Application of Approximate Matching on Industrial Control System (ICS) Network Communication Using Ssdeep Algorithm

Mutua, Nelson Makau

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in Information System Security at Strathmore University

Faculty of Information Technology

Strathmore University

Nairobi, Kenya

June, 2020

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by Strathmore University or any other institution. To the best of my knowledge and belief, the dissertation contains no material previously published or written by any other person expect where due reference is made in the dissertation itself.

Mutua Nelson Makau

18th June 2020

Approval

The dissertation of Mutua Nelson Makau was reviewed and approved by the following:

Dr. Petr Matoušek,
Faculty of Information and Technology,
Strathmore University.

Dr. Joseph Orero,
Dean, Faculty of Information and Technology,
Strathmore University.

Dr. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University.

ABSTRACT

Industrial Control Systems (ICSs) are significant for functioning of numerous critical infrastructures for instance power plants, water treatment facilities and gas pipelines. In spite of the fact that security of such systems deserves attention, application of thorough security intelligence approaches to ICS is not a standard practice. ICS are becoming more and more connected, so they require heightened security. Intrusion Detection Systems (IDSs) do not work well to secure ICSs because they mostly work on a signature basis and there are not many known signatures to detect attacks on ICSs.

Network communication is associated with many security challenges. Changes in Internet technologies have allowed for an increase in networked devices, the complexity of cybercrimes and the transfer of huge amounts of data, which can easily be intercepted and manipulated by attackers. Due to vulnerabilities in IDS used in ICS, there is need for a solution that can detect attacks at a higher rate. There have been several real-world documented incidents and cyber-attacks affecting ICSs which clearly illustrates critical infrastructure vulnerabilities. These reported incidents demonstrate that cyber-attacks on ICSs might cause a variety of financial damage and harmful events to humans and their environment.

Based on the aforementioned challenges, the solution was actualized by implementing a technique for International Electrotechnical Commission (IEC) 60870-5 also known as IEC 104 network communication protocol analysis based on approximate pattern matching. This protocol was intentionally selected in this study because it is crucial for the communication control and the controlled stations in many ICSs. ICS profile was computed from normal ICS network communication. To detect anomalies, unknown ICS communication was compared to the profile using approximate pattern matching algorithm.

This prototype applied Agile Software Methodology, for building of an evaluation tool. It provides opportunities to assess the project progress and direction throughout the development lifecycle. This is achieved though iterations and more frequent release with subsequent feedback. A python-based application was developed, tested and validated.

Keywords: *Approximate matching, Vulnerability, Network traffic analysis, Industrial Control System (ICS), International Electro technical Commission (IEC 104).*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

ACKNOWLEDGEMENTS

First, I give thanks to God Almighty for giving me the strength, health and capacity to do the dissertation. Secondly, I am grateful to my supervisor Dr. Petr Matoušek for the handholding, patience and constant encouragement throughout the project. To my father and sisters for their constant support and motivation in my studies. Lastly, I would like to acknowledge my friends, colleagues, students, teachers, archivists, and other librarians as well who assisted, advised, and supported my research and writing efforts over the years.

DEDICATIONS

I dedicate this dissertation to my parents and siblings who saw that I never lacked when I was schooling, not forgetting all other parties involved in the project including and in particular the management of @iLabAfrica for the award of the scholarship for Master of Science in Information System Security.

ABBREVIATIONS / ACRONYMS

APCI	-	Application Protocol Control Information
ASDU	-	Application Service Data Unit
BBH	-	Block-Based Hashing
BBR	-	Block-Based Rebuilding
BUT	-	Brno University of Technology
CTPH	-	Context Triggered Piecewise Hashing
CTPH	-	Context-Triggered Piecewise Hashing
DCS	-	Distributed Control Systems
ERP	-	Enterprise Resource Planning Systems
FOD	-	Functional Oriented Design
IACS	-	Industrial Automation and Control Systems
ICS	-	Industrial Control System
IDS	-	Intrusion Detection System
IEC	-	International Electro technical Commission
MELISSA	-	Mining Event Logs for Intrusion in SCADA Systems
MES	-	Manufacturing Execution Systems
MIS	-	Management Information Systems
MITM	-	Man in-the-Middle
MRSH	-	Multi-resolution Similarity Hashing
PCL	-	Programmable Logic Controllers
PE	-	Pattern Engine
RTU	-	Remote Terminal Unit
SCADA	-	Supervisory Control and Data Acquisition
SDHASH	-	Similar Digest Hash
SFI	-	Statistically Improbable Features
TDD	-	Test Driven Development
UML	-	Unified Modelling Language

CHAPTER 1: INTRODUCTION

1.1 Background of the Study

Industrial Control Systems (ICS) incorporates numerous kinds of systems, among them, Engineering Workstations, Programmable Logic Controllers (PLC) and Supervisory Control and Data Acquisition (SCADA). Generally, ICSs are utilized at the core of critical infrastructure and usually exposed to public networks that are at a high risk of cyber-attack. The control procedures of critical infrastructures such as power plant, oil and gas facilities, chemical processing plants, traffic control systems, among others are becoming more and more vulnerable to external network threats (Leith & Piper, 2013). Computer security has moved past ordinary office systems, and attackers are currently likewise focusing on ICS (Bolzoni et al., 2012). The key reason as to why organisation networks are targets for hackers is to obtain confidential information. The contemporary network system is voluminous to such an extent that manual inspection for malware is illogical and a costly task for an organisation to perform.

There are many serious consequences of cyber-attacks against ICS including disruption of critical infrastructure, physical damage to plants, environment, and humans (Stouffer, Falco & Scarfone, 2011). An ICS-CERT report shows that attacks targeted to ICS are continuously increasing in the recent past. In the year 2013, trusted industrial partners reported 73 incidents to ICS-CERT. In 2014 and 2015 a total of 245 and 295 incidents were also reported respectively (Stouffer, Falco & Scarfone, 2015).

In recent history, the most critical attack against ICS is the Stuxnet attack (Langner, 2011), which is known as the first digital/cyber fighting weapon. The attack was meant to physically sabotage the Iranian nuclear program. In response to such kind of attacks, the U.S government in 2007 demonstrated how attackers could possibly destroy a power plant with only 21 lines of code. In addition, a similar attack was carried out in late 2015 and early 2016, whereby hackers sabotaged

two power distribution companies in Ukraine with an aim of shutting down electricity to more than 80,000 people in the country. Another case was witnessed in Germany in 2015 where attackers compromised a steel mill and tampered with a blast furnace, leading to massive destruction on the plant (Zetter, 2016).

Detection systems are used in most ICS communications. These detection systems build a baseline of normal traffic and system events. The baseline is normally built on how events should occur, how long they should run, how they should work, how they are configured and so forth. The baseline is used for purposes of comparing activities in the system, such that when there is a detection of a deviation in behaviour, the detection engine is activated (Vries et al., 2012). In ICS communication, physical process control variables might exhibit some noisy behaviours by nature, that might lead to a high possibility of false-positive rates for anomaly detectors and low detection rates for attacks.

Considering the attacks mentioned above, advanced security solutions capable of good ICS protection need to be implemented to prevent any critical damages. ICS operators are more cautious as any disruptions and stipulates that security measures are system specific and should be robust. Taking this into consideration, there is need for analysis of network capture files and creation of a standard network communication profile as a viable method for achieving improved ICS security.

Approximate pattern matching is a useful technique for similar input detection (e.g. different versions of a file) and embedded object detection (e.g. an image within a word document), although it is a new working field. Approximate pattern matching technique can also be utilized to detect a fragment. A fragment is a small piece of a file (Breitinger et al., 2014). It can also be used to identify similarities between digital objects such as storage media, files and network streams and so on. This is because the technique is based on the logic of analysing and selecting unique attributes of each

object, then compare them. The fingerprint of the object under being analysed is usually the collection of attributes.

For each network packet file analysed, a profile is generated. This packet's profile is compared to a normal communication profile. If the normal and the packet profile score differ with a high margin, it signifies that the payload of the packet might contain attacks hence an anomaly is reported. The proposed technique is the best because when compared to the existing ones, it is less complex to use and do not entail extensive configuration. The technique can be deployed and maintained in an easy manner because the only requirement is network packet capture files. Moreover, the proposed approach (approximate pattern matching) can be utilized for detecting malware, in that, it can detect a malicious code that adversaries usually modify a previously known version to evade IDS or virus scanners.

1.2 Problem Statement

The system security of IEC 104 has been proven to be problematic. According to recent research on this protocol, serious security vulnerabilities have been identified. These vulnerabilities include lack of data encryption and authentication mechanisms. This means an attacker can hijack and inject network communication with malicious information to IEC 104 protocol that can damage or cause harm to ICS communication. The current ICS security systems in place are unable to detect injected malicious packets in network communication because they do not analyse data at the binary level.

A network threat affects the integrity, confidentiality and the availability of data as well as the stability, reliability, and safety of physical devices. These are critical situations that traditional security solution such as intrusion detection systems and firewall have failed to consider, making them not to effectively protect ICS by preventing abnormal flow of data that would cause greater damage/harm. Therefore, it is important to ensure promotion of other security mechanisms under the precondition that the data flows in the industrial control network remain unaffected.

1.3 Research Objectives

i. To find out which approximate matching algorithm can be used to profile ICS network communication.

ii. To profile ICS communication using approximate matching algorithm.

iii. To design and develop an ICS network profiling tool using approximate matching algorithm.

iv. To test and validate the effectiveness in terms of accuracy of the solution.

1.4 Research Questions

i. Which approximate matching algorithm can be used to profile ICS network communication?

ii. How can ICS communication be profiled using approximate matching algorithm?

iii. How can approximate matching algorithm be used to design, develop and test an ICS network profile?

iv. How can the effectiveness in terms of accuracy of the solution be tested and validated?

1.5 Justification

This research leads to the development of an ICS network profiling tool. This tool was used to match known network profiles against other network profiles to identify anomalies in a network communication packet file. Approximate matching algorithm was used to compare the computed network profiles. This tool will help to improve the information security for ICS systems and network security in general. Network administrators will be in a position to easily check for network communication variations and determine the need to analyse packet capture files further to identify the specific malwares contained in a network.

1.6 Scope

This research applies a passive monitoring approach in IEC 104 communication protocol to analyse ICS network packet capture files from ICS. The problems identified under our research problem was

addressed through the development of a tool which was be able to analyse IEC 104 protocol and detect anomalies in ICS network packet capture files using ssdeep algorithm. The tool ought to be intuitive and effective for skilled ICS network administrators. With proper analysis of the ICS network packet capture files, the tool generates a matching score between the compared network packets capture files to show the percentage of attack packets contained in the capture files.

The major limitation is the study will focus on one family of ICS protocol, the IEC 104 protocol. The tool will also focus of on ICS network packet capture files to detect anomalies in network communication and not analyses of a live ICS network communication.

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

The objective of this chapter is to give details of related work performed by other researchers in a similar field as the topic of this dissertation. This chapter also gives an overview of ICS communication and basic operations of an ICS. Lastly, discussed in this chapter are the security threats facing ICS, recent attacks, security systems and approximate matching algorithms.

2.2 Industrial Control Systems

ICS is an inclusive terminology that describes the various types of related instrumentation ranging from devices, networks, systems, protocol and control systems that are necessary in industrial automation (Badiru, Ibidapo-Obe, & Ayeni, 2016). According to the Guide on Information Security by the National Institute of Standards and Technology, Information Control Systems are information systems that are used in the control of industrial procedures. These procedures include production, the manufacturing process, handling of products and distribution.

ICSs are vital systems to a company and the nation at large; vital in this case means that they must be handled very carefully in terms of their infrastructure and all the assets either virtual or physical, lack of which the result can be deadly and of great losses (Rieger et al., 2019). If compromised, ICSs and other critical systems for that matter, the results of the attack can have a huge impact on matters national security, public health and safety and can cripple the national economy (Thames & Schaefer, 2017). Compromise under the organizational level can lead to trespassing and disclosure of confidential information to the public and competitors, data manipulation, stealing of passwords as well as weakening of the security infrastructure of the organization, access denial and data loss all of which can affect the reputation of the organization as well as leading to huge losses. Figure 2.1 below shows a sample architecture of an ICS.

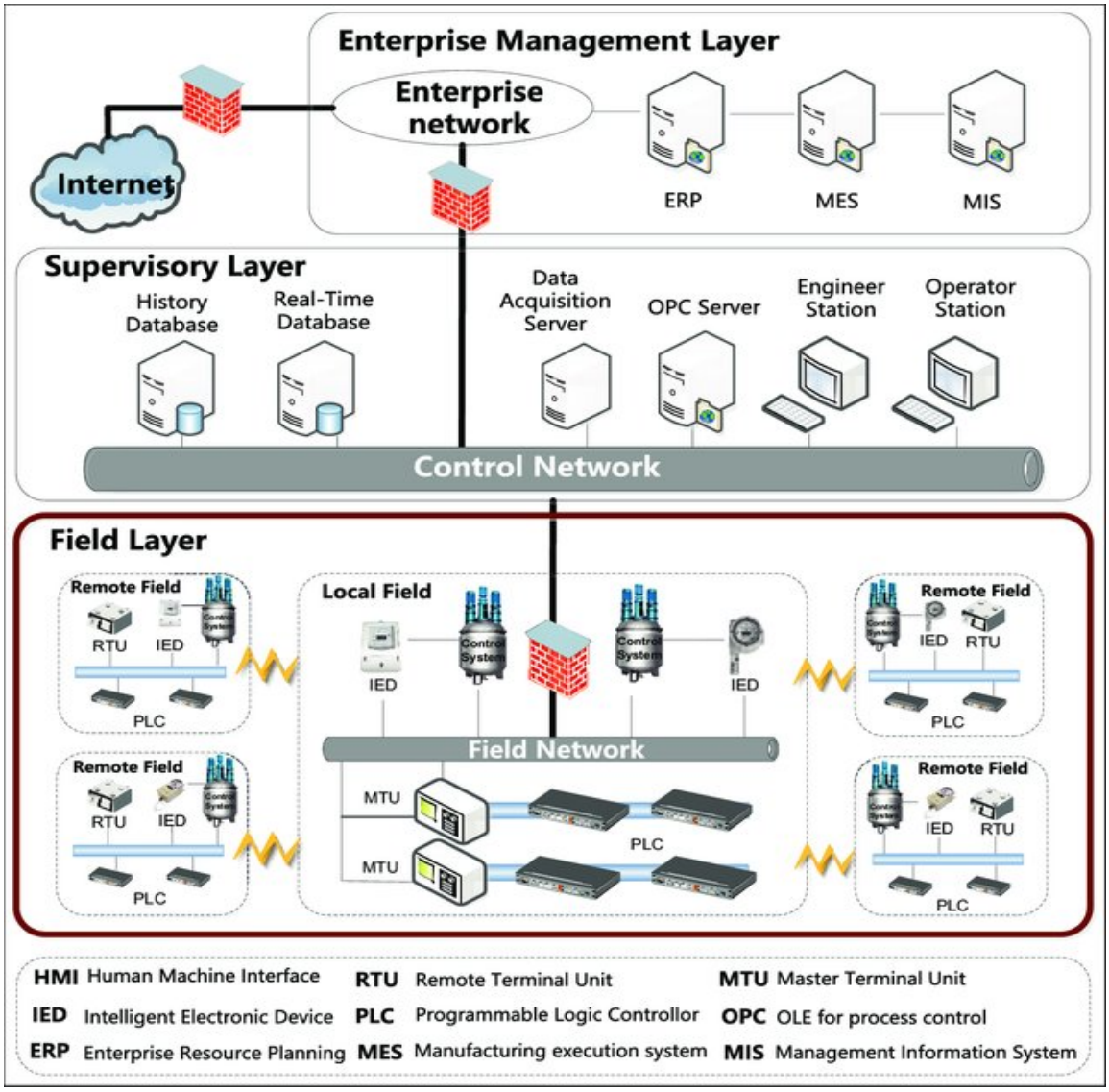


Figure 2.1: Architecture of Industrial Control System (Yang, 2018)

2.3 IEC 60870 Communication

IEC 104 messages are exchanged between the controlled and the controlling station. The controlled station is also called RTU slave. A master station commands the slave station. The controlling station (master station) performs controls of outstation. IEC 104 communication is delivered in the monitoring direction i.e. from the controlled station (slave) to the controlling station (master) or in the control direction i.e. controlling station (master) to the controlled station (slave). Figure 2.2 shows a topology of IEC 104 router connected with 104 SCADA monitoring systems using IEC104 protocol

over TCP/IP and IEC101 sensors communicating via Modbus RTU with the router (Matoušek, 2017).

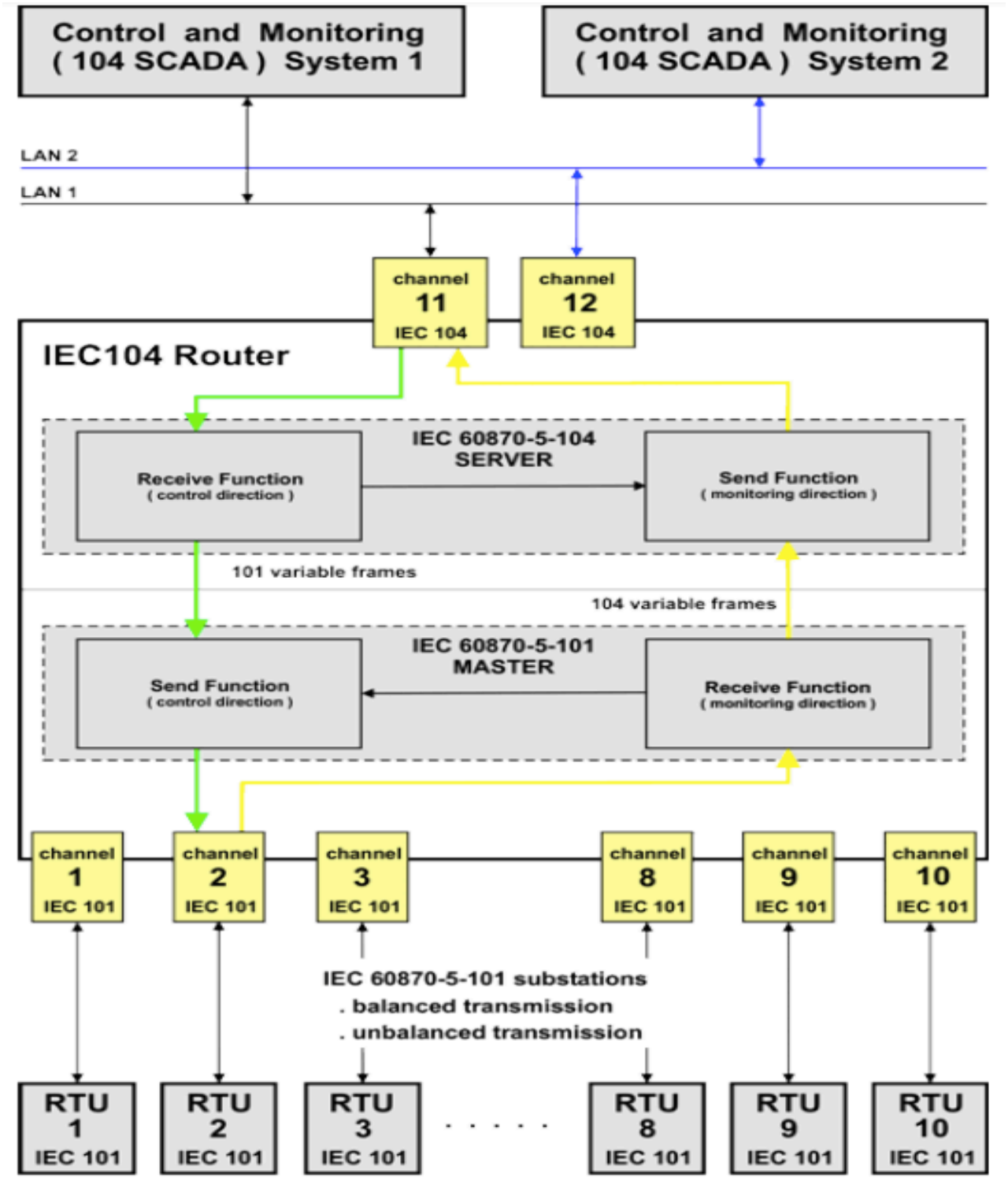


Figure 2.2: Network Topology of SCADA Monitoring System (Warner, 2012)

2.4 IEC 60870-5 - 104 Protocol

IEC 60870-5-104 is also commonly referred to as IEC 104. This SCADA protocol is defined within a collection of standards called IEC 60870. This protocol is a TCP/IP adaptation of the long-standing IEC 101 serial protocol, also known as IEC60870-5-101. IEC 101 defines the remote-control functionalities needed in extensive areas. Within the electrical industry, both IEC 104 and IEC 101 are used extensively to help in establishing communications links that connect the electrical control stations to the substations (Haverkort & Remke, 2018). The following section focuses on IEC 104 protocol.

The application layer of the IEC 104 consists of sub-layers. They are the Application Protocol Control Information (APCI) and Application Service Data Unit (ASDU) sub-layers. Above the TCP layer lies a sub-layer, the APCI, which defines three 104 message types. The formats consist: the (I-format) short for information transfer format, (S-format) known as the numbered supervisory functions and finally the (U-format) unnumbered control functions. U-format type of messages are used for initiating, halting and checking the connection statuses. On the other hand, the I-format type of messages transfer data two or more IEC 104 devices. Finally, S-format messages are crucial in acknowledging I-format messages previously received (Isakov et. al., 2014).

IEC 104 protocol makes use of APCI to establish the start and the end of the ASDUs. APCI starts with a start byte of 0x68 followed by the 8-bit length of APDU and four 8-bit control fields. APDU is made up of an APCI or an APCI with ASDU. An APCI is usually 6 bytes in terms of length as illustrated in figure 2.3 below.

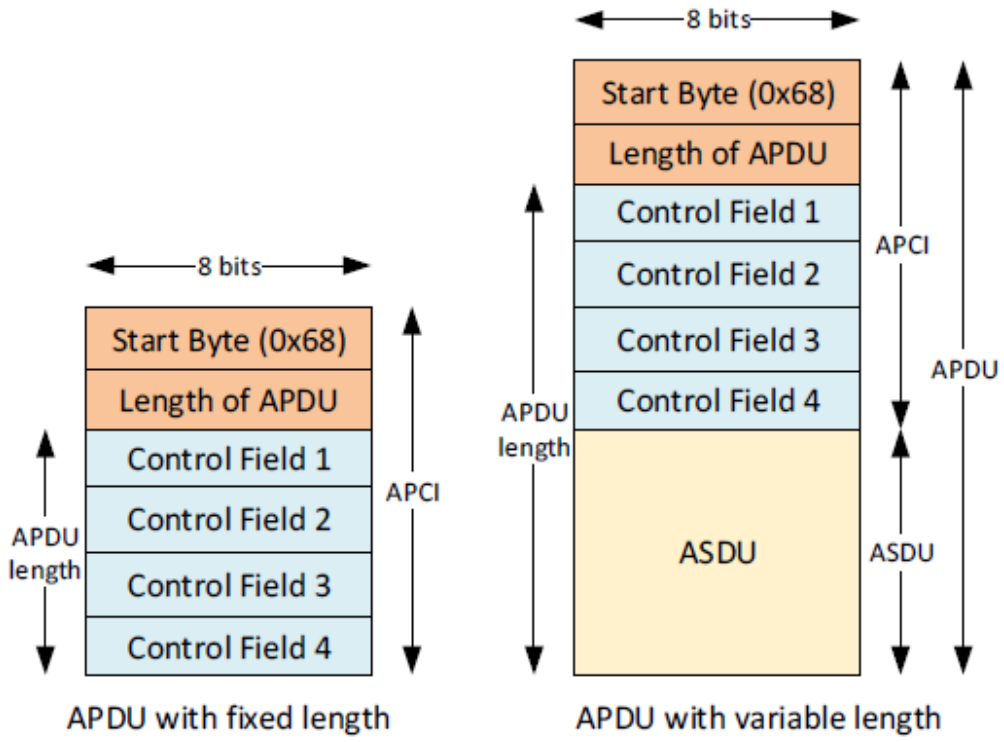


Figure 2.3: APCI Frame Format (Matoušek, 2017)

Data is arranged in terms of information objects in IEC 104. That means that each and every object has its own number of elements of information. The structure of information elements is defined and how they form each type of information object. The originator address used to identify the source stations is specified by the data unit identifier. This happens when more than one control station and an ASDU address fields are involved (Kerkers et. al., 2018). They establish the target station or even the address of the broadcast. Data unit identifier comes alongside several information objects, which are associated with one or more informational elements, an Information Object Address (IOA) and a non-compulsory time tag. To determine the formats of the information elements and their count, the fields in the data unit identifier are used. Figure 2.4 below illustrates the structure of an IEC 104 ASDU.

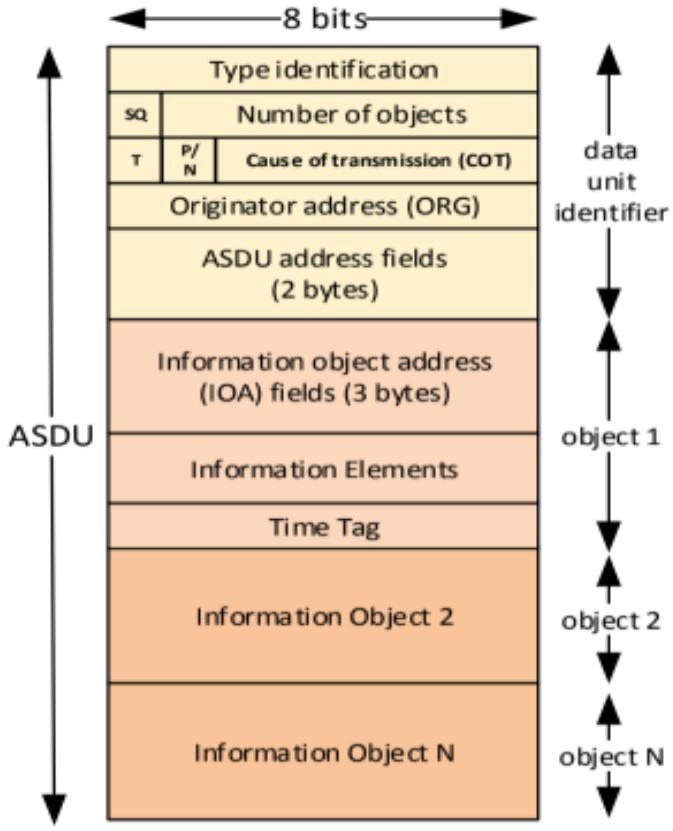


Figure 2.4: Structure of an IEC 104 ASDU (Matoušek, 2017)

2.5 Vulnerabilities Associated with IEC 104

The existing cyber vulnerabilities and attacks found within the physical and application layer of IEC 104 protocol are as follows;

2.5.1 Plaintext Mode Message Transmission

Due to data transmission in the form of clear text in IEC 104, information transfer between substations and the control centre is conceivably prone to high level cyber-attacks such as information alteration, tapping and espionage. For instance, a cyber-attacker can unveil a Man in-the-Middle (MITM) attack to help in sniffing as well as gathering remote signals, remote measurement values and remote-control commands (Papakonstantinou, 2015). For every case packet can be intercepted, modified and re-injected to the communications channel in order to compromise

the security and stability of the SCADA system. These malicious attacks expose the SCADA to future attacks.

2.5.2 Lack of Authentication Mechanism

Due to lack of authentication mechanism in ICS, malicious attackers can gain unauthorized access and compromise information integrity and availability. The attackers can also launch different type of attacks such as; MITM attacks, spoofing attacks or replay attacks. This is a huge system security loophole because lack of authentication provides hackers with a chance to gain easy access and this pauses a serious system vulnerability, which can lead to system compromise thus affecting operations, safety and may result into catastrophic damage (Case, 2016). For example, a false remote-control command such as "open the circuit breaker" could lead the power system to shed load affecting power supply reliability and at the same time exposing the operators and the large population to safety concerns.

2.6 Recent ICS Communication Attacks

ICS has experienced many cyber-attacks in the recent past. The number of ICS attack incidents is increasing according to ICS-CERT report 2018. Despite the difficulty of gaining process and control system understanding, the 2014 German Steel Mill Cyber Attack illustrated that adversaries learned sufficient details about the environment, leveraged specialized ICS knowledge, and caused multiple control system failures that ultimately led to a plant outage and consequential physical equipment damage (Lee, Assante, & Conway, 2014).

One of the notable attacks happened on December 23, 2015 "BlackEnergy" in a Ukrainian power company, causing power disruption to 225,000 customers, lasting up to 6 hours (Alert, 2016). This BlackEnergy attack was a multi-staged attack, starting with "spear phishing" emails targeting staff to gain access to the corporate network of the power company. Once inside the power company

network, attackers gathered credentials, and used VPNs to enter the ICS network. Attackers then used remote access tools to gain control of an HMI, and send commands to the ICS devices (Case, 2016). This example shows the potential for attacks to be conducted on ICS devices connected to the Internet through corporate networks.

Another notable attack on ICS devices is the Stuxnet worm. The Stuxnet worm used a series of zero-day vulnerabilities to gain access to operating systems running the software used to monitor and control frequency-converter drives used to power centrifuges in a nuclear power plant in Natanz, Iran. These centrifuges were used in the process to concentrate uranium-235 isotopes. The Stuxnet worm was used to alter the speed of the drives, causing them to alternate between fast and low speeds, causing the centrifuges to fail (Farwell & Rohozinski, 2012). Zero-day vulnerabilities are vulnerabilities which were previously unknown, meaning no patches, or signatures of these vulnerabilities are known to the public, increasing the likelihood the vulnerabilities can be exploited. In addition, the Stuxnet worm was spread through the use of USB-drives, showing that even offline ICS are at risk of attack.

According to Falliere, (2011), Stuxnet's multiple spreading techniques and the exploitation of four zero-day vulnerabilities manifests its level of sophistication. The Stuxnet malware infected Siemens PLCs to perform centrifuge cascade overpressure and centrifuge rotor speed operational manipulations at the Natanz Fuel Enrichment Plants marked a decisive change in the history of the ICS community (Langer, 2013). Stuxnet executed its attack by controlling all requests sent to Siemens Simatic PLCs by simply wrapping a library that was used to communicate with the PLCs. This action allowed the malware to install itself on the PLCs and take-over the communication between the PLC and WinCC. In this case, the attacker was able to gain a foothold on the PLCs and then directly issued commands to actuators. The malware effectively blinded any IDS deployed in

the supervisory layers. Since Stuxnet, ICS owners, operators and vendors have begun to take notice of their vulnerable systems and incorporate increased security controls.

As an example, Rockwell Automation ControlLogix PLCs now can encrypt their programs and lock access to routines (Bradley, 2018). Additionally, they have included new CIP security enhancements in their communication modules to encrypt Ethernet/IP protocol traffic between PLCs and drives (Rockwell Automation, 2019). For their S7 PLC product line, Siemens has implemented password block protection and three staggered CPU protection levels (Siemens, 2016).

These attacks on ICS show that the exposure to switched and routed networks to the Internet, places them at risk of cyber-attacks as in the case of the "BlackEnergy" attack on Ukraine. As such, there needs to be methods of detecting these cyber-attacks on the ICS. To detect these cyber-attacks, IDS have been used in industrial environments. In the next section, we outline the current ICS security systems.

2.7 Current ICS Security Systems

Many cyber-attackers have been targeting ICSs because an attack on ICS can destroy a company's rating, cause monetary loss and cause serious damages on the firm equipment well as the environment. Engineers design the logic flow of ICS controllers, with a primary aim of maintaining system stability and safety. All the same, the controllers are attached to a wide global network, which makes them vulnerable to many attacks (Gao, 2014). According to Bécue et al., (2016), the evolution of modern technology has brought with it a fair share of advantages and disadvantages. The migration of ICS from the traditional enclosed environment into an open world posed severe challenges to ICS. This prompts for development of measures to counter the rapid change in environment. This section discusses the current security mechanisms used in many ICS.

2.7.1 Intrusion Detection in Industrial Control Systems

Intrusion detection techniques are some of the security mechanisms to have been applied in the area of ICS security, primary in the area of signature-based intrusion detection systems. In this section we outline the current literature on the application of intrusion detection in ICS environments. Intrusion Detection Systems (IDS) are security systems that are intended to improve security of networks and devices by detecting behaviour such as cyber-attacks on a computer system or network (Gaiceanu et al., 2020).

Prevailing IDS have been utilized for classifying irregular ICS behaviour. Udd, et al., 2016, presented a technique of enhancing the intrusion detection tool Bro for the duty of noticing abnormalities on ICS. Concentrating on the IEC104 protocol, the technique of detecting abnormalities utilizes two major mechanisms, applied in the Bro IDS; a learning segment over a whitelisting framework to characterize accepted system traffic that passes through the IDS segment to trigger alerts and network once traffic in the network control system is not equal to the distinct whitelists.

Accepted traffic is organized over two whitelists, first for Address Resolution Protocol (ARP) traffic, where Media Access Control (MAC) addresses are matched to align with Internet Protocol (IP) addresses guaranteeing solely permitted devices are permitted on the network and 2nd for Transmission Control Protocol (TCP) traffic, guaranteeing only hosts acquire the permitted ports and IP addresses. The whitelists are formed via listening to standard network traffic, which are then utilized by the detection engine to equate network packets to produce whitelists. In the event a network packet doesn't contest the whitelists, an alert is triggered for additional analysis (McDaniel et. al., 2017).

Notwithstanding the IDS utilization for detecting abnormal actions on ICS systems, "process-oriented" approaches have remained a current concern for intrusion detection for ICS. In 2016,

Colbert, et al., shared a technique which was regarded as process-oriented for intrusion detection by manipulating domain control knowledge framework, operatives and concentrating on the “critical process variables” of the industrialized process. Colbert, et al., outlines critical process variables as factors that are essential to the industrialized operation process like sensor observing the temperature of a device. Through working with the ICS specialists and system security engineers of the domain, scope of basic acceptable value is both defined and useful to sensors under check. When this sensor touches one defined value, the alarm is activated and channelled to an expert who uses a web interface to receive the alerts for both detailed reaction and analysis.

In 2017, Cheung et al., presented a strategy for “model-based” intrusion detection ICS networks that creates models the Modbus protocol, anticipated system behaviour and the anticipated condition of both services and servers on the system. The models are implemented as guidelines for the regularly utilized IDS Snort that sends a notification to the operation in case unanticipated behaviour occurs. The research also presented specification mining, plus a three-stage method for utilizing specification mining on the ICS system to create rules of IDS device, like Bro. Firstly, the discovery stage of the framework recognizes network devices being checked. Secondly, documentation of devices gotten from the Internet or provided by the devices is gathered. Lastly, this documentation is investigated, and IDS controls are created.

Log mining, another process-oriented approach to IDS, has recently been used to identify threats in ICS. In 2012, Hadžiosmanović developed Mining Event Logs for Intrusion in SCADA Systems (MELISSA). Therefore, MELISSA was intended to search for “undesirable behaviour” constituting of anomalous attacks, behaviour and misconfiguration. Making use of its “Data Preparator” (DP) engine, utilized for combining and interpreting data to peruse data mining pattern, similar tasks to those of pre-processing stage, in mining process to formulate event logs for Pattern Engine (PE) and mining based analysis. This utilizes an algorithm mining for steady patterns, MELISSA searches

inconsistent events and returns them for a manual analysis. Through MELISSA utilization on event logs from SCADA record of 14 days' time span comprising 101, 025 occasions. Most importantly, MELISSA discovered 198 procedures for assessment for domain specialists. Out of the 198 assessment events, 23 of them were suspicious while 5 of them were termed irregular behaviour (Hadžiosmanović, et al., 2012).

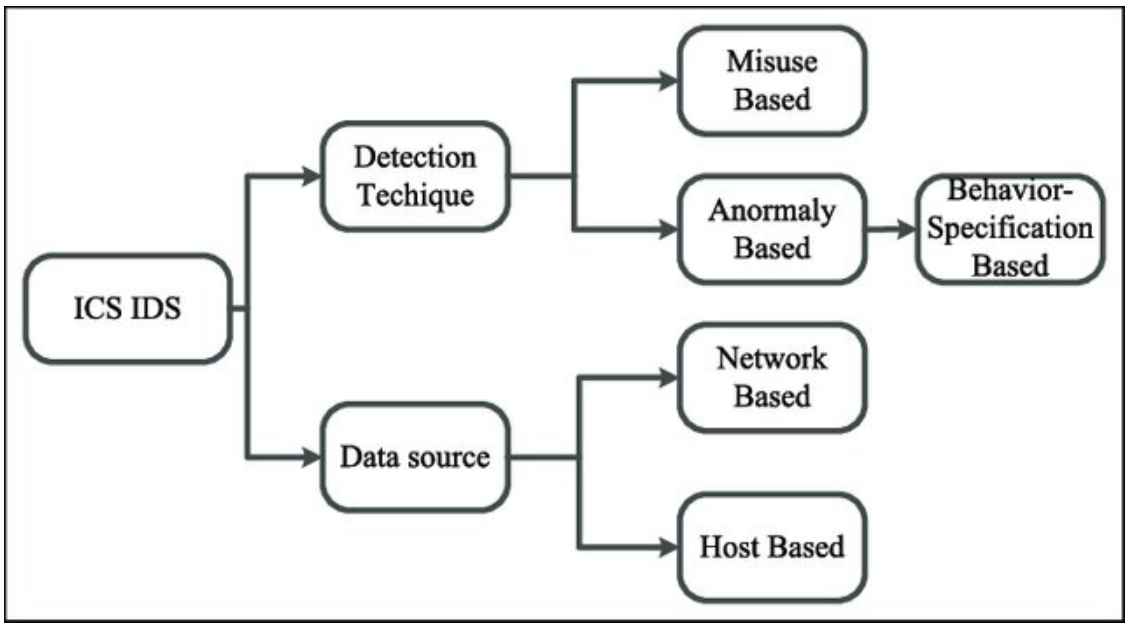


Figure 2.5: ICS IDS Taxonomy (Yang, 2018)

2.7.2 Firewalls

The security of the system highly depends upon different elements that play an explicit task. Firewalls are considered as the first line security perimeter in the ICS network environment. These system security elements restrict the attacker from gaining access into the system by allowing the passage of approved traffic necessary to run the organization. Therefore, the notion of network division is only applicable to the network in zones to protect resources at the different levels. Firewalls go about as sentinels or in other words watchmen, between the zones (Fabro, 2016). When precisely configured, they will just allow for basic traffic cross security zones. In the event the firewall rules are not well configured, unauthorised attackers can easily gain access into the system.

Firewall rules analyse the network traffic and grant access to whitelisted users and devices only. This is effective when the firewall rules are well configured.

Firewalls are border protection devices and control communication between different levels of trust (CISA, 2020). In the ICS space, firewalls are commonly used to define the boundary between the corporate network and ICS network, create the ICS DMZ and further segregate levels of trust deeper in the control network (NIST, 2015). Firewalls inspect and control traffic at different layers of the TCP/IP stack, and thus can have varying degrees of granularity in their applicability. Host Firewall ensures that a network that emanates from a given host is secure. It can be described to be part of the operating system, or it can be a device directly in line with the host. On the other hand, Network Firewall is a kind of firewall used to provide security for the network system using one of the techniques as provided below:

Packet filtering

Packet filtering is a type of firewall system that is configured based on certain rules. The monitoring and analysis of traffic highly depends on three primary levels of the Open Systems Interconnection (OSI) model. This include the IP addresses and MAC address with filtering capacity within the transport later.

Circuit level gateways

This category includes different types of firewalls that permit only explicit sessions to communicate.

Proxy level gateways

This is a type of firewall that filters traffic at the application layer. In that capacity, it entails type of application, and protocols that exchange data across security limits such as File Transfer Protocol (FTP), Hypertext Transfer Protocol (HTTP), etc.

Stateful inspection

Cyber security experts refer to this firewall as stateful firewall because it maintains "state" of the connections experienced across the firewall. They are resistible for aligning the packets with the different connection types indication the type of packets to accepted or reject.

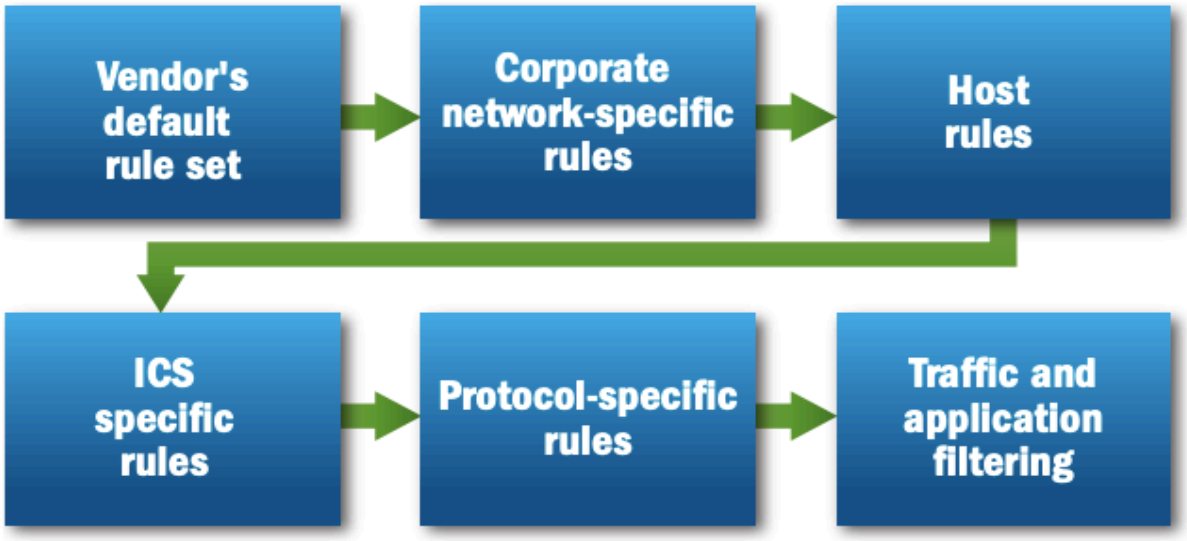


Figure 2.6: ICS Firewall Rule Set Layers (Fabro, 2016)

2.8 ICS Security Requirements

The security necessities of ICS vary fundamentally from the ones of normal information systems. Taking into consideration information systems, security implies that unapproved individuals or affiliations cannot uncover, change, take or harm of valuable and sensitive information. Nevertheless, it is believed that security in traditional systems enclosed ICS is safety. This translates that the traditional ICS can avoid adverse effects of hardware, software or system failure for safety purposes. This means that ICS have both security and safety requirements as summarized below:

Real-time

It has controlled operation time on each physical device in a way a slight change may have significant negative impact like damaging the physical device or it may result to industrial accidents.

Limited computing resources

The sensors and actuators in ICS have constrained computing and storage resources, which makes it difficult to run system security applications.

Fixed business logic

For ICS to realize specific objectives, it should follow explicit business rationales, failure to which there might be serious accidents.

Legacy systems

It is difficult to upgrade ICS because there are legacy subsystems in it. This makes it very likely that during and ICS operation, field devices will encounter persistent security threats.

Hard refreshing and updating of industrial equipment

Due to the impossibility of stopping ICS from running so as to fix bug or make software updates, letting it operate continuously, for it to achieve its socio-economic benefits ensures the stability of ICS.

Poor security of industrial control protocols

The introduction and rapid growth of the Internet technology protocols that were originally secure in closed environments become vulnerable to cyber-attacks when exposed to the Internet. This increases the possibility of sensitive and important data to get exposed to unauthorised users or attackers.

2.9 Approximate Matching Algorithm

In evaluation to cryptographic hashing functions, approximate pattern matching algorithm, otherwise referred to as fuzzy hashing algorithm, attempts to detect the similarity between two or more files by connecting similar inputs to similar outputs, indistinctly known as a fingerprints or profile. Approximate matching algorithms analyse files at byte level making them suitable to compare a large

chuck of data and identify similar texts or embedded objects (e.g., an image in a file) or binary fragments (e.g., a virus contained inside a file or a specific data packet in a network communication).

By utilizing a variety of strategies, approximate pattern matching algorithms are able to identify if even a single byte has been changed (NIST, 2014). A confidence score is determined upon successful similarity matching. The confidence score is normally determined by the number of attributes in common between the objects in correlation. The score is anticipated to be high when similarity between the shared content is high and low when less content in the two files is similar. Fuzzy hashing can be categorized into the following four categories: -

Block-Based Hashing (BBH)

Block-based hashing technique produce and store cryptographic hashes for given block of number with fixed byte sizes such as 512 bytes. In more subsequent stages of block-based hashing, it compares two different inputs by evaluating and establishing the number of common blocks on these inputs and providing the measure of the degree of similarity between them. This fuzzy function was designed by Nicholas Harbour under a program that he termed as dcfldd (Harbour, 2002). This fuzzy function splits the input data into several blocks of a fixed length and then computes the corresponding cryptographic hash value for each block. Despite the fact that the hash-based approach is efficient, more reliable and exceptionally proficient. A single byte change at the beginning of a file can change all the other block hashes, making this scheme very vulnerable.

Context-Triggered Piecewise Hashing (CTPH)

This method was developed by Andrew Tridgell, who actualized spamsum, a CTPH application focused on the identification of spam e-mails (Tridgell, 1999). The primary idea behind this concept was to help in locating content markers, known as contexts, within a binary data. This method calculates the hash function of each segment of the file delimited by the contexts and save the

sequence of hashes into a file. Therefore, the boundaries of each segment are depended on the content of the object and not determined by a subjectively fixed block size. In 2006, Jesse Kornblum implemented ssdeep based on spamsum. Ssdeep is one of the principal programs for computing CTPH.

Statistically Improbable Features (SFI)

This approach was designed based on the idea of identifying several common attributes for every object under study and then doing a compare whether the features are different. The feature to be identified in this case refers to a sequence of consecutive bits that are identified based on the criteria from the file containing the object. For practical implementation, Vassil Roussev chose to utilize entropy in with a reason to find measurably improbable features. Based on this idea and findings, he developed an algorithm known as sdhash. The objective of this algorithm was to highlight object features that have a rear chance of occurring in other data objects (Oliver, Forman, & Cheng, 2014). The score generated by this algorithm range between zero and one hundred. This range refers to the confidence value to show how certain the algorithm is that the two data objects being compared have non-trivial amount of similarity.

Block-Based Rebuilding (BBR)

This function utilizes external data. The data is made up of arbitrary picked blocks, consistently or in a fixed way, in order to rebuild a file. The process computes the dissimilarity between the bytes of the initial file to the chosen blocks using the Hamming distance or any other measurement (Breitinger & Baier, 2012). Two of the best-perceived usage of this method are SimHash and bbHash.

2.9.1 Ssdeep

In 2006, Jesse Kornblum released ssdeep. Ssdeep is a fuzzy hashing algorithm that employs similarity digest to be able to determine if the hashes that represent two or more files have similarity.

This tool produces fuzzy hashes, which are considered Context Triggered Piecewise Hashes (CTPH). CTPH considers two files similar if they have some identical sub-parts. Fuzzy hashes and other rolling hash methods are useful because they generate a continuous stream of hash values for a rolling window over the binary (Jesse, 2006). This algorithm computes a matching score ranging in between zero to one hundred percent. This score is translated as a weighted level of the similarity degree between the input files, whereby a higher score implies a more similarity between the input files (Martínez, et al., 2015). The signature format produced by ssdeep comprises of a header followed by one hash as indicated below. The following represents the content of the header in the latest versions is:

ssdeep,1.1—blocksize: hash: hash, filename,

The word ssdeep in the header format above identifies the algorithm, 1.1 identifies the version of the file format, -- is a separator while the remaining part of the header points out the elements displayed below the header in this order (block size, primary hash, secondary hash and filename).

To compute similarity in file signatures, this tool takes signatures as regular strings and utilize Levenshtein distance measure to determine the degree of correlation. The Levenshtein distance is the number of operations required to change one string into another. There are various ways to characterize the Levenshtein distance, depending on which tasks are permitted. Ssdeep technique of calculating similarity is based on the Levenshtein distance between two strings. This distance differentiates two strings and determines the least number of operations required to transform one string to another. The edit operations allowed in the distance comparison are deletions, insertions, and substitutions of a single character and transpositions of two adjacent characters (Kim, et al., 2017). The example below demonstrates how this algorithm can be used to compute distance

between two strings “Saturday” and “Sundays”. The character in bold format shows the characters being added or removed at each progression. The Levenshtein distance in this example is 5.

S**a**turday $\xrightarrow{\textit{deletion}}$ S**t**urday $\xrightarrow{\textit{deletion}}$ Su**r**day $\xrightarrow{\text{deletion}}$ Suday $\xrightarrow{\text{insertion}}$ Su**n**day $\xrightarrow{\text{insertion}}$ Sunday**s**

Levenshtein distance between two strings, *a* and *b* (of length |a| and |b| respectively) is given by equation 2.1 below: -

Lev *a, b* (|a|, |b|).

Where:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Equation 2.1: Levenshtein Distance Formula

From this equation, 1(ai≠bi) is the indicator function equal to 0 while ai≠bi and equal to 1 otherwise, and lev a, b (i, j) is the distance between the first *i* characters of (a) and the first *j* characters of (b). The primary component inside minimum corresponds to deletion (from a to b), the second to insertion and the third to match, depending on whether the separate symbols are equivalent.

In forensics, ssdeep is a widely known and used byte wise approximate matching application and it is considered by a couple of researchers as the de facto standard in a couple of cyber security areas (Breitinger, Stivaktakis, & Baier, 2013). To compute similarity in file using ssdeep, hashes of the input files are computed then used for comparison with another hash or with a data file. The outcome in both scenarios are the same, using one technique or the other is highly depended on the data available to the user. Ssdeep has increased enormous fame in malware detection and analysis in the recent past.

One reason for ssdeep popularity is grounded on the fact that it helps experts to rapidly decide if a suspected piece of file is similar to a known file. For instance, to date Malware.lu, VirusTotal, VirusSign, VirusShare, Malwr, Malshare.com, and vicheck.ca all use ssdeep hashes for their malware test (Pagani, Dell'Amico & Balzarotti, 2018). Another reason for ssdeep popularity is attributed by the high speed at which it computes hashes compared to sdhash and mrsh-v2 (Sarantinos et al., 2016). Ssdeep represents an essential achievement in similarity detection systems and it is relatively up to date. At the time of writing this dissertation, the latest version was 2.14.1 accessible through https://github.com/ssdeep-project/ssdeep/releases/tag/release-2.14.1 (Last accessed on 13th March 2020).

2.9.2 Similarity Digest Hash (SDHASH)

Vassil Roussev proposed sdhash in 2010. Sdhash is the successor to ssdeep, despite the fact that it employs totally unique approach to detect and match similarity between files. Sdhash uses Bloom Filters to determine block size and computes file difference using Hamming distance (Roussev, 2011). Preferably, sdhash utilizes the concept of similarity digest hashing and works by extracting attributes that are statistically improbable using the Shanon entropy, where a feature is a byte sequence of sixty-four bytes (Vassil, 2010). All the selected features are then hashed using a cryptographic hash function SHA-1.

Sdhash calculates a normalized Shannon entropy measure, since it is not possible to directly estimate empirical probability of encountering 64-byte feature and such observation cannot be practically stored and looked up. In particular, these standardized features are placed into comparability of 1000 classes.

2.9.3 Multi-Resolution Similarity Hashing (MRSH)

Mrsh-v2 is the modernized version of mrsh that was developed by Breitinger & Baier (2012) and depends on the concept of multi-resolution similarity hashing and context triggered piecewise

hashing. With regards to identifying triggers, mrsh-v2 does as such by pointing in the input byte sequence so as to divide it into chunks. This application divides a file into chucks and then uses a pseudo random function *pr f* and a modulus called block size b. A window of fixed size 7 slides through the whole input, byte for byte, and *pr f* generates a pseudo random number *r* at each step over the window. If r = −1 mod b, the byte sequence in the window is a trigger point and thus the end of the chunk. The implementation aims at having a fingerprint length of 0.5% and hence of b = 160 bytes. In view of format, mrsh-v2 allows a maximum of 160 chunks per Bloom filter, after which a new Bloom filter is created when the limit is reached. This implies that the final fingerprint obtained is a sequence of Bloom filters. Unlike the traditional hash functions, mrsh-v2 generates variable length fingerprints.

2.9.4 Approximate Matching Algorithm Comparisons

A comparison of mrsh-v2, sdhash and ssdeep is introduced in this section. To start with, mrsh-v2 is a significant improvement over ssdeep and sdhash. Mrsh-v2 is preferred for detecting fragments embedded in other files. It is identified to have the ability to find fragments, as its design is more inclined to finding detecting file fragments rather than comparing similarity on complete files. Another drawback for this algorithm is that its chuck sizes are alterable. When this tool computes similarity using large chuck sizes, it computes similarity of files much faster but produces inaccurate results. When this algorithm uses a smaller chuck, it takes more time to compute similarity but computes more reliable results (Lee & Atkison, 2017). Mrsh-v2 should be utilized with caution when it comes to analysing whole files as it may generate unreliable results.

On the other hand, a detailed analysis of the implementation and security analysis of sdhash revealed several implementation bugs in the algorithm and demonstrated a possible way to beat the similarity score by tampering with a given file without changing the perceptual behaviour of this file (e.g. image files look almost same despite the tampering) (Breitinger et al., 2012).

Ssdeep is considered by many network security and forensic investigators as the de facto fuzzy hash algorithm but that does not signify that it has no limitations. The well-known limitation of ssdeep is that it experiences a considerable degree of slowness as it processes files larger than 2 gigabytes (Lee & Atkison, 2017). However, this slight drawback is over-weighed by the main advantages that relate to ssdeep given its hash computation speed and similarity match accuracy.

Additionally, ssdeep is easy to install and configure compared to the other algorithms. Ssdeep also presents the matching score results in percentage form making it simpler to understand and interpret (Sarantinos et al., 2016). These specific advantages are missing from the other algorithms. Ssdeep performance is based on its design and highly depends on the large and continuous data to detect similarity. Considering the fact that IEC 104 communication involves machine to machine exchanging common data in a predictable manner, ssdeep was found suitable for use in this study.

2.10 Gap Analysis

Detection of anomalies or attacks in network communication is an important task for any network administrator. A number of network security applications have been developed to detect network communication anomalies in ICS. The key features for these security applications includes detecting of attacks and sending an alert to the administrators for further analysis. While the tools reviewed in this chapter have demonstrated the ability to detect attacks, they have drawback in that they have high false rate, some cannot detect a new attack, while others are not open source.

The gap identified is the need for development of ICS network profiling tool that is more accurate, open source, can analyse network data at binary level, detect similarities using ssdeep and compute the matching score. Analyses of network data at binary level helps to ensure that the application can detect any slight change in network communication pattern which in return makes the application more accurate.

2.11 Conclusion

This chapter has outlined a number of security applications used to detect attacks in ICS. However, these security solutions are confirmed to have high false rates making them not effective for detecting attacks in ICS network communication. The tools are also commercial in nature and require licence. There is need for coming up with an application that can easily analyse network communication, create a baseline for each network packet capture file and do comparison to detect anomalies in ICS network communications.

CHAPTER 3 : RESEARCH METHODOLOGY

3.1 Introduction

This chapter elaborates the adopted methodology in this research process. Focus falls on detailing the method that was utilised for conducting this study and their feasibility are described in this section. Besides, this chapter concentrates on the methodologies applied in system design, architect, development, analysis, and implementation and testing.

3.2 Research Design

Research design is the format used in the research in terms of collection of data, measurement and analysis in effort to get answers to research questions. Wanjugu further reiterates that the research design should bring out the relevance of the research with the economy in procedure (Wanjugu, 2015). The selected approach is partly theoretical in nature through literature review and the development of a tool used to validate the proposed solution. A review of relevant research documents on profiling ICS communication using approximate matching algorithm is sufficient to answer the first two objectives of this study. The theoretical research was intended to provide a deeper theoretical understanding of the research area. This forms a basis for the development and validation of the proposed solution.

3.3 Data Collection

Data Collection explains and demonstrates the methods, techniques and approaches adopted in social research projects (Denscombe, 2014). ICS network packet capture files were the main source of data for this research. The packets were captured from ICS systems in Brno University of Technology (BUT) during the research period. The packets were provided to the researcher by BUT for internal research purposes only with no rights to publish in detail because of the University's ICS security purposes.

3.3.1 Data Sets

The data used in this research was captured from a SCADA to substation network communication using Wireshark. A total of four packet capture files were generated for IEC 104 analysis in this research. They include:

i. SCADA_to_substation_normal. pcapng – This packet capture file was captured while the network communication between the SCADA and the substation was operating in normal conditions with no human interference. The network communication was between two nodes with IP 172.16.1.1 and 172.16.1.100. This communication took a period of 15 minutes 72 seconds and total of 172 packets were captured.

ii. SCADA_to_substation_attack1.pcapng – This packet capture file was captured while simulating Denial of Service attack (DoS). This attack was simulated by continuously switching on and off one of the communication nodes. This was achieved by using ASDU with C_CD_NA messages (double command). Not all commands were accepted. Out of the 12 commands that were received, only 4 replies were send back. This simulation took 15 minute and 83 seconds. A total of 217 packets were captured in this packet capture file.

iii. SCADA_to_substation_attack2.pcapng – This packet capture file was captured while simulating DoS attack within many substations. This attack involved network nodes ranging from 172.17.1.20 to 172.17.1.119 and was caused by ASDU messages C_IC_NA_1 (interrogation command). During this attack, a total of 688,979 packets were captured and it took a period of 16 minutes and 30 seconds.

iv. SCADA_to_substation_attack3.pcapng – This packet capture file was captured file simulating a scanning attack. This attack was meant to retrieve values from various objects on the target device. This attack used ASDU M_SP_NA messages (single point information). This communication was between two network nodes with IP 172.16.1.1 and 172.16.1.100.

This attack took a period of 14 minutes and 17 seconds. A total of 147 packets were captured in this packet capture file.

3.4 Software Development Methodology

The system design method used in this case study was the agile software development methodology. This methodology gives the research an opportunity to assess the direction of a project all the way through the development lifecycle (Harvin, 2016). A Manifesto for Agile Software Development signed by Beck et al. (2001), shows that agile development is the most preferred way of developing software that promotes sustainable development. In particular, the agile software development method allows for faster iteration and more frequent release with subsequent user feedback. The processes involved in the agile method enable for release schedule and give opportunities for user feedback, which allows faster and more controlled improvements (Dybå & Dingsøyr, 2008).

Figure 3.1 below illustrates the steps followed in this study in order to attain the outlined objectives for this dissertation, the first step was planning which involved the collection of the intended product specification or features and specifying what it should do or how it should do it. The second step was the architecture and design, which included defining the architecture and design of the system. Development of the application was the third step, which involved the implementation of the application. Test and feedback were the fourth step, which allowed room for product improvement. After feedback collection, the steps should be repeated if changes are being implemented on the system. The developed tool was tested independently in every development iteration.

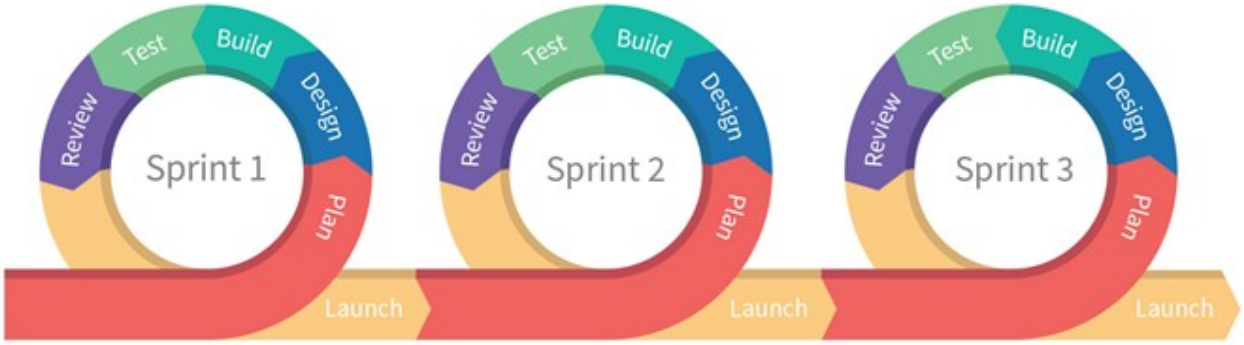


Figure 3.1: Agile Methodology (Maruping, Venkatesh & Agarwal, 2009).

3.4.1 System Planning

There are three approaches in the information system development section and that is: data, process and function-oriented approaches. Unlike its two predecessors that lay emphasis either on data or process, the object-oriented method combines processes and data into single entities called objects (Dennis et al., 2015). Wireshark is an open-source packet analyser tool that was used to capture client to server communication packets for analysis in this research (Sanders, 2017). Tshark is a command based oriented version of Wireshark that was used to filter data from the packet capture files.

Functional Oriented Design (FOD) is the concept used in this research. FOD escalates the comprehension of problem areas in light of the fact that FOD develops a smooth move from the analysis stage to the design stage and offers a more customary method for setting up details (Zimmerman, 2008).

This research concentrated on class and use case modelling to investigate the different methodologies that are conducted in the system analysis. In the system development life cycle using object-oriented, case modelling is used at the analysis stage. Use case modelling was carried out at the initial stage of system development to help the designer pick up an impeccable comprehension of the useful

prerequisite of the framework without agonizing over how those necessities are connected (Gemino & Parker, 2009).

3.4.2 System Requirement Analysis

In this step, data collected through analysing systems in the planning step was analysed. This helped in formulating the requirements of the application that was developed. This stage also included the creation of the system architecture. The design step involved coming up with the application designs based on the requirements gathered.

3.4.3 System Design

This involves the sectioning of the system to be developed into components for purposes of studying how they work together to achieve system functionality (Gemino & Parker, 2009). The use case diagram models the system functionalities and gives an illustration of how system actors interact with the system processes known as use cases (Mishra & Mohanty, 2012). Functional Oriented Design (FOD) techniques was used to refine the functional requirements identified during system analysis and to decompose the design into sets of interacting units where each unit has clearly defined functions.

In addition, Unified Modelling Language (UML) was used for purposes of specifying, visualising, constructing and documenting the artefacts of the system. To show the required system usages, Use Case Diagrams were used under design or analysis. Use Case Diagrams captured what the system is supposed to do. System Sequence Diagrams was used to illustrate a scenario of a use case, the events that external actors generate, their order and possible inter-system event. These artefacts were developed by the use of draw.io tool.

3.4.4 System Implementation

Python programming language was used because it provides several libraries that can be used to achieve low-level functionalities and has many resources supported by a large community. Ssdeep was used as the pattern-matching algorithm. By design, ssdeep performance is suitable for a large chunk of data, which makes it suitable for network analysis. This algorithm is also preferred because it is open source and cross-platform. There are other algorithms that were considered during the initial phases of this study including mrsh v2 and bbhash. However, these two algorithms have issues when it comes to performance. For instance, bbhash performs slower, hence, it cannot be used to analyse network packet (Breitinger & Baier, 2012). Mrsh v2 is file depended and thus not further considered.

3.4.5 System Testing

The functionality of the application was conducted through unit testing. As a practical methodology or simply a component of Test-Driven Development (TDD), unit testing usually adopts a rigorous strategy that helps in building a product by method for consistent testing as well as adjustment. First, TDD requires the system developer to write the fizzling unit tests whereby code is written, and the application refactored till the test passes. Normally, a TDD brought about a clear and obvious code base. This application was tested by modifying data in a packet capture file and recomputing similarity with the original version. This acted as a simulation of how potential attacks are performed in ICS.

3.4.6 System Review

The objective of a systematic review is to determine whether the tool was built in conformity with professional standards and to ensure the user complying with the system usage (Shamseer et al., 2015). In this research, peer review was used.

3.5 Research Quality Aspects

A research quality aspect is a degree to which the research was carried out correctly (Fellows & Liu, 2015). Validity and reliability were used to test the quality aspects.

3.5.1 Validity

Validity decides whether the research is truthfully and accurately what it was planned to measure or the openness of the research findings (Kimberlin & Winterstein, 2008). Content validity was utilized to validate the research by examining systematically the content to be tested in order to determine whether it covers a chosen sample of the behaviour area to be measured (DeVon et al., 2007).

3.5.2 Reliability

Reliability is the degree to which there is consistency in results over a specified time and a correct illustration of the aggregate population under research. If the outcomes of research can be replicated in a comparable approach at that point the research mechanism is considered to be reliable (Kimberlin & Winterstein, 2008).

3.6 Conclusion

This chapter has highlighted the methodology that was used in undertaking the research. The methods for data collection and analysis are proposed to guide the researcher in making deductions from the findings to be observed and thus helps in answering the research questions defined in the first chapter of this research. The tools used in this research are also highlighted. In the event any of these phases change during implementation stage, the details will be indicated during the project documentation stage.

CHAPTER 4 : SYSTEM DESIGN AND ARCHITECTURE

4.1 Overview

This chapter details the design and architecture of the proposed system to deepen the understanding of the system. Testing the ssdeep algorithm formed the basis for designing the architecture of the proposed application. As a result, a detailed view of the ICS network-profiling tool was designed as outlined in section 4.3.

4.2 Requirement Analysis

Requirement analysis reviewed user expectations for the ICS network profiling tool ensuring that it takes care of all the stakeholders need. In line to the set objective, the section below discusses the functional and non-functional requirements of the application in this research.

4.2.1 Functional Requirements

The functional requirements specify what the application should do to support tasks, activities or goals through function, behaviour input and output. The functional capabilities of the application include the Extracting payload (extraction of TCP payload from the packet capture files), Hashing (ability to hash the extracted file using ssdeep to form profiles), Pattern matching (calculating the similarity between the normal profile against any other unknown profile extracted from the packet capture files).

4.2.2 Non-Functional Requirements

Hardware Requirements

i. Machine: Intel Core i3 or higher.
ii. Clock speed and processor: 2.5 GHz or higher.
iii. System Memory: 4GB or higher.

Software Requirements

i. Operating Systems: Windows 10/8/7, Linux.

ii. Python 2.7.

4.3 System Architecture

System architecture is a conceptual model that outlines the structural design of the system (Giraldo et al., 2019). The system architecture addresses how various system component interact to achieve the system functionality. This section provides an understanding of how the system design, structure and user requirements must be supported by the application. The developed ICS network profiling tool is a command line-based application that comprises of three python scripts. They include:

i. Create_profile.py Script – This python script is the first to be executed while running the application. It is integrated with Tshark to extract TCP payload from the packet capture files and save the data into a text file.

ii. Hash_profile.py Script – This is the second python script to be executed. It is used to hash the payload files extracted from the packet capture files. Ssdeep is integrated with this script for to be used for fuzzy hashing.

iii. Compare_profile.py scripts – This is the third python script in the application. This script is used for pattern matching the fuzzy hashes against the comparative files and generated a matching score. Ssdeep is used in this script to compute similarity between the input files.

The steps below discuss the processes and functionality of the scripts in detail;

Step 1: Import the Packet Capture File

This is the first step that takes place when the user runs the create_profile.py script. The main goal of this step is to import packet capture files into the application for further analysis of the IEC 104 protocol. This is achieved by specifying the names of the files to be analysed.

Step 2: TCP Payload Data Extraction

The second step involves the use of Tshark to filter the payload data from the packet capture files imported into the application. Tshark attribute filters used at this stage will determine the data to be extracted from the packet capture files. The Tshark command used in this application to filter IEC 104 payload is: *tshark, -r, pcap_file, -T, fields, -e, tcp. payload, 104apci.*

Step 3: TCP Payload Data Hashing

The third step uses the hash_profile.py script to hash the extracted text file in the stage above. Ssdeep is utilized at this level to fuzzy hash the files. So as to hash a file, ssdeep only utilizes a rolling hash algorithm to generate a pseudo-random value based on the current context of the input. A sliding window is overridden as input and every time the generated rolling hash matches the trigger; the algorithm computes a hash on the current piece. Originating from the spamsum algorithm, ssdeep uses block size for the trigger value.

Step 4: Profile Pattern Matching

The fourth step includes calculating the similarity between files using compare_profiles.py script. During comparison of two or more ssdeep hashes, ssdeep begins by determining the block size. Ssdeep computes the block size for the files to be compared. Following stage is the elimination of recurring sequences. Sequences indicate patterns in the input file which hold little information and therefore have little impact on the content. Finally, ssdeep computes a weighted Levenshtein distance between the two resulting hashes.

The Levenshtein distance represents the minimal number of edits required to change from one input to the other. The standardized match score represents a conservative weighted percentage of how much two inputs are ordered homologous sequences which represents how many of the bits are identical and in the same order. At last, the application generates a matching score in the range 0 – 100. The score is a proportion of how similar the input files are, where a higher result implies a

greater similarity of the files and lower score is an indication of dissimilarity. Figure 4.1 below illustrates the processes and interconnections for the ICS network profiling tool.

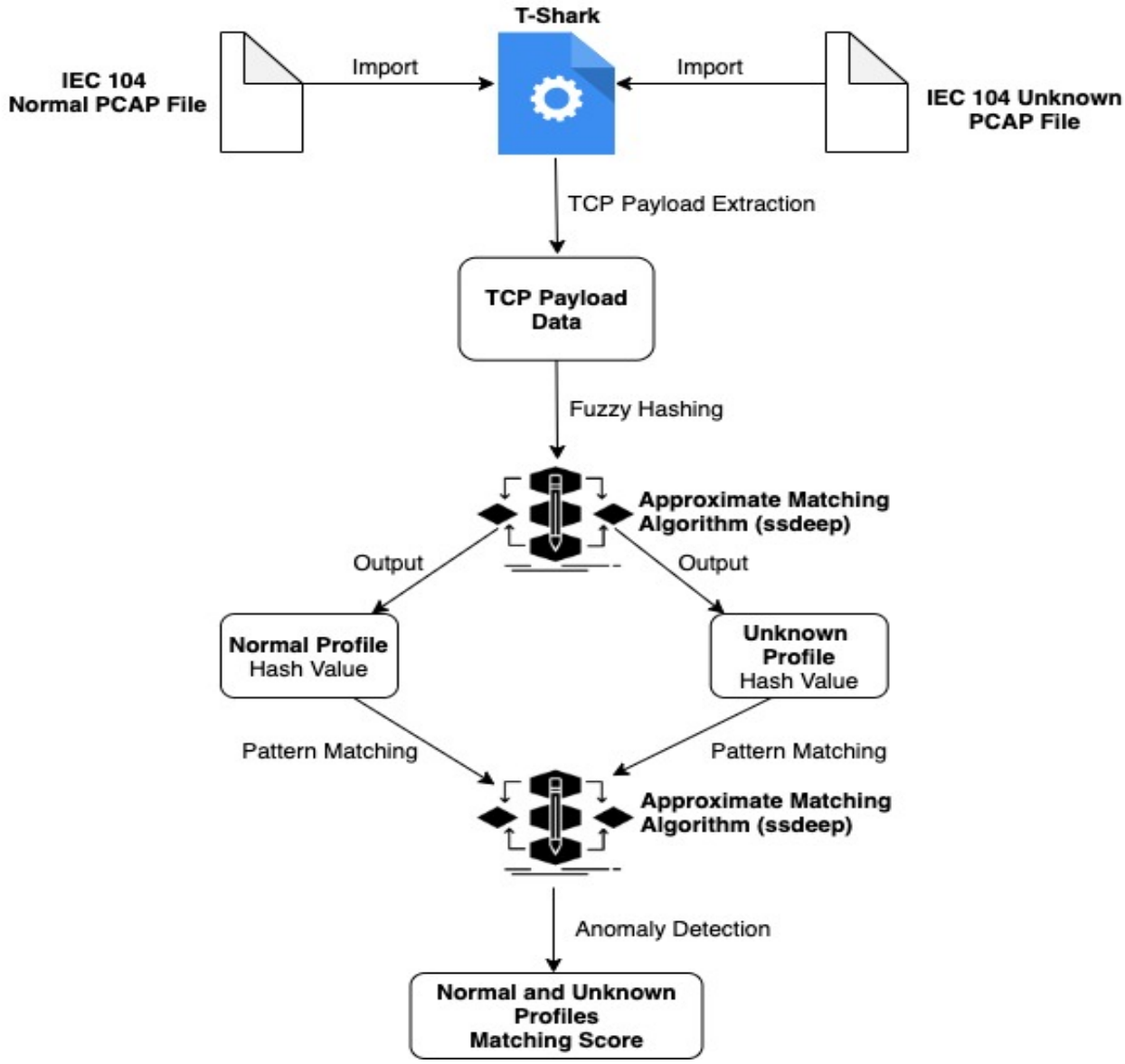


Figure 4.1: System Architecture for the ICS network-profiling tool

4.4 System Design

Since ICS network communication has proved stability within long term monitoring, approximate pattern matching algorithm was used in this research to detect anomalies in unknown ICS communication. The developed tool was designed to provide a standard ICS network profile that

would be used to identify any anomalies in ICS network communication. From this, ICS network administrators are better placed to monitor network communication based on solid facts. This section will expound the design structure of the proposed solution using various design diagrams.

4.4.1 Use Case

The application users and interactions of various functionalities of the ICS network profiling tool is illustrated in the use case diagram in figure 4.2 below. This use case describes the key features offered by the developed prototype. All the external tools and users interacting with the developed solution are also highlighted.

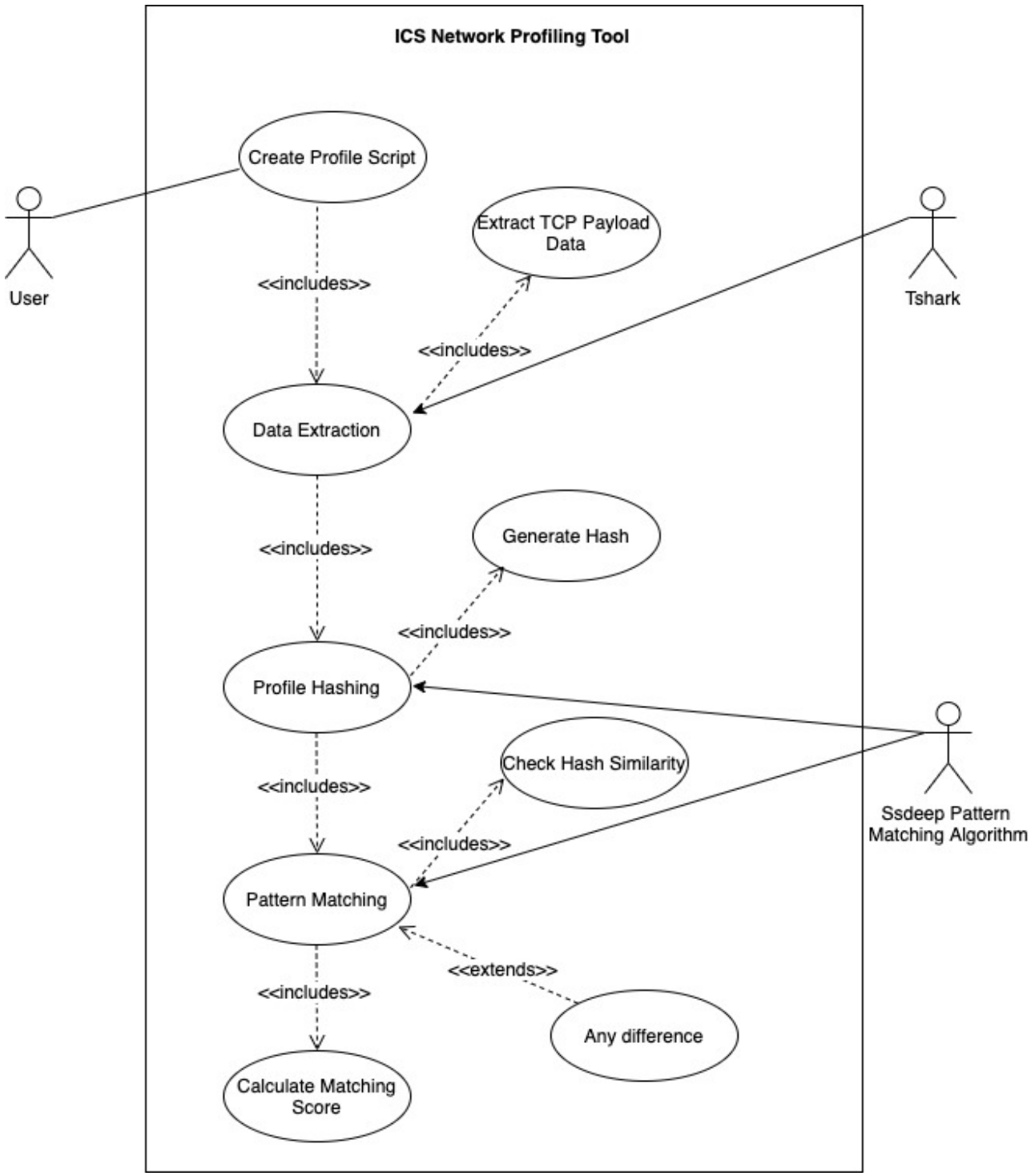


Figure 4.2: Use Case Diagram

4.4.2 Use Case 1 Create-Profile.py Script

Table 4.1: Create-profile.py Script Use Case

Use Case Name:	**Create-profile.py script**
Description:	- This use case contains instructions to extract TCP payload data from the PCAP file using Tshark, hashing the network profile and matching the known profile against the unknown profile.
Primary Actor:	- User
Secondary Actor:	- None
Include Use Cases:	- Data Extraction - Profile Hashing - Pattern Matching - Calculate Matching Score
Extend Use Cases:	- None
Precondition:	- ICS communication PCAP file must have been captured.
Post Condition:	- The tool matches the known profile against the unknown profile.
Main Flow:	- The system triggers TCP payload data extraction use case and waits for it to complete. - The system triggers hashing use case and waits for the communication profiles to be hashed using ssdeep. - The system compared the two profiles using pattern matching algorithm. - The system calculates the matching score.
Alternative Flows:	- Failed data extraction.

Use Case Name:	Create-profile.py script
	- The application restarts Tshark. - The use case ends.

4.4.3 Use Case 2 Data Extraction

Table 4.2: Data Extraction Use Case

Use Case Name:	Data Extraction
Description:	- This use case contains procedures for the application to extract TCP payload data from the ICS captured packet capture file using Tshark as the analysis tool.
Primary Actor:	- None
Secondary Actor:	- Tshark
Include Use Cases:	- Extract TCP payload data
Extend Use Cases:	- None
Precondition:	- Tshark must be preinstalled.
Post Condition:	- The application extracts the TCP payload from the ICS packet capture files and saves the results into a text format.
Main Flow:	- The application uses tshark to filter IEC104 payload using the following command: *tshark –r packet file.pcapng –T fields –e iec60870_104.type –e tcp.payload 104apci.* - TCP payload is extracted and save to a text file for further processing.
Alternative Flows:	- Tshark fails to extract payload. - Rerun the script.

4.4.4 Use Case 3 Profile Hashing

Table 4.3: Profile Hashing Use Case

Use Case Name:	**Profile Hashing**
Description:	- This use case contains the procedures for the system to hash the text file saved after extraction of the payload. Ssdeep is used for generating the fuzzy hash.
Primary Actor:	- None
Secondary Actor:	- Ssdeep Pattern Matching Algorithm.
Include Use Cases:	- Generate Hash
Extend Use Cases:	- None
Precondition:	- Save a text file containing TCP payload.
Post Condition:	- Fuzzy hash for the saved TCP payload.
Main Flow:	- This use case computes the fuzzy hash of the input file and scores the results.
Alternative Flows:	- File hashing fails. - Rerun ssdeep.

4.4.5 Use Case 4 Pattern Matching

Table 4.4: Pattern Matching Use Case

Use Case Name:	**Pattern Matching**
Description:	- This use case compares the known and unknown profiles and computes the matching score.
Primary Actor:	- None

Use Case Name:	Pattern Matching
Secondary Actor:	- Ssdeep Pattern Matching Algorithm.
Include Use Cases:	- Check hash similarity.
Extend Use Cases:	- Any difference.
Precondition:	- Compute profile hash.
Post Condition:	- Matching score of the profiles.
Main Flow:	- Ssdeep matches the unknown profile against a file of known profile. - The similarity match is calculated between the two input files.
Alternative Flows:	- Pattern matching algorithm fails to run. - Rerun ssdeep.

4.5 Sequence Diagram

This is an interaction diagram that depicts time ordering of messages between different system objects. Creating a normal profile for an ICS network communication and comparing the normal profile against an unknown communication to identify anomalies in the network communication are amongst the main features of the ICS network profiling tool. Figure 4.3 below indicates the main flow of data and events with regards to the main functionalities of the proposed system.

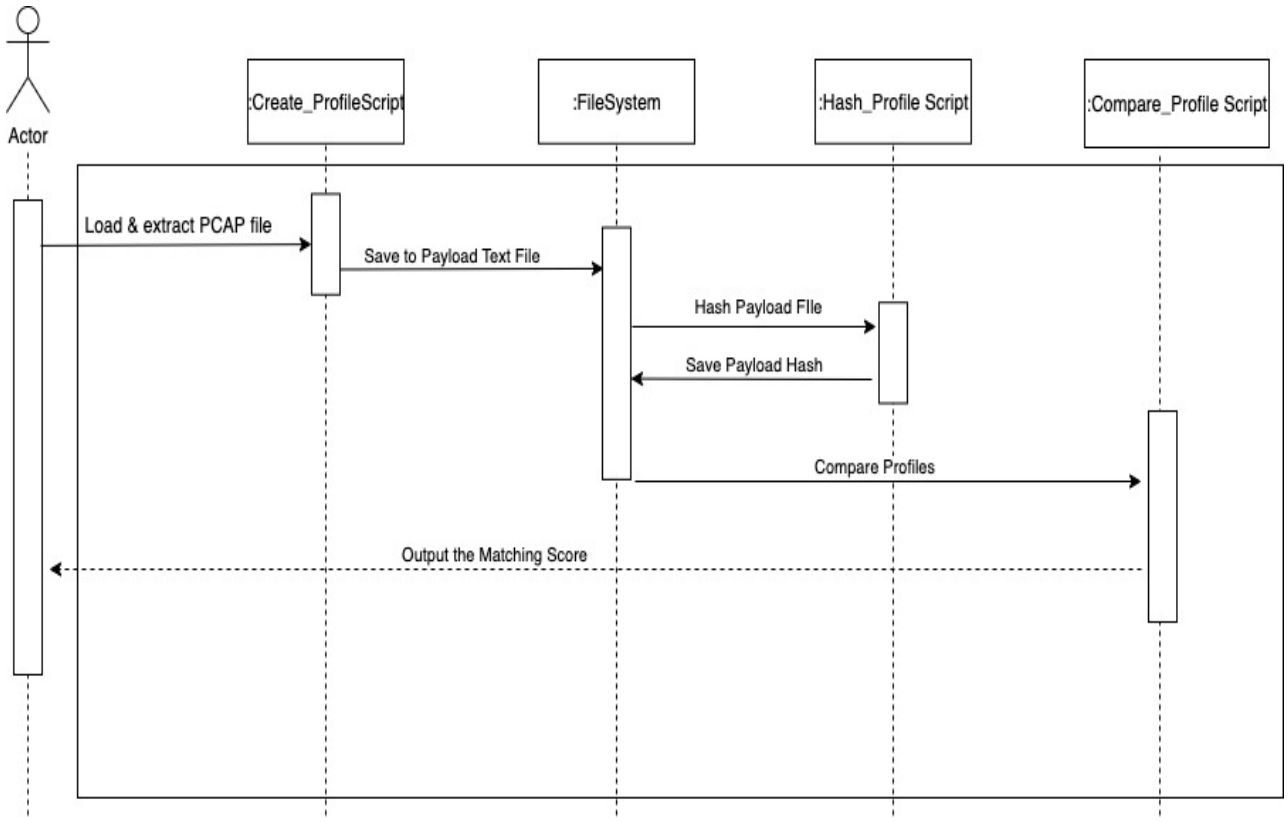


Figure 4.3: Sequence Diagram

Load and Extract PCAP File

Create-profile python script starts by triggering Tshark. Tshark network protocol analyser tool is used to extract network communication TCP payload from the packet capture files. This tool goes through the ICS packet capture files to analyse the IEC 104 payload data according to the specified attribute filters. The extracted data from the TCP payload is then saved into a text file in the file system.

Hash Payload File

The application uses ssdeep algorithm to fuzzy hash the TCP payload data extracted from the ICS packet capture file. The hash file is also saved in a text file in the file system.

Compare Profiles

Compare-profile python script is used to compute similarity between the input files. The comparison can be between the normal profile hash against the unknown profile hash to compute similarity between the profiles. Ssdeep pattern matching algorithm computes the similarity between the profiles to generate the matching score. The higher the similarity score between the profiles, the fewer anomalies can be detected in the compared profiles.

CHAPTER 5 : SYSTEM IMPLEMENTATION, TESTING AND VALIDATION

5.1 Overview

Designing of the proposed application was followed by its development and implementation as per system design done in chapter 4 of this study. This chapter concentrates on the implementation and the testing process of the application. On the implementation section, the implementation environments are explored together with the main functionality of the application. The testing part focuses on usability testing and functional testing to validate if the application attains the objectives of the proposed solution. The relevant screenshots of the prototype are provided.

5.2 Software Environment

The developed application is an ICS network-profiling tool. It is implemented using TCP payload for ICS network communication. The system was built in different phases using python programming language and integrated with other tools as detailed below.

5.2.1 Python Programming Language

Python is a high-level, interpreted, general purpose programming language. It has low-level capabilities that make it faster than other high-level languages (Cruz et al., 2015). Python was selected because it works well with other integrated tools.

5.2.2 Wireshark

Wireshark is a network analyser tool. This tool provides a graphical interface for both live packet capture and analysis, as well as for offline packet capture analysis. It can parse and interpret the IEC 104 protocol from the application layer. In this research, it is used to capture traffic in ICS and inspect the packets IEC104 payload.

5.2.3 Tshark

Tshark is the command line-based version of Wireshark. This tool goes through the packet capture files and analyses their content according to specified filters. It can filter packet capture file and output payload data as per the filter specified fields such as IEC 104 data type, IP address, APCI or ASDU data, etc.

5.2.4 Ssdeep

Ssdeep is a fuzzy hashing algorithm that employs similarity digest in order to determine whether the hashes that represent two or more files have similarity. This algorithm works by computing fuzzy hash for each file or string supplied to it. By design, ssdeep performance is highly depended on the presence of large, continuous chuck of common data and thus making it fit for pattern matching. The output of a compute function in ssdeep is a ssdeep hash, which looks like the following:

ssdeep,1.1--blocksize: primary hash: secondary hash, filename.

24:i1PSFEGReaRbVgD5kjSo97lFEGRea45bV36ruFEGRea+crbV9jod:i5gRFTNR

mqrERjpI,"Normal.txt"

In the sample ssdeep hash above, 24 is an integer that describes the size of the chucks in the ssdeep hash. In primary hash, each character of the chuck represents a part of the original file of size 24. The secondary hash is computed over the same data as primary hash but computed with block size*2.

5.3 Ssdeep Set-up and Installation

Ssdeep-2.14.1 is the latest version accessible in GitHub as of the time of documenting this research. The source code can be downloaded through this link: https://github.com/ssdeep-project/ssdeep/releases/tag/release-2.14.1(Last accessed on 13th March 2020). After downloading the algorithm, the user can change into the downloaded directory by using this command: *cd ssdeep-2.14.1.*

The next installation step is executing the *./bootstrap* file so as to generate the configure script. Once the script is generated, it is executed by *./configure*. Lastly, the ssdeep program can be complied by running make command, *make* and install it by *make install*. For the installation to work, the user logged into the computer must be a root user on most operating systems to be able to install the program to its default location, /usr/local/bin. This can also be achieved by running *sudo make install command*.

5.4 Prototype Implementation

5.4.1 IEC 104 Packets Analysis

One of the main features of Wireshark is that it enables packet filtering and analysis. A user can filter the captured traffic to display only a specific protocol or attributes in the communication packets.

In this implementation, Wireshark was used to filter IEC 104 packets from ICS network packet capture file as shown in Figure 5.1 below. IEC 104 is used as a standard for controlling equipment and processes with coded bit serial data transmission in TCP/IP based network. IEC 104 protocol involves machines to machine communication. This communication is known to be stable since it involves different network nodes sharing fixed operational objects and business processes. This stability communication results to relatively stable network traffic pattern while operating under normal circumstances since no interference is brought about by human behaviour.

IEC 104 consists of two layers, that is; APCI and ASDU. Each APCI starts with a byte with value 0x68 followed by the 8-bit length APDU and four 8-bit control fields. APDU contains an APCI or an APCI with ASDU. Generally, the length of APCI is 6 bytes. The TCP payload is not encrypted. This makes packets analysis to find useful information much easier. As shown in the figure 5.1, the APCI for the first packet is “680443000000”.

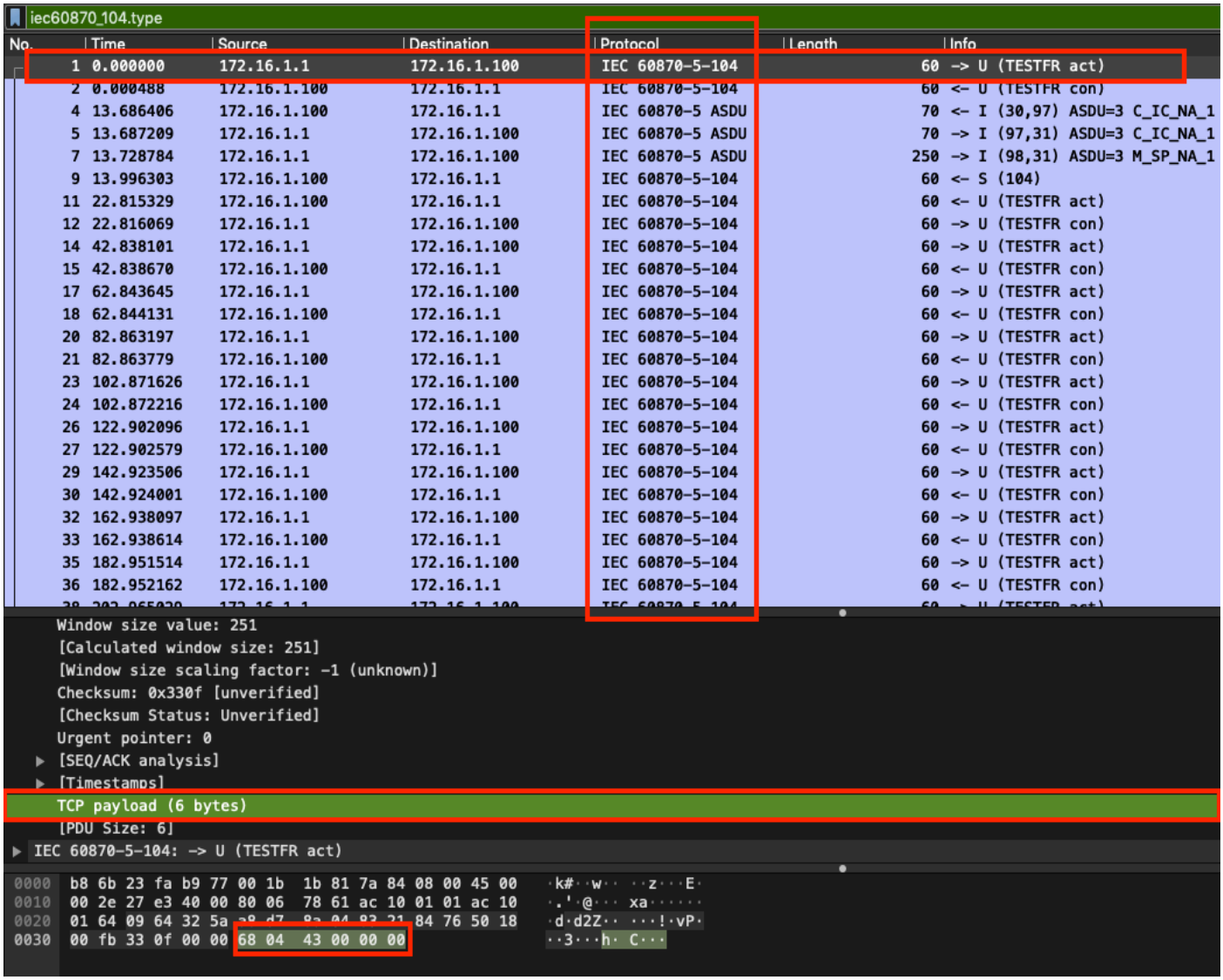


Figure 5.1: IEC104 Packets Filtered Using Wireshark

Figure 5.2 below shows a graphical representation of normal ICS network communication. It is notable that ICS communication is stable and exhibits a predictable pattern. Packets are transmitted in real time with no delays in sending the response.

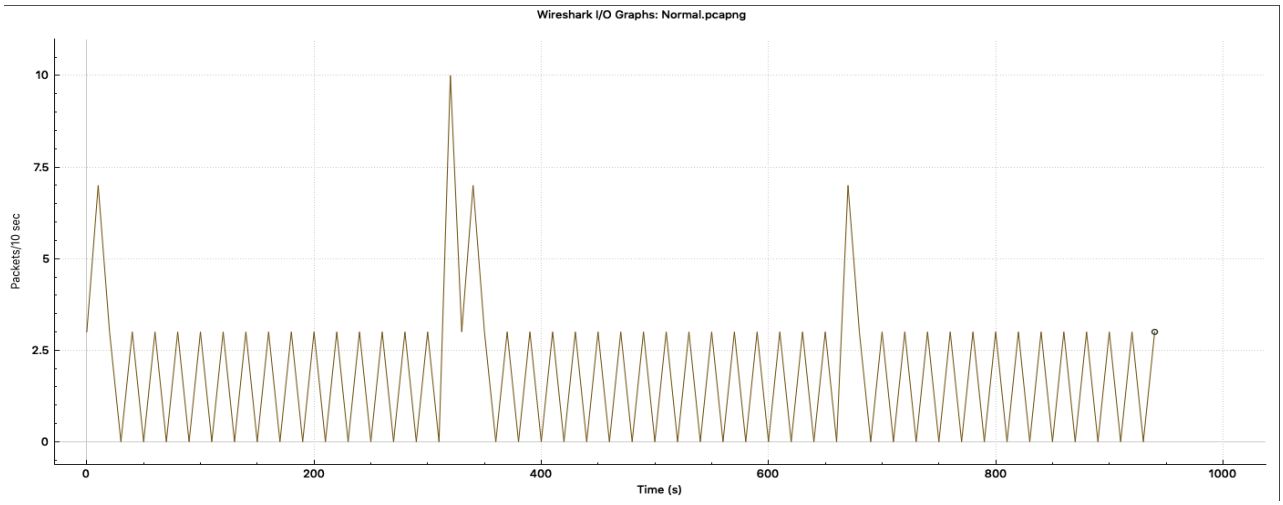


Figure 5.2: Wireshark Input/ Output Graph for Normal Communication

Figure 5.3 shows a graphical representation of SCADA_to_substation_attack 1 network communication. From the graph, it is notable that at after around 500 seconds, too many packets were being sent and dropped at the same time, which is an indication of an attack (DoS). This attack was simulated by continuously switching on and off a network device as detailed in section 3.3.1.

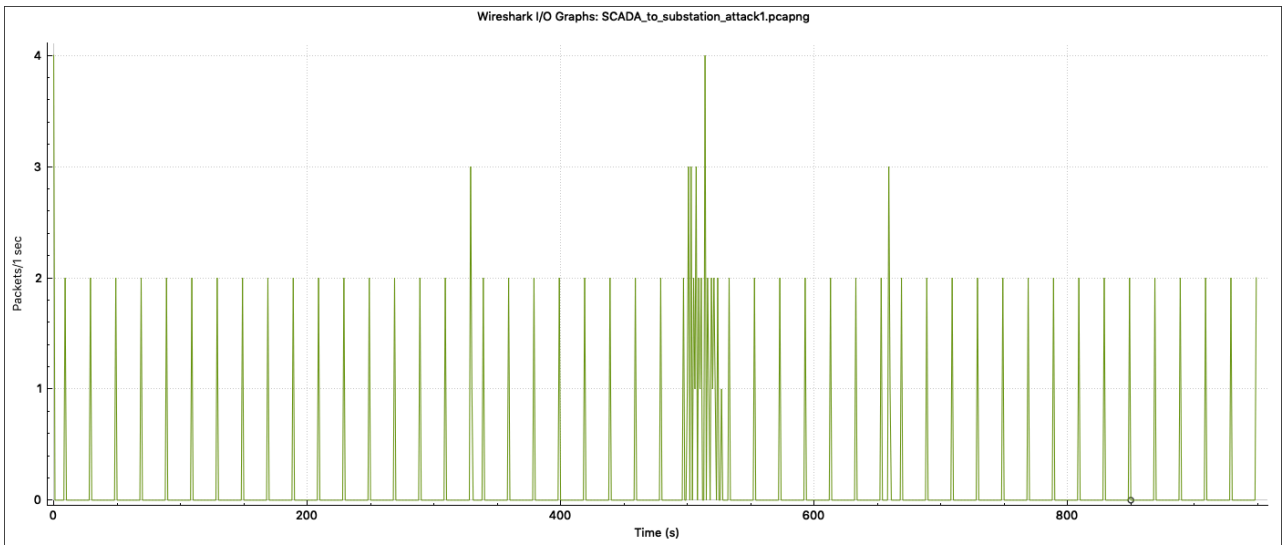


Figure 5.3: Wireshark Input/ Output Graph for Attack 1

5.4.2 TCP Payload Extraction

Figure 5.4 and 5.5 shows TCP payload extraction process from SCADA_to_substation_normal file. The user is required to run create_profile.py script in order for the TCP payload data to be extracted. Tshark is integrated with the script to extract specific attributes from the ICS packet capture file. In the create_profile.py script, flag -cp is a predefined argument for creating profile while -o is an argument for outputting the results into a text file. Extracting TCP payload from the normal communication packet capture file was achieved through this command: *python create_profile.py -cp SCADA_to_substation_normal.pcapng -o Normal_Payload.txt*.

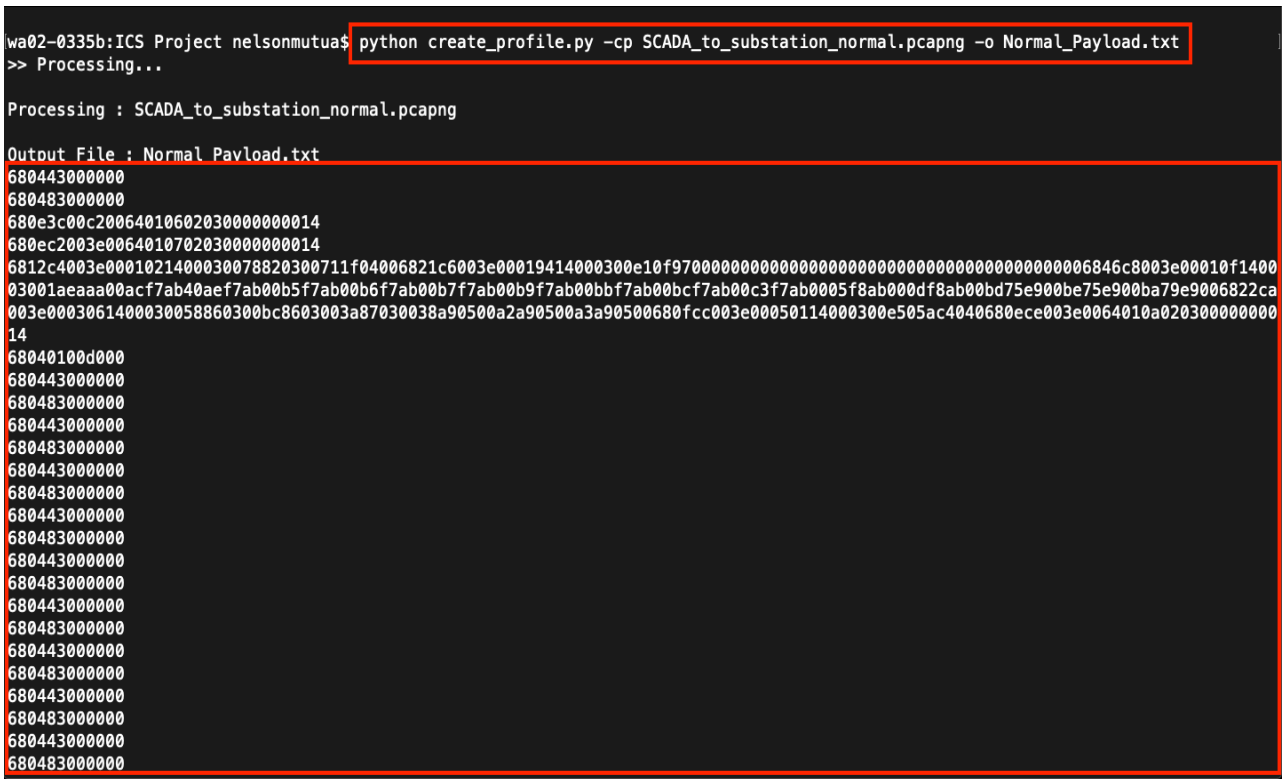


Figure 5.4: Payload Extraction from Normal File

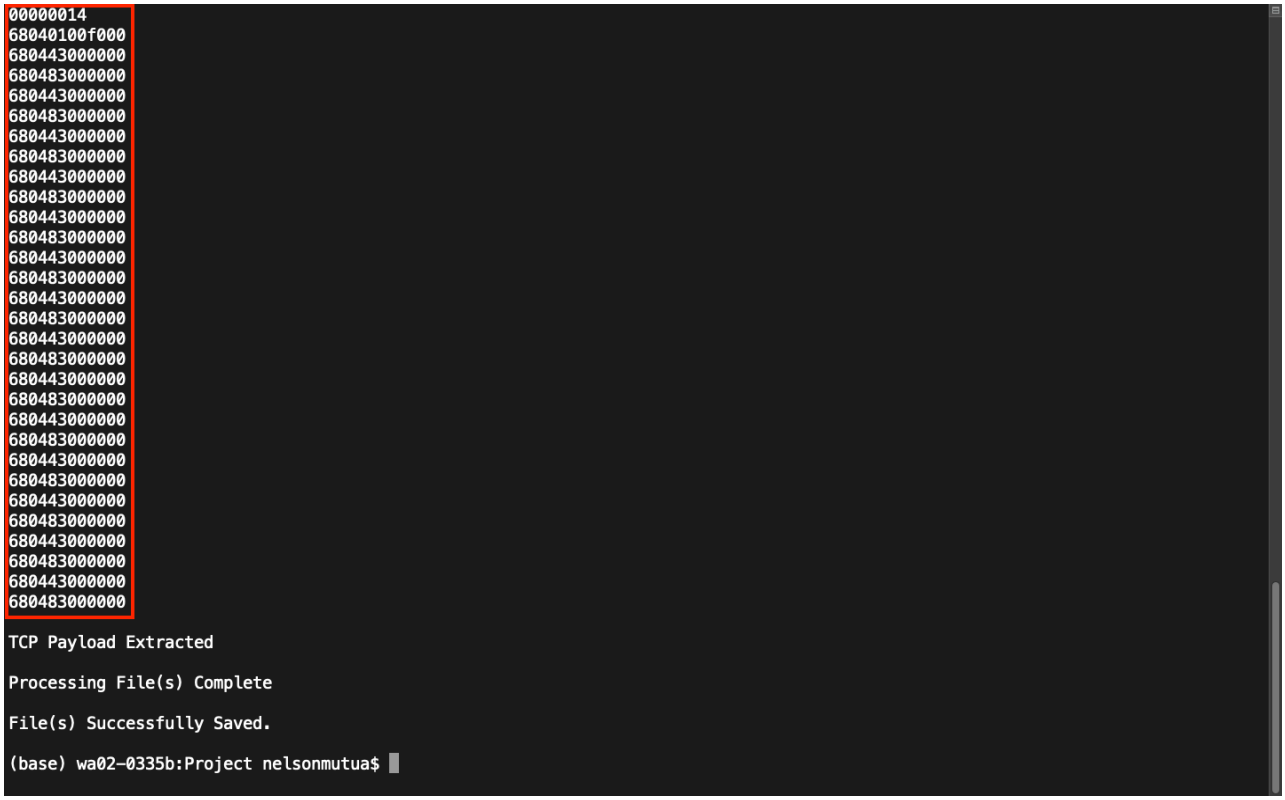


Figure 5.5: Successful TCP Payload Extraction from Normal File

Figure 5.6 shows the extracted TCP payload saved in a text file, Normal_Payload.txt.

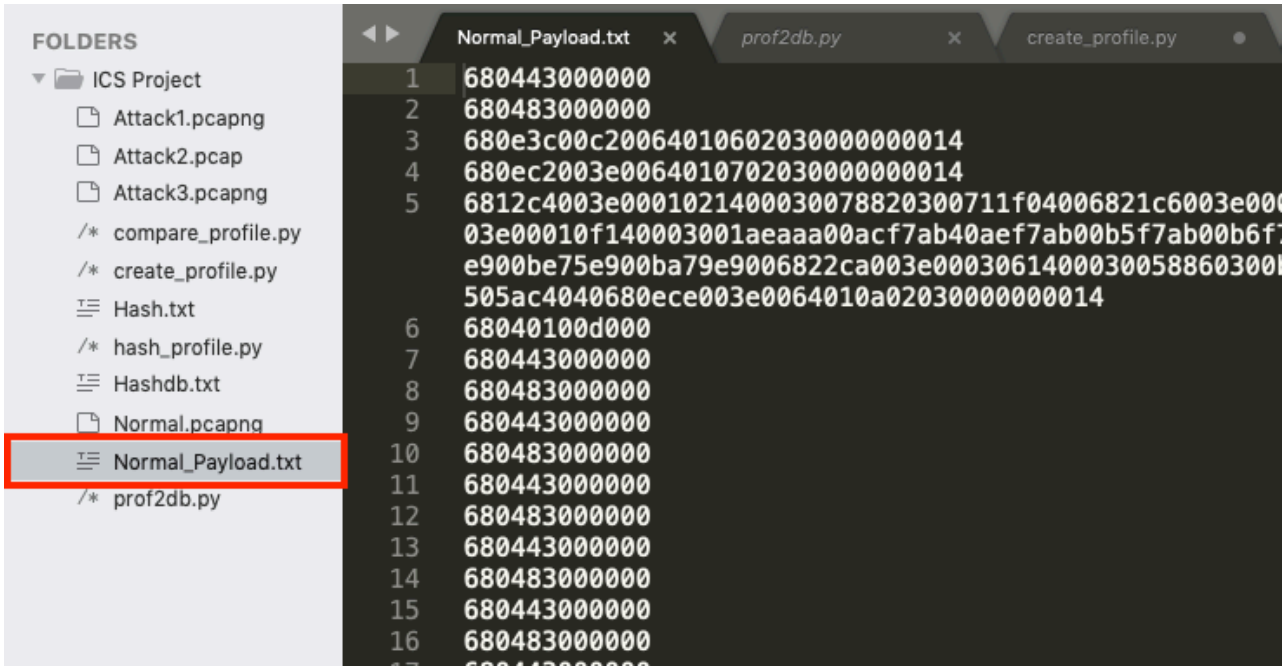


Figure 5.6: Normal Payload File

Network communication in IEC 104 involves 2 network nodes. In SCADA_to_substation_attack 2 simulation network nodes ranging from 172.17.1.20 to 172.17.1.119 were used. This attack was caused by ASDU messages C_IC_NA_1 (interrogation command). In order to use the ICS network profiling tool in attack 2, there is need to narrow down to communication between 2 nodes. IP 172.17.1.20 and 172.17.2.135 were selected for testing. Network packets between these IP were filtered and exported using Wireshark. Figure 5.7 illustrates attack 2 packets filtering.

Figure 5.7: Filtering Attack 2 Packets

Figure 5.8 below illustrates successful TCP payload extraction from all the SCADA_to_substation packet files. i.e. Normal, Attack 1, 2 and 3 packet capture files. The command below was used in this research to extract payload from the different SCADA packet capture files.

Python create_profile.py -cp SCADA_to_substation_normal.pcapng -o Normal_Payload.txt -cp SCADA_to_substation_attack1.pcapng -o attack1.txt -cp SCADA_to_substation_attack2.pcap -o attack2.txt -cp SCADA_to_substation_attack3.pcapng -o attack3.txt

```
wa02-0335b:ICS Project nelsonmutua$ python create_profile.py -cp SCADA_to_substation_normal.pcapng -o normal.txt -cp SCADA_to_subs
tation_attack1.pcapng -o attack1.txt -cp SCADA_to_substation_attack2.pcap -o attack2.txt -cp SCADA_to_substation_attack3.pcapng -o
 attack3.txt
>> Processing...

Processing : SCADA_to_substation_normal.pcapng

Output File : normal.txt
TCP Payload Extracted

Processing File(s)Complete


Processing : SCADA_to_substation_attack1.pcapng

Output File : attack1.txt
TCP Payload Extracted

Processing File(s)Complete


Processing : SCADA_to_substation_attack2.pcap

Output File : attack2.txt
TCP Payload Extracted

Processing File(s)Complete


Processing : SCADA_to_substation_attack3.pcapng

Output File : attack3.txt
TCP Payload Extracted

Processing File(s)Complete
```

Figure 5.8: TCP Payload Extraction from Packet Files

Figure 5.9 below shows a screenshot of all the four output text files.

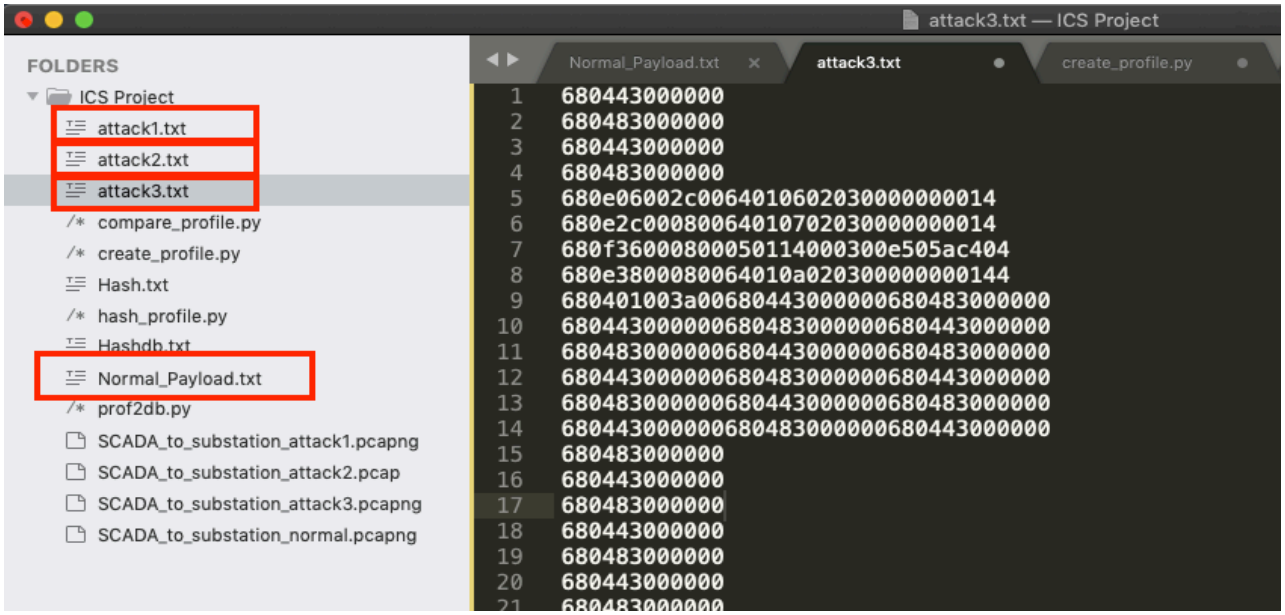


Figure 5.9: TCP Payload Output Files

5.4.3 TCP Payload Hashing

Figure 5.10 below shows a screenshot of successful hashing of the payload files extracted in figure 5.7 above. The user is required to run Hash_profile.py script in order to hash the input files. Flag -ch is defined in the script as an argument for creating hash using ssdeep while -o is defined as an argument to output the hash into a text file. The command below was used to hash all the payload files extracted in the stage above in this research.

Python hash_profile.py -ch Normal_Payload.txt -ch attack1.txt -ch attack2.txt -ch attack3.txt -o HashFile.txt

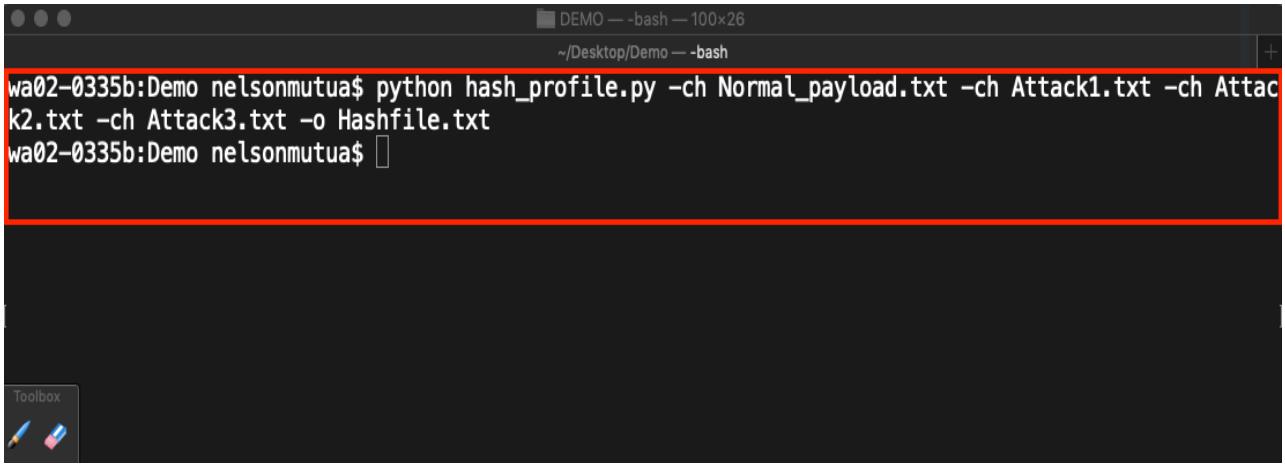


Figure 5.10: Payload Hashing Command

Figure 5.11 below shows the hash values generated from the ICS payload files and saved into a text file.

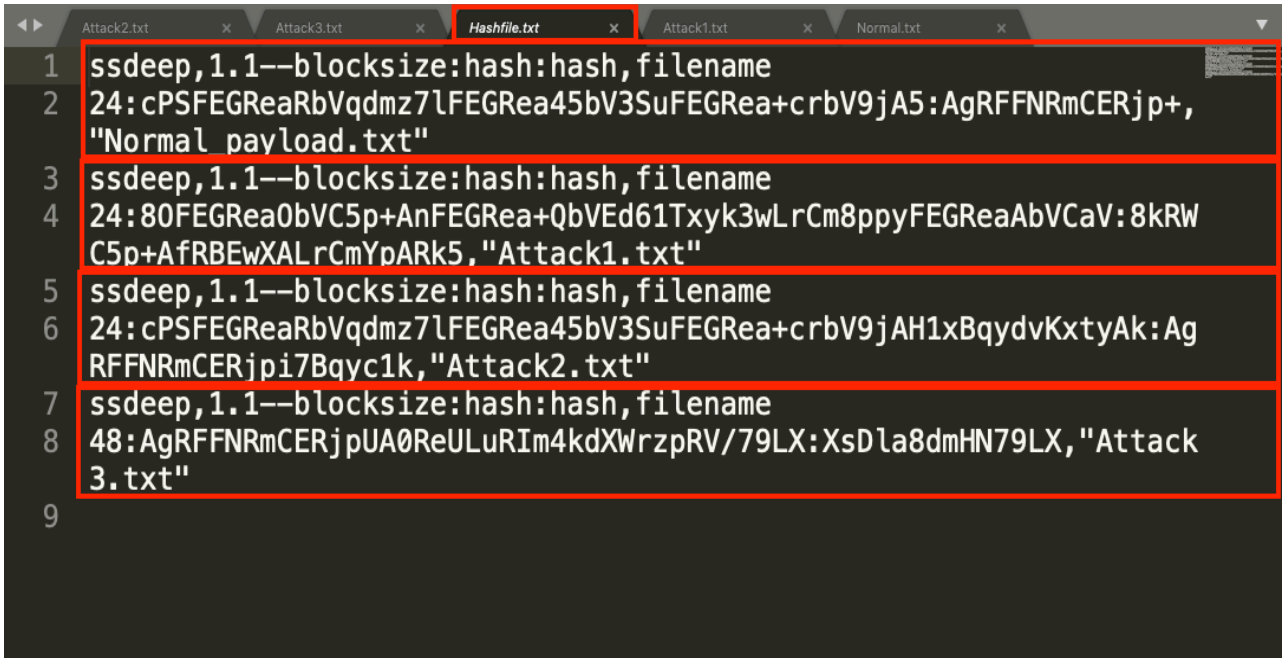


Figure 5.11: Normal and Attacks Hash Value

5.4.4 Pattern Matching

Compare_profile.py script is used for pattern matching. The user can compare one or more payload files against the hash values using –c flag. The file containing the hash values to be used for

comparison against the input files can flagged using -hf. The output result is the similarity percentage between the hash values and the input files.

Figure 5.12 shows a screenshot of the compare_profiles.py script calculating the similarity between the ICS normal communication profile and attack 1 payload files. Normal_Payload.txt was extracted from SCADA_to_substation_normal.pcapng while attack1.txt was extracted from SCADA_to_substation_attack1.pcapng packet capture file as illustrated in figure 5.8 above. The terminal command below was used to pattern match the hash file with Normal_Payload.txt and attack1.txt files. *Python compare_profile.py -hf Hash1.txt -c Normal_Payload.txt -c attack1.txt*

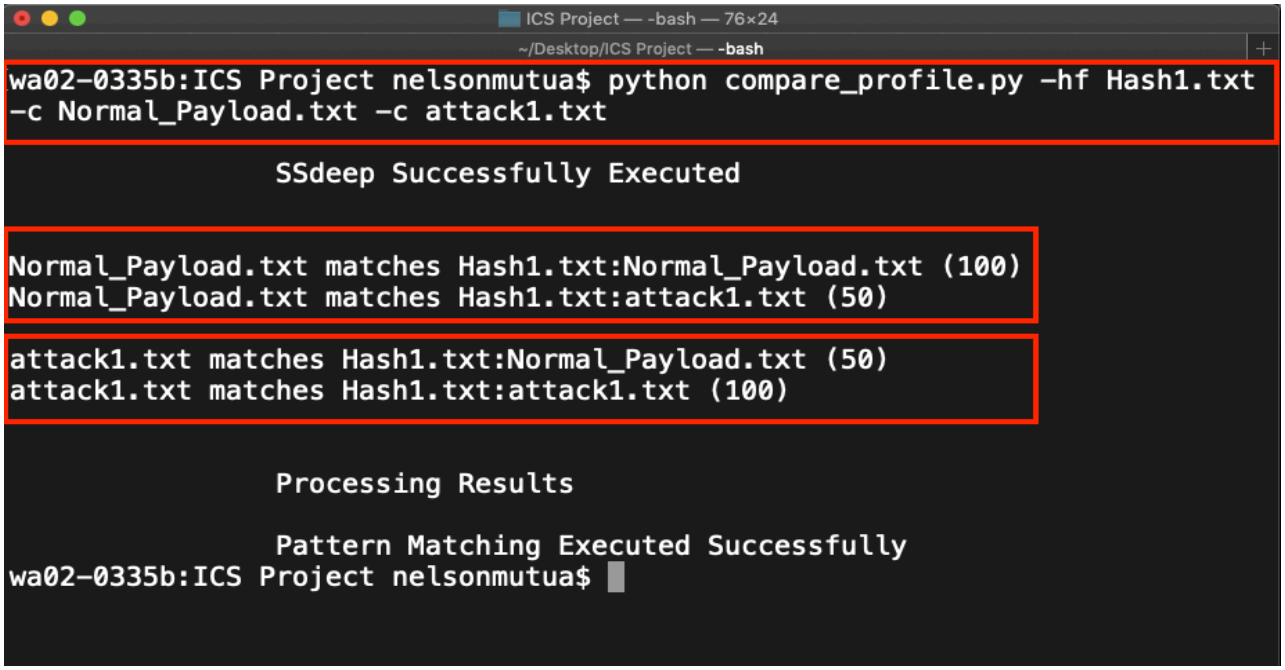


Figure 5.12: Attack 1 Pattern Matching

The results obtained show that the normal profile hash value matches Normal_Payload.txt file 100%. Attack 1 hash value matches Normal_Payload.txt file 50%. Since SCADA_to_substation_attack1 contained packets of DoS attack simulation, the attack 1 communication profile could not match the normal communication 100%. The attack capture file contained an ASDU messages with double commands. The attack simulation caused a malfunction between the communication nodes. This

resulted to instability in the IEC 104 network communication. The application was able to detect the threshold of the malicious packets contained in attack 1 file.

The network profiling tool can also compare one hash file against many files. Figure 5.13 below shows a screenshot of the compare_profiles.py script calculating similarity of the normal ICS hash value against attack 1, 2 and 3 payloads extracted from the packet capture files. The similarity matching score between the normal profile and attack 1, attack 2 and attack 3 is 50%, 86% and 50 respectively. The application was able to detect malicious communication in all the packet files. The command below was used to pattern match the normal hash against attack 1, attack 2 and attack 3 payloads. *python compare_profile.py -hf Hashfile.txt -c Normal_payload.txt -c Attack1.txt -c Attack2.txt -c Attack3.txt.*

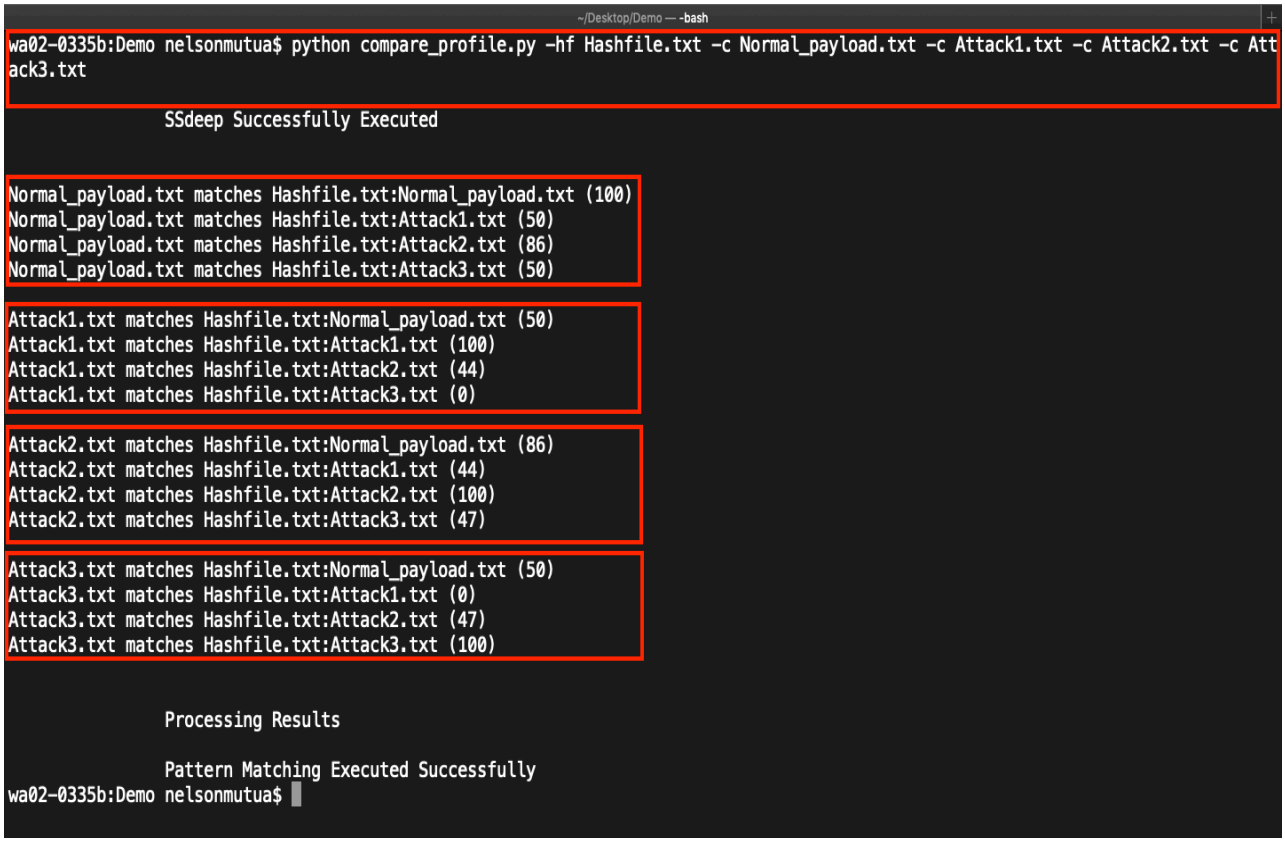


Figure 5.13: Pattern Matching Different Attack Files

5.5 Man-In-The-Middle Attack Simulation

Through the MITM attack, an attacker can inject or alter readings and commands in the communication stream in real time. While intercepting packets, some packet can be dropped, altered or a new packet can be injected with arbitrary outcome. Attackers can inject malicious data into the ICS communication to cause harm. In this study, MITM attack was simulated by artificially modifying the first APCI control field byte for the first 10 IEC 104 packets in the normal packet capture file. This binary modification was performed by using a binary tool editor known as Hex Editor. For instance, the control field in a packet binary data 6804**43**000000 was modified to 6804**50**000000. The bolded characters represent the changes. Figure 5.14 below illustrates binary data manipulation using Hex Editor.

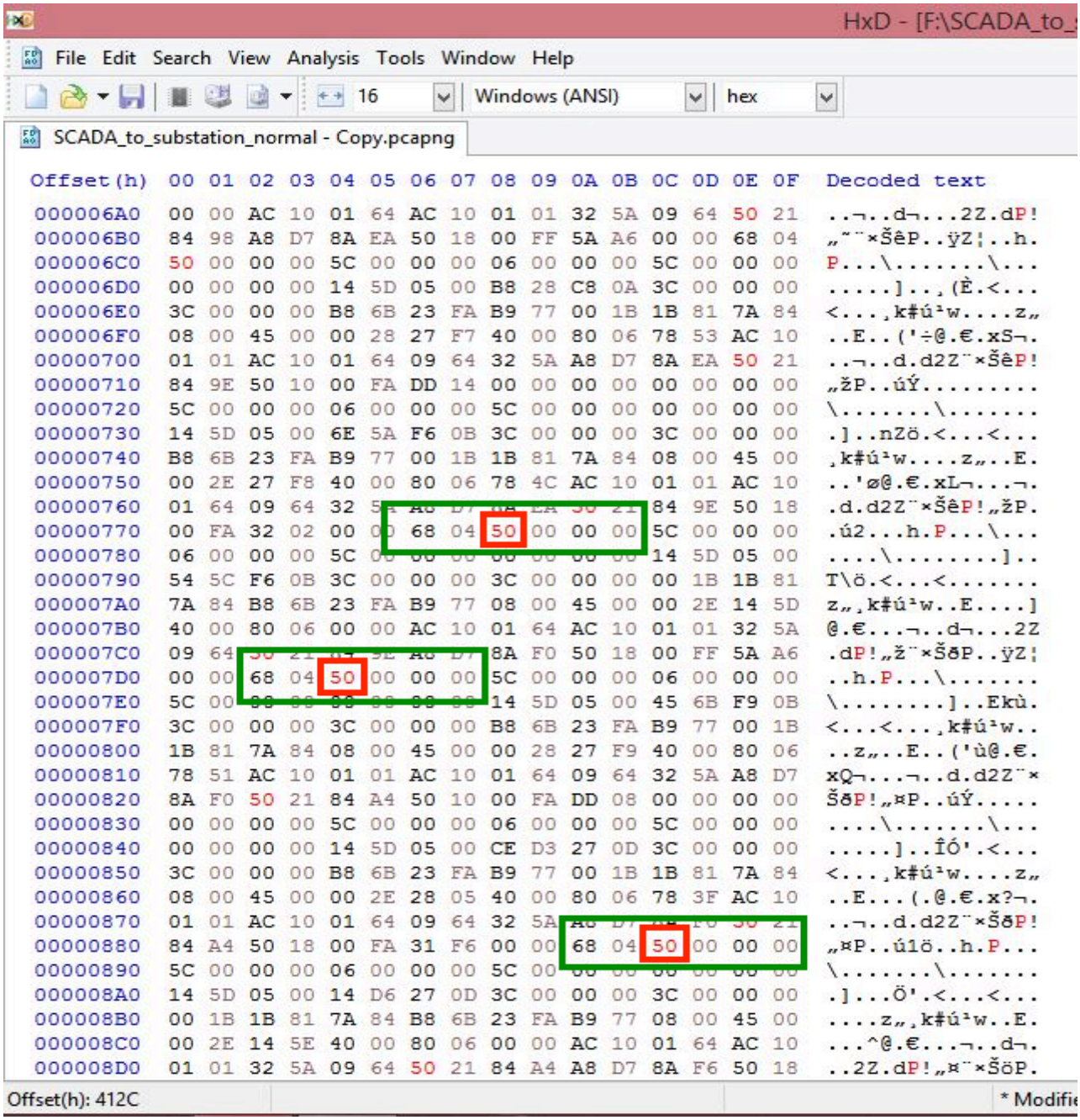


Figure 5.14: Binary Data Manipulation

After manipulating the first APCI control fields in the first 10 packets, the payload is reextracted from the edited SCADA_to_substation_normal.pcapng using create_profile.py script. Figure 5.15 below demonstrates the extraction of TCP payload from the edited normal packet capture file. The

terminal command used to extract the payload is: *python create_profile.py -cp SCADA_to_substation _edited.pcapng -o Edited_Payload.txt*

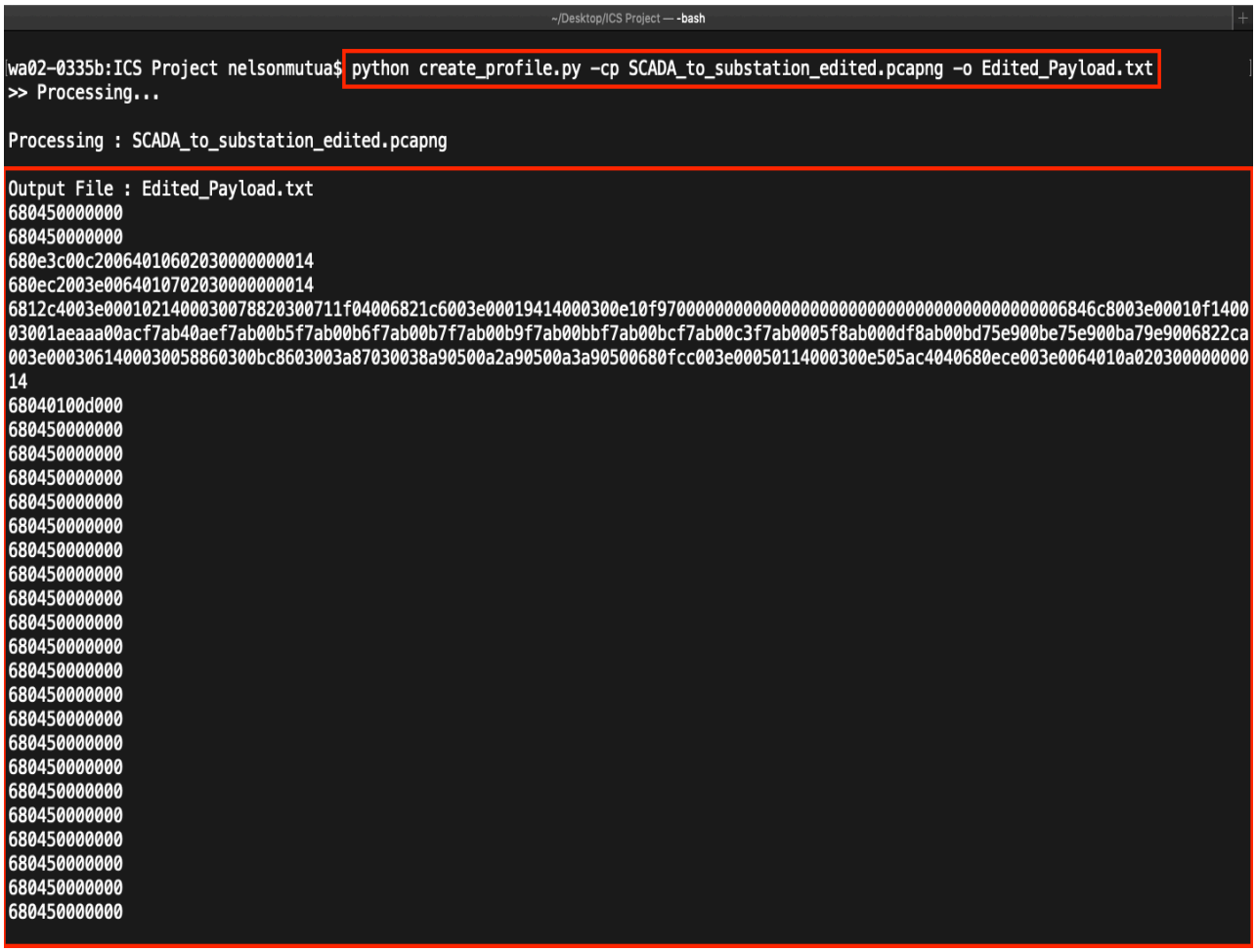


Figure 5.15: Extracting Payload from Edited Packet File.

The normal and edited payloads were then hashed using hash_profile.py script. The python command used to hash these normal and edited payloads is: *Python hash_profile.py -ch Normal_payload.txt -ch Edited_payload.txt -o Editedhash.txt*

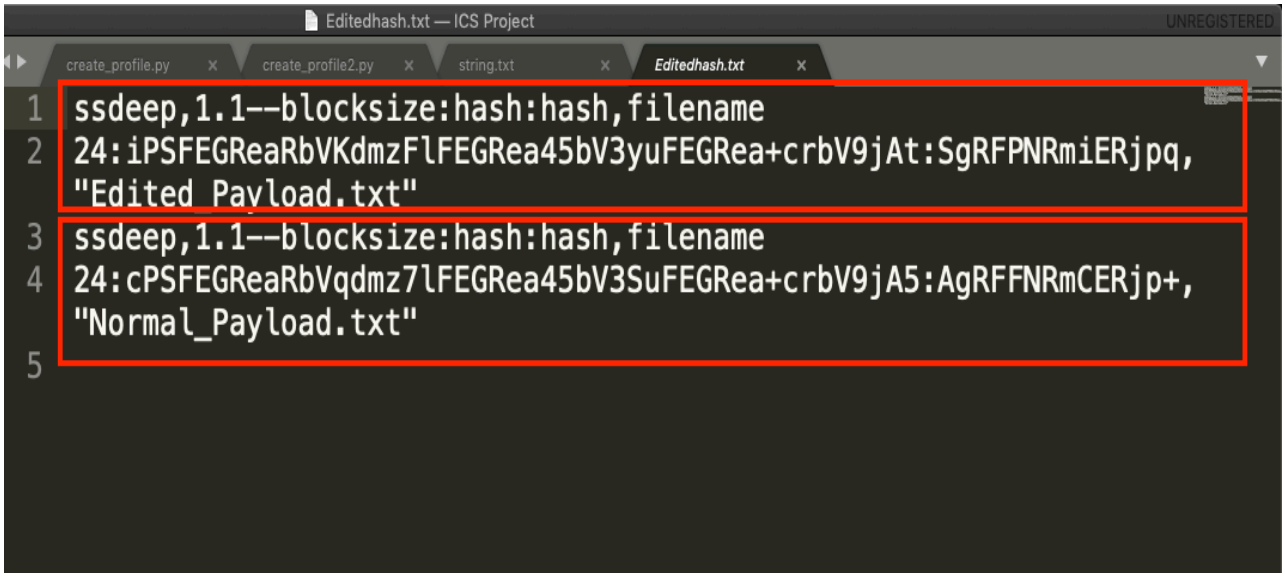


Figure 5.16: Normal and Edited Payloads Hash

Similarity was then computed to compare the original normal payload file with the manipulated payload. Figure 5.17 below is a screenshot of similarity matching between the original and the manipulated packet capture file. The matching score between these packet capture files is 91%. This is a proof that this tool can be used to detect for any manipulation of binary data in packets. The python command below was used. *Python compare_profile.py Editedhash.txt -c Normal_payload.txt -c Edited_payload.txt*

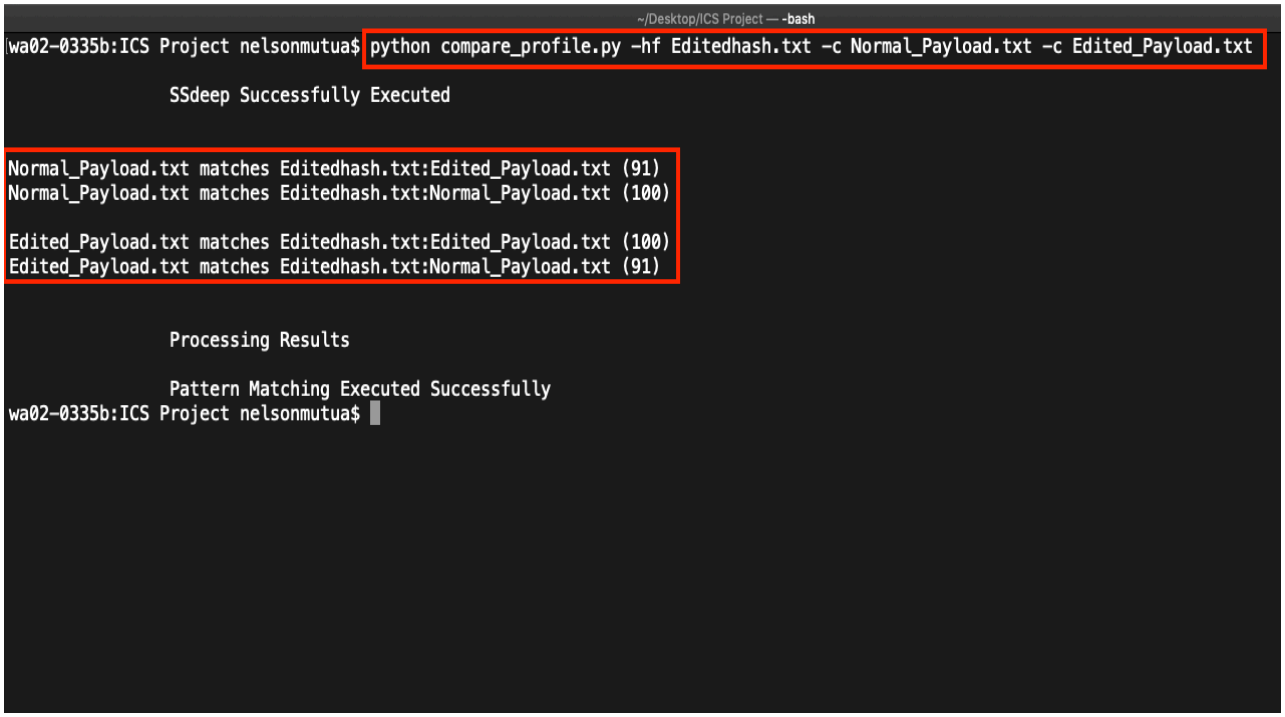


Figure 5.17: Attack Simulation Results

5.6 System Testing

5.6.1 Introduction

To ascertain that the ICS network-profiling tool meets the functional and non-function requirements, several tests were done. Testing was an important part of the system's lifecycle as it helped the researcher to determine whether or not the research objectives were achieved. Through testing, the researcher was also able to determine what enhancements are needed in the subsequent version of the system. Finally, testing helped to verify the functionality of the application thus ascertaining whether or not the entire application worked seamlessly. The application testing was conducted in two broad areas: system functionality and system usability.

5.6.2 Functionality Testing

Functionality testing was done to ensure that the system met all the specifications and requirements defined in chapter 3 of this research. The sample test cases below illustrate how the developed

application matched the proposed requirements. In this section, a detailed description of all the functions of the application is done. The tests were carried out on Mac OS, Linux and Windows operating systems.

Table 5.1: Test Plan

Item	**Value**
Objectives	- To define the tools to be used through the testing process. - To identify the items to be tested and the expected outcome. - To define the environmental needs. - To provide a procedure of how the tests will be conducted.
Test Items	- Functional Testing. - Usability Testing.
Features to be Tested	- Payload extraction. - Fuzzy hashing. - Pattern Matching
Approach	- Manual tests will be carried out on all scripts of the application. - Test cases will be developed with step-by-step procedures for testing the features. - Sample ICS network packets capture files to be analysed and profiled by the application will be availed to the tester. - The user testing the application will fill in prepared manual test cases with the results of the testing. - User is required to repeat testing the application using different ICS network packets capture files to ensure the results are repeatable.

Item	Value
Item Pass/Fail Criteria	- All core functionality of the system should function as expected and outlined in the individual test cases. - There must be no critical defects found. - 95% of all test cases should pass.
Suspension Criteria	- Testing should be paused immediately of either part of the application fails to execute successfully.
Test Deliverables	- Test plan (this table). - Test Cases. - Test Reports.
Test Environment	- Linux or Windows operating system. - Python version 2.7. - Testing tools include ssdeep.

The table 5.3 below shows the use case summary results filled in by the user testing the tool. From the actual results obtained, it is evident that all test cases succeeded and that the tool is working as required. The main functionality of the application was to profile ICS network communication using approximate pattern matching algorithm. Normal ICS network communication profile is used as a baseline for comparison with other unknown ICS network communications. Ssdeep is used to calculate the similarity between the profiles. Table 5.3 below shows the steps performed in testing the ICS network-profiling tool.

Table 5.2: Profiling ICS Network Packets Test Case

Test Case Name: Profiling ICS Network

Date Tested: 24th March 2020.

Tested By: Nelson Mutua

Test Description: Step by step test for profiling ICS network packets

TEST STEPS

Pre-Condition:

Python 2.7 should be preinstalled to enable execution of the python scripts.

The computer in use must have Wireshark installed for supporting Tshark, which is required for extracting the payload from the packet capture files. Ssdeep pattern matching algorithm should also be installed to be used for hashing and pattern matching the profiles. Further, sample ICS packet files should be available to analysis.

Post-Condition:

The application calculates the matching score between the two or more profiles imported into the python script.

Step	Action	Expected Response	Pass/Fail	Comments
1.	Run create_profile.py script to extract TCP payload from the packet capture file.	A text file containing the payloads is generated and is generated and saved.	Pass	None
2.	Run hash_profile.py script to fuzzy hash the extracted payloads into one file.	A text file containing hash values for all the	Pass	None

		payloads is generated and saved.		
3.	Run compare_profiles.py script to pattern match the hash values against the payload files.	A matching score between the hash value and the payload files is calculated using ssdeep.	Pass	None

5.6.3 Usability Testing

Usability testing was conducted to ensure that the application met the required aesthetic values. This testing focused on the ease of use, efficiency, usefulness, responsiveness and the execution speed. This was an important test since application users always like applications that are simple to understand and use. Table 5.4 below describes the tests that were done to ensure the application developed from this study met the required usability.

Table 5.3: Usability Testing

Test Case Name: Application Usability **Date Tested**: 24th March 2020. **Tested By**: Nelson Mutua **Test Description**: Step by step test for application usability.
TEST STEPS
Pre-Condition: All python scripts must have executed successfully.
Post-Condition:

User can seamlessly run the application.				
Step	**Action**	**Expected Response**	**Pass/Fail**	**Comments**
1.	User can execute all the python scripts.	All the python scripts are executable.	Pass	None
2.	The script terminal commands are simple.	Terminal commands are simple to understand and easy to memorise.	Pass	None
3.	The user can interpret the results.	The results generated are in percentage measure and easier for the user to interpret.	Pass	None

As illustrated below in figure 5.18, out of the 5 respondent who participated in application usability, 4 rated "efficiency" attributes excellent and 1 rated very good. On "ease of use" and "responsiveness" attribute, all the 5 respondents rated excellent. On "speed" attribute, 3 respondents rated excellent while 1 rated very good. On "usefulness" attribute, 4 respondents rated excellent while 1 respondent rated very good. Generally, this was a good feedback from the users.

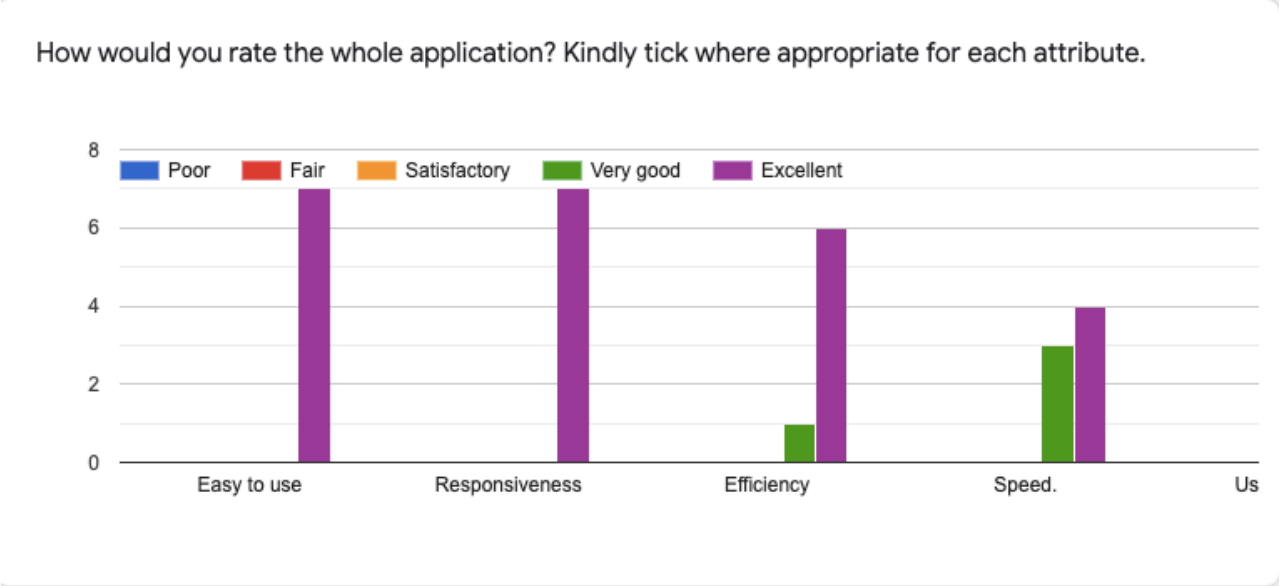


Figure 5.18: Usability Results

5.7 System Validation

System validation is defined as the set of activities and processes aimed at verifying that the application is performing as expected in line with the set objectives. The main objective of this research was to develop an application that profiles ICS communication using approximate matching algorithm. The implemented application was able to achieve the above in that it has been able to extract IEC 104 payload from ICS packet capture files. The application can then hash the payload using ssdeep and compare different communication profiles to compute the matching score.

Testing was done to ensure the developed application did not have bugs. A bug is defined as an error or flaw in a computer application that causes the software to give an incorrect or unexpected result (SteelKiwi, 2015). Testing for bugs was done through conducting several tests repeatedly to determine the result given was consistent. NIST's guidelines state that the results of an application should be repeatable and reproducible (Brunty, 2017).

i. **Repeatability** refers to obtaining the same results when using the same method on identical test items in the same environment by the same operator using the same equipment within

short intervals of time. The tests were performed five times using the same packet capture files and produced the same results.

ii. **Reproducibility** refers to obtaining the same results being obtained when using the same method on identical test items in different laboratories with different operators utilizing different equipment. The application was installed in Mac OS, Linux and Windows operating systems and peer reviewed using the same packet capture files produced the same results for all instances.

The performance of the developed application was determined by using three metrics namely accuracy, sensitivity and specifying. Accuracy is the overall success rate of the application. It gives the proportion of the input test that are correctly identified. Sensitivity also known as true positive is the rate at which attack profiles were detected by the normal profile to ascertain attacks packets. Specifying also known as true negative is the rate at which normal communication was correctly detected as normal and attack free. Therefore, sensitivity quantifies the avoidance of false negative (attack profiles incorrectly detected as normal profile) while specifying quantifies the avoidance of false positive (normal communication incorrectly detected to contain an attack).

Figure 5.19 below illustrates normal profile pattern matching. From the results, the normal profile hash value matched the normal payload 100%. Out of the five tests done on every operating system, all the results turned out to be true negative (normal communication was correctly detected).

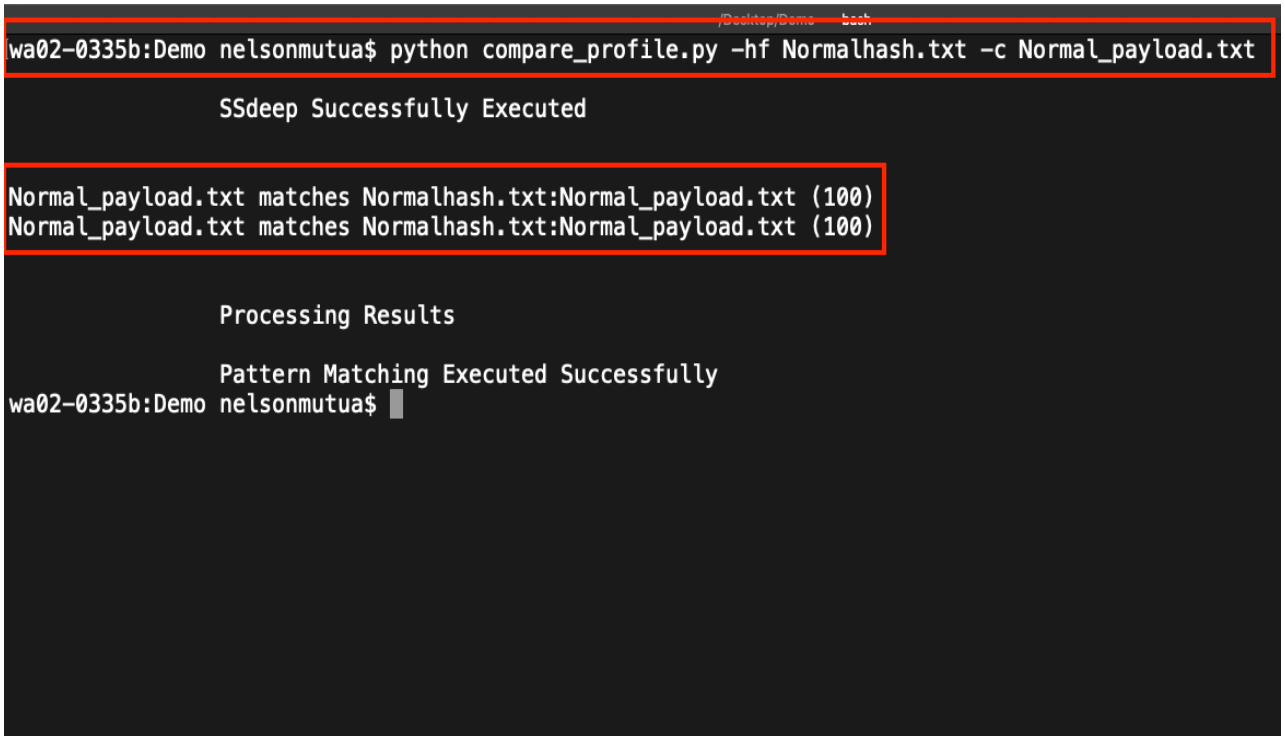


Figure 5.19: Normal Profile Matching

On the other hand, figure 5.20 below demonstrates pattern matching of attack 2 profile. This was achieved by pattern matching the normal profile with attack 2. Attack 2 payload was positively identified to contain 86% of normal communication profile, translating to 14% attack packets. As demonstrated in figure 5.13, attack 1, 2 and 3 profiles were positively identified in comparison with normal profile with matching scores of 50%, 86% and 50% respectively. Out of all the 5 tests done on every operating system, all the results turned out to be true positive (attack profiles were detected).

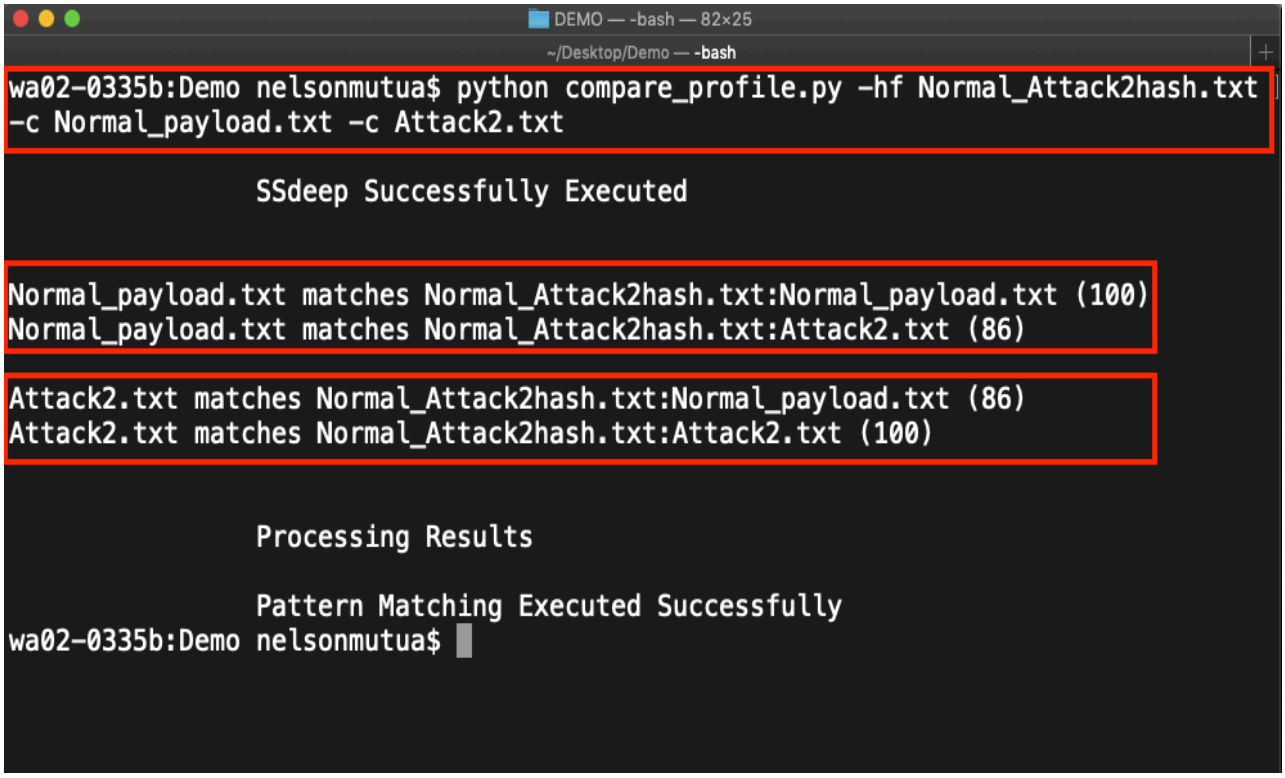


Figure 5.20: Attack 2 Profile Positively Identified

In order to interpret the results in a more logical manner, the researcher did experimental tests to objectively establish a threshold value. A threshold is defined as a maximum or minimum value which serves as a benchmark for comparison guidance (Han, 2015). A threshold acts as the point above which we should ignore any positive results. From experiments done in this research, a threshold point was determined to be 98% because the rise in false positives will overwhelm the rise in true positives. The matching scores was contrasted with the predefined threshold and a score greater than threshold is identified as normal communication. Likewise, if the score is less than threshold, the traffic indicates the existence of attack packets.

5.8 Conclusion

The ICS network profiling application passed the functionality and usability tests conducted. The positive tests were a clear indication that the network administrators without any user dissatisfaction fit the application for use. From the tests, the users provided positive feedback and recommended the application for use.

CHAPTER 6 : DISCUSSION OF RESULTS

6.1 Overview

The results of the study formed the basis on which ICS network-profiling application was developed. According to the study, the application was moreover tested to determine its functionalities. This chapter analyses the study findings in relation to its objectives. It also evaluates to what extend the findings agreed with the literature review.

6.2 Objective One

The first objective in section 1.3 was to find out which approximate matching algorithm can be used to profile ICS network communication. From the study findings, ssdeep pattern-matching algorithm was found suitable for use in this study. During the hashing process, ssdeep breaks up a file into chucks using the result of a rolling hash function. It then uses a different hash function to produce a small hash for each piece and then concatenate the results to produce the hash signature for the whole file. This algorithm is highly dependent on the presence of a large, continuous chuck of common data. Files that have common content will exhibit some level of similarity in their signature while unrelated files would not.

Exchange of data in IEC 104 is considered stable since it involves machine to machine communication and data exchanges is in a predictable manner. This makes ssdeep suitable for this research since ssdeep it can easily detect a slight change in the input bytes. In addition, ssdeep was found advantageous in this research since it has a fast computation time compared to other fuzzy hashing algorithms as discussed in chapter 2 of this research. The developed ICS network-profiling tool uses ssdeep algorithm for hashing the TCP payloads to generate hashes and also pattern matching the known profile against the unknown profiles to determine the matching score.

6.3 Objective Two

The second objective sought to profile ICS communication using approximate matching algorithm. Section 5.4.3 of this study helped to achieve this objective. Ssdeep generated profiles for each input file, which were used to match the profile against the known communication profile and find any possible similarity or matches. The profile of the normal ICS network communication was saved into a text file and used for comparison with other unknown profiles generated from the sample packet capture files containing network attacks. In this research, any input file with a matching score lower than 98% when compared to the normal profile is an indication of malicious packets in the ICS network communication.

6.4 Objective Three

The third objective looked at designing and developing an ICS network-profiling tool using approximate matching algorithm. This objective was partly achieved in chapter four of this research, which assisted the researcher to come up with the system design and architecture of the proposed application. This objective was also partly met under section 5.1 where the application was developed and implemented to meet the desired requirements of this research. The ICS normal profile computed from the normal network communication was used to determine the similarity with other unknown packet capture files. Thus, this study provided a good basis for designing and developing a tool to profile ICS network communication.

6.5 Objective Four

The fourth and final objective of this study was to test and validate the effectiveness in terms of accuracy of the solution. Testing was extensively done under section 5.5 of this study. Functionality and usability testing were all done on the application. All the respondents who participated in the usability testing did not experience any problems using the application. Out of the 5 respondents who participated in the application usability testing, 4 rated navigability ‘excellent’ and 1 rated ‘very

good'. The entire testing process as a whole, helped the researcher to identify future work that could be added onto the application as discussed in Section 7.3.

6.6 Advantages of the Developed Solution Compared to Existing Tools

6.6.1 Multiplatform

The application implemented in this study is developed on python programming language, which is supported in many operating systems. This study focuses only on IEC 104 communication protocol, but the application has the capability to work with different protocols.

6.6.2 Open Source

The network-profiling tool is an open source application that can be downloaded via https://github.com/nelsonmutua/ICS-Network-Profiling-Tool.git. It is distributed under the MIT License whose conditions only require preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

6.6.3 Integration

The application was integrated with ssdeep for pattern matching purposes. Once the payload files to be pattern matched are selected, the application will automatically check for the similarity without any user interventions.

6.7 Conclusion

The objectives of the study sought to understand the ICS network communication and IEC 104 protocol. Consequently, the study sought to analyse different pattern matching algorithms and find out how similarity is computed in files. Further, how to profile ICS network communication using pattern matching algorithm. Extensive research in this study, along with testing different algorithms in files, helped to inform the need of the developed ICS network profiling tool. This tool was built to inform the ICS network administrators and researchers on how unknown ICS network

communication can be analysed and determine whether network packet files contain attacks. Unknown network communication profile highly differing with the known profile can be an indication of malicious packets or attacks contained in a network communication. Encouragingly, the application was received with positive feedback from the need for its functionality and usability.

CHAPTER 7 : CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORK

7.1 Conclusions

Profiling network communication using approximate matching algorithms is a relatively new concept when it comes to ICS communication. The study reveals that ICS use IDS and firewalls to monitor and detect attacks in a network communication. Such security mechanisms increase the attack surface of the ICS, which controls critical systems in an industry. These security mechanisms also have a high false rate, which can disrupt the normal operations of the system.

Using the development tools discussed in chapter 4, the ICS network-profiling tool was developed. During system development phase, agile methodology was used to implement the application. This allowed for more and frequent release with subsequent user feedback, which led to development of a usable and more reliable application. The system usability testing was performed thoroughly and respondents found it useful and satisfying. The aim of the developed application is to analyse IEC 104 protocol, create a profile for ICS network communication under normal circumstance and compare this known ICS network profile against other ICS unknown profile. The application should then compute the similarity between the two profiles using ssdeep algorithm and output the matching score. The matching score will determine the need for further analysis to identify the attack packets contained in the network communication.

Findings from the research conducted in this study indicated that ICS network administrators and researchers could benefit from the proposed solution especially when they need to check for attacks in packet capture files. Moreover, the findings were found to be relevant to researchers from all avenues, be it students, network administrators, etc., in need of analysing ICS network communication using approximate pattern matching algorithm.

7.2 Recommendations

The findings of this study were key in understanding the ICS network communication. The ICS network profiling tool was of great importance to researchers and network administrators. From this research, it was noted that use of one communication protocol did not reach the satisfaction of the users. To make the users more receptive, this study recommends the following:

i. There is need for public awareness to the existence of this open source application - The tool can be advanced by collaboration with organizations using ICS to encourage network administrators to use it and avoid overreliance on manual inspection of network packet capture files which is proven to be tedious and time consuming. The open source community is encouraged to test, evaluate, critique and give recommendations to improve this tool.

ii. Multi-protocol evaluation – Analysis of network packets from other protocols in ICS network communication will help to ensure most of the protocols are covered in the ICS network profiling tool.

7.3 Future Work

ICS is known to have relatively fixed operational objects and business processes which result in relatively stable traffic pattern under normal conditions. Fluctuations of network traffic generally indicate the status change in ICS communication. The scope of this study was to detect anomalies in ICS network packet capture files using approximate pattern matching algorithm with focus on IEC 104 protocol. Future work will concern studying how to analyse and detect anomalies in ICS network communication in a real time manner. This can also be enhanced by incorporating other network attributes such as the source and destination IP, port number and average time interval between adjust packets. Pattern matching algorithm can then be used to match to match different attack patterns. Finally, SMS and email alert capability feature and identification of the attack packets can also be developed for the tool to offer real time monitoring and more comprehensive security service.

REFERENCES

Akselsen, B. (2016). *Intrusion Detection Systems*. Scitus Academics LLC.

Alert, D. (2016). Cyber-attack against Ukrainian critical infrastructure. Cybersecurity Infrastruct. Secur. Agency, Washington, DC, USA, Tech. Rep. ICS Alert (IR-ALERT-H-16-056-01).

Breitinger, F., & Baggili, I. (2014). File detection on network traffic using approximate matching.

Breitinger, F., & Baier, H. (2012). A fuzzy hashing approach based on random sequences and hamming distance.

Breitinger, F., Stivaktakis, G., & Baier, H. (2013). FRASH: A framework to test algorithms of similarity hashing. *Digital Investigation*, *10*, S50-S58.

Brunty. (2017). *Validation of Forensic Tools and Software: A Quick Guide for the Digital Forensic Examiner.*

Case, D. U. (2016). Analysis of the cyber-attack on the Ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, *388*.

Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., & Valdes, A. (2017, January). Using model-based intrusion detection for SCADA networks. In Proceedings of the SCADA security scientific symposium (Vol. 46, pp. 1-12).

Combs, M. M. (2012, September). Impact of the Stuxnet virus on industrial control systems. In *XIII International forum Modern information society formation-problems, perspectives, innovation approaches, St.-Petersburg, RUSSIA, September* (pp. 5-10).

Cruz, T., Barrigas, J., Proença, J., Graziano, A., Panzieri, S., Lev, L., & Simões, P. (2015, May). Improving network security monitoring for industrial control systems. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (pp. 878-881). IEEE.

D. Hadžiosmanović, D. Bolzoni, S. Etalle, and P. Hartel, "Challenges and opportunities in securing industrial control systems," in *2012 Complexity in Engineering (COMPENG). Proceedings*, June 2012, pp. 1–6.

Denscombe, M. (2014). The good research guide: for small-scale social research projects. McGraw-Hill Education (UK).

De Vries, J., Hoogstraaten, H., van den Berg, J., & Daskapan, S. (2012, December). Systems for detecting advanced persistent threats: A development roadmap using intelligent data analysis. In 2012 *International Conference on Cyber Security* (pp. 54-61). IEEE.

DeVon, H. A., Block, M. E., Moyle-Wright, P., Ernst, D. M., Hayden, S. J., Lazzara, D. J., & Kostas-Polston, E. (2007). A psychometric toolbox for testing validity and reliability. *Journal of nursing scholarship*, *39*(2), 155-164.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, *50*(9), 833-859.

Elavarasan, G., & Suresh, Y. (2017). Intrusion Detection in Industrial Control System by Packet Behaviour Based Analysis. *International Journal for Research in Science Engineering & Technology*, *4*(4), 33-39.

Fabro, M., Gorski, E., & Spiers, N. (2016). Recommended practice: improving industrial control system cybersecurity with defense-in-depth strategies. *DHS Industrial Control Systems Cyber Emergency Response Team*.

Falliere, N., Murchu, L. O., & Chien, E. (2011). W32. Stuxnet dossier. *White paper, Symantec Corp., Security Response*, *5*(6), 29.

Farwell, J. P., & Rohozinski, R. (2012). The new reality of cyber war. Survival, 54(4), 107-120.

Fellows, R. F., & Liu, A. M. (2015). *Research methods for construction*. John Wiley & Sons.

Frank Breitinger and Harald Baier. A fuzzy hashing approach based on random sequences and hamming distance. *7th annual Conference on Digital Forensics, Security and Law (ADFSL)*, pages 89–101, 2012.

Frank Breitinger and Harald Baier. Similarity preserving hashing: Eligible properties and a new algorithm mrsh-v2. 4th International ICST Conference on Digital Forensics & Cyber Crime (ICDF2C), 4, 2012.

Gaiceanu, M., Stanculescu, M., Andrei, P. C., Solcanu, V., Gaiceanu, T., & Andrei, H. (2020). Intrusion Detection on ICS and SCADA Networks. In *Recent Developments on Industrial Control Systems Resilience* (pp. 197-262). Springer, Cham.

Gemino, A., & Parker, D. (2009). Use case diagrams in support of use case modeling: Deriving understanding from the picture. *Journal of Database Management (JDM)*, *20*(1), 1-24.

Gemino, A., & Parker, D. (2009). Use case diagrams in support of use case modeling: Deriving understanding from the picture. Journal of Database Management.

Giraldo, J., Urbina, D., Cardenas, A. A., & Tippenhauer, N. O. (2019, June). Hide and seek: An architecture for improving attack-visibility in industrial control systems. In *International Conference on Applied Cryptography and Network Security* (pp. 175-195). Springer, Cham.

Goel, S., Hong, Y., Papakonstantinou, V., & Kloza, D. (2015). *Smart grid security*. London: Springer.

Hadžiosmanović, D., Bolzoni, D., & Hartel, P. H. (2012). A log mining approach for process monitoring in SCADA. International Journal of Information Security, 11(4), 231-251.

Hadziosmanovic, D., Bolzoni, D., Etalle, S., & Hartel, P. (2012, June). Challenges and opportunities in securing industrial control systems. In *2012 Complexity in Engineering (COMPENG). Proceedings* (pp. 1-6). IEEE.

Han, S. (2015). *Collection of schedule quality metrics and application to projects of the office of facilities planning and construction (OFPC)*.

Harbour, N. (2002). Dcfldd. defense computer forensics lab. In *Net5. 5.2* (Vol. 1).

Ho, T., Oh, S. R., & Kim, H. (2017). A parallel approximate string matching under Levenshtein distance on graphics processing units using warp-shuffle operations. *PloS one*, *12*(10).

Hu, Y., Yang, A., Li, H., Sun, Y., & Sun, L. (2018). A survey of intrusion detection on industrial control systems. *International Journal of Distributed Sensor Networks*, *14*(8), 1550147718794615.

Kerkers, M., Chromik, J. J., Remke, A., & Haverkort, B. R. (2018, February). A Tool for Generating Automata of IEC60870-5-104 Implementations. *In International Conference on Measurement, Modelling and Evaluation of Computing Systems*.

Kimberlin, C. L., & Winterstein, A. G. (2008). Validity and reliability of measurement instruments used in research. *Am J Health Syst Pharm*, *65*(23), 2276-84.

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, *3*, 91-97.

Langer, R. (2013, November). Resources. Retrieved from Langner: Retrieved 13th March 2020 from (langer.com): https://www.langner.com/wp-content/uploads/2017/03/to-kill-a-centrifuge.pdf

Lee, A., & Atkison, T. (2017, April). A comparison of fuzzy hashes: evaluation, guidelines, and future suggestions. In *Proceedings of the SouthEast Conference* (pp. 18-25).

Leith, H. M., & Piper, J. W. (2013). Identification and application of security measures for petrochemical industrial control systems. *Journal of Loss Prevention in the Process Industries*, *26*(6), 982-993.

Martínez, V. G., Álvarez, F. H., Encinas, L. H., & Ávila, C. S. (2015). A new edit distance for fuzzy

hashing applications. In *Proceedings of the international conference on security and management (sam)* (p. 326). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377-399.

Matoušek, P. (2017). Description and analysis of IEC 104 Protocol. *Faculty of Information Technology, Brno University of Technology, Tech. Rep.*

Matoušek, P., Ryšavý, O., & Grégr, M. (2019, September). Increasing Visibility of IEC 104 Communication in the Smart Grid. In *6th International Symposium for ICS & SCADA Cyber Security Research 2019 6* (pp. 21-30).

Mishra, J., & Mohanty, A. (2012). Software Engineering. New Delhi, India: Dorling Kindersley.

NIST, S. (2014). 800-168," *Approximate Matching: Definition and Terminology," Jul*.

Oliver, J., Forman, S., & Cheng, C. (2014, November). Using randomization to attack similarity digests. In *International Conference on Applications and Techniques in Information Security* (pp. 199-210). Springer, Berlin, Heidelberg.

Pagani, F., Dell'Amico, M., & Balzarotti, D. (2018, March). Beyond precision and recall: understanding uses (and misuses) of similarity hashes in binary analysis. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy* (pp. 354-365).

PowerMag, "What you need to know (and don't) about the aurora vulnerability," 2013. Retrieved 17th March 2020 from (powermag.com): http://www.powermag.com/ what-you-need-to-know-and-dont-about-the-aurora-vulnerability.

Remke, A., & Haverkort, B. R. (2018, February). A Tool for Generating Automata of IEC60870-5-104 Implementations. In Measurement, Modelling and Evaluation of Computing Systems:

19th International GI/ITG Conference, MMB 2018, Erlangen, Germany, February 26-28, 2018, Proceedings.

Robert Udd, Mikael Asplund, Simin Nadjm-Tehrani, Mehrdad Kazemtabrizi, and Mathias Ekstedt. Exploiting Bro for Intrusion Detection in a SCADA System. Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security - CPSS '16, pages 44–51, 2016.

Roussev, V. (2010, January). Data fingerprinting with similarity digests. In *IFIP International Conference on Digital Forensics* (pp. 207-226). Springer, Berlin, Heidelberg.

Sanders, C. (2017). *Practical packet analysis: Using Wireshark to solve real-world network problems*. No Starch Press.

Sarantinos, N., Benzaïd, C., Arabiat, O., & Al-Nemrat, A. (2016, August). Forensic Malware Analysis: The Value of Fuzzy Hashing Algorithms in Identifying Similarities. In *2016 IEEE Trustcom/BigDataSE/ISPA* (pp. 1782-1787). IEEE.

Sarantinos, N., Benzaïd, C., Arabiat, O., & Al-Nemrat, A. (2016, August). Forensic Malware Analysis: The Value of Fuzzy Hashing Algorithms in Identifying Similarities. In *2016 IEEE Trustcom/BigDataSE/ISPA* (pp. 1782-1787). IEEE.

Scarfone, K. A., & Mell, P. M. (2007). Sp 800-94. Guide to intrusion detection and prevention systems (idps).

Sidney Colbert, Edward; Sullivan, Daniel; Hutchinson, Steve; Renard, Kenneth; Smith. A Process-Oriented Intrusion Detection Method for Industrial Control Systems. 11th International Conference on Cyber Warfare and Security, page 497, 2016.

Skoko, V., Atlagic, B., & Isakov, N. (2014, November). Comparative realization of IEC 60870-5 industrial protocol standards. In *2014 22nd Telecommunications Forum Telfor (TELFOR)* (pp. 987-990). IEEE.

Stouffer, K., Falco, J., & Scarfone, K. (2011). Guide to industrial control systems (ICS) security. *NIST special publication*, *800*(82), 16-16.

Stouffer, K., Falco, J., Scarfone, K *Incident response activity November 2014*, Retrieved 20th March 2020 from (cert.gov) : https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Nov-Dec2015_S508C.pdf.

Tian, D. J., Butler, K. R., Choi, J. I., McDaniel, P., & Krishnaswamy, P. (2017). Securing Arp/Ndp from the ground up. *IEEE Transactions on Information Forensics and Security*, *12*(9), 2131-2143.

Tridgell, A. (1999). Efficient algorithms for sorting and synchronization.

Wanjugu, E. W., & NGUGI, K. (2015). Role of information and communication technology investment on project performance of large supermarkets in Kenya; a case of Nairobi county.

Wired, "A cyberattack has caused confirmed physical damage for the second time ever," 2015. Retrieved 12th February 2020 from (wired.com): https://www.wired.com/2015/ 01/german-steel-mill-hack-destruction/

Wired, "Everything we know about Ukraine's power plant hack," 2016. Retrieved 12th February 2020 from (wired.com): https://www.wired.com/2016/01/ everything-we-know-about-Ukraines-power-plant-hack/

Zetter, Kim. (2016). Inside the cunning, unprecedented hack of Ukraine's power grid. Wired Magazine. Retrieved 12th February 2020 from (wired.com) https://www.wired.com/2016/03/inside-cunningUnprecedented-hack-ukraines-power-grid/

Zhang, S., Hu, Y., & Bian, G. (2017, March). Research on string similarity algorithm based on Levenshtein Distance. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (pp. 2247-2251). IEEE.

Zimmerman, B. (2008). Object oriented analysis. *Journal of Computing Sciences in Colleges*, *23*(3).

APPENDICES

Appendix A: Python Script Code Snippet

```
3
4  import argparse
5  import sys
6  import subprocess
7  from subprocess import *
8  import json
9
10
11 all_data=[]
12
13 parser = argparse.ArgumentParser()
14 parser.add_argument('-cp', '--create_profile', dest='create_profile', action='append', help="provi
15 parser.add_argument('-o', '--output_file', dest='output_file', action='append', help="provides nam
16 args = parser.parse_args(args=None if sys.argv[1:] else ['--help'])
17
18
19 read_file_ext=['pcap','pcapng']
20 out_names_ext=['json','txt']
21 read_files = args.create_profile
22 out_names = args.output_file
23
24 if len(read_files)!=len(out_names):
25     print '[+] Number of Input Files Should be the same as Output Files'
26     print '[+] Exiting... :('
27     sys.exit(0)
28
29 for file in read_files:
30     if file.split('.')[-1] in read_file_ext:
31         pass
32     else:
33         print '>> Only .pcap or pcapng are allowed'
34         print '>> Exiting... :('
35         sys.exit(0)
36
```

Figure A.1: Create_profile.py script

```
59
60  def _104pcap_json(pcap_files,json_out_names):
61      for pcap_file,json_file in zip(pcap_files,json_out_names):
62          print '\nProcessing : %s '%(pcap_file)
63          print '\nOutput File : %s'%(json_file)
64          cmd=["tshark","-r",pcap_file,"-T","fields","-e","tcp.payload","104apci"]
65          proc=subprocess.Popen(cmd,stdin=PIPE,stdout=PIPE,stderr = PIPE)
66          data=[]
67          while proc.poll() is None:
68              pkt=proc.stdout.readline().strip('\r').strip('\n')
69              if len(pkt)>0:
70                  data.append(pkt)
71          #print data
72          if json_file.endswith('.txt'):
73              fp=open(json_file,'w')
74              #for data_pkt in json_list(data).split(','):
75              #fp.write(data_pkt+'\n')
76              for i in data:
77                  fp.write(i)
78              fp.close()
79          else:
80              fp=open(json_file,'w')
81              #for data_pkt in json_list(data).split(','):
82              #fp.write(data_pkt+'\n')
83              fp.write(json_list(data))
84              fp.close()
85          print 'TCP Payload Extracted\n'
86          print 'Processing File(s)Complete\n'
87      print 'File(s) Successfully Saved.\n'
88      return str(json_list(data))
89
90
91  _104pcap_json(read_files,out_names)
92
```

Figure A.2: Create_profile.py script

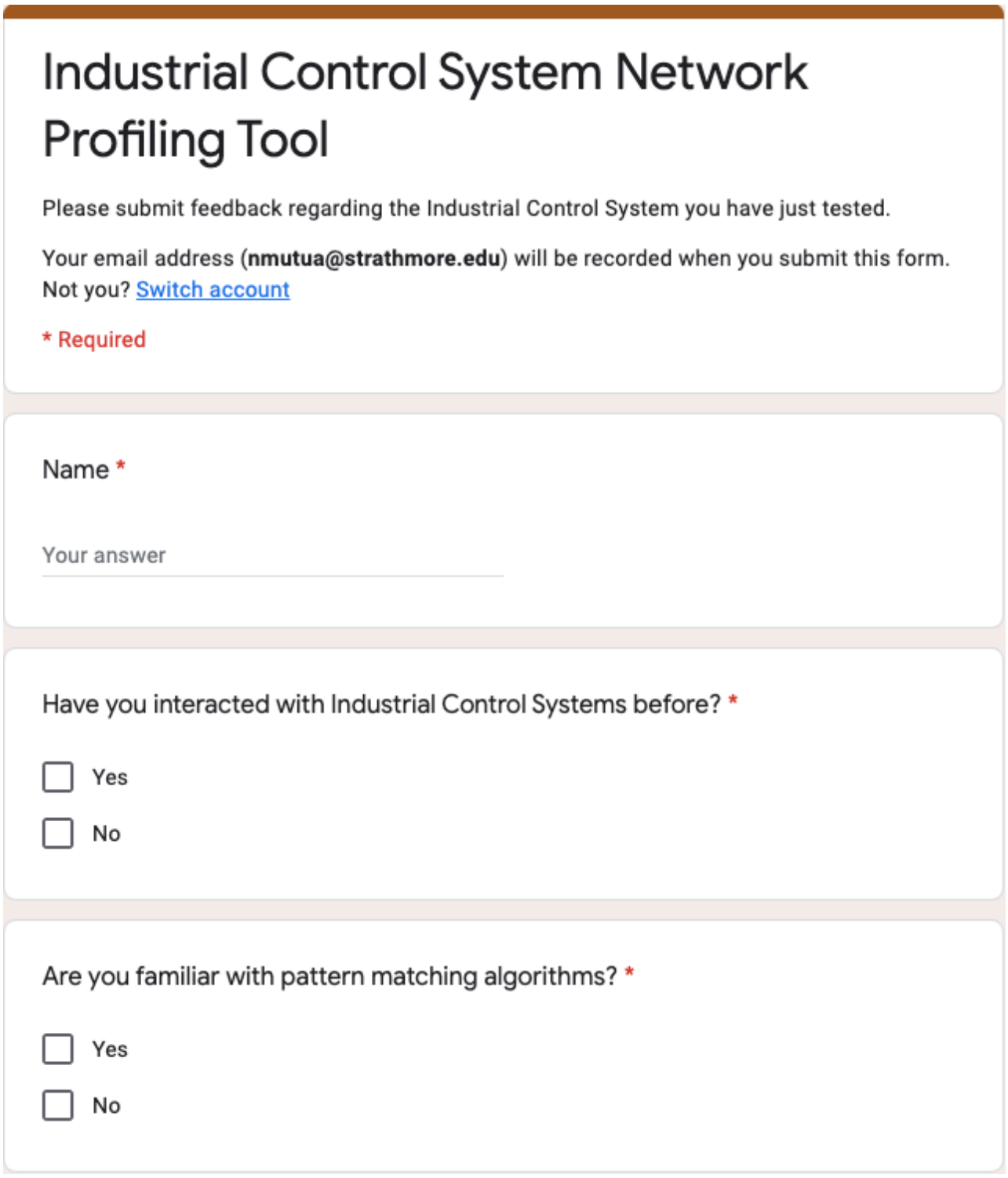
```
parser = argparse.ArgumentParser()
parser.add_argument('-ch', '--create_hash', dest='create_hasher', action='append', help="Provides n
parser.add_argument('-o', '--output_file', dest='outfile', action='append', help="Provides names to
args = parser.parse_args(args=None if sys.argv[1:] else ['--help'])

filestoshash=args.create_hasher
storehash=args.outfile


def ssdeep(ssdeep_file):
    cm1=['ssdeep',"-a",'-s','-b',ssdeep_file]
    proc=subprocess.Popen(cm1,stdin=PIPE,stdout=PIPE,stderr = PIPE)
    data=''
    while proc.poll() is None:
        data += proc.stdout.read()
    return data

if len(filestoshash)==len(storehash):
    for i,j in zip(filestoshash,storehash):
        outdata=ssdeep(filestoshash)
        fp=open(storehash,'w')
        fp.write(outdata)
        fp.close()


elif len(filestoshash)!=len(storehash):
    fp=open(storehash[0],'w')
    for i in filestoshash:
        outdata=ssdeep(i)
        fp.write(outdata)
    fp.close()


```

Figure A.3: Hash_profile.py Script

```

def ssdeep(ssdeep_file,hash_txt):
    cm2=['ssdeep','-a','-s','-b','-m',hash_txt,ssdeep_file ]
    proc2=subprocess.Popen(cm2,stdin=PIPE,stdout=PIPE,stderr = PIPE)
    data=''
    while proc2.poll() is None:
        data += proc2.stdout.read()
    return data


def _init_ssdeep(sdf,hf):
    #brute force results, ssdeep keeps returning >> ssdeep: No matching files loaded
    while 1:
        cont = ssdeep(sdf,hf)
        if cont!='ssdeep: No matching files loaded' or cont=="":
            if cont.strip('\n').strip('\r')!="":
                return cont
        else:
            print cont


print '\n\t\tSSdeep Successfully Executed\n\n'

if len(hash_files)==len(compare_files):
    for x,z in zip(hash_files,compare_files):
        print _init_ssdeep(z,x)

if len(hash_files)!=len(compare_files):
    for i in compare_files:
        print _init_ssdeep(i,hash_files[0])

print '\n\t\tProcessing Results'
print '\n\t\tPattern Matching Executed Successfully'
```

Figure A.4: Compare_profile.py Script

Appendix B: Usability Testing Questionnaire



Industrial Control System Network Profiling Tool

Please submit feedback regarding the Industrial Control System you have just tested.

Your email address (**nmutua@strathmore.edu**) will be recorded when you submit this form. Not you? Switch account

* Required

Name *

Your answer

Have you interacted with Industrial Control Systems before? *

- [ ] Yes
- [ ] No

Are you familiar with pattern matching algorithms? *

- [ ] Yes
- [ ] No

Figure B.1: Usability Questionnaire

If yes, name any pattern matching algorithm you have used before.

Your answer

How would you rate the whole application? Kindly tick where appropriate for each attribute. *

	Poor	Fair	Satisfactory	Very good	Excellent
Easy to use	○	○	○	○	○
Responsiveness	○	○	○	○	○
Efficiency	○	○	○	○	○
Speed.	○	○	○	○	○
Usefulness	○	○	○	○	○

Anything else you would like to comment about the usability of the application?

Your answer

Figure B.2: Usability Questionnaire

Appendix C: Turnitin Report

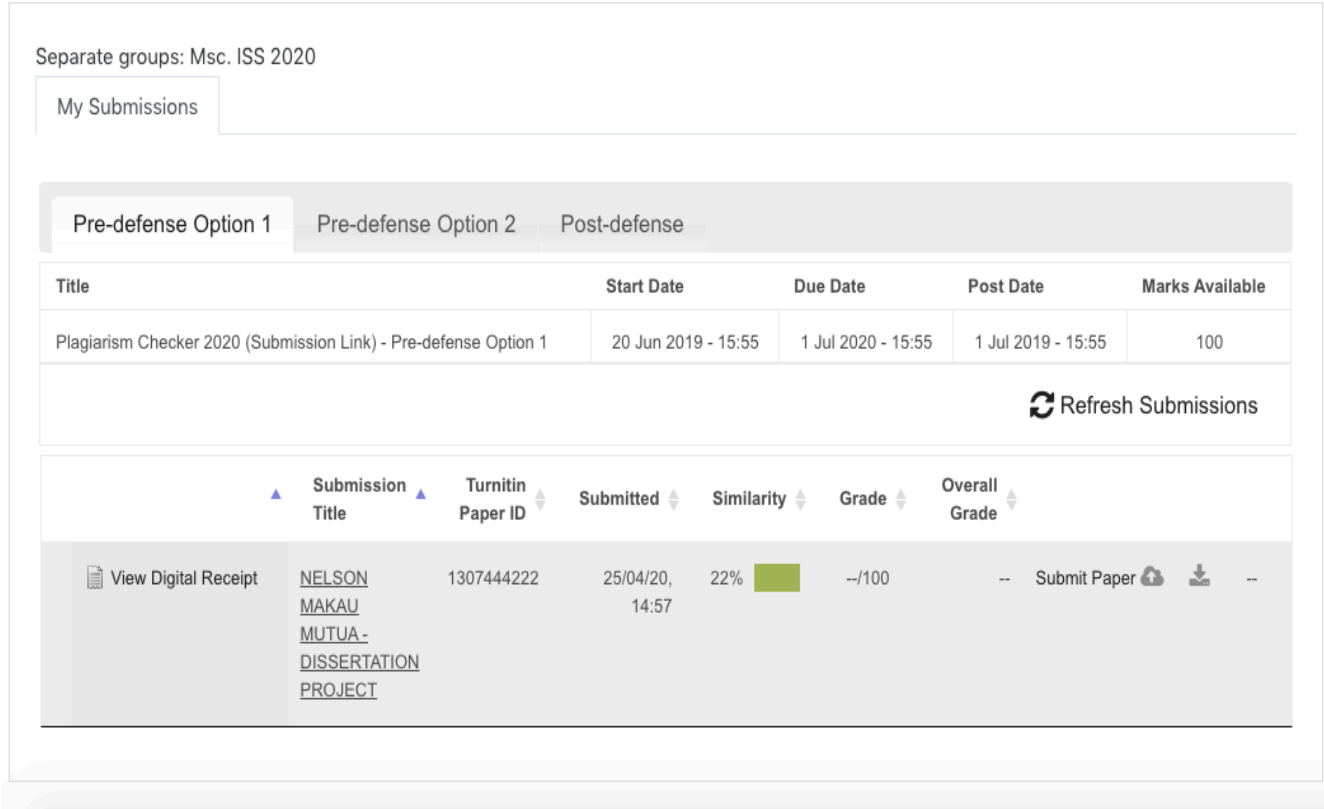


Figure C.1: Turnitin Report