT, rather than tailored to OT. Selective Non-Invasive Active Monitoring for ICS Intrusion Detection (SENAMI) is an IDS built specifically for OT [45]. The approach involves passively observing PLC data packets on the network to spot suspicious behaviour and specific attacks, as well as actively monitoring a select number of PLC variables. The vectors used in SENAMI detection include:

- Timing - both frequency and difference to legitimate timing
- IP addresses
- Packet content
- Values in the PLCs themselves

SENAMI addresses the issue of false positives by not automatically reporting an intrusion when the upload of PLC logic code through packet content analysis is detected - since uploading PLC logic code is a valid operation that an operator may carry out. SENAMI instead combines this with other vectors such as source IP addresses to form a decision.

2.5 Asset Discovery Techniques

There are often thousands of PLCs and other devices connected on an OT network. This means that when operators are configuring and interacting with a network, they require some method of digitally requesting information from the devices. Asset discovery and detection is also an essential part of the reconnaissance phase of the ICS kill chain that we mentioned earlier. Many tools have been developed to perform the task of asset discovery. We can categorise scanning tools into two methodologies - active and passive which we will discuss next.

2.5.1 Passive Scanning

Bartlett et al. describe passive scanning as "[finding] services on a network by observing traffic generated by servers and clients as it passes an observation point and is generally invisible to the hosts running the services" [46]. Gonzalez and Pappa provide a passive technique that extracts Modbus traffic in order to gain information about the devices and transactions occurring on the network [47]. Other examples of passive scanning techniques designed for ICS are Dragos's tools and the NSA's GrassMarlin [48, 49]. Passive scanning techniques do not create any traffic, so are undetectable by means of traffic analysis, thus our work will focus on the analysis of active scanning techniques.

2.5.2 Active Scanning

Active scanning techniques send packets to a hosts, and then analyse the response (or lack of response) returned. It is these tools which we wish to be able to classify. Active scanning tools are used to ascertain whether a device is active, whether a port is open or service running, or to gain some additional information about the system. Some active scanning hosts are designed to scan a large range of hosts, whilst others are designed to narrow in on a small number of targets and obtain deeper, more complex amounts of information. Active scanning has the benefit of being instant and not needing to wait for traffic as with passive techniques, but are able to be detected by IDS. A vast number of active scanning tools are available, with different specialisations. Coffey et al. focus extensively on the use of Nmap and Nessus in OT settings, and discusses the negative impact that these scanning tools can have on ICS networks [4, 40, 50].

The core functionality of Nmap is to detect hosts at IP addresses and check whether ports are open. To detect whether a port belonging to a trasport protocol (TCP, UDP, STDP) is open, probing packets are sent to target port, and the response indicates the status of the port. Each transport protocol has a specific mechanism for requesting a response. To detect whether a host is available, Nmap uses combinations of multi-port TCP SYN/ACK, UDP, SCTP INIT and ICMP probes. Nmap has a huge number of other features such as detecting whether ports are filtered, service detection and OS detection.

Many IDS are capable of detecting that a scan is occurring. The SiLK toolset by NetSA at CERT uses Threshold Random Walk (TRW) and Bayesian Logistic Regression (BLR) techniques with a number of features to detect the presence of an asset discovery scan from live or captured traffic [5, 51]. Floodeen and Van Wyk argued to expand beyond detection and proposed that network traffic should be stored and used to classify the scanner that generated the traffic.

Chapter 3

Methodology

3.1 Overview

With the aim of being able to fingerprint asset discovery scanning tools, we first needed to obtain a suitable set of data. We used a physical test bed to simulate a real ICS system, and captured the traffic generated from the use of 10 different scanning tools. Table A.1 shows the individual commands we used to generate the traffic from each scanner. We then processed the data to extract features, before building classifiers and evaluating them on a new set of data. In this section we expand and discuss our methodology and the choices we made.

3.2 Scanning Tools

3.2.1 Nmap

Nmap will be many security professionals first thought when it comes to network and asset discovery [40]. Nmap is free and open source, and included in many popular security distributions of Linux. Nmap has the capacity to scan an entire range of IP addresses, or narrow down onto a single address. The basic functionality of Nmap is sending packets to different ports on a host, in order to identify if they are open. Nmap allows users to write and share scripts, extending its functionality and allowing automation.

Many of these scripts are included in the base install, such as **s7-info** and **modbus-discover**, both of which are designed to enumerate devices using their respective protocols. When using the s7-info script, we found errors prevented the script from running against our test enviorenment, but repaired these by removing the lines shown in Figure 3.1.

```
output["System Name"] = string.unpack("z", response, 40 + offset)
output["Module Type"] = string.unpack("z", response, 74 + offset)
output["Serial Number"] = string.unpack("z", response, 176 + offset)
output["Plant Identification"] = string.unpack("z", response, 108 + offset)
output["Copyright"] = string.unpack("z", response, 142 + offset)
```

Figure 3.1: The lines to be removed to ensure s7-info runs without error.

3.2.2 Ping

Ping was written by Mike Muuss in what is now the US Army Research Laboratory, as a utility to test whether a host can be reached on an IP network [52]. It is a struggle to find an operating system that does not include Ping, since it is so widely used. Ping works by sending an ICMP echo packet (that is an ICMP packet with the type header set to 8), and waiting for an ICMP echo reply (type header set to 0). We chose to include Ping because whilst it is perhaps the most simple tool used for network discovery, it alongside Nmap, are the most widely used.

3.2.3 Nessus

Nessus is a vulnerability scanner, and is the most cited in literature at the time of writing [50]. Originally free and open source, Nesuss is now distributed as proprietary by Tenable Inc. Similar to Nmap's scripting functionality, Nessus allows plug-ins written in Nessus Attack Scripting Language (NASL) to expand functionality. Nessus is constantly updated and 24/7 customer support is provided. The cost may prohibit adversaries from using Nessus, though illegal cracked versions are available.

3.2.4 OpenVAS

OpenVAS began as a fork of Nessus, when it become proprietary [53]. OpenVAS is very similar to Nesuss as a result of this, differing in that it is free and open source. OpenVAS is a collective term for the scanner, user interface, and daily-updated vulnerability test feed.

3.2.5 PLCscan

PLCscan is a free tool developed by Dmitry Efanov at Positive Technologies [54]. PLC-scan is designed to enumerate devices over S7Comm or Modbus protocols, and can be used on a single device or a range of addresses at a single time. Packets are sent to the PLC over COTP, which firstly identify whether S7Comm or Modbus is running, and then send requests in the respective protocol to read information from the devices memory. The information retrieved by PLCscan is related to the module name, type, ID, and software version.

3.2.6 S7scan

S7scan was developed by Kaspersky Lab Security Services and is another free and open source tool used to enumerate PLCs [55]. S7Scan is functionally very similar to the above PLCscan, but only supports S7Comm devices. S7scan provided the most detailed results of any of the open source tools other than OpenVAS and Nessus, possibly because the code has been updated recently, or perhaps it was designed specifically for the same model of PLC that would used.

3.2.7 Industrial Exploitation Framework

Industrial Exploitation Framework (ISEF) is an exploitation based framework based on RouterSploit and similar to MetaSploit but designed specifically for ICS [56]. ISEF contains modules for scanning, exploiting and decoding traffic from devices of many different ICS protocols. Interestingly, ISF has a relatively large number of interactions on GitHub in contrast to some of the other scanners, but only one mention in academia at the time of writing, highlighting how fast security in industrial control systems is progressing [57].

3.2.8 ModScan

ModScan was developed by Mark Bristow, and is another free, open source tool, similar to PLCscan and S7scan [58]. ModScan only supports devices using the ModBus protocol, and can operate on a range or individual IP address.

3.2.9 ZMap

ZMap is a tool designed to perform host discovery on internet scale ranges of addresses in small amounts of time [59]. ZMap has the capacity to scan the entire public IPv4 address space in under 45 minutes with a gigabit connection.

3.2.10 Simatic Automation Tool

Simatic Automation Tool (SAT) is a proprietary piece of software used for operating and maintaining Siemens ICS devices [60]. SAT can retrieve system information from supported devices, as well as start, stop and modify the software running on them.

3.3 Packet Capture

There are several tools available for packet sniffing and analysis. Asrodia and Patel discuss several common packet sniffing solutions [61]. Some tools focus on live analysis of traffic over a large network whilst others have functionality for anomaly detection. For the purpose of this research, we need only inspect the traffic over small periods of time between two devices, use filtering, and have the desire to inspect individual packets. As such, the tools Wireshark and tcpdump are suitable. Since Wireshark has a GUI which we are able to utilise and tcpdump is only command line based, we will opt to use Wireshark for packet collection and analysis.

3.4 Hardware

When an adversary or engineer is performing a scan on a PLC, they would in most cases be doing so from a device on the same local network as the PLC. We used the popular Siemens S7 1200 CPU 1212C PLC, and Siemens KTP700 HMI connected by a Westermo RFI-219-T3G switch. The devices were contained in a Bristol Cyber Security Group ICS teaching box (Figure 3.4), but we replaced the simple switch with the more advanced Westermo. We then connect the computer with the scanning tools to the switch, and begin testing each one. The connection between the scanner and switch is mirrored and fed as input into Wireshark (shown in Figure 3.3), this is such that there are no duplicate packets as there would be when monitoring the connection between switch, PLC and HMI.

Due to the egregious circumstances surrounding the 2019-20 coronavirus pandemic, the data collected using the previous setup was incomplete, meaning that a remote system must be assembled in order to collect new data. The PLC and HMI remain connected to the switch in the same manner as before, but the Westermo RFI-219-T3G switch was replaced with a Mikrotik routerboard hex lite. The switch was then connected to a raspberry Pi running OpenVPN, allowing the scanning computer to communicate across the internet as if it were a LAN connection (shown in Figure 3.3. The issues that arise from this are discussed in section 3.6.

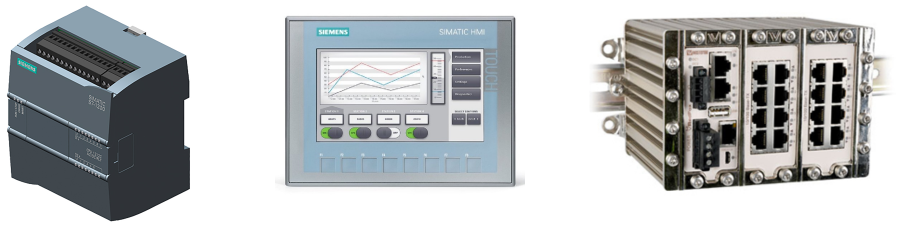


Figure 3.2: Hardware used for this paper. From left to right: Siemens S7 1200 CPU 1212C PLC, Siemens KTP700 HMI, and Westermo RFI-219-T3G switch.

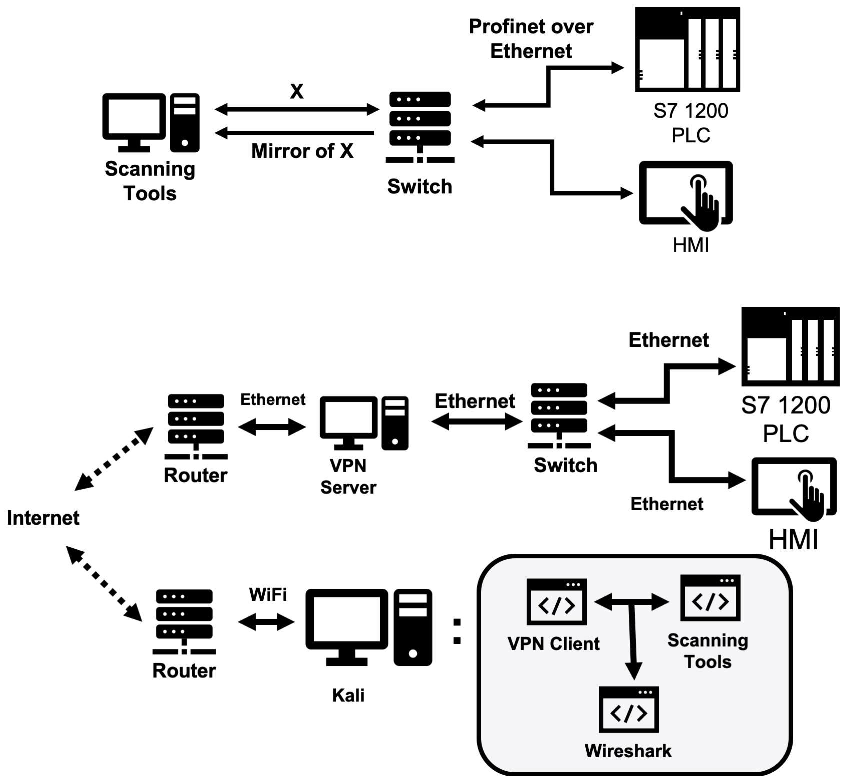


Figure 3.3: Diagrams showing the original experimental setup in the lab (top), and the new, remote experimental setup (bottom)

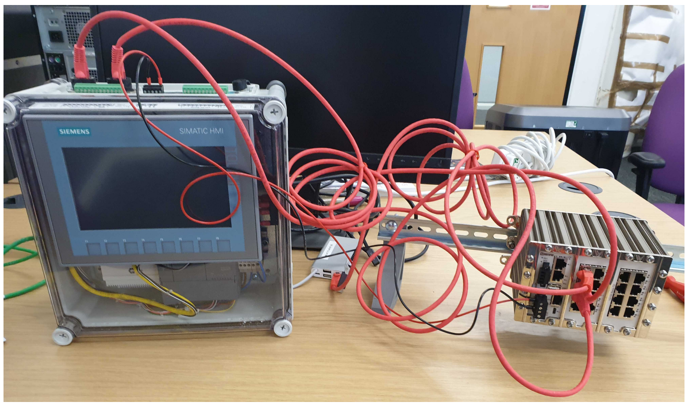


Figure 3.4: A photograph of the test environment we used for the experimentation.

3.5 Analysis Methods

Here we briefly outline the different methods of analysis that we will use to investigate toolset detection. In Chapter 4 we dig deeper into the details of each of these.

1. Number of Packets - If the number of packets sent by a tool is the same each time it is used, and has large variance with the packets sent by other tools, then it the number of packets is a great candidate for toolset detection.

2. Packet Type and Content - It may be that certain tools send certain packets in a specific order or send specific types of packet that others do not. Examples we will investigate are IP addresses, protocols, ports and packet features.

3. Flow - If the number of packets is not a viable indicator, will look at the bandwidth of the traffic across the connection. This may differentiate between tools that send a similar number of packets. Shah et al. used the bandwidth consumed by Nmap in order to fingerprint its different options [62].

3.6 Known Limitations

In this section we will discuss the limitations that arose in carrying out our experimentation, and provide some solutions. Firstly, differences in hardware and network architecture will have an effect on the operation of a program. Take for example, wired and wireless connections - there would be a great difference in latency and bandwidth, meaning that the scan over the wired connection would happen much quicker than that of the scan over the wireless connection. We will minimise this effect by using the tools in the same virtual machine, on the same computer, and the same network set up. Due to the nature of modern networks, there is will always be some amount of variance in the output when an operation is repeated, even if the hardware remains the same [63, 64]. Connections may be lost, packets may be dropped or resent multiple times, execution of tasks may take longer or shorter. In order to further minimise this effect, we will repeat each experiment 10 times and take the mean.

Many of the tools we use in this paper have different options and different versions. In this paper we use the latest stable version of each tool where possible and run with default parameters unless otherwise specified, but it is worth noting that different options and different versions will change the behaviour of the scanners. There is no way around this issue, other than testing every different version with different options, which could be automated once a suitable classifier is trained.

As previously mentioned, the physical setup of the experiment needed to be changed before the tests could be fully carried out. Since the setup changed so dramatically, new results could not be used alongside old results, as there was too much variance between them. This meant that all the tests needed to be amended. The connection now relies on WiFi and a VPN connection across the internet, meaning that the noisiness of the connection will have increased, making it even more important to take a mean.

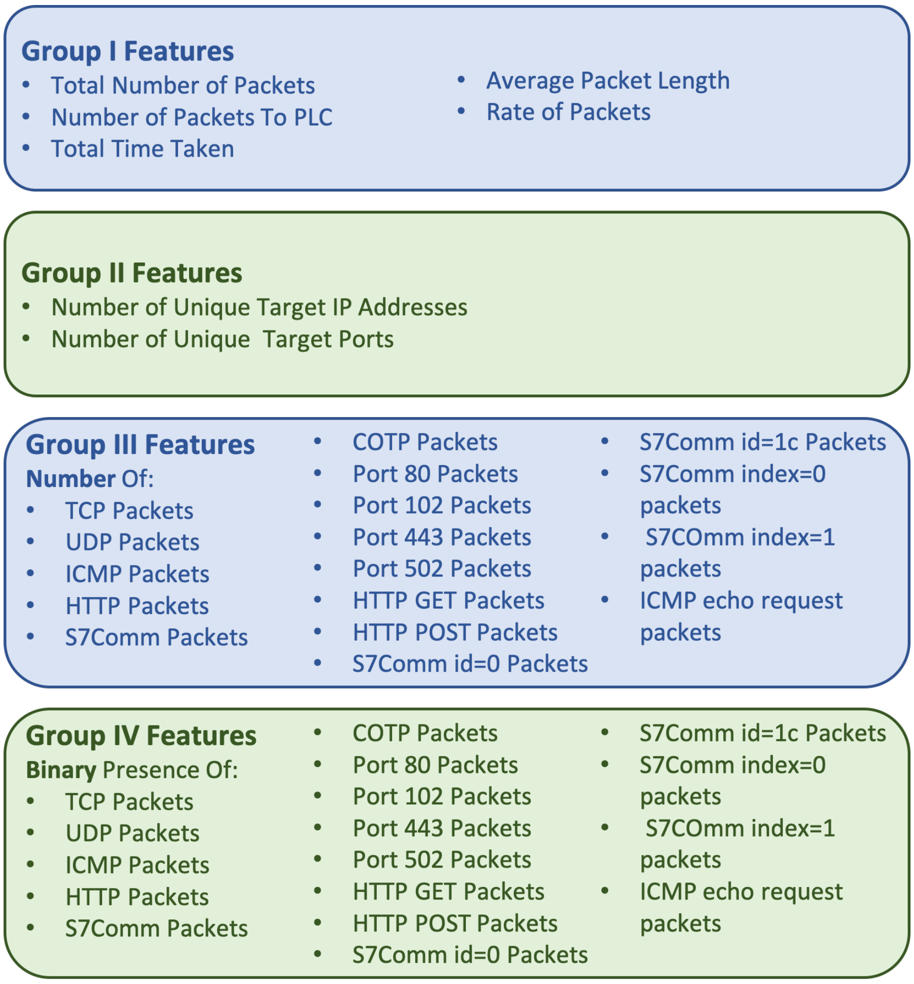
3.7 Methodology of Classification

3.7.1 Preprocessing

The purpose of this paper is to investigate the whether we can detect the scanner that generated some traffic. The next step in the process of the experiment is to build a classifier that can do exactly this. After we have collected 10 sets of traffic from each scanning tool, we undergo an analysis of each capture, with the aim of extracting sets of features. The features that we are aiming to extract are those that lead to the greatest information gain, that is the greatest increase in entropy from parent to children over a split. We used Wireshark to quickly and efficiently skim through a capture and perform filtering, and used the PyShark library to perform more complex tasks and automate the process[65, 66].

An important part of preprocessing data for classification is feature selection - removing redundant and irrelevant features. Redundant features are those that are highly covariant in terms of only another feature or features, and thus may hinder the performance of the classifier by causing overfitting to the train data. We handle redundant data by splitting our data into separate groups, then comparing the accuracy of each both individually and combined together to form the most significant set of features. Irrelevant data is data which is not needed and does not fit in under the context of the problem. In the case of our experiment, the importance of irrelevancy is small since virtually all elements of our data vary across tools and provide information. Features such as actual IP address and time of day have been counted as irrelevant and redacted - although it is worth noting that time of day is a viable indicator for IDS where operations typically only occur at certain times of day (e.e. nine to five).

To avoid over-fitting and achieve generalisation, we split our feature into groups and trained models on each of these sets. Groups III and IV represent the same data, the former represented by integer counts of the respective features whilst the latter is the binary presence of those features. As such, they cannot be used in conjunction with one from another. Figure 3.5 shows the different groups of features we used.



Group I Features
- Total Number of Packets
- Number of Packets To PLC
- Total Time Taken
- Average Packet Length
- Rate of Packets

Group II Features
- Number of Unique Target IP Addresses
- Number of Unique Target Ports

Group III Features
Number Of:
- TCP Packets
- UDP Packets
- ICMP Packets
- HTTP Packets
- S7Comm Packets
- COTP Packets
- Port 80 Packets
- Port 102 Packets
- Port 443 Packets
- Port 502 Packets
- HTTP GET Packets
- HTTP POST Packets
- S7Comm id=0 Packets
- S7Comm id=1c Packets
- S7Comm index=0 packets
- S7COmm index=1 packets
- ICMP echo request packets

Group IV Features
Binary Presence Of:
- TCP Packets
- UDP Packets
- ICMP Packets
- HTTP Packets
- S7Comm Packets
- COTP Packets
- Port 80 Packets
- Port 102 Packets
- Port 443 Packets
- Port 502 Packets
- HTTP GET Packets
- HTTP POST Packets
- S7Comm id=0 Packets
- S7Comm id=1c Packets
- S7Comm index=0 packets
- S7COmm index=1 packets
- ICMP echo request packets

Figure 3.5: The different features belonging to each group. Group IV is the binary presence of those features in Group III - where the feature in IV is *True* if the corresponding feature is > 0 in Group III, and *False* otherwise.

3.7.2 Learning Methods

Then, after extracting suitable features and forming a dataset, we can begin to build and train classifiers. Many different types of classifier are used in IDS in academia and industry. Hasan et al. compare the use of Support Vector Machine and Random Forest classifier methods applied to IDS, finding that both performed well, with SVM achieving slightly higher accuracy [67]. Our dataset is multi-class, and SVM is binary so would not be suitable without extensions such as reducing our multi-class problem to multiple binary classifications. The random forest classifier in that work still achieved a similar accuracy, and has been shown to be effective in intrusion detection methods in other work such as that by Farnaaz et al. and by Zainal et al. [68, 69]. Before we explain our implementation and the methodology of random forest classifiers, we first briefly explain decision trees.

Decision Trees

Decision trees are predictive models composed of decisions about observations of some data represented by nodes, and classes represented by leaves. Data to be classified is passed through a series of these decisions or nodes, down the tree until the data is assigned a class at the leaves. Decision trees are excellent for visualisation in data mining, but are limited in that they often overfit training data because they create decisions that learn the noise. To build our decision tree, we chose to use the sci-kit learn Python library, which uses the CART algorithm in which decisions in the tree are picked to provide the greatest reduction in Gini impurity[70].

Random Forests

Random forests are a method for classification that uses multiple decision trees in order to combat overfitting to the training data. Samples from the training data are drawn at random, and drawn with replacement (bootstrapping), meaning that samples may be reused in the same tree. This means that since each tree is trained on different subsets of the training data, different trees' bias to their respective subsets will cancel out, and reduce overfitting of the whole forest. The other random aspect of random forests is that for each node in each tree, only a random subset of features is considered. This allows the model to learn relationships between features that would not be possible due to other splits that caused a greater reduction in impurity. Typically, the size of the subset of features is set to the square root of the number of features, a standard which we will conform to.

When the random forest is presented with new data to be classified, the average of the predictions of each tree is taken to provide the forest's prediction - this is called bootstrap aggregation or bagging. Random forest classifiers perform well on data sets like ours, and are supported in the aforementioned sci-kit learn Python library, making them a suitable choice to use to classify the scanning data.

Chapter 4

Results and Analysis

4.1 Analysis

In this section we will be exploring and analysing the traffic generated by each run of the scanning tools against the PLC.

4.1.1 Total Number of Packets

The first obvious set of features that we can extract from the traffic is the total number of packets sent from the scanner, and the packets sent specifically to the PLC. We see in Table A.2 that there is an enormous difference in the number of packets generated between some scans. Take the number of packets generated by Modscan and OpenVAS's full fast scan, in contrast to the much smaller difference between s7scan and SAT's service data scan.

The intra-class variance[1] in some cases is notably sizeable. On a further inspection of the capture files, we can attribute this to retransmission. This could be problematic in classifying the traffic generated by a scanning tool. Let us imagine 100 packets have been retransmitted. In the case of comparing the packets sent by the OpenVAS Full and Modscan, a difference of 100 packets is negligible, however if we were to compare s7scan with SAT service data scan, a difference of 100 could mean that we confuse one with the other. In our experiments we found that this is the case, Table A.2 shows the high variances in the number of packets recieved, up to 67% in the case of s7scan. The connection from the scanner to the PLC that we used in our experiment relied on WiFi and the internet, which would result in many more transmissions than a local, wired connection between PLC, scanner, and switch.

4.1.2 Size and Timing

There are other general features that we can extract from the traffic received by the PLC which will be useful in classifying traffic:

- The **total time** taken by a scanner, that is the difference in time between the first packet sent from the scanner and the last packet sent from the scanner.

[1] Intra-class variance is the variance of instances from the same class

- The **rate**, which is the the number of packets divided by the total time taken above.
- And the **average packet length**, calculated as the total size of the traffic, divided by the number of packets.

Table A.3 shows these features alongside each other. With the total time taken, there is a very large intra-class variance in some cases. Taking Modscan as an example, there is a 200% CoV, we can also see that in Table A.2 there was a relatively high (33%) CoV in the number of packets. Naturally we would expect there to be a positive correlation between number of packets and time taken, and this shows up in the results here - the scanners that had a higher number of packets typically have a higher total time. We can also see that those scanners with a high variance in number of packets have a high variance in time taken, reinforcing the fact. However, not all scanners that have a high variance in time taken have a high variance in number of packets, meaning that the relationship seems to be one way. We can attribute this to the fact that a variance in time taken may not only be caused by re-transmission, but also by noise on the network or the computer running the scanners.

4.1.3 IP Addresses and Ports

The number of destination addresses and number of ports of the packets sent by a scanner are also suitable features to be used to fingerprint a scanner. Some scanning tools such as a ping are are targeted at a single IP address, whereas others use a whole range, for example the Nessus host discovery scan. Similarly for ports, some scanners use a select subset, whereas others like the OpenVAS Full scan target a whole range. Table A.4 shows that there is 0 intra-class variance of the number of unique target address and ports for each scanner. This is expected, since the scanners code would have to be quite erroneous to accidentally scan an address not intended. The OpenVAS host and ping scans have 0 unique ports, since they use the ICMP protocol which does not use specific ports.

4.1.4 Specific Ports and Protocols

Many of the scanners access the exact same number of ports (shown in Table A.4). We can differentiate between these by picking out the frequently used ports A.6 and protocols A.5 and using these as individual features. These were selected by identifying the most frequent ports and protocols in the traffics generated by the scanning tools, excluding those that performed a complete sweep of ports. Some of these protocols are used on top of each other - HTTP for example uses TCP (or now UDP with HTTP/3) for the underlying transport protocol, for our feature extraction this packet would count as both TCP and HTTP, attributing to the counts of each.

All of the protocols we have selected have a common port for TCP traffic. We can see that in some cases, such as the NMap scans - packets are sent to ports of the PLC which are just plain TCP, and do not attempt to use the protocol associated with that port. These scanners are able to ascertain whether the port is open, but are unable to establish whether the protocol is running.

4.1.5 Special Packets

Many of the aforementioned protocols have different options in the packet headers that can be used in order to differentiate between packets of the same protocol. HTTP packets each have a specific method, we will use the GET and POST, the most common two. The former is used to retrieve data whilst the latter is used to alter or add, as well as retrieve data. ICMP packets also have different options - the most common is type 8, which specifies an echo request. By comparing tables A.5 and A.7, we can see that not all of the ICMP packets are echo requests.

Many of the scanners use the S7comm protocol in order to gain information about the target PLC. Specifically, the packets contain requests to read a device's System-ZustandsListen (in English: system status list). The header that forms these requests includes an index and ID. The ID refers to which SZL is being requested, whereas the index refers to which part of the SZL to be read. An index of 0 requests the whole list whilst other indices relate to specific parts of the list. On inspecting the S7Comm traffic in different scanners, it was apparent that the index and ID of the requests varied, making them a suitable feature to use for our classification.

Now that we have sets of features selected from each of the scanners, we can inspect a few interesting characterics. The OpenVAS host discovery scan is identical to the Ping scan in terms of the presence of packet types, this indicates that the OpenVAS scan just uses ping with nothing else built in. Since they are the same, it should be impossible for a classifier to differentiate one another.

The SAT scans use COTP on top of TCP, and the traffic received is identical above and including this layer. We can assume that there is some protocol working on top, but it is not understandable by PyShark. Comparing the data streams generated by the SAT scans, we found that there was data common to the streams of each scan. A classifier would not be able to detect each scanner from one another based on the presence of packet types, but one could either pursue a 'deep' packet analysis or use other features of the traffic to differentiate between them.

4.2 Classification

Now that we have analysed and extracted features, we are ready to build a classifier. The machine learning approaches described earlier are supervised learning techniques, so we split our data into 140 records for training, and 60 for testing.

Each combination of the feature groups was used to build a decision tree from the training data, and then evaluated on the test data. We computed the exact match ratio accuracy (Subset accuracy), which is the percentage of samples that the classifier has assigned labels to correctly. Summarised in Graph 4.1 and Table A.8.

$$ExactMatchRatio = \frac{1}{n}\sum_{i=1}^{n} X_{predicted} == X_{label}$$

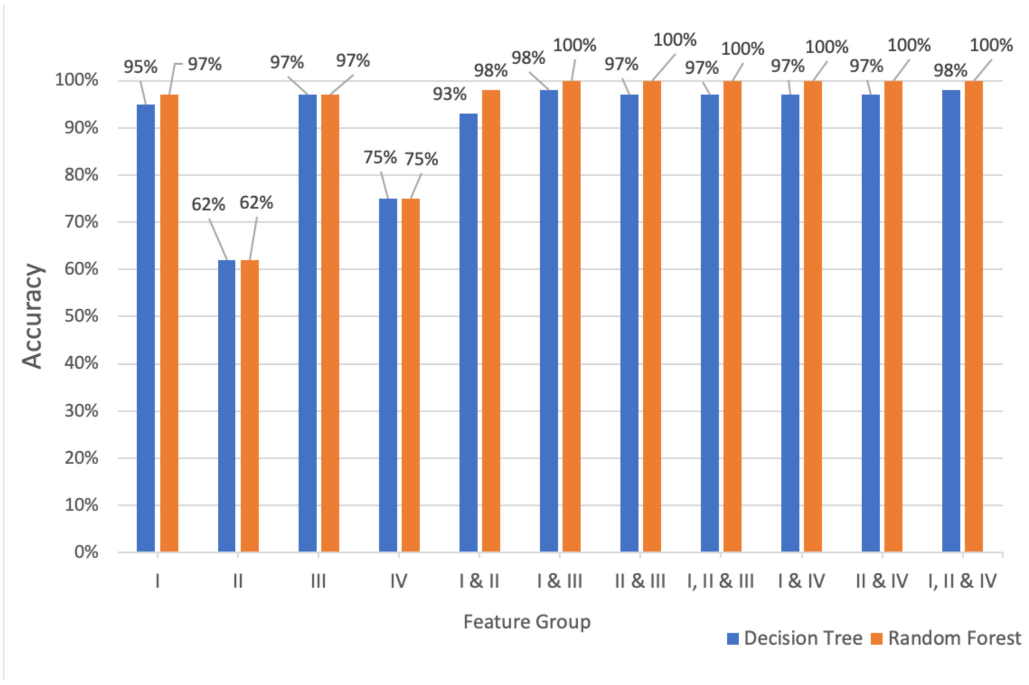


Figure 4.1: Graph showing the accuracy of the decision tree and random forest classifiers built with different groups of features on the test data

We achieve accuracy scores of above 90% on our test data with many singular decision trees and random forest classifiers. The lowest accuracy is for the classifiers using group II. By looking at the confusion matrix (Figure 4.2), we can see that much of this error comes from features labeled incorrectly as the SAT Service scan. If we look again at the data from group II (Table A.4), we can see that the classes incorrectly labeled as SAT Service are all alike in that they only have one IP and one port, meaning that with just these features it would be impossible for any classifier to reliably label them accurately.

Group III achieved an accuracy score 22% higher than it's binary twin, group IV. Just detecting the binary presence of certain packet types would likely be considerably faster in terms of computation, especially over large, highly active networks. However, here we can see that the unbalanced trade-off between speed and accuracy indicates that it is better to use the number of packet types.

4.2.1 Kepware

In a real scenario, the PLC would be receiving additional noise from many other devices in the network. For example Kepware, a data historian, collects data from devices by regular reading variables from them. We simulated this 'noisy' traffic by creating a Python script that connected to the PLC and sent request packets to read the PLC memory once every second (Figure 4.3). We then reran the scanners alongside this noisy traffic, and collected the results. We tested each of our already built classifiers on this new data to see how well it would generalise. The Kepware traffic generated added packets that used S7Comm on top of COTP which in turn was on top of TCP.

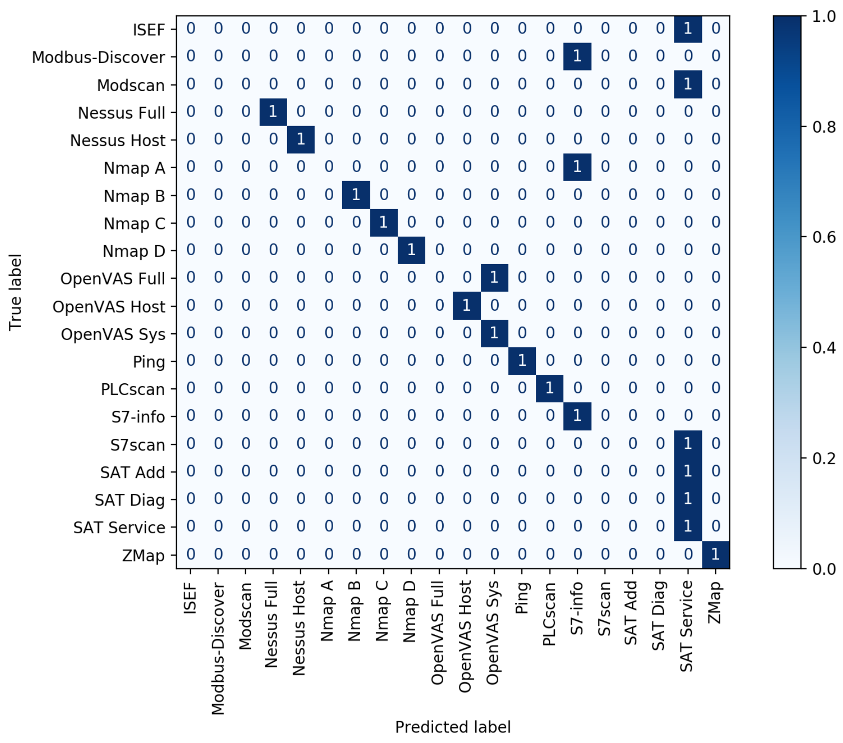


Figure 4.2: Confusion matrix for random forest classifier using group II

The accuracy of the decision tree classifiers dropped considerably when presented with packet captures including the Kepware traffic (Figure 4.4). Looking at the confusion matrix of group II - one of the most accurate classifiers (Figure 4.2), we found that the traffics classified incorrectly all shared the fact that before the Kepware noise was added, they had no S7Comm packets. Furthermore, we can verify this hypothesis by looking at one of the decision trees taken from the forest (Figure A.1) where we can see that a node at a high level in the tree used S7Comm packets.

The random forest classifiers consistently outperformed the decision tree classifiers. When presented with the Kepware traffic, this out-performance was increased further. Random forest classifiers use a combination of bootstrap aggregation and a random selection of features, which reduces overfitting, hence the greater improvement with Kepware traffic where the captures were less similar.

```python
import snap7
from snap7 import *
import time
plc = snap7.client.Client()
plc.connect("192.168.2.10", 0, 1)
area = 0x82     # area for Q memory
start = 0       # location we are going to start the read
length = 1      # length in bytes of the read
bit = 0         # which bit in the Q memory byte we are reading
while True:
    byte = plc.read_area(area,0,start,length)
    # print("Q0.0:", get_bool(mbyte,0,bit))
    time.sleep(1)
plc.disconnect()
print(plc.get_cpu_state())
```

Figure 4.3: The script we used to generate simulated Kepware traffic

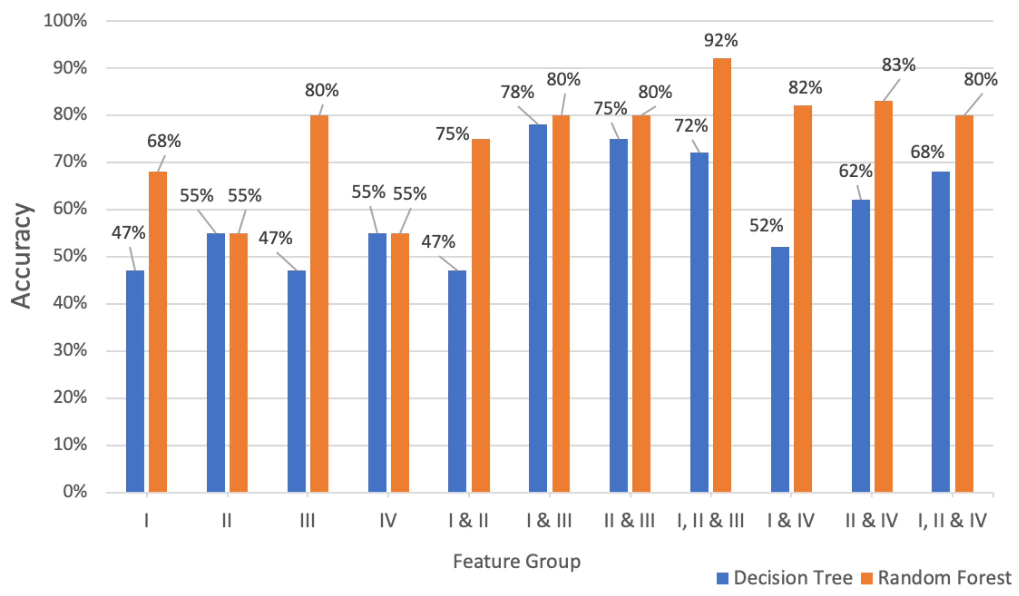


Figure 4.4: Graph showing the accuracy of the decision tree and random forest classifiers built with different groups of features on the Kepware data

Chapter 5

Discussion

The security of ICS is vital to the critical industrial infrastructure of our nation. One factor that remains weak in ICS security is intrusion detection. We propose a novel technique to use with existing IDS solutions, that would strengthen security by reducing the number of false positives and providing additional information points about a possible intrusion. As with other IDS techniques, this method can be carried out in parallel to the running of ICS, so we would expect no additional overhead.

We captured traffic from 10 different ICS asset discovery scanning tools against a real PLC in a testbed simulating a real ICS system. We then extracted features from each of the traffic captures, and used different subsets of these to train random forest classifiers, which we evaluated on a set of unlabeled test data. We found that with an exact match accuracy of 62%, the least accurate of these subsets was the group that included only the number of unique target ports and target IP addresses, whilst the most accurate of these individual groups used the number of many different types of packet achieving an accuracy of 97%, whilst the subsets that used a combination of groups achieved accuracy scores of 100%. We then simulated Kepware traffic that would be found in a real system, by writing a script that would send S7Comm packets to the PLC. We tested our existing classifiers with this new traffic, and found that the accuracy dropped to around 80%. The classifier with the greatest accuracy of 92% on this new traffic used several general flow features, along with the number of unique IP addresses, ports, and different packet types. Whilst the accuracy score of 92% is desirable, if the classifier was trained with the Kepware traffic, we anticipate that the error would shrink towards 0%. This highlights an important point: where this technique is to be implemented into a system, the classifier should be trained on the system, and possibly retrained when changes are made in order to achieve highest accuracy. Although this is an undesirable constraint, this could be carried out automatically, and would still provide the same benefits to security regardless.

Linking back to our original research aim, we have found that it is in fact possible to detect the asset discovery tool that generated some scanning traffic, and here we discuss the viability of using this technique as an additional vector in IDS. First, we explain the process of how one could integrate our findings with existing IDS, so that the type of asset discovery scanning tool could be used to aid intrusion detection. Firstly, the IDS must be given some prior knowledge of which scanning tools are typically used by engineers for legitimate purposes on the network, and similarly, which scanning tools are not. Then when some probing activity is detected, the IDS can use our technique to ascertain which scanning tool is being used. If the scanning tool is on the white-list provided by the operators, the risk is lowered since it is more likely be an engineer performing some innocuous operation, however if the scanning tool is not on the white-list, the risk is raised since it is more likely to be from an adversary. This would reduce the chance of successful attacks by limiting the large number of false positives that SOC analysts must trawl through, reducing the time before an intrusion is noticed, and providing more information about potential attacks.

There are however, limitations to this. Firstly, as we mentioned earlier in the introduction, we can only detect active scanners on the network, so any form of passive scanner would not be able to be detected. We are also limited in that we used a small number of scanners - though more scanning tools are available, and could trivially be added to the classifier.

The technique we described relies on the accurate detection and separation of asset discovery scanning traffic. Detection tools such as SiLK, and IDS such as Snort and Bro all have a capacity for the detection of asset discovery scans [5]. Jammes and Papadaki discuss the ability for one of the scanning tools we used - NMap - to evade detection by Snort [71]. They found that some of the evasion techniques were successful in evading detection, whilst others could be detected by appending additional rules to Snort.

On the topic of evasion, Gardiner and Nagaraja put forward a survey focusing on the evasion resilience of the ML component(s) of command and control detection systems [43]. Gardiner and Nagaraja detail the increasing number of evasion techniques used to evade ML detection, which contrasts the lack of literature around the evasion resilience of these ML detection techniques. There are several techniques used in the paper which specifically can avoid random forest classifiers (Mimicry, Wagner and Soto and Tree ensemble evasion, Kantchelian et al. [72, 73]). Whilst these techniques have not yet been used in literature to evade scanning, they are certainly viable.

To understand this next limitation, let us imagine that an adversary is performing an attack on our network, with the PLC as the target. After entering our network and traversing to the part of our network with PLCs, the adversary could use some form of passive observation or man-in-the-middle to obtain traffic passing over the network. If the adversary is able to collect the network traffic, they could use a technique similar to ours to ascertain which scanning tool(s) are used by engineers and admins. This may take a long period of time, and the adversary would need to go unnoticed. 'Low and slow' attacks are not all too uncommon from somewhat sophisticated adversaries. The adversary could now use the same tool that the engineers use, which would hit the white-list of the IDS. This may prove to hide the adversary's behaviour on the network. As such, an IDS using this technique should not just completely allow traffic that matches the white-list of tools, but instead a non-matching scanning tool should assign some weight that increases the risk if not on the whitelist.

Chapter 6

Conclusion

6.1 Future Work

The unique merge of modern and legacy OT equipment with IT and IIoT devices means that there is an expanse of interesting and vital work to be done in and around ICS security, and due to our reliance on CNI, it is now more important than ever. This paper showed that fingerprinting scanning tools is a viable vector to use in IDS, but further work is certainly possible. In this final part of the dissertation, we will outline some possible routes for future work.

Wider Range of Tools

We attempted to use a variety of common tools used to probe ICS networks, but the complete list of asset discovery tools is vastly expansive. A general investigation into a more complete list of ICS scanning tools is being carried out by M. Samanis at the University of Bristol. It would be suitable to expand the classification to include these tools.

Integrate with existing IDS

Earlier we mentioned existing IDS tools such as SENAMI, Zeek and Snort [45, 74, 6]. Our technique relies on a scan being detected and the traffic already presented. If our technique was combined with these IDS tools, we would be able to see how well it performs in tandem with these.

More Accurate Traffic

The Kepware traffic we introduced in Chapter 4 provided a suitable simulation of the real noise that may be present in an ICS network, but the best way of representing this traffic in our data would be to collect the traffic in a real system. Further work could include repeating experiments using a test bed where real Kepware or other SCADA system traffic is occurring.

6.2 Conclusion

In this dissertation we aimed to investigate the viability of fingerprinting the asset discovery tools used to probe an OT network as an additional vector to use in IDS. First, in Chapter 2 we discussed the background of asset discovery alongside OT and the security of OT. In Chapter 3 we discussed the methodology for the proposed technique of fingerprinting asset discovery tools by using a random forest classifier using features extracted from network traffic. Next, in Chapter 4 we presented and analysed the results from the features extracted from the traffic collected and evaluated our different classifiers, and in Chapter 5 we discussed our findings and how this new vector could be used for IDS.

In answer to our first research aim, **can we identify which asset discovery tool is being used to probe an OT network** - with just traffic from each of the asset discovery scanning tools we were able to achieve exact match accuracy scores of 100%, and with added Kepware traffic (simulating real traffic on a network) we achieved an accuracy score of 92% - showing that it is indeed possible to fingerprint the asset discovery tools used to probe an OT network. In Chapter 5 we answered our second question, **can we use intelligence about which asset discovery tool being used to probe an OT network as an additional vector to use for intrusion detection?** Through our results, we show that an IDS module could feasibly be built to identify different OT scanners.

By using this technique to reduce the number of false positives generated by an IDS, the difference in time between an intrusion and action against the intrusion would be reduced, strengthening OT security.

Bibliography

[1] D. Kushner, "The real story of stuxnet," *IEEE Spectrum*, vol. 50, pp. 48–53, 2013.

[2] FireEye, B. Johnson, D. Caban, M. Krotofil, D. Scali, N. Brubaker, and C. Glyer, *Attackers Deploy New ICS Attack Framework "TRITON" and Cause Operational Disruption to Critical Infrastructure.* 2017.

[3] CISA, *Cyber-Attack Against Ukrainian Critical Infrastructure.* 2016.

[4] K. Coffey, R. Smith, L. A. Maglaras, and H. Janicke, "Vulnerability analysis of network scanning on scada systems," *Security and Communication Networks*, vol. 2018, pp. 3794603:1–3794603:21, 2018.

[5] CERT, *SiLK RWScan.* https://tools.netsa.cert.org/silk/rwscan.html.

[6] M. Roesch, *Snort.* Cisco Systems.

[7] M. Deere, *Battersea Power Station Photos.* 2014. http://mikedeere.com/battersea-power-station/.

[8] *Thames Barrier Control Room.* Port of London Authority.

[9] *Rockwell CompactLogix 5380.* Rockwell Automation. https://twitter.com/rokautomation/status/831572529529622530.

[10] Siemens, *Photograph of Water Pumps.* https://new.siemens.com/global/en/products/automation/industrial-controls/media/hybrid-3rw5-pump-stopping-function.html.

[11] D. Beresford, "Exploiting siemens simatic s7 plcs," *Black Hat USA*, vol. 16, no. 2, pp. 723–733, 2011.

[12] H. K. Hui and K. McLaughlin, "Investigating current plc security issues regarding siemens s7 communications and tia portal," 2018.

[13] B. Lim, D. Chen, Y. An, Z. Kalbarczyk, and R. Iyer, "Attack induced common-mode failures on plc-based safety system in a nuclear power plant: Practical experience report," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 205–210, Jan 2017.

[14] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Communications surveys & tutorials*, vol. 15, no. 2, pp. 860–880, 2012.

[15] A. Pauna, K. Moulinos, M. Lakka, J. May, and T. Tryfonas, "Can we learn from scada security incidents," *White Paper, European Union Agency for Network and Information Security, Heraklion, Crete, Greece*, 2013.

[16] J. Matherly, *Shodan.* 2009. https://www.shodan.io/.

[17] L. M. Corporation, E. M. Hutchins, M. J. Cloppert, and R. M. Amin, *Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains.* 2015.

[18] SANS, M. J. Assante, and R. M. Lee, *The Industrial Control System Cyber Kill Chain.* 2015.

[19] CISA, *ICS-ALERT-14-176-02A.* Jun 2014. https://www.us-cert.gov/ics/alerts/ICS-ALERT-14-176-02A.

[20] "The state of industrial cyber security," *Network Security.*

[21] J. Tidy, *How a ransomware attack cost one firm £45m.* Jun 2019. https://www.bbc.com/news/business-48661152.

[22] S. McClure, "Operation cleaver," *Cylance Report, December*, 2014.

[23] *Critical National Infrastructure.* CPNI. https://www.cpni.gov.uk/critical-national-infrastructure-0.

[24] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 199–212, 2009.

[25] *BigFix - Systems-management software product.* IBM. www.bigfix.com.

[26] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *40th IEEE Symposium on Security and Privacy (S&P'19)*, 2019.

[27] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018.

[28] J. Nivethan and M. Papa, "A linux-based firewall for the dnp3 protocol," in *2016 IEEE Symposium on Technologies for Homeland Security (HST)*, pp. 1–5, IEEE, 2016.

[29] B.-K. Kim, D.-H. Kang, J.-C. Na, and T.-M. Chung, "Abnormal traffic filtering mechanism for protecting ics networks," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pp. 436–440, IEEE, 2016.

[30] M. W. Boyce, K. M. Duma, L. J. Hettinger, T. B. Malone, D. P. Wilson, and J. Lockett-Reynolds, "Human performance in cybersecurity: A research agenda," in *Proceedings of the Human Factors and Ergonomics Society annual meeting*, vol. 55, pp. 1115–1119, SAGE Publications Sage CA: Los Angeles, CA, 2011.

[31] W. J. Lynn III, *Defending a New Domain.* Sep 2010.

[32] N. Nissim, R. Yahalom, and Y. Elovici, "Usb-based attacks," *Computers & Security*, vol. 70, pp. 675–688, 2017.

[33] B. Green, D. Prince, J. S. Busby, and D. Hutchison, "The impact of social engineering on industrial control system security," in *CPS-SPC '15*, 2015.

[34] *TalkTalk cyber attack – how the ICO's investigation unfolded.* ICO. https://ico.org.uk/about-the-ico/news-and-events/talktalk-cyber-attack-how-the-ico-investigation-unfolded/.

[35] D. Norman, *Design of Everyday Things.*

[36] D. I. Buffey, D. R. Piggin, and Atkins, *Active defence using an operational technology honeypot.*

[37] M. Crosbie, B. Dole, T. M. Ellis, I. Krsul, and E. H. Spafford, "Idiot - users guide," 1996.

[38] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A rule-based intrusion detection approach," *IEEE Trans. Software Eng.*, vol. 21, pp. 181–199, 1995.

[39] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, P. Neumann, H. Javitz, and T. Garvey, "A real-time intrusion-detection expert system," 01 1992.

[40] G. Lyon, *Nmap: The Network Mapper.* https://nmap.org/.

[41] T.-H. Cheng, Y.-D. Lin, Y.-C. Lai, and P.-C. Lin, "Evasion techniques: Sneaking through your intrusion detection/prevention systems," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 1011–1020, 2011.

[42] T. H. Ptacek and T. N. Newsham, "Insertion, evasion, and denial of service: Eluding network intrusion detection," tech. rep., Secure Networks inc Calgary Alberta, 1998.

[43] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware cc detection: A survey," *ACM Comput. Surv.*, vol. 49, Dec. 2016. https://doi.org/10.1145/3003816.

[44] T. Pietraszek, "Using adaptive alert classification to reduce false positives in intrusion detection," in *Recent Advances in Intrusion Detection* (E. Jonsson, A. Valdes, and M. Almgren, eds.), (Berlin, Heidelberg), pp. 102–124, Springer Berlin Heidelberg, 2004.

[45] W. Jardine, S. Frey, B. Green, and A. Rashid, "Senami: Selective non-invasive active monitoring for ics intrusion detection," in *CPS-SPC '16*, 2016.

[46] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Understanding passive and active service discovery," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, (New York, NY, USA), p. 57–70, Association for Computing Machinery, 2007. https://doi.org/10.1145/1298306.1298314.

[47] J. Gonzalez and M. Papa, "Passive scanning in modbus networks," in *Critical Infrastructure Protection* (E. Goetz and S. Shenoi, eds.), (Boston, MA), pp. 175–187, Springer US, 2008.

[48] Dragos. https://www.dragos.com/.

[49] *GrassMarlin*. The (United States) National Security Agency. https://github.com/nsacyber/GRASSMARLIN.

[50] *Nessus*. Tenable, Inc. https://www.tenable.com/products/nessus.

[51] R. Floodeen and K. van Wyk, "Identifying network scanning tools," in *Identifying Network Scanning Tools*, 2008.

[52] M. Muuss, *Ping*. ping, 1983.

[53] Greenbone Networks, GmbH. https://www.openvas.org/.

[54] E. Dmitry, *PLCscan*. https://code.google.com/archive/p/plcscan/.

[55] Kapersky Lab Security Services. https://github.com/klsecservices/s7scan.

[56] L. B. Pig, *Industrial Exploitation Framework*. https://github.com/dark-lbp/isf.

[57] D. Kapellmann and R. Washburn, "Call to action: Mobilizing community discussion to improve information-sharing about vulnerabilities in industrial control systems and critical infrastructure," in *2019 11th International Conference on Cyber Conflict (CyCon)*, vol. 900, pp. 1–23, 2019.

[58] M. Bristow, *ModScan*. https://code.google.com/archive/p/modscan/.

[59] T. Z. Team, *ZMap*. https://zmap.io/.

[60] *Simatic Automation Tool*. Siemens.

[61] P. Asrodia and H. Patel, "Analysis of various packet sniffing tools for network monitoring and analysis," *International Journal of Electrical, Electronics and Computer Engineering*, vol. 1, no. 1, pp. 55–58, 2012.

[62] M. Shah, S. Ahmed, K. Saeed, M. T. Junaid, H. Khan, and A. ur rehman, "Penetration testing active reconnaissance phase – optimized port scanning with nmap tool," *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pp. 1–6, 2019.

[63] J. Chen and J. Revels, "Robust benchmarking in noisy environments," *arXiv preprint arXiv:1608.04295*, 2016.

[64] C. Moreno and S. Fischmeister, "Accurate measurement of small execution times—getting around measurement errors," *IEEE Embedded Systems Letters*, vol. 9, no. 1, pp. 17–20, 2017.

[65] G. Combs, *Wireshark*. The Wireshark Team. `www.wireshark.org`.

[66] K. Newt, *PyShark*. `https://github.com/KimiNewt/pyshark/`.

[67] M. A. M. Hasan, M. Nasser, B. Pal, and S. Ahmad, "Support vector machine and random forest modeling for intrusion detection system (ids)," *Journal of Intelligent Learning Systems and Applications*, vol. 2014, 2014.

[68] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, no. 1, pp. 213–217, 2016.

[69] A. Zainal, M. A. Maarof, S. M. Shamsuddin, *et al.*, "Ensemble classifiers for network intrusion detection system," *Journal of Information Assurance and Security*, vol. 4, no. 3, pp. 217–225, 2009.

[70] D. Cournapeau, *scikit-learn*. `https://scikit-learn.org/`.

[71] Z. Jammes and M. Papadaki, "Snort ids ability to detect nmap and metasploit framework evasion techniques," *Adv. Commun. Comput. Netw. Secur*, vol. 10, p. 104, 2013.

[72] D. Wagner and P. Soto, "Mimicry attacks on host-based intrusion detection systems," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 255–264, 2002.

[73] A. Kantchelian, J. D. Tygar, and A. Joseph, "Evasion and hardening of tree ensemble classifiers," in *International Conference on Machine Learning*, pp. 2387–2396, 2016.

[74] V. Paxson, *Zeek*. Open Source.

Appendix A

Appendix

Table A.1

	Version	Target IP	Command/Options
ICSEF	0.1.0	Single	s7comm_scan, min_slot=0, min_rack=0, max_slot=2, max_rack=2
Modbus-Discover	7.80	Single	nmap –script modbus-discover.nse –script-args='modbus-discover.aggressive=true' -p 502 192.168.2.10
ModScan	0.1	Single	python modscan.py 192.168.2.10
Nessus Full	8.9.1	Single	Basic Network Scan, Ports = Common Ports
Nessus Host	8.9.1	Range	Host Discovery Scan, Ports = Host Enumeration
NMap A	7.80	Single	nmap -p 102 192.168.2.10
NMap B	7.80	Single	nmap 192.168.2.10
NMap C	7.80	Range	nmap 192.168.2.0/24
NMap D	7.80	Range	nmap -p 102 192.168.2.0/24
OpeVAS Full	7.03	Single	Scan Config=Full & Fast, Scanner=OpenVAS default, Port List=OpenVAS default
OpeVAS Host	7.03	Range	Scan Config=Host Discovery, Scanner=OpenVAS default, Port List=OpenVAS default, Targets=192.168.2.0/24
OpeVAS Sys	7.03	Range	Scan Config=System Discovery, Scanner=OpenVAS default, Port List=OpenVAS default, Targets=192.168.2.0/24
Ping	s20190709	Single	ping 192.168.2.10, Ran for 20 iteratios
PLCscan	0.1	Single	python plcscan.py 192.168.2.10
s7-info	7.80	Single	nmap –script s7-info.nse -p 102 192.168.2.10
S7scan	1.03	Single	python s7scan.py –tcp 192.168.2.10
SAT Add	V03.01.04.01	Single	Edit->Insert->New Device
SAT Diag	V03.01.04.01	Single	Options->Show Diagnostics
SAT Sys	V03.01.04.01	Single	Options->Retrieve Service Data
ZMap	2.1.1	Range	zmap 192.168.2.0/24 -p 102

Table A.1: The versions and options of the asset discovery scanning tools used to generate traffic.

For the following tables of data, the means of different features were too varied, so we decided to show relative variances by using the coefficient of variance (**CoV**) which is calculated as $c_v = \frac{\sigma}{\mu}$. The accuracy is calculated as the exact match ratio (Equation 4.2).

Scanner	Total Packets	CoV	Packets to PLC	CoV
ISEF	46.9	2%	46.9	2%
Modbus-Discover	9.1	3%	9.1	3%
Modscan	1.2	33%	1.2	33%
Nessus Basic	9669.4	1%	9669.4	1%
NMap A	12.2	30%	12.2	30%
NMap B	1083.1	3%	1083.1	3%
NMap C	12111.4	3%	1272.6	5%
NMap D	4963.8	28%	9.8	37%
OpenVAS Full	58006.0	31%	58006.0	31%
OpenVAS Host	504.0	0%	1.0	0%
OpenVAS Sys	11171.1	6%	11171.1	6%
Ping	10.0	0%	10.0	0%
PLCscan	6.5	16%	6.5	16%
S7-info	27.6	5%	27.6	5%
S7scan	114.3	67%	114.3	67%
SAT Add	261.8	2%	261.8	2%
SAT Diag	416.5	0%	416.5	0%
SAT Service	1313.3	0%	1313.3	0%
ZMap	2018.1	5%	3.7	24%
All	5476.9	238%	4175.0	313%

Table A.2: Table showing the average number of packets generated by each scanner, alongside the respective coefficient of variance (CoV)

Scanner	Time	CoV	Rate	CoV	Packet Length	CoV
ISEF	37.0	2%	1.3	1%	57.7	0%
Modbus-Discover	0.2	119%	57.5	28%	52.2	1%
ModScan	0.2	200%	0.4	200%	68.0	0%
Nessus Basic	446.6	1%	21.7	1%	109.5	0%
Nessus Host	20.7	1%	376.6	1%	51.2	0%
NMap A	0.4	165%	62.4	34%	50.8	2%
NMap B	3.5	62%	381.8	35%	67.7	0%
NMap C	51.2	55%	272.9	28%	67.9	0%
NMap D	9.7	62%	592.6	25%	68.0	0%
OpenVAS Full	2425.9	101%	43.8	45%	130.8	10%
OpenVAS Host	60.9	0%	8.3	0%	32.0	0%
OpenVAS Sys	317.2	2%	35.2	6%	130.0	2%
Ping	9.0	0%	1.1	0%	88.0	0%
PLCscan	0.5	98%	46.9	73%	57.9	1%
S7-info	0.5	17%	55.0	12%	54.5	1%
S7scan	4.2	79%	27.9	31%	58.4	1%
SAT Add	30.5	5%	8.6	4%	66.3	1%
SAT Diag	32.5	2%	12.8	2%	61.6	1%
SAT Service	46.5	1%	28.2	1%	63.6	0%
ZMap	8.0	20%	259.5	16%	68.0	0%
All	175.3	309%	114.7	146%	70.2	37%

Table A.3: Table showing the average time taken, rate and packet length of the traffic generated by each scanner, alongside the respective coefficient of variance (CoV)

Scanner	Unique IP Addresses	CoV	Unique Ports	CoV
ISEF	1	0%	1	0%
Modbus-Discover	1	0%	3	0%
Modscan	1	0%	1	0%
Nessus Basic	1	0%	16	0%
Nessus Host	256	0%	23	0%
NMap A	1	0%	3	0%
NMap B	1	0%	1000	0%
NMap C	256	0%	1000	0%
NMap D	256	0%	2	0%
OpenVAS Full	1	0%	4481	0%
OpenVAS Host	254	0%	0	0%
OpenVAS Sys	1	0%	4481	0%
Ping	1	0%	0	0%
PLCscan	1	0%	2	0%
S7-info	1	0%	3	0%
S7scan	1	0%	1	0%
SAT Add	1	0%	1	0%
SAT Diag	1	0%	1	0%
SAT Service	1	0%	1	0%
ZMap	256	0%	1	0%
All	64.7	175%	551.1	250%

Table A.4: Table showing the average number of unique destination IP addresses and unique destination ports accessed, alongside the respective coefficient of variance (CoV)

Scanner	TCP	CoV	UDP	CoV	ICMP	CoV	HTTP	CoV	S7COMM	CoV	COTP	CoV
ISEF	46.9	2%	0	0%	0	0%	0	0%	6	0%	15	0%
Modbus-Discover	9.1	3%	0	0%	0	0%	0	0%	0	0%	0	0%
Modscan	1.2	33%	0	0%	0	0%	0	0%	0	0%	0	0%
Nessus Basic	9583.9	1%	77.5	2%	8	0%	1284.3	0%	0	0%	1	0%
Nessus Host	53.1	8%	10.4	12%	1	0%	5	0%	0	0%	0	0%
NMap A	12.2	30%	0	0%	0	0%	0	0%	0	0%	0	0%
NMap B	1083.1	3%	0	0%	0	0%	0	0%	0	0%	0	0%
NMap C	1272.6	5%	0	0%	0	0%	0	0%	0	0%	0	0%
NMap D	9.8	37%	0	0%	0	0%	0	0%	0	0%	0	0%
OpenVAS Full	58004	31%	0	0%	2	0%	8805.1	26%	3	0%	7	0%
OpenVAS Host	0	0%	0	0%	1	0%	0	0%	0	0%	0	0%
OpenVAS Sys	11169.1	6%	0	0%	2	0%	1124.3	0%	3	0%	7	0%
Ping	0	0%	0	0%	10	0%	0	0%	0	0%	0	0%
PLCscan	6.5	16%	0	0%	0	0%	0	0%	0	0%	0.7	65%
S7-info	27.6	5%	0	0%	0	0%	0	0%	4	0%	6	0%
S7scan	114.3	67%	0	0%	0	0%	0	0%	6	0%	36.8	67%
SAT Add	246.7	2%	15.1	2%	0	0%	0	0%	0	0%	136.6	0%
SAT Diag	407.5	0%	9	0%	0	0%	0	0%	0	0%	224.6	0%
SAT Service	1304.3	0%	9	0%	0	0%	0	0%	0	0%	749.5	0%
ZMap	3.7	24%	0	0%	0	0%	0	0%	0	0%	0	0%
All	4167.7	313%	6.1	288%	1.2	230%	560.9	352%	1.1	189%	59.2	291%

Table A.5: Table showing the presence of protocols in each scanner averaged across the sample data, alongside the respective coefficient of variance (CoV)

Scanner	Port 80	CoV	Port 102	CoV	Port 443	CoV	Port 502	CoV
icsef	0	0%	46.9	2%	0	0%	0	0%
Modbus-Discover	4.1	7%	0	0%	4	0%	1	0%
Modscan	0	0%	0	0%	0	0%	1.2	33%
Nessus Basic	7254.1	1%	69.3	7%	2241.5	0%	0	0%
Nessus Host	22.6	4%	0	0%	5.2	12%	0	0%
NMap A	4.6	39%	3	0%	4.6	39%	0	0%
NMap B	9.3	42%	0	0%	9	33%	0	0%
NMap C	14	23%	0	0%	4.1	7%	0	0%
NMap D	6.4	50%	3.4	35%	0	0%	0	0%
OpenVAS Full	33819.9	24%	80.3	4%	18996.4	125%	1	0%
OpenVAS Host	0	0%	0	0%	0	0%	0	0%
OpenVAS Sys	2624.2	0%	72.6	2%	3423.5	1%	1.6	75%
Ping	0	0%	0	0%	0	0%	0	0%
PLCscan	0	0%	5.4	17%	0	0%	1.1	27%
S7-info	4.4	18%	19	0%	4.2	14%	0	0%
S7scan	0	0%	114.3	67%	0	0%	0	0%
SAT Add	0	0%	246.7	2%	0	0%	0	0%
SAT Diag	0	0%	407.5	0%	0	0%	0	0%
SAT Service	0	0%	1304.3	0%	0	0%	0	0%
ZMap	0	0%	3.7	24%	0	0%	0	0%
All	2188.2	349%	118.8	250%	1234.6	346%	0.30	182%

Table A.6: Table showing the presence of target ports in traffic of each scanner averaged across the sample data, alongside the respective coefficient of variance (CoV)

Scanner	HTTP GET	CoV	HTTP POST	CoV	S7 ID=0	CoV	S7 ID=1c	CoV	S7 Index=0	CoV	S7 Index=1	CoV	echoRequests	CoV
ISEF	0	0%	0	0%	2	0%	0	0%	4	0%	0	0%	0	0%
Modbus-Discover	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Modscan	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
Nessus Basic	1253.1	0%	17	0%	0	0%	0	0%	0	0%	0	0%	1	0%
Nessus Host	5	0%	0	0%	0	0%	0	0%	0	0%	0	0%	1	0%
NMap A	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
NMap B	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
NMap C	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
NMap D	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
OpenVAS Full	8160.9	26%	536.6	51%	1	0%	0	0%	0	0%	2	0%	2	0%
OpenVAS Host	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	1	0%
OpenVAS Sys	1124.3	0%	0	0%	1	0%	0	0%	0	0%	2	0%	2	0%
Ping	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	10	0%
PLCscan	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
S7-info	0	0%	0	0%	1	0%	0	0%	0	0%	3	0%	0	0%
S7scan	0	0%	0	0%	0	0%	2	0%	4	0%	0	0%	0	0%
SAT Add	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
SAT Diag	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
SAT Service	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
ZMap	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%	0	0%
All	527.2	348%	27.7	433%	0.3	220%	0.1	447%	0.4	308%	0.35	250%	0.85	265%

Table A.7: Table showing the average number of packets that have different HTTP methods, S7Comm packet types and echo requests in each scanner, alongside the respective coefficient of variance (CoV)

Group	DT	RF
I	95%	97%
II	62%	62%
III	97%	97%
IV	75%	75%
I & II	93%	98%
I & III	98%	100%
II & III	97%	100%
I, II & III	97%	100%
I & IV	97%	100%
II & IV	97%	100%
I, II & IV	98%	100%

Table A.8: The exact match accuracy of the decision tree and random forest classifiers built with different groups of features on the test data

Group	DT	RF
I	47%	68%
II	55%	55%
III	47%	80%
IV	55%	55%
I & II	47%	75%
I & III	78%	80%
II & III	75%	80%
I, II & III	72%	92%
I & IV	52%	82%
II & IV	62%	83%
I, II & IV	68%	80%

Table A.9: The exact match accuracy of the decision tree and random forest classifiers built with different groups of features on the kepware data

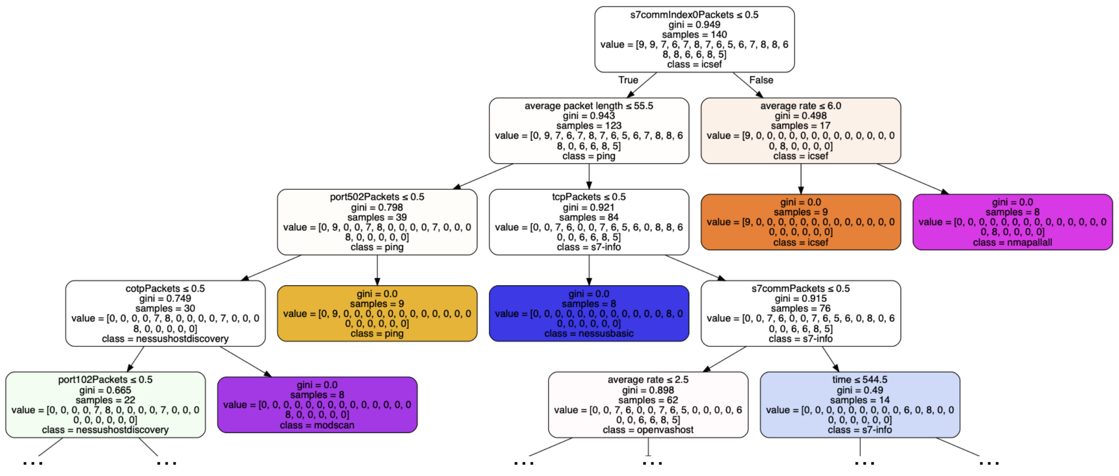


Figure A.1: A tree selected at random from the random forest classifier built using groups I, II & III, limited to a depth of 5 out of 11. The coloured nodes represent nodes with low or zero impurity. We can see a node in the fourth layer using the number of S7comm packets, demonstrating its importance.