A NOVEL COMPONENT BASED FRAMEWORK FOR COVERT DATA LEAKAGE DETECTION

Hanaa Nafea

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the Degree of Doctor of Philosophy

November 2019

Abstract

Cyber-attacks are causing billions of dollars of losses every year and data breaches are one of the major causes of these losses. The problem of data breach/leakage is attributed as a serious threat to organisations where any incident can inflict cost that is not only limited to monetary value but also can cause damage to organization goodwill, branding and reputation. Steganography is the practice of writing hidden messages via a medium in such a way that only the sender and the intended recipient know about the hidden message. Steganography is categorised into different forms including text, image, audio, video and network/protocol steganography. Network steganography is increasingly being used by malwares to facilitate the data leakage. This study focuses on aspects of network steganography at different levels of network packets.

The existing tools for data leakage prevention and detection are often bypassed by the use of sophisticated techniques such as network steganography for stealing the data. This is due to several weaknesses of the existing detection systems. First, these techniques have high time and memory training complexities as well as large training data sets. These are challenging issues as the amount of data generated every second becomes very large in many realms. Secondly, the number of their false positives is high, making them inaccurate. Finally, there is a lack of a framework catering for needs such as raising alerts as well as data monitoring and updating/adapting of a threshold value used for checking packets for covert data.

To overcome these weaknesses, this study proposes a novel framework that includes elements such as continuous data monitoring, threshold maintenance and alert notification. The study also proposes a model based on statistical measures to detect covert data leakages especially with regard to non-linear chaotic data. The main advantage of the proposed framework is its capability of providing more efficient results with tolerance/threshold values. Experiment outcomes indicate that the proposed framework performs better in comparison with state-of-the-art techniques in terms of accuracy and efficiency. Additionally, the proposed mathematical model can also be used for on-the-fly detection of covert data as opposed to offline processing methods.

Keywords: Data Leakage, Data Leakage Prevention, Steganography, Covert Channel, Phase Space Reconstruction, TCP/IP protocol.

Acknowledgement

First, great thanks to my Almighty God, Allah, the most merciful and the most generous, for his blessings that have given me the ability, strength and determination to complete a PhD at this prestigious institution and for everything else he has given me.

And I thank Allah, because without HIS support, blessing, love, and guidance, this work would not have been possible.

I am extremely grateful for the mentorship of Prof. Kashif Kifayat and Prof. Qi Shi my director of studies, my supervisors Prof. Abir Hussain and Dr. Bob Askwith. I am very fortunate that I have the opportunity to be guided by this immensely skilful and diverse team who have enabled me to think from various perspectives that I would not be capable of and for that I am highly appreciative and gracious.

I would also like to thank my husband Dr. Sultan Hijazi, who has supported me not just throughout my research but also throughout my life. Without him, I would never have reached this far. His endless support, encouragement, patience and confidence in me have never gone unappreciated, and it has played a huge part in the completion of this research.

Also, I feel a deep sense of gratitude to the spirit of my father (may Allah have mercy on him); and my mother. Also I extend sincere esteem, gratitude and a special thanks to my sons Faisal and Abdulmajeed and my daughter Elyana for encouragement, support and patience during the period of this research.

I want to dedicate this thesis to dearly beloved Husband and my sons and my daughter.

Table of Contents

Table of Figures

Table of Equations

Table of Tables

List of Symbols

Symbol	Description
ISN(i)	Initial Sequence Number Where *i = 1, 2, ... N – m +1*. *N* is the number of ISNs and *m* is the dimension.
x(n), y(n), z(n) and w(n)	Coordinates that is used to create a phase space dataset
R^m	Realm in m dimensions
r_i	The four dimensional phase vector
$d_{i,j}$	Euclidean distance between two vectors
d(k)	Matrix holding distance between vectors. Where $d_{k,k}$=0
μ_k	The mean value of row vector $d(k)$.
σ_k^2	Represents variance of Euclidean distances between the specified vector r_i and all other vectors in *R*
σ^2	Matrix holding variance of all distance in d(k)
var_σ	Variance of variance vectors in σ^2
M	Number of readings (vectors)
$Threshold_{ISN}$	Threshold for normal ISN
Lag-order	the number of lagged difference
p-value	probability value or significance

List of Acronyms

Symbol	**Description**
ICT	Information Communication Technology
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
RTP	Real-Time Transport Protocol
RTCP	Real-Time Transport Control Protocol
ISN	Initial Sequence Number
DNS	Domain Name Service
DMA	Direct Memory Access
SCTP	Stream Control Transmission Protocol
ACK	Acknowledgement
RST	Reset
URG	Urgent Pointer
SYN	Synchronize
FIN	Final

Chapter 1 Introduction

1.1 Overview

This chapter presents an introduction to cyber security and its practice to protect systems and networks. This includes cyber-attacks and data leakage challenges for the understanding of the attack processes and main causes. The chapter also discusses data leakage detection and its current issues, and presents the research gap and motivation for proposing better solutions. It then describes the research questions and objectives set out by this project as well as the methodologies proposed. This is followed by the summary of the main project contributions and the outline of the thesis structure.

1.2 Cyber Security: Scale of the Problem

The rapid expansion of Information and Communication Technologies (ICT) and its integration with other technologies have changed traditional networks into more smart and intelligent networks. These technologies provide efficient services to businesses at affordable cost through the Internet. However, the highly interconnected critical infrastructures and businesses have created many cyber security threats and challenges, which become main concerns for online businesses and their operations. Due to cyber-attacks, businesses have faced millions of dollars' loss every year and this loss is increasing day by day. There are around 24,000 malicious mobile applications blocked every day. In 2017, there is a 92% increase in new types of malwares [1]. According to an official annual cybercrime report in 2019, it is predicted that cybercrime will cost the world $6 trillion annually by 2021 [2]. These losses have been encountered due to data damage, theft, alteration, misuse of confidential personal and financial data, online fraud, and unavailability of businesses caused by the attacks, detection and restoration of hacked data and systems as well as reputational damage. In all data breaches, malicious attacks are on the highest position as shown in Figure 1.1 [3].

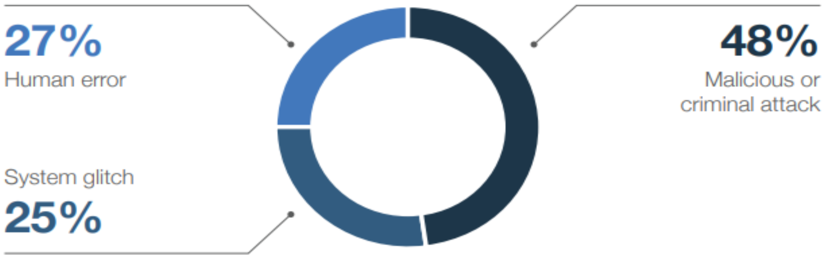
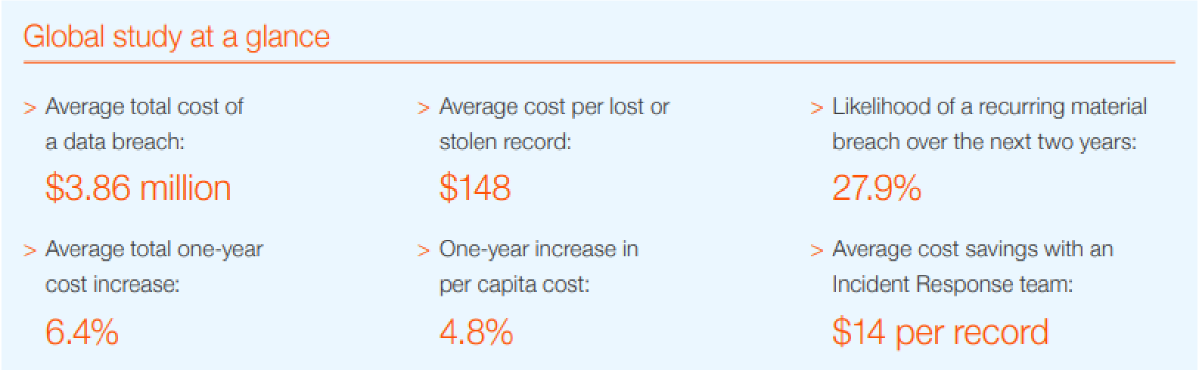


Figure 1.1: Distribution of the root cause of the data breaches (Source: [3])

According to IBM [3], the average individual cost due to data lost or stolen is $148 and it is higher in the USA and Canada, as shown in Figure 1.2. An estimated average cost of one million record breaches in 2018 is $39.49 million as illustrated in Figure 1.3.



Global study at a glance

> Average total cost of a data breach:
$3.86 million

> Average cost per lost or stolen record:
$148

> Likelihood of a recurring material breach over the next two years:
27.9%

> Average total one-year cost increase:
6.4%

> One-year increase in per capita cost:
4.8%

> Average cost savings with an Incident Response team:
$14 per record

Figure 1.2: Global Data Breach Statistics (Source: [3])

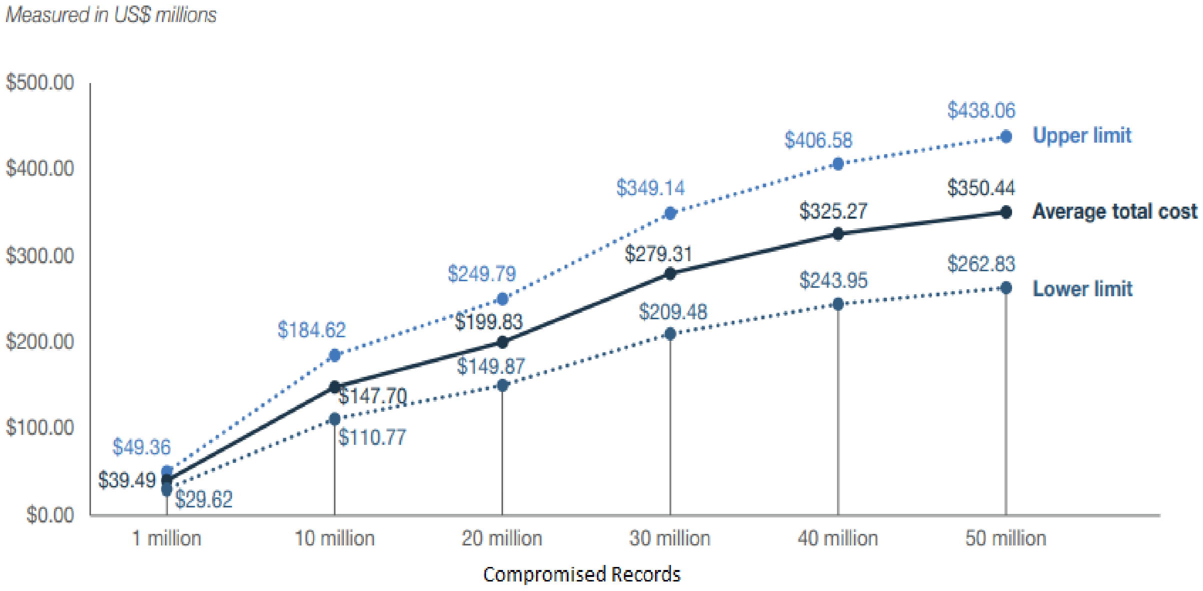


Figure 1.3: A total average of mega data breaches (Source: [3])

In February 2016, the (US) Inland Revenue Service (IRS) reported a massive data breach where more than 700,000 social security numbers and other sensitive information were stolen [4].

According to [4], the consolidated average per capita cost for all countries is $154 compared to $145 average from the previous year. The US and Germany as in previous years have a trend towards the highest per capita cost which is at $217 and $211, respectively, while India and Brazil have the lowest costs at $56 and $78, respectively (See Figure 1.4 for details). The statistics are skewed as western developed countries are having more data available in electronic format as opposed to developing countries.

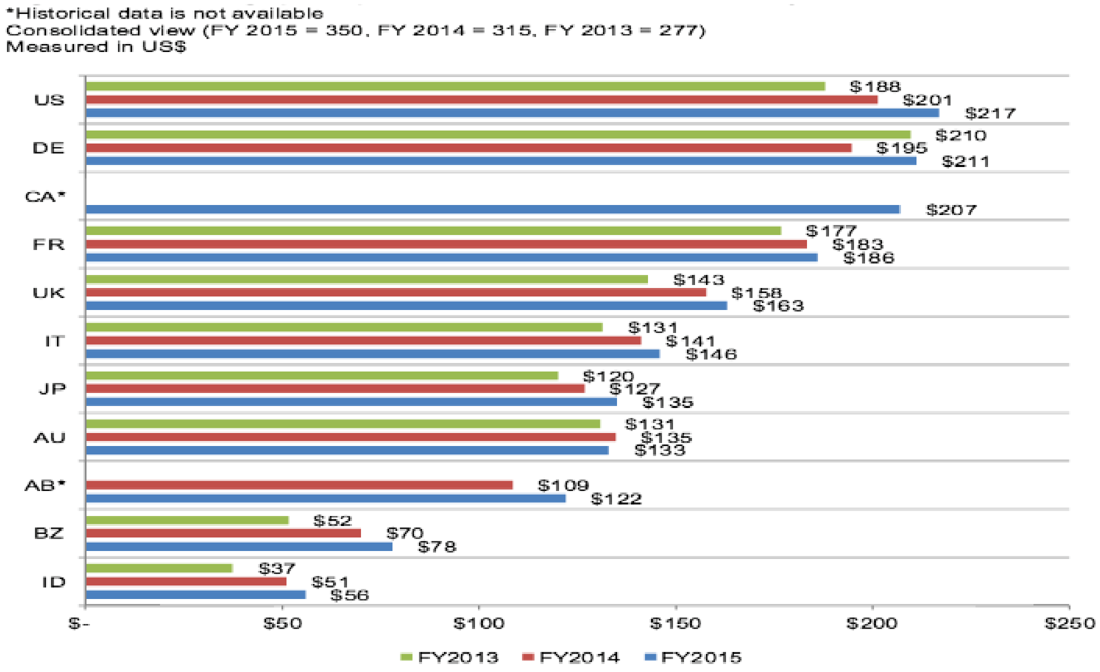


Figure 1.4: A total average of mega data breaches (Source: [4])

In Figure 1.4, though the amount of data available electronically may be smaller, the average number of records lost due to data breaches is high for Arabian, Indian and US regions.

1.2.1 Cyber Attacks

There is a range of cyber-attacks to compromise the confidentiality, integrity and availability of the systems. Confidentiality and integrity are important attributes that are associated with a shared resource. For example, confidentiality indicates who has access to a resource, whereas the integrity indicates who can alter a resource. In the context of shared resources, knowing where data can flow becomes a critical piece of information that can help to define the security parameters of shared resources. The top of the list is DDOS (Distributed Denial of Service), SQL injection, phishing and malware attacks that are explained further below:

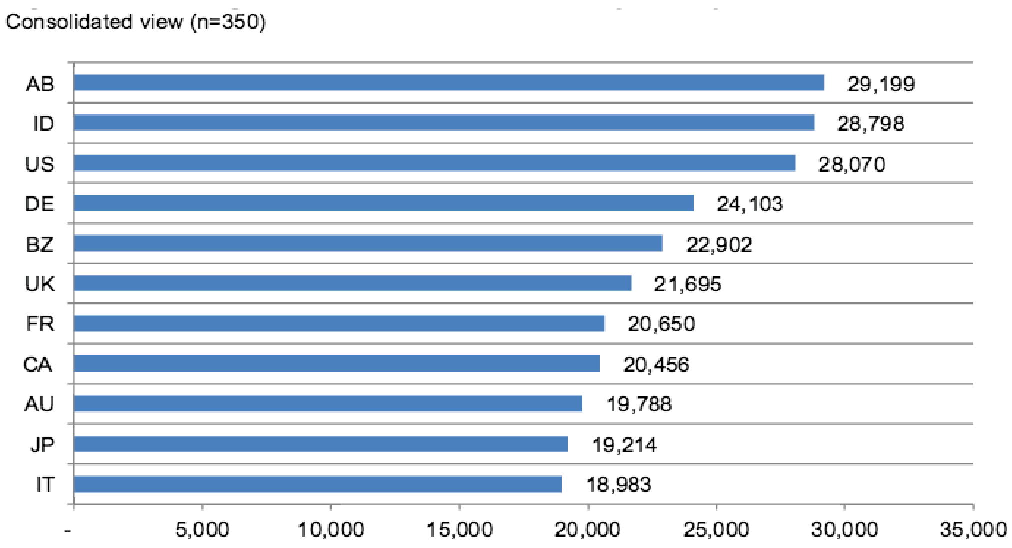


Figure 1.5: The average number of breached records by country in 2018 (Source: [4])

- Malware attacks are based on malicious programs developed to destroy specific computing systems or steal private information and money [4]. Malware attacks include Viruses, Spyware, Worms, and Trojans.
- Password attacks allow attackers to bypass the authentication procedures in secure systems. Password attacks can be divided into two types. Firstly, brute-force attack is where an attacker uses a specific tool executing on a target machine for all possible combinations of letters, numbers and symbols to get a correct password. Although this method may take too long to get the correct password, the systems with short passwords can be easily cracked by this method. Secondly, dictionary attack is where an attacker uses a malicious tool containing a dictionary of different common word combinations that users usually use, e.g. birthday dates or names, to obtain a correct password.

- In denial-of-service attacks, an attacker sends a high volume of data traffic to a targeted network system to interrupt or bring down its services.
- In SQL injection attacks, an attacker uses malicious code for database manipulation to access the information that may contain sensitive company data or user information. This type of attack causes more damage to business database systems.
- Advanced Persistent Threats (APT) are one of the most dangerous attacks where attackers use different techniques such as socially engineered Trojans, phishing emails and malicious websites to exploit the weaknesses and vulnerabilities of computer systems. The risk of these APT attacks is increasingly extreme when exploiting a new unknown vulnerability (zero-day vulnerability) where vendors and system developers have not got any information about this vulnerability and do not know how to patch it. Also an attacker could create a fake website that contains malicious links to exploit vulnerabilities on client devices [5]. Once a vulnerability on a client device is exploited, the attacker could steal credentials and gain unauthorised access to the device [6].
- Backdoors are one of the biggest security threats on organizations' networks as they can exist in networks without the network administrator's knowledge and be used to launch the other attacks mentioned above [7]. There are two common ways to create backdoors on computer systems: using shellcode and using covert channel communication. This study focuses on the backdoor attacks created using shellcode, which are referred to as Backdoor Shells [8].

Attacks such as SQL injection, phishing attacks, key loggers, man-in-the-middle, backdoors and steganography-based covert channels are used for data breaching.

1.3 Steganography for Data Breaches/Stealing

Steganography is an act of hiding the data in another message. It is also used to hide a message within a digital image [9, 10]. Every image has the least significant bits, these bits are mostly empty or altered, and no significant change in the image can be encountered by casual viewers. Steganography has been observed for data breaches for the past 15 years but now it is getting more complicated as hackers use this method to send malwares. When the attackers deploy steganography, they modify the simple images in order to hide the original data [11]. According to an FBI investigation held in 2017 [12], around 19k important data files had been stolen from a known US company. These kinds of threats have been marked as the major problem for US companies. Around $225 million to $600 million losses are faced by US companies every year due to such thefts.

Attackers never want their paths to be discovered. In order to hide their tracks, the attackers try to keep the communication masked by using covert channels. Steganography is one technique for covert communication. Information hiding techniques are known as a five-fold goal in which finding out the target, scanning for vulnerabilities, gaining unprivileged access, staying undetected and finally covering their tracks are performed one by one. Figure 1.6 shows the classification of information-hiding techniques. It shows how different attack phases are used by malware software [13]. The most recent malware found in the last few years makes use of steganography to steal valuable or private data.

These types of data hiding techniques make it difficult for intrusion detection mechanisms to identify data leakages. Furthermore, these techniques can be randomly used by attackers to make it harder to detect. Therefore, this area has great potential in terms of research and development. Table 1.1 [14] shows some examples of information hiding malware. Information hiding methods are also increasingly used in advanced persistent threats.

According to Lori Cameron [14], in the near future, cybercrimes will get more stealth and commoditization of malware, and focus on exploitation of the Internet of Things (IoT) related devices. It means that the cybercriminals are developing new techniques to make it harder to detect and trace back the malware to its origin. This is the main reason for them to use steganography for information hiding.

1.4 Covert Channels

Covert channels are gaining more popularity than encryption techniques among cybercriminals. It is a newly discovered technique that is used to perform the communication more secretively [15]. Stenographic techniques are being used by the covert method of communication to hide the information among original text or images as explained in section 1.2. In [16], the SCTP (Stream Control Transmission Protocol) stenographic method's idea is presented that can pose a threat to network security. It describes possible security attacks in SCTP with countermeasures. SCTP-specific stenographic methods can be divided into three groups: (1) methods that modify the content of SCTP packets, (2) methods that modify how SCTP packets are exchanged, and (3) methods that modify both the content and the way they are exchanged. Using INIT and INIT ACK chunks, Data chunks, SACK chunks, AUTH chunks, PAD chunks and Variable parameters can modify the SCTP content via a stenographic method. For modifying the SCTP packet exchange routine, Multi-Homing and Multi-Streaming techniques are used. As a result, this analysis emphasises how important it is to investigate further over the network protocols. The art of covert sending of information has also been applied to many other protocols such as TCP/IP, UDP, ICMP, HTTP and others. TCP is more reliable as it provides the delivery reliability of packets to the receiver while UDP is non-reliable and does not give information about the packets. A covert channel is a communication channel that is not designed and/or not intended to exist, and that can be used to transfer the information in a manner that violates the existing security policy.

The past researches have shown that the covert channels are divided into three major types on the basis of the following characteristics:

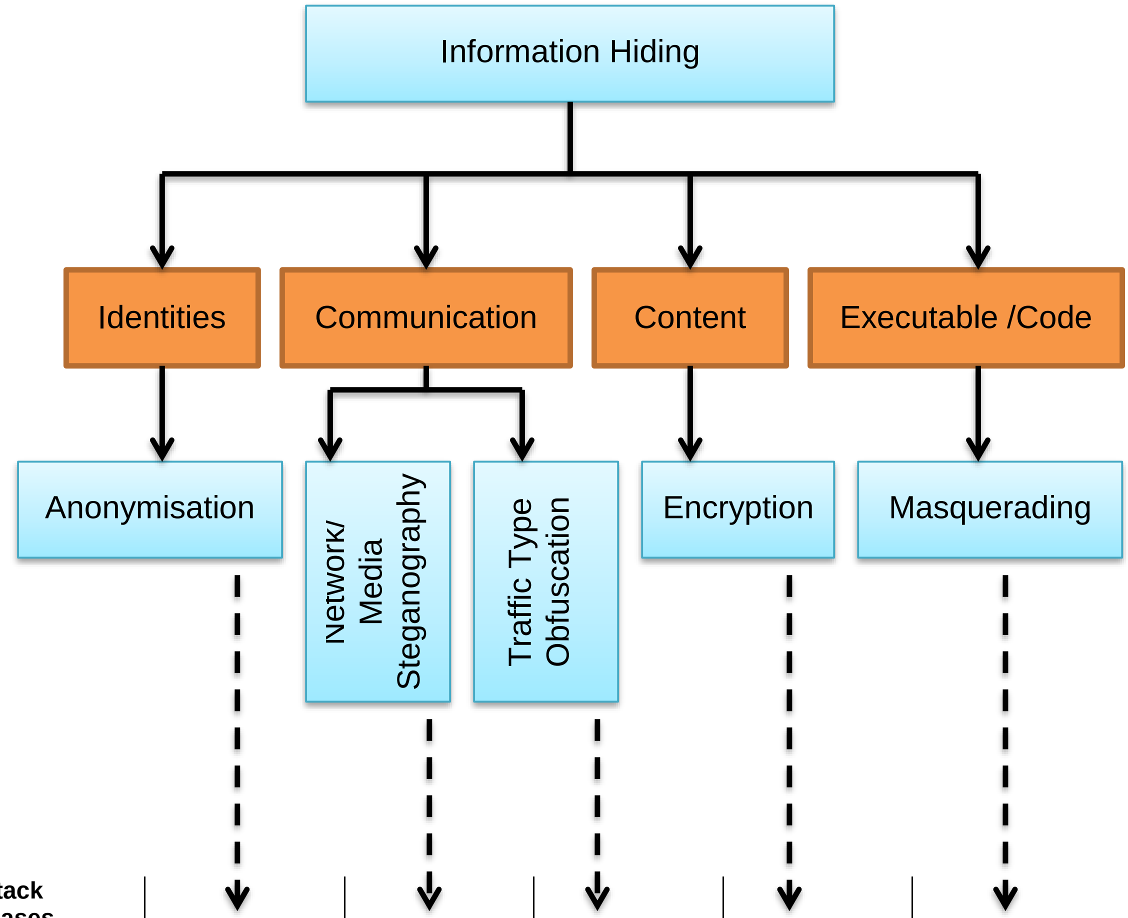
- Covert Timing Channels
- Covert Storage Channels
- Covert Network Channels

Covert Timing Channels

Covert timing channels take the advantage of system performance and components to send a secret message. They may also use the system resources and timing to share confidential data from one source to another. Attackers gained these advantages for detecting the covert channels [17].

Covert Storage Channels

Storage based covert channels use the method of sending intended hidden information by using the common area between processes in the main memory of the system. In this type of covert channel, some processes write directly or indirectly into certain shared resources while other processes read them [17].



Attack Phases					
Scanning	Annonymisation of names, IP spoofing, location spoofing		Hiding scanning		
Gaining Access		Hiding infection process	Hiding infection process	Hiding infection process	Masquerading malicious apps
Maintaining	Hiding C&C		Hiding data	Hiding C&C	Masquerading

Access	traffic		exfilteration or C&C traffic, tunnelling	traffic	processes & apps

Figure 1.6: Attack phases of information hiding techniques (Source: [18])

Covert Network Channels

The main aim of covert network channels is to hide secret data inside network data carriers/packets, i.e. normal network traffic of users. In an ideal situation, the hidden data exchange cannot be detected by third parties who are unaware of the covert channel's usage [19]. Recently attackers have used the network protocols to perform the covert channels-based communications. The main advantage of this technique is the stealthy nature of communication possibility over the network. The HTTP network protocol and TCP/IP are known types of covert network channels. HTTP based network channels have the following techniques to perform the covert communication over the internet [20]:

Malware/exploit kit	**Information- hiding method**	**Purpose**
Vawtrak/ Neverequest	Modification if the Least-significant bites (LSBs) of Favicons	Hiding URL to download a configuration file
Zbot	Appending data at the end of a JPG file	Hiding configuration data
Lurk/Stegoloader	Modification of the LSBs of BMP/PNG files	Hiding encrypted URL for downloading additional malware components
AdGholas	Data hiding in images text and HTML code	Hiding encrypted malicious Java Script code
Android/Twitter. A	Impersonating a pornography player or an MMS app	Tricking users into in stalling malicious apps and spreading infection
Fakem RAT	Mimicking MSN and Yahoo Messenger conversation traffic	Hiding command and control (C&C) traffic
Carbank/Anunak	Abusing Google cloud-based services	Hiding (C&C) traffic
Spy Note Trojan	Impersonating Netflix app	Tricking users into installation malicious app to gain access to confidential data
Tesla Crypt	Data hiding in HTML comments tag of the HTTP 404 error message page	Embedding C&C commands

Cerber	Image steganography	Embedding malicious executable
SyncCrypt	Image steganography	Embedding core components of ransomware
Stegano/Astrum	Modifying the colour space of the used PNG image	Hiding malicious code within banner ads
DNSChanger	Modification of the LSBs of PNG file	Hiding malware AES encryption key
Sundown	Hiding data in white PNG files	Exfiltrating user data and hiding exploit code delivered to victims

Table 1.1: Data hiding malwares (Source: [14])

- Redirects
- Cookies
- Referrer
- HTML tag elements
- Active Content
- Arbitrary headers
- HTTP timing channel
- Webpage browsing pattern
- Reordering of header files

While in the TCP/IP type of covert network channel, TCP/IP based modification is performed for a stenographic covert channel over network protocol header values to perform secret communication [16, 17, 21, 22].

There is no doubt that timing channels are the most common covert channels. Because of this, a large amount of research has been carried out to propose solutions to their detection and prevention, so it is a relatively mature topic from the research point of view. However, covert network channels based on steganography, which do not rely on timing to pass secret information, are increasingly attracting cybercriminals to exploit their potentials, as there is little research on effective and efficient methods for their detection. This is the main reason for this study to focus on network steganography based covert channels, instead of covert timing channels, to bridge the existing gap in the research area.

1.5 Intrusion Detection System (IDS)

Intrusion is a set of actions that make an attempt to challenge the integrity, discretion or accessibility of a resource [23]. Bace and Mell [24] provide a more comprehensive definition of intrusion detection “as a collection of practices and mechanism used to detect the errors that may lead to security failures with use of anomaly detection, misuse detection and diagnosing intrusions and attacks”. The objective of an IDS is to alert administrators of suspicious activities and also in some cases to avert any attacks.

Historically, network security administrators conducted Intrusion Detection (ID) manually to monitor the network traffic for anomalies. This early form of ID is ineffective in the implementation of an automated audit log file that allowed quick searching for irregularities [23]. The introduction of these logs is helped with the manifestation of ID into forensic techniques where the administrator is responsible for collecting the information and identifying issues only after incidents have occurred rather than when the process of intrusion is taking place. Researchers in [25] proposed an IDS that reviewed the audit data and produced the first version of a real-time IDS to block the cyber-attack in real-time.

Now the focus has shifted towards the enhancing of security measures as newer attack techniques continue to spawn data on the network. Due to this ever-changing situation, the researchers and developers realised that defending against changing threats must include seeking to improve the methods for threat identification.

The use of an IDS for detecting the data leakage is the right approach, i.e. the IDS should have the ability to detect the covert channels where important data can be leaked. Different methods such as misuse detection [26] can be used to detect such malicious activities. The main shortcoming of these methods is that industrial standard tools are still using the payload field as a target for detecting the data leakage while trends are indicating data leakage through covert channels.

1.6 Research Gaps and Motivation

Data leakage is one of the major issues in Cyber Security and leads to companies' financial and reputational losses. In most cases, data leakage happens when the confidential data is stolen in bulk or entire databases are hacked. For this purpose, hackers have adopted different security bypassing techniques for stealing confidential files, and covert channels are one of them [27], which are very hard to detect as the stolen data is blended in a form of small binary values in some components of network packets such as ISN (Initial Sequence Number) and Flags. An ISN refers to a unique 32-bit sequence number assigned to each new connection on a Transmission Control Protocol (TCP)-based data communication, which is commonly used for steganography-based covert channels. This is called network steganography. It is one of the areas, which almost all existing tools have failed to address as it is very dynamic and random in nature. Furthermore, now hackers use the same methods to deliver infected payloads to victim machines [28, 29].

Based on existing literature [30-32], covert channel detection techniques are categorised into pattern, machine learning and statistics-based approaches. The pattern-based approaches have a common denominator of matching a captured packet header against a known pattern in order to identify if the packet has covert data in the header [33]. These techniques provide the high accuracy and speed of detection but fail to cope with new threat vectors and incomplete data sets and incur high maintenance requirements. The machine learning-based approaches [30-32, 34] have used a set of rules created from the complex data sets for classification. These methods overcome some shortcomings of the pattern-based approaches, improve the ability to cope with incomplete data sets, and offer further insights into latent information within the data sets. However, the benefits are offset by the shortcomings of large system footprints, low speed of detection and high maintenance requirements. Finally, the statistical approaches [35] measure deviations from expected stochastic behaviour, which are effectively capable of handling incomplete data sets. Additionally, these approaches offer the speed of detection and ability to detect new threat vectors. The main shortcoming of these approaches is the possibility of high false positive results.

It has been identified that the common problem with existing approaches is the inability to detect new attack vectors, particularly when the attack vectors are masked, i.e. the data leakage commences via the field in the packet header where the data can be non-linear [36-38]. A non-linear type of covert data means that the change of output is not proportional to the change of input. Unlike linear covert data with rather simple dynamics, non-linear covert data has very complex dynamical behaviour, so it is not easy to reverse-engineer from the covert data to its original data. When non-linear data is sensitive to initial conditions, it becomes non-linear chaotic data. Sensitivity to the initial conditions means that each point in a chaotic system is arbitrarily and closely approximated by other points. A classic example is double pendulums where if the starting point of releasing a pendulum is slightly different, then the trajectory taken by them is very different. Effectively, this means that machine learning or pattern finding-based approaches may not be suitable as they work on the realms of fixed pattern locations. Thus the pattern and machine learning-based approaches fail immediately in such cases, as there is no fixed pattern and the classification of non-linear data is mathematically not viable [37, 39].

Chaos theory focuses on the study of different states of dynamic systems which are highly random and irregular in nature. Chaotic approaches have been widely used to investigate and describe highly complex and nonlinear systems which have vastly different outcomes [40-42]. It has also been acknowledged that the concept of chaos is radically different from statistical randomness [43]. Chaos theory is used to check how a system evolves over time in a nonlinear, deterministic, and dynamic manner [44]. On the other hand, statistical approaches have some degree of effectiveness. However, the approaches similar to machine learning-based ones have failed due to the chaotic nature of non-linear data being leaked via packet headers [45]. In addition, it is difficult for existing techniques to differentiate between normal data and the data masked to appear as normal data. Additionally, it is also observed that various techniques suggested in the literature have failed to cope with the diversity of fields in the TCP header and new unknown threats [39].

After the above investigation, it is clear that there is a need to develop a more efficient technique to detect the data leakage in the network data streams. This research gap is the main motivation to propose the novel technique in this thesis to handle data leakage in the network data streams. More specifically, there is a need to design a novel component-based monitoring framework to detect data leakage, analyse the TCP traffic for covert data and quantify the data leakage without the use of data signature patterns. In addition, it is necessary to establish thresholds that can be used to identify outliers while performing data leakage detection. This research has taken all these requirements into account for the framework design.

1.7 Research Aim and Objectives

The main aim of this research is to propose a novel component-based framework to detect data leakage at the transport layer. This study proposes a framework based on statistical measures to detect covert data leakages more efficiently with adaptable threshold values. It focuses on TCP because TCP is widely used to provide reliable Internet communications and becomes a highly preferable target for attackers to exploit for cover communications but the current research has limitations in tackling such threats. That is why this study is motivated to rectify the problems. The significance of the proposed framework is about offering continuous data monitoring, threshold maintenance, alert notification and capability of providing more efficient results with tolerance/threshold values than the existing work.

In order to achieve the above aim, the project objectives and adopted methodology are given as follows:

Objectives	**Methodology**
1. To do a detailed background study of different types of data leakage attacks, use of steganography for data leakage, covert channels, available tools and technologies, and different protocol parameters that can be used for covert communication.	This objective is successfully completed by studying a varied range of existing books, magazines, research papers and PhD theses. The outcome of this objective is

	presented in Chapter 2.
2. To perform detailed literature research in different covert channel detection methods and identify research gaps. The scope of covert channel detection on protocols such as TCP and UDP is very large, which is defined in the header fields and various combinations of header fields used for the purpose of covert communication. Existing research into the detection of covert channels on the header fields of those protocols is assimilated in order to understand the weaknesses of existing techniques.	A detailed study of different research papers has been conducted in the area of data leakage detection, which is further expanded toward covert channel detection in different protocols such as TCP, IP and others. This objective has been successfully completed, and the details are presented in Chapter 3.
3. To design a novel component-based monitoring framework for the detection of data leakage at the transport layer of network traffic, which should have the following abilities: a. To analyse TCP traffic for covert data and quantify data leakage without the use of data signature patterns. b. To establish the thresholds used to identify outliers while performing data leakage detection.	A complete framework has been produced. It has different components, offering different processes to perform different operations locally and interact with other components. Each component is described in detail in Chapter 4.
4. To implement and evaluate the proposed solution based on a selected data set and live experiments.	All components of the proposed framework to detect covert channels have been implemented. Furthermore, its

This includes developing an experimental environment to test different data leakage scenarios through covert communication, and creating a simulation testbed where a socket-based client-server system is used to send covert data in the header field (specifically Initial Sequence Number) of the TCP protocol. The environment is also extended using virtual machines where a client and a server are over a virtual network. Every received packet is passed through the covert channel detection algorithm proposed in this thesis in order to indicate if the packet contains covert data with an associated probability.	evaluation and testing have been completed using a selected data set and live network traffic. As part of the implementation, a client server testbed has been developed, which exchanges covert communication. Overall, the objective has been successfully fulfilled, and the details are presented in Chapter 5.
5. Write up the thesis.	The thesis is completed.

1.8 Research Scope and Novelty

The scope of this study is limited to detecting data leakage through the method of covert channels in TCP packet ISN and other values. There are many other methods used for data leakage such as encryption and steganography. This study focuses on aspects of network steganography at different levels of network packets. The existing techniques are not able to detect new attack vectors, particularly when the attack vectors are masked. The proposed framework provides multiple aspects for detecting the non-linear chaotic covert data leakages.

The main contributions of this research in the detection of covert data in the context of TCP packet headers are as follows:

1. The first contribution of this research is the design of a data monitoring framework that can detect the real-time concealed data in a multitude of fields on a TCP header, whilst having a very small system footprint. It features a

flowchart that evaluates covert channels for buffered data collection. The existing frameworks in literature can neither handle network based covert data monitoring nor detect the non-linear covert data in header fields. The existing studies do not provide a satisfactory level of accuracy, especially with regard to the detection of the concealed data within chaotic background noise such as in the case of initial sequence numbers.

2. A statistical threshold calculation technique that adopts a hybrid (based on regressive auto-correlation and classical statistics) approach to the calculation, which addresses the existing studies' weaknesses. The resultant calculated threshold value is based on expected values across multiple operating systems to achieve the longevity with regard to the threshold validity and hence ensure that various flavours of operating systems are not susceptible to leakage. The literature has shown a number of successful techniques that work in a dichotomous manner in order to indicate if the packet/data being scanned is covert or not. However, these techniques suffer from high false positive rates as well as low performance, which have been eliminated by the proposed statistical threshold technique where the result of packets containing covert data is presented with a degree of threshold probability.

3. A statistical technique quantifies the level of covert communication by computing the deviation of observed data from the expected threshold value. The technique is a two-stage analysis process where a deviation score is calculated in real-time over a buffer of collected TCP headers as well as over a larger collection of TCP headers. This offers insights into temporal data leakage detection for larger data collections informing the analyst of time-based indicative analysis as well as on-the-fly outlier analysis which are improvements on the existing detection techniques that analyse the packets based on a strict set of rules and miss deviation computation over buffered and collection data.

4. Furthermore, the proposed framework is novel because not all deviation scores produced are tagged as data leakage incidents but instead the proposed solution focuses on further analysis using a novel approach to the quantification of skewed results, which informs about the probability of data leakage as detailed in Section 4.8 of Chapter 4. This process uses moving average for decision making about data leakage, thereby increasing the level of accuracy as well as reducing the likelihood of Type I and Type II errors. Type I and Type II are two types of errors used for statistical testing to check whether test results are true or false. A Type I error refers to a false positive finding whereas a Type II error is a false negative finding. The moving average offers the smoothing of erratic data and provides the ability to observe trends that facilitate in reducing the likelihood of the aforementioned errors. This forms a basis for the continuous improvement of a threshold value rather than a strict value, which strives to reduce Type I and Type II errors, thus forming a closed loop. Most of the existing techniques do not offer this feature and are hence categorised as open loops.

1.9 Research Findings

This research has recognised a security weakness with regard to data leakage that occurs over non-conventional channels also known as covert channels. This weakness is primarily caused by the presumption that the existing data leakage monitoring techniques and tools should be applied to TCP headers, which is essentially a limitation of the existing work as it allows such covert communication to go undetected and unnoticed.

This study has also identified other limitations of existing techniques in order to discuss their inherent problems. It has been observed that existing techniques do not offer sufficient evaluation in the context of cases where data in the header field is complex. The application of these techniques to covert data in fields where background noise is chaotic produces flawed and incorrect results. Besides, this research also highlighted the reliability and accuracy in a dynamic real-time TCP-based communication.

This research has devised a novel component-based monitoring framework that can accurately detect the covert data leakage over TCP headers despite chaotic background data. However, it has been found that some of the techniques proposed in this thesis provide deviated results in certain known cases, so further research is required to apply statistical techniques such as multi-modality for further improvements.

1.10 Thesis Organization

The remainder of the thesis is arranged into five chapters as follows:

Chapter Two: Data Stealing

This chapter starts with an overview of data leakage and its terminologies as well as the description of different types of attacks that cause data leakage/stealing/breach. The detailed background of steganography and covert channel techniques in computer network protocols are also discussed. The covert channel details include how malicious users use different types of networks packets for covert communication.

Chapter Three: Related Works

This chapter discusses the related work and existing studies in the domain to analyse their advantages and limitations to find the research gap stated earlier. The existing studies have been evaluated in terms of their measures and metrics required for the detection and prevention of data leakage.

Chapter Four: Covert Data Monitoring Framework and Statistical Detection Model

The chapter presents the design of the proposed covert data-monitoring framework for TCP header fields. The sections of this chapter present the novel algorithm for the detection of concealed data within non-linear chaotic data, threshold value calculation, deviation quantification from the threshold, and outlier scoring in real-time for buffered data collection and larger data sets. This chapter also presents an autocorrelation method for computing dimensions and forming a vector for random data. Moreover, it discusses the software solutions detailing how the framework and statistical detection techniques are implemented and how the software is used to evaluate the proposed techniques.

Chapter Five: Implementation and Evaluation

This chapter details the implementation and evaluation of the work proposed in the previous chapter to check the success of the covert data monitoring framework. This allows the validation of the project claims, aims and objectives as well as requirements discussed in the previous chapters.

Chapter Six: Conclusion and Future Works

This chapter enumerates the conclusions of the research work presented in this thesis, including the summaries and characteristics of the proposed systems and contributions. Future research directions are also discussed in this chapter for further improvements on the proposed work.

Chapter 2 Data Stealing

2.1 Overview

This chapter presents the details of data leakage and the possible types of attacks. After a detailed discussion on data leakage attacks to understand the main causes behind these attacks, the steganography background, techniques and types are discussed to understand their technical processes. This includes TCP/IP based steganography and its characteristics. Covert channel details and data leakage through a covert channel in computer network protocols are also discussed. In the last section, data leakage prevention tools and technologies are discussed to understand the domain and possibilities to design more optimal solutions.

2.2 Data Leakage

Data leakage is one of the concepts where any confidential or private data is disclosed intentionally or unintentionally without the permission of the data owner [46]. Other terms of data leakage include data spill, information disclosure and information leakage [47].

Data leakage is one of the emerging security threats to organisations' internal or external sources. According to authors in [35], attacks can be classified based on the source of attacks. Attacks initiated by an entity inside the security perimeter (an "insider") are known as insider attacks. In these attacks, the attacker accesses the system resources without their authority's approval. Attacks initiated by an entity outside the security perimeter (an "outsider") are known as outside attacks. On the Internet, potential outside attackers exist in different forms such as amateur pranksters, organized criminals, international terrorists, and hostile governments.

According to authors in [46], there are three physical players (organisation, attackers and organisational assets) and four logical players (security mechanisms, vulnerabilities, threats and security risks) in the security domain. The organisation assets are mainly based on data and information that are store in databases. The organisation tries to minimise the security risk by applying appropriate security mechanisms such as firewalls, intrusion detection and prevention tools. For example, the Symantec solution [48] is one of the data leakage tools deployed in Amazon Web Services or as Single Server support, and Kaspersky's data leakage prevention plugin [49] is adopted by Microsoft Exchange.

Security patches are applied to patch the vulnerabilities exposed by software from a certain vendor. However, the integration of multiple vendors' patches further opens vulnerable areas (channel), which can be used for data leakage. For example, Microsoft windows regularly provide Windows Updates to close identified security vulnerabilities. However, Samba is a very popular file and print service that uses the SMB/CIFS protocol. It has 114 published vulnerabilities [50] that cannot be patched by Microsoft and could be used for data leakage. The DROWN vulnerability was announced on the 11th March 2016 and patched by Microsoft on the 12th April 2016 [50], which was previously unknown due to missed integration tests.

2.3 Data Stealing Attacks

Google, University of California Berkeley, and the International Computer Science Institute conducted detailed studies to assess the risks of credential threats [51]. The studies conducted in March 2016 and March 2017 identified 788,000 potential victims of off-the-shelf key loggers and 12.4 million potential victims of phishing attacks, resulting in 1.9 billion usernames and passwords exposed via data breaches and merchandised on the black market. The studies also found that there were 4,069 different phishing attack tool kits and 52 different key loggers responsible for the data breaches. Around 3,000 black hat hackers used phishing kits which emulated Gmail, Yahoo, and Hotmail logins to steal 1.4 million credentials. The HawkEye was the most popular tool used by 470 hackers to generate 409,000 reports from injected devices [52].

There are a number of different attacks that cause data breaches or leakages, such as SQL injection, phishing, backdoors and keyloggers. Some of them are further described in the subsections below.

2.3.1 SQL Injection Attacks

SQL Injection Attacks (SQLIAs) are considered as a threat to vulnerable database systems. In this type of attack, the attacker manipulates the input box by using different SQL statements in an application to gain unauthorised access to databases and change the data.

Applications without validated input become prone to such attacks easily. Different classical types of SQL injection cause different types of damage, e.g. tautologies-based, piggy-backed query, logically incorrect, union query, stored procedure, and inference-based attacks. More advanced types include fast flux [53] and compounded SQL injection attacks, where the compounded attacks include SQL injection + DDoS attacks, SQL injection + DNS hijacking, and SQL injection + insufficient authentication [54]. Research in different directions has been done to improve the SQLIA detection, e.g. using machine learning and role-based access control [55]. In [56], anomalistic test queries are classified using a trained support vector machine for data sets with illegitimate queries (e.g. those including quotation marks) to improve detection accuracy. Furthermore, in [57], a new approach is introduced where the instruction set is randomized due to instances of the language which are created and cannot be predicted by an attacker. Hence, when the attacker injects malicious queries, the database parser terminates untranslated queries.

2.3.2 Data Stealing Malwares

Malware is generally a piece of malicious software designed to pose different types of threats. Nowadays, these are widely used to steal data. Data-stealing malware is often used by attackers for financial benefits and get valuable information from the Internet. It is usually the second or third component of a sequential multi-pronged attack. According to trends in micro reports, data-stealing malware is also used for terrorism to launch attacks against government targets [58].

Researchers discussed in [59] that Bitdefender has shown a new multi-staged malware attack termed as “Operation PZChao” that is spreading in Asia and the USA. The initial delivery of the malware is carried via a phishing email. This email has two parts including a VBS file attachment and a 7zip file that is self-extracting. Both parts are defined based on their functionality and details in the provided link in the phishing email.

Some more examples on data stealing malwares are as follows:

Bot site scripting: In this type of attack, a bot duplicates a website and scrapes usernames from the website with poorly designed logins. The attacker develops a large number of bots from different IP addresses to avoid the application firewall which detects multiple failed attempts [60].

Credential stuffing: This is one of the biggest security problems launched in 2017 where 2,227 breaches were reported. Such attacks are automated and launched using special malware such as Sentry MBA that repeats attempts to take over corporate or personal accounts [60].

Beta bot: This is a Trojan that maligns computers to try to block user access to security websites and also disables their antivirus and malware scan software. This malware is known as Neurevt. It is a sophisticated information stealer that began as a banking Trojan, but now it includes features that allow its operators to practically take over a victim’s machine, steal sensitive information and shut down more than 30 popular anti-malware products. Beta bot’s main features are a browser form grabber, an FTP and mail client stealer, a robust rootkit, the ability to download additional malware, and the ability to execute commands [61].

2.3.3 Phishing Attacks

A phishing attack is a way of tricking users to provide private information without their knowledge. Commonly, such attacks use electronic messaging like email or web links to provide a fake link to the original site for malicious purposes. Phishing is a hybrid attack combining social engineering and technological aspects [62].

Phishing is more effective when it is launched by someone who is close to or at least known to a targeted user. Such phishing is known as social phishing. It is reported in [63] that 72 percent of users responded to a forged phishing email that looked like it was from friends.

2.3.4 DNS Poisoning

Domain Name System (DNS) cache poisoning is an attack type that usually exploits the loopholes in DNS to redirect the Internet traffic from an intended system. The intended system is a system controlled by the attacker or a system that the attacker wants the traffic to go to. The fact that this poisoning can carry and grow from one DNS server to another makes it more dangerous [64]. It is also known as DNS spoofing since some kind of false DNS data is fed into the DNS resolver’s cache which ultimately makes the server return the results that are not actually asked. This includes false IP addresses and website addresses.

2.3.5 DMA Attacks

Direct Memory Access (DMA) is a process that enables an I/O device to send or receive the data from the main memory to enhance the operation performance by bypassing the CPU. It is normally controlled by a DMA controller which is installed as a chip [65]. DMA attacks come under the category of side-channel attacks in which an attacker intrudes into a machine or any other device by exploiting DMA. The attacker uses vulnerabilities present in high-speed expansion ports.

I/O attacks are launched by an attacker through read or write access to DRAM through DMA. Morgan, et al. [66] discussed a design weakness in the configuration of the I/O Memory Management Unit (IOMMU) that can be exploited to facilitate such attacks. Stewin and Bystrov [67] discussed DMA malware and its effectiveness and implemented DAGGER to show how stealthy DMS malware works.

2.3.6 Backdoors

A backdoor in a computer system environment is a term used as a point of entry that is not documented or intended to use under normal circumstances. It is a portal through which an attacker gains access to a system’s resources, thus evading security mechanisms through the doors that are not normally monitored. A backdoor is used by an administrator for maintenance or for other purposes by attackers or intelligence agencies [68].

Clements and Lao [69] discussed backdoor attacks in neural network operations. Neural networks are a growing field of research where the attackers find ways to attack and gain access to resources and cause misclassification. Detecting a backdoor is a difficult task as it establishes secret channels for communication. Thomas and Francillon [70] presented a framework that is able to detect eleven diverse types of backdoors.

2.3.7 Key Loggers

A key logger is a spying program that is used to record every keystroke made by a user on a system’s keyboard. The recorded data can be used for gaining un-authorised or fraudulent access to private and secret information, e.g. passwords, PINs, etc. Key loggers are normally integrated into viruses and malware too [71].

New technologies have opened new ways of implementing key loggers too. Hussain, et al. [72] showed that built-in motion sensors in smartphones can serve as a key logger to infer user taps on smartphone touchscreens and use the recorded data for malicious purposes. Furthermore, Solairaj, et al. [73] described certain common techniques to detect key logger software, e.g. Anti-Hook techniques, HoneyID, Spyware detection, bot detection, safe access to password-protected accounts and the dendritic cell algorithm. The US Department of Homeland Security conducted research to investigate the crimeware and discussed that crimeware is an illegal action [74]. This research concluded that crimeware is a ubiquitous fact and one of the online interactions. After installation of crimeware, it benefits attackers in terms of denial-of-service extortion, click fraud, spamming and theft of confidential data.

2.4 Steganography

Steganography is the art of writing a hidden message through a covert medium where only the sender and the intended recipient know about the hidden message. Unlike cryptography, steganography messages are hidden and undetectable through human eyes. Essentially, it is not only the art of hiding data but also hiding related facts. However, steganography has been widely used by hackers to leak the data or steal information [75]. This strategy makes it very difficult to detect data leakage. The following sub-sections discuss steganography in more detail. Steganography refers to a technique for hiding secret data in other contents, which plays an important role in modern cyber-attacks. In particular, it is used as an underlying technique by malware to conceal their contents and/or command & control communication (C2) channels.

2.4.1 Types of Steganography

Depending on the nature of a covert object, steganography can be categorised into five types:

1. **Text Steganography:** This type of steganography involves modifying a text layout based on rules such as using the nth character or altering white space between words or lines [76]. Several methods are available for this type of steganography that include linguistic, random and statistical methods.
2. **Image Steganography:** This type of steganography hides information behind an image, i.e. message insertion may be encoded and every bit of the information in the image is selectively embedded in the noisy areas to avert attention. The message may also be randomly distributed in the image. Authors in one paper [77] have presented image steganography to prevent data leakage in certain scenarios.
3. **Audio Steganography:** The data in this type of steganography is embedded by slightly altering the binary sequence of a sound file. This type of steganography is more difficult than other media methods. It ranges from simple to complex algorithms that insert information in the form of signal noise. There are different

categories of audio steganography used to hide the information into auditory data such as least significant bit coding, echo coding, phase coding and spread spectrum coding. These methods are different in implementation, covertness and bandwidth. This is a more powerful method that uses principles of digital signal processing.

4. **Video Steganography:** Video files are interlaced images and audio files where the image and audio techniques apply to video files. A large amount of data can be leaked using a steganography method, as small distortions in a video file are often unobserved by humans.
5. **Network/Protocol Steganography:** This type refers to embedding information within messages and network control protocols in network communication. In the OSI layered network model, covert channels are used for steganography. Figure 2.1 shows how steganography can be implemented at different levels [78]. This study also focuses on a similar domain, i.e. detection of network steganography.

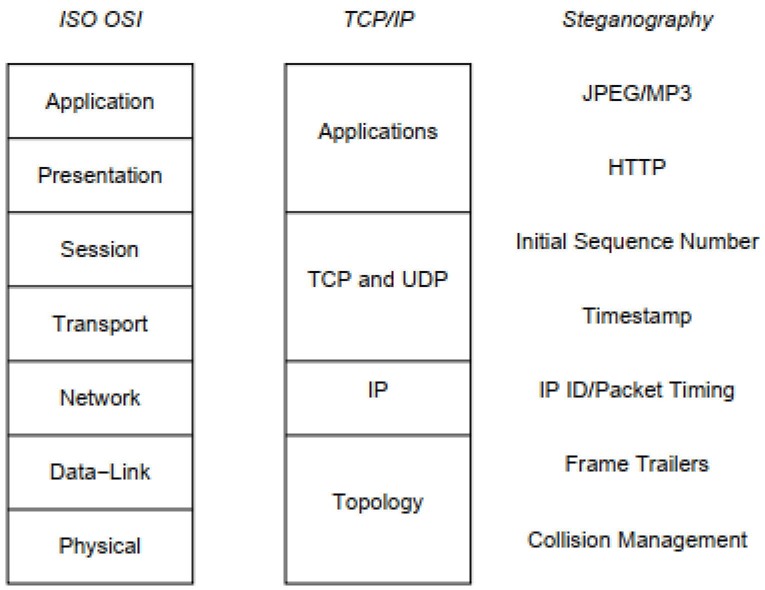


Figure 2.1: Steganography at different levels of OSI Model (Source: [78])

Network steganography is increasingly being used by malware in today’s world to facilitate data leakage. Authors in [79] discussed network steganography and threw light on malicious purposes of attackers for data leakage using steganography. Some steganography techniques and countermeasures are also discussed in the paper.

2.4.2 Steganography Techniques

1. **Spatial Domain Methods:** This type of method is applicable to image and video contents where secret data is embedded directly in the intensity of pixels. The pixel value of an image is changed during the process of hiding the data. The spatial domain methods can be categorised as follows:
   a. Least Significant Bit Insertion: This is the simplest approach where the Least Significant Bit (LSB) of a byte is replaced with a bit of a covert message. The technique is useful for both audio and video steganography and the result looks identical to the human eye.
   b. BPCP: In this method, the segmentation of an image is used to measure its complexity. The complexity is then used to determine a noisy block that is then replaced with a binary pattern mapped from the secret data to be transferred.
   c. PVD: In this method, consecutive pixels are selected for embedding covert data. The payload is determined by computing the difference between two consecutive pixels.
2. **Spread Spectrum Technique:** In this method, covert data is spread over a wide frequency of bandwidth. The ratio of signal to noise in every frequency band is made small enough so that it becomes difficult to detect the presence of the data.
3. **Transform Domain Technique:** This method of steganography uses algorithms such as Discrete Fourier Transformation, Discrete Cosine Transformation, Discrete Wavelet Transformation, Lossless or Reversible Methods for hiding a message in the frequency domain.
4. **Distortion Technique:** In this technique, a secret message is stored by distorting signals. A sequence of modifications is applied to an image by an encoder. The corresponding decoder measures differences between the original cover and the distorted cover to detect a sequence of modifications and hence recover the secret message.

5. **Masking and Filtering:** Masking and filtering techniques are mostly applied on 24 bit and greyscale images. Information is hidden in a way similar to a watermark. Masking images entails changing the luminance of the masked area. A small luminance change is less detectable thus averts unwanted attention. Masking is more robust than LSB in the context of compression, cropping and image processing.

2.5 TCP/IP-based Steganography

Steganography can be implemented in different fields of TCP/IP. However, in the past, stenographic techniques failed in TCP because of the different probability distributions of unmodified TCP/IP implementations [80]. In some cases, the implementations of these protocols are outside the relevant specification as in stenographic implementation. It is important to know the applicable standards and implementation details. The upgraded versions of TCP/IP have extra scope for stenographic coding as compared to the older versions. For example, RFC 1323 is an extension of RFC 793 and RFC 791 and has additional header options to use for stenographic purposes. However, the IP part still has limited scope due to its properties. Figure 2.2 shows the structure of a TCP/IP header.

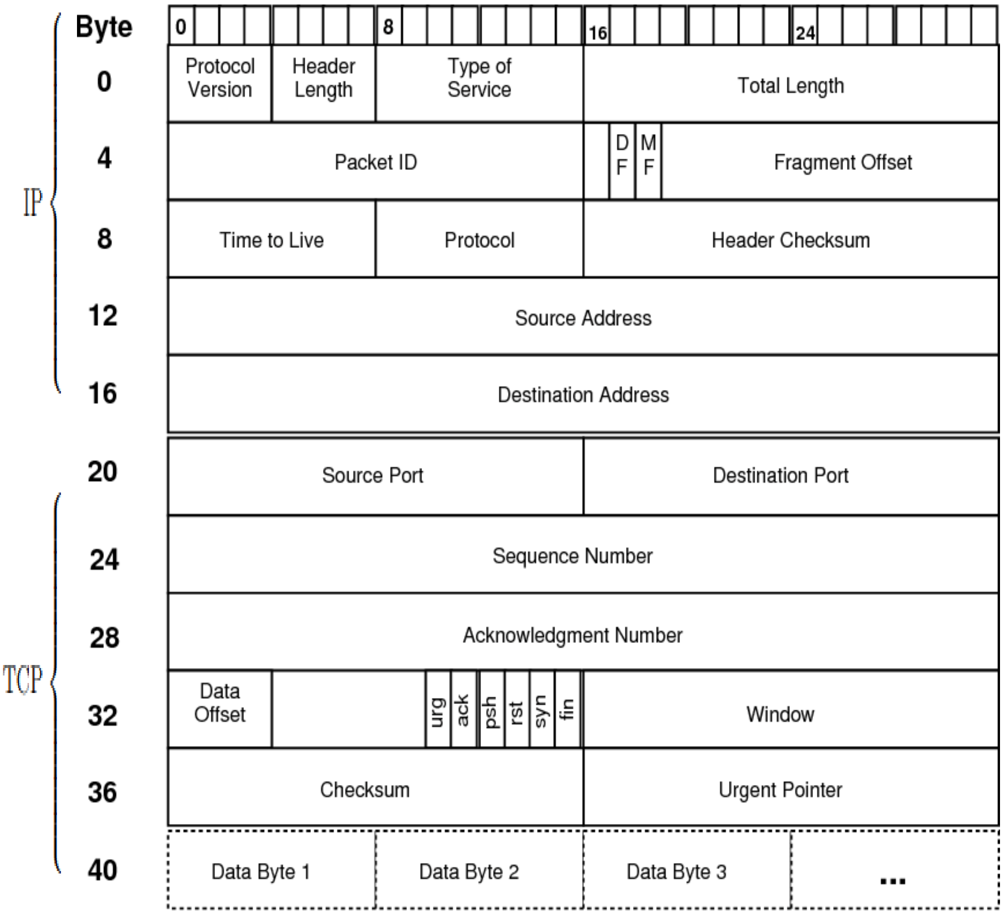


Figure 2.2: Basic IPv4 TCP/IP Header (Source: [78])

The further fields of the TCP/IP header are discussed below, which are used in steganography [78]:

a) **Type of service (ToS):** It is the second 8 bits field in the IPv4 packet header. The precedence field denotes the importance of the packet with three bits for delay, throughput and reliability as shown below. Most networks do not use these fields, so they can be used as a carrier for steganography [78]. However, it can be easily detected as the unused field is set to zero in almost all operating systems.

7 6 5	4 3 2	1 0
Precedence	Type of Service	Unused (0)

b) **IP flags:** IP packets have two flags, ***Do Not Fragment*** **(DF),** and ***More Fragments*** **(MF)**. DF indicates that if the packet cannot be sent without being fragmented, it should be discarded. MF shows 0 if the packet contains the last fragment or has not been fragmented. The DF bit can be used for stenographic signalling as mentioned in [81].

c) **Fragment offset:** IP fragment offset fields facilitate a packet receiver to reorganize the fragmentation of the packet in the right order. This field can be used for TCP/IP based steganography.

d) **IP options:** This field can also serve as a means for carrying stenographic materials. However, since IP packets rarely contain “options”, their potential for use in steganography is limited.

e) **TCP timestamp:** The TCP timestamp option enables a host to measure the round trip time of a path, and also solves problems related to sequence number wrap-around. The timestamp option consists of two 32 bit fields, TS Value and TS Echo reply. Giffin, et al. [82] proposed a covert channel based on the TCP timestamp.

f) **Packet order:** This field can also be used as a means for carrying hidden and stenographic contents. However, since packets are very rarely reordered, that is why this field is easily detectable when used for steganography.

g) **Packet timing:** A type of covert channel can be created by manipulating timing between packets. These are termed as timing covert channels. Packet timing can carry covert information.

h) **TCP sequence number:** The TCP sequence number is a 32-bit field used in the TCP header. It supports features such as flow control and reliability, provided by TCP. Each packet of data transmitted over a TCP stream is assigned a unique sequence number. This field is one of the best possible carriers of steganography, because of the random nature that this field possesses and its size that is large enough to carry a big chunk of information easily. However, the data transmission rate has to be chosen carefully to avoid detection. ISNs are used which must not be easily guessable by those who are not involved in the connection [83]. ISNs are generated in such a way that they do not overlap in sequence number spaces. Similarly, if a connection is in the TIME-WAIT state and ready to start again, the ISN for that tuple pair should increase monotonically too.

The framework proposed in this thesis uses the ISN number. Therefore, it is important to understand how it is generated and used. Rowland described a practical implementation of steganography using TCP ISNs, named as covert TCP, in [84] with further details discussed below.

ISNs are generated from different operating systems. The Linux 2.0 ISN producer (shown in Figure 2.3) is based on RFC 1948 [83]. Linux is an open source operating system designed for embedded servers and systems. It is based on a modular design, which schedules applications and processes, handles network access and oversees file systems and services. Linux 2.0 offers a superior terminal for developing and experimenting new security solutions due to its openness that is lacking in other operating systems such as Windows. Most existing security solutions work on the Linux platform although they can be adapted to other operating systems as well.

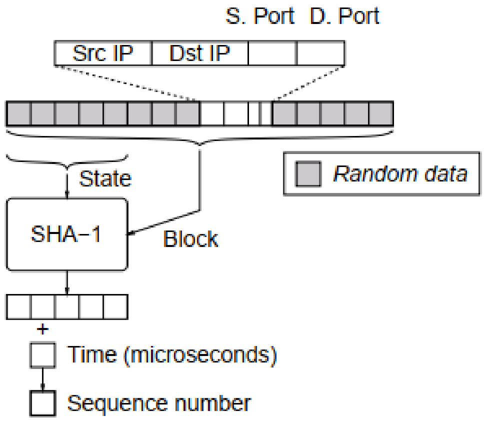


Figure 2.3: Linux 2.0 ISN generator (Source: [22])

It uses SHA-1 to hash a block of sixteen 32-bit words, including words 9–11 set to the source and destination IP addresses and ports of a packet, and the other 13 words filled with a secret initialised during the computer boot process. Instead of using the values predefined in the SHA-1 standard for the initial state, the first 5 words of the block are used. To obtain an ISN, the second word of the hash is selected and the current time (in microseconds) added.

In terms of the importance of ISN values in detecting covert channels, different studies [85] [22] discussed the behaviour of ISN random number generation in different operating systems. This helps to find hidden patterns in random data.

2.6 Covert Channels

Covert channels are communication channels that are used to transfer sensitive or private data between two parties covertly. The concept of the covert channel does not lie in the data hiding but the hiding of the entire communication channel. Covert channels are broadly divided into storage and timing covert channels. Storage covert channels exploit access to system resources while timing covert channels exploit the timing between sending and receiving packets. Various fields where such channels can be designed are discussed further below.

2.6.1 Data Leakage through Covert Channels

It is understood that data leakage can be caused as a result of:

- Data leakage from actual payload where the payload of a TCP/IP stream is extracted or listened to by two parties when they are communicating.
- Data leakage where actual payload is held on covert channels such as in the TCP/IP packets header field. The payload held in covert channels is extremely dangerous as leakage solutions mainly monitor the payload section of a packet, not its header.

According to authors in [86], covert channels are described as a technique for secretly communicating between two or more parties, assuming that the normal data channel is under continuous surveillance due to low-level messages containers modifications. In paper [87], the channels are divided into two main types:

1. Covert Storage and Timing Channels: A covert channel, which involves direct or indirect writing to a storage location by one process followed by direct or indirect reading by another process, is called a covert storage channel. A covert timing channel involves one process signalling information to another by modulating its system resources. This manipulation results in real-time observations from the second process to receive the information.
2. Noisy and Noiseless Channels: A noiseless channel uses resources that are exclusively available for two parties to conduct uninterrupted communication between them. On the other hand, a noisy channel utilises shared resources, so it is required to distinguish between data from a particular sender and other data from other senders.

2.6.2 Application Layer as Covert Channels

In the TCP/IP stack, the top layer is called an application layer that is provided by a software application, not the operating system. Any data exchange between a user and the transport layer (the layer underneath the application layer) is performed via the application layer. Since messages created at this level use the transport layer for data delivery purposes, these high-level data streams are not altered by security applications and hence have the highest susceptibility to covert channels hidden in the messages. A message sent by *Alice* is guaranteed to reach *Bob* unaltered and un-scanned. Additionally, in the cases where networks employ proxy servers, the syntax of messages transferred remains unchanged, does not alter control fields or data in a significant way, and allows the creation of noiseless covert channels.

Figure 2. and Figure 2. illustrated the structure of the TCP and IP header and possible covert channel fields.

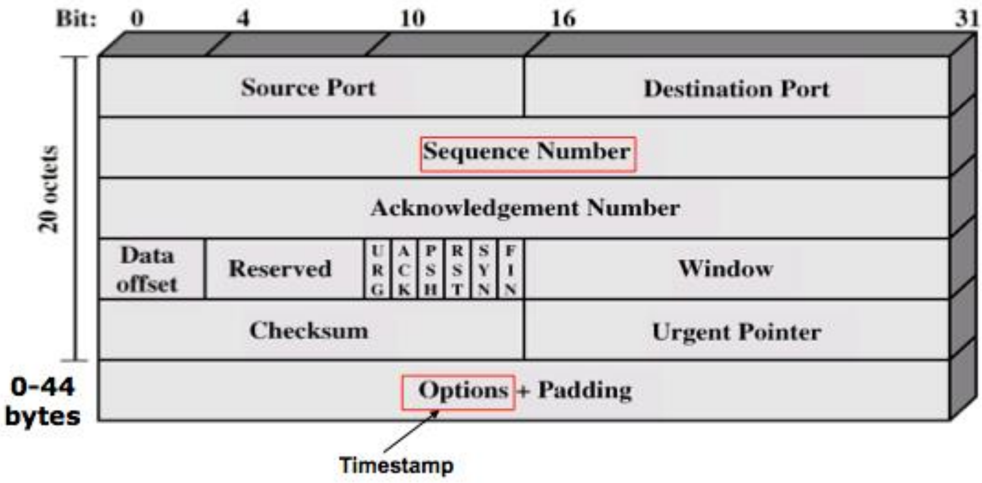


Figure 2.4: TCP Header Structure (Source: [45])

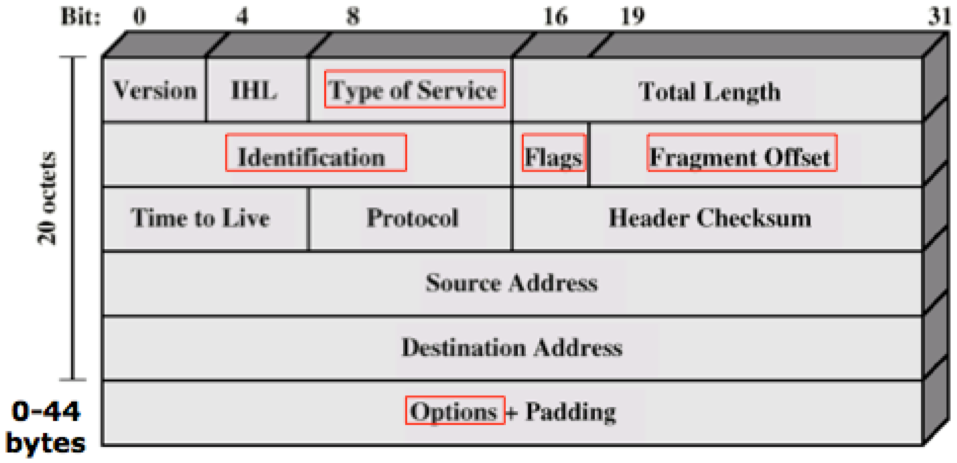
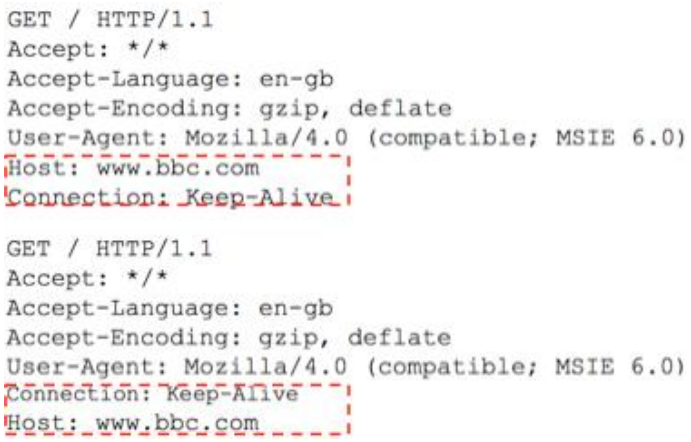


Figure 2.5: IP Header Structure (Source: [45])

According to [88, 89], there are five different methods for implementing a covert channel with the application layer header as explained below:

1. Reordering of Fields

According to authors in [87], headers in the HTTP envelope change from implementation to implementation. This variation could be used to encode a covert payload. The concept of pattern variation is changed which allows the automatic adoption of the generic hiding method without the re-implementation of the method itself. Authors claimed that a pattern catalogue allows the researchers to modify and extend the covert channel patterns collection in a moderated process. Figure 2. shows HTTP header reordering.

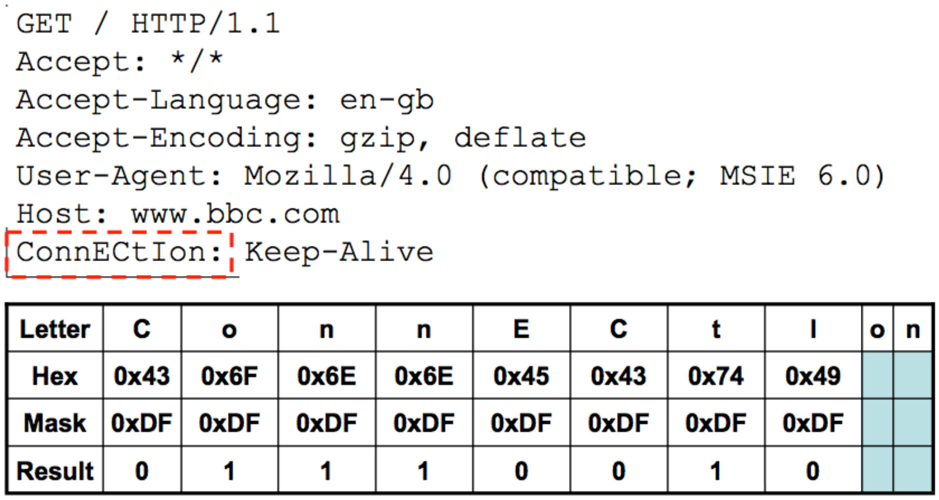


```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.bbc.com
Connection: Keep-Alive

GET / HTTP/1.1
Accept: */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Connection: Keep-Alive
Host: www.bbc.com
```

Figure 2.6: HTTP Header Reordering (Source: [90])

2. Modification of Letter Cases

Fields in the protocol header are case insensitive [87, 91]. This means that the field name case is ignored by the standard HTTP specification. Figure 2. shows the encoded bits of the ASCII code in lower-case (binary ones) and upper-case (binary zeros) letters.



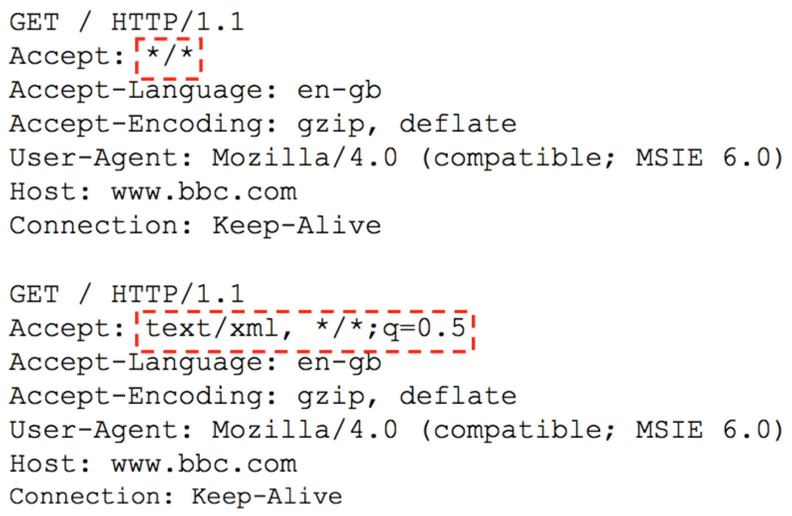
```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.bbc.com
ConnECtIon: Keep-Alive
```

Letter	C	o	n	n	E	C	t	I	o	n
Hex	0x43	0x6F	0x6E	0x6E	0x45	0x43	0x74	0x49		
Mask	0xDF	0xDF	0xDF	0xDF	0xDF	0xDF	0xDF	0xDF		
Result	0	1	1	1	0	0	1	0		

Figure 2.7: Case modification resulting in encoded message (0x72) (R) (Source: [90])

3. Use of Optional Fields and Flags

Figure 2. shows a possible covert channel implementation where a wildcard in the header indicates that any file type is allowed as a response, while a filtered list of responses indicates an encoding. Here, a normal wildcard is considered as binary zero, and filtered file types are considered as binary ones [87]. This is indicated with red encircles in Figure 2..

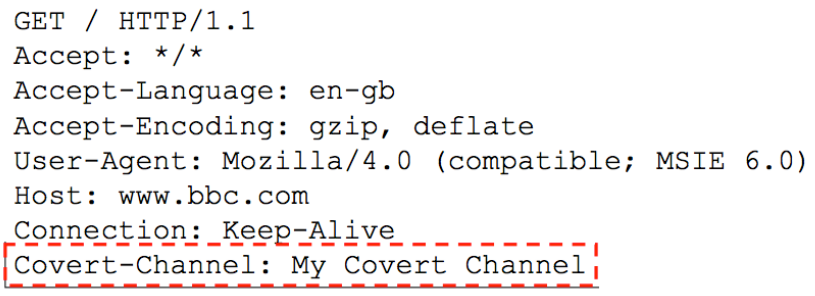


```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.bbc.com
Connection: Keep-Alive

GET / HTTP/1.1
Accept: text/xml, */*;q=0.5
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.bbc.com
Connection: Keep-Alive
```

Figure 2.8: Use of optional fields for covert communication (Source: [90])

4. Adding of New Fields

Figure 2. shows additional fields in a covert implementation. Most applications are configured to ignore unrecognised headers and hence will treat the request and response with a new field which is not there. In this covert implementation, the covert payload is held in a new field [92].

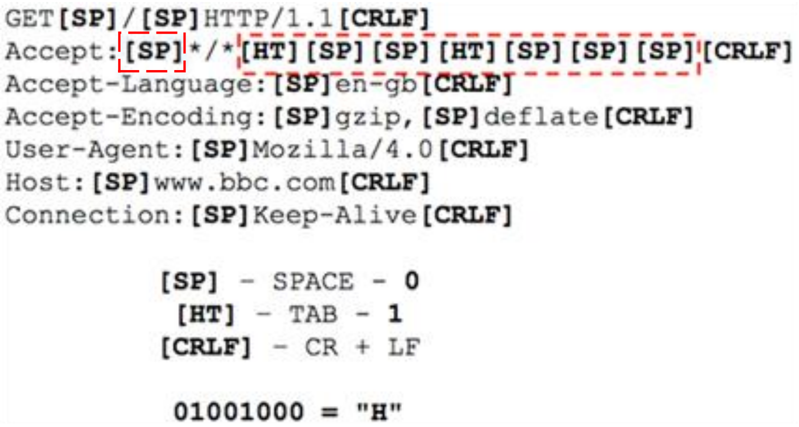


```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-gb
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0)
Host: www.bbc.com
Connection: Keep-Alive
Covert-Channel: My Covert Channel
```

Figure 2.9: Addition of new fields for covert communication (Source: [90])

5. Use of White Spacing Characters

Web browsers do not differentiate between one or more spaces separating HTTP header values. Hence, by changing the number of white spaces, a covert channel can be created. Figure 2. shows [SP] as binary zero and [HT] as binary one, making a covert message as 01001000 which is equivalent to ASCII H [87].



```
GET[SP]/[SP]HTTP/1.1[CRLF]
Accept:[SP]*/*[HT][SP][SP][HT][SP][SP][SP][CRLF]
Accept-Language:[SP]en-gb[CRLF]
Accept-Encoding:[SP]gzip,[SP]deflate[CRLF]
User-Agent:[SP]Mozilla/4.0[CRLF]
Host:[SP]www.bbc.com[CRLF]
Connection:[SP]Keep-Alive[CRLF]

          [SP] - SPACE - 0
           [HT] - TAB - 1
          [CRLF] - CR + LF

           01001000 = "H"
```

Figure 2.10: White-spacing characters for introduction of encoded message (Source: [21, 90])

2.7 Summary

Chapter 2 presented the details of data leakage and the possible types of attacks. The detailed discussion is presented based on the existing literature in the area of covert communication using network protocols. After the detailed discussion, this chapter describes data leakage attacks, the steganography background, its types and techniques, including TCP/IP based steganography and its characteristics. Covert channels and data leakage through them in computer network protocols are also discussed. In the last section, data leakage prevention tools and technologies are outlined to understand the scope and possibilities for designing more optimal solutions. This chapter provides the background information on the thesis research topic and lays down a foundation to conduct a more focused review of the research area in Chapter 3 and subsequently the proposed research in the other chapters. The next chapter will provide a critical review of the related work on existing data leakage frameworks and their limitations and also set out necessary metrics for the framework design.

Chapter 3 Related Works

3.1 Overview

This chapter discusses the related work on deep packet inspection and deep content inspection for data leakage detection. Related algorithms are critically reviewed based on the existing literature to measure the detection and prevention metrics of data leakage through network protocols. Covert channels are also discussed in terms of their involvement in the transport control protocols: User Datagram Protocol (UDP), Hypertext Transfer Protocol (HTTP), Real-time Transport Protocol (RTP) and Real-time Transport Control Protocol (RTCP). The findings from the review provide a solid rationale for the development of the framework proposed in this thesis to detect and prevent data leakage.

The main focus of our research is on a data leakage security weakness. Covert channels are one of the attacks that transfer stolen information through networks. These attacks occur over non-conventional channels known as covert channels. A covert channel is exploited by a process to transfer the information which violates the security policy [93]. This issue leads to security violation and loss of sensitive or private data together with other losses such as reputation. To address this issue, the existing data leakage monitoring techniques and tools have applied to TCP headers. However, these tools and techniques have their limitations and are not able to detect certain covert channels.

A literature review was conducted to identify the limitations of existing detection and prevention techniques and find out research gaps. One of the limitations identified from the literature review is that the existing solutions are unable to properly deal with covert data in the header field of network data packets [94]. The accuracy and reliability of deviation from norms in dynamic real-time TCP-based communication are also assessed.

The data leakage prevention approaches are categorised into host-based and network-based solutions [95]. Network-based solutions are used to scan the data traffic at network access points and examine the data traffic of various applications while further filtering and blocking the inbound traffic that can cause potential leaks. The solutions can also block access to URLs containing inappropriate content and stop confidential data loss over IP network connections. These solutions block the rules and prevent the data access. Although these solutions block the rules for data access, it is believed that the real value of the solutions comes from their ability to collate forensic information by providing a feature to track the data traffic [95]. On the other hand, host-based solutions are applied at the endpoint, e.g. PCs, laptops and other user devices. Host-based solutions are often based on data encryption on the persistent store (when not in use), monitoring of host operations and setting up security policies to restrict the transfer of sensitive data to a limited number of authorised users. Host-based solutions are managed by centralised data security policies, where the solutions can track, discover, notify and protect the data at rest. These solutions can be used in an isolated manner or can be used to complement each other.

Network-based data leakage and host-based data leakage solutions are used to prevent data leakage at access points and host sides. The subsequent subsections will discuss existing solutions to deep packet and content inspection and data leakage detection as well as their advantages and limitations.

3.1.1 Deep Packet Inspection and Deep Content Inspection

Network-based data leakage detection is further complemented with host-based approaches. These approaches focus on the analysis of un-encrypted network data through a Deep Packet Inspection (DPI) approach [96]. The DPI solution inspects every packet in order to check the occurrence of sensitive data as defined in a sensitive data database [95]. This solution generates alerts when sensitive data is found in outgoing data traffic from the network. The solution offers good data leakage protection. However, it requires to store the sensitive data in plaintext in the detection system. It is undesirable to hold the plaintext for comparison purposes as the machine performing the comparison could be compromised, thus revealing all the sensitive data. It also does not support the outsourcing of data leakage detection to a third party because it would learn the sensitive data [35]. It is noteworthy that the data processing on data leakage detection consumes heavy computation power. This leads to involving the outsource of the task to a third party. However, the sensitive network data traffic and sensitive databases are in plaintext format, so the data security is at risk for outsourcing [92, 95]. DPI checks network packets for anomalies based on rules that operate on the layer 3/4 of the OSI stack, to pass, block or raise alerts about the packets [96].

Another technique for data leakage prevention is Deep Content Inspection (DCI). The DCI method keeps track of contents across multiple packets where signature matching is needed across these packets. Signature matching is a technique for organizing, retrieving, and navigating the software libraries. DCI analyses the passing data based on rules that operate on the layer 7 of the OSI stack. Studies [97, 98] mention various methods for content matching with their root in the field of data mining. For instance, content matching at layer 7 can be achieved by holding attributes such as keywords, regular expressions, file types, file sizes, senders and recipients. Another DCI technique is machine learning such as a vector space model. Vectors represent documents, and vector features represent their frequency. These vectors are used to create probabilistic models based on decisions made on the fly regarding leakage prevention. Table 3.1 illustrates differences between the DCI and DPI techniques as well as their common attributes. Table 3.2 presents the comparison of host and network-based data leakage prevention methods.

Attribute	**DCI**	**DPI**
OSI Stack Layer	Layer 7 (Application)	Layer 3 / 4 (Network / Transport)
Inspection	Ability to understand the information in packets as it works across multiple packets.	Ability to read packet headers in order to read elements such as sender, destination, port, application type and other packet level information.
Protection	Data inspection	Low level packet inspection
Prevention	Rule based analysis where rules are based on data keywords, file types, etc. Also, a strict set of conditions is in place (set by administrators)	Rules are based on packet headers.

	with regard to the holding of data on devices.	
Known leakage causes	Malicious insider loss of data due to failing of data storage conditions.	Network attackers (generally external).

Table 3.1: Difference between Deep Packet Inspection and Deep Content Inspection in the context of common attributes.

Approach Type	**Prevention/Analysis Method**	**Work By**	**Issue**
Host Based	Encrypt Data	All shared data is encrypted and decrypted by users with private keys.	Key exchange problem, Trust between two parties, Authentication and integrity checking.
	Policy based restriction on shared data	In order to secure the data, policies are created on shared folders for protecting the access.	Legal and policy restrictions.
	Monitoring Tools	Monitoring tools are used to log methods for accessed data, time, etc.	Lack of network visibility, Balancing active and passive network monitoring.
Network Based	Deep Packet Inspection (DPI)	DPI works by inspecting network traffic for sensitive data and comparing it against sensitive data database. It triggers alerts when a match is found between the sensitive data and database.	• Works on plaintext sensitive data and can be easily read from network packets. • Data inspection consumes more processing power. • Outsourcing it to a third party is risky due to the plaintext sensitive network traffic and database. Both sensitive data and sensitive data database can be compromised.
	Entropy Analysis or Theoretic Analysis	This technique is used to hide information.	Process complexity, low entropy data sources.

Table 3.2: Comparison of Host and Network based Data Leakage Prevention Methods

3.1.2 Data Leakage Detection

To prevent the data leakage, authors in [99] formulated the data leakage prevention problem as Partially Observable Markov Decision Processes. The model created encodes a mechanism for monitoring an invisible recipient’s data. The technique is called digital watermarking. The authors presented an optimal information sharing strategy for data leakage and monitoring the malicious data on the recipient side. The paper provides a test case where only one recipient either leaks all the received data or no leakage data. Another test case considers a fuzzy recipient who leaks a certain percentage of data that it receives. The final test case allows the sender to share the data with a set of fuzzy recipients. The experimental results from these test cases show the effects of different settings on the data leakage cost, fuzziness level and trust of the sender on recipients. This model is tightly coupled with the monitoring mechanism; the efficiency of the monitoring mechanism decides the effectiveness of data leakage detection.

In [47, 100], the authors provided an introduction to leakage detection systems. The specific case discussed in this study is a collection of evidence in case of data leakage. The authors proposed the use of a strategy that includes the distribution of data by a distributor to agents upon explicit or sample-based requests. The data distribution includes a recording of the data which is given to agents, the creation of fake objects that act as a watermark for forensic analysis in case of leakage, and creation of a guilt module for finding out a guilty agent(s) when leakage occurs. The authors also discussed the methods of watermarking used for embedding unique objects in non-significant sections of data. They looked into relevant documents and media and surveyed various associated watermarking methods. Through experiments, the authors concluded that the distribution of fake objects along with an old method of watermarking can provide security as a detection method. Furthermore, this method can find guilty agents to assist in forensic analysis.

In [101, 102], the authors implemented a security analysis method for certifying the security of an application by analysing its information flow by using a program in a static state. The technique is based on a lattice model as described by [103] that guarantees that the application never leaks information to any insecure storage. In [102], the technique is extended to include a method or procedure calls as well as access to global variables of an application. The technique can be used as a guiding model or a testbed for applications during their software development phase. However, the authors did not discuss the case where an application is already deployed in a corporate environment.

In [104], the authors used the strategy of honeypots for detection, identification and gathering information about malicious senders in the context of data leakage. A honeypot is a trap set to detect, deflect or in some cases counteract the attempts of unauthorised usage of production systems. It appears to be part of a network but remains isolated and protected. Its value lies in being probed, attacked and compromised. Hence, honeypots have no production value and should not work with any legitimate traffic or events. According to [104], the main difference between a regular honeypot and the authors' implementation is that the honeypot is inside a corporate network rather than being invisible and it is openly advertised so that all the network traffic goes through it. The authors have concluded that the honeypot can positively contribute towards the early detection of data leakage.

Another approach to data leakage prevention is discussed in [105] where authors proposed shadow execution. This means that two instances of the same application run but with different initialisation strategies. One of them is public with access to the Internet, while the other is private with no access to the Internet. The public copy is initialised with different inputs and restrictions, whereas the private copy is initialised with actual user data. The results obtained from the public copy are used by the private copy in order to successfully communicate over the network while ensuring that no data leakage occurs.

Statistical techniques are the ones involving data capturing followed by the measurement of deviation from expected stochastic behaviour, i.e. they are appropriate for the cases where observations are not reproducible exactly. Other measurements include the audit of records, categorisation of various activities, amount of activity and also system resource usage. A deviation is calculated by comparing a current result to a pre-known/calculated stochastic expectation. If the measure is outside a postulated threshold, the result is categorised as a failure/leakage activity. There are other approaches that link a failure to the amount of irregularity for the data under observation, which is considered as an outlier if it is not within a postulated threshold such as guilt model. In [47], the authors discussed a guilt model in detail and explained with an instance where, upon the distribution of a data set (*T*) by the distributor to agents, the distributor discovers that a subset (*S*) of *T* is leaked. A target (*t*) is found in the possession of *S*. Since agents U_1 to U_n are in the possession of *T*, each of them is effectively suspected. However, a set (*U*) of agents may argue that they are innocent and *S* is obtained by *t* through other means, for instance guessing or from publicly available data. In this case, the goal is to estimate the likelihood/probability that *S* came from *U* rather than from other sources. Clearly, the bigger the *S*, the more difficult for *U* to argue that the data is created from publicly available sources. Conversely, the smaller the *S* and rarer the object in *S*, the more difficult for U to argue that these are not involved in the leakage. So the goal is not only to find evidence that U is involved in the leakage, but also which one of *U* is more likely to be the leaker. Hence, U_i is guilty if it contributes to one or more objects to target *t*. This guilt event is denoted by G_i for agent U_i and the event that agent U_i is guilty of leaked set *S* is expressed by $G_i|S$. Thus the objective is to calculate the probability of $P(G_i|S)$, i.e. the probability that agent U_i is guilty, given evidence *S*.

Although the guilt model presents a solution to identifying a possible agent(s) which caused data leakage. It does not provide a facility to prevent data leakage in the first place. Also, the model lacks the ability to completely prevent data leakage.

3.2 Detection Algorithms in Network Protocols

This section covers the existing research work on network covert channel detection for different protocols.

3.2.1 Covert Channels in Internet Protocol

Internet Protocol (IP) works within the TCP/IP protocol to provide the encapsulation of segments received from the transport layer with the IP header and ensures its delivery to the next layer using IP addresses from one endpoint to another endpoint over the Internet. IP types fall into two forms including version 4 (IPv4) and version 6 (IPv6). The following literature summarizes covert channels within the IP, alongside different strategies used to achieve steganography.

Whenever attackers attack, the main task of traffic normalizers and protocol scrubbers is to eliminate the ambiguities that may be found in the traffic stream. The authors in [106] described how active wardens, or traffic normalizers, perform the following tasks one by one:

- Modifying the incoming/outgoing packets
- Modifying reserved fields to zero
- Setting all the padding to zero

Such traffic normalizers or active wardens are parts of end hosts over the network. Traffic normalizers also use the header fields for steganography sometimes, setting them to zero or rewriting according to different situations. The Time to Live (TTL) field is usually eliminated by the active wardens and set the same for all the packets in the flow. The authors in [107] used Jitterbug, a type of keyboard, to exfiltrate the typed data. It is done by putting some kind of device between the keyboard and the victim machine. Each keyboard press passes the code to transfer a packet over the network, while the delays between key presses are considered as covert timing channels. The Jitterbug keyboard encodes binary 0 and 1 values. When 0 is added, the delay is multiplied by a window value w, whereas for 1 the delay is multiplied by w/2. Timing channels are known to be the most commonly used covert channel for steganography. The authors in [37] used an improved timing channel that is based on replay attacks, in which saved real IPDs (Inter-Packet Delays) are used as samples. The samples are divided into two groups: binary 1 and binary 0. 1 is sent for replaying IPDs randomly and 0 for replaying the groups. Timing channels with reduced robustness are a new attraction in the steganography field. It also overcomes the challenges like volatile network traffic and imprecise timekeeping mechanisms. This approach also works well within dynamic network protocols.

The authors in [108] proposed a new method that is based on a packet timing channel. It takes inter-packet times of any normal traffic and IPDs of continuous packets at inter-transmission times: T1; T2; … Tn, and then masks the message to be shared by performing one-time pad encryption. This method provides better synchronization using a key hashed with random numbers that make it stealthier for encoding covert bits into the least significant bits of IPDs. This improved approach does not only give a lower false-positive rate but also shows the capacity increase by hundreds of bits per second depending on the network jitter.

In [109, 110], a new covert channel is introduced in which the authors used the traceroute command to record the IP header's record route option. It presents a protection strategy by steganalysis and uses a sniffing module to help in hiding a message in the valid IP header's record route option. To keep the selection procedure of IP packets, a hidden algorithm is generated to choose them randomly. This approach's covert memory option makes it unique from all the previous contributions to steganography techniques.

Allix [111] proposed a new approach for a timing covert channel where an attacker has two control machines A and B with each having a connection to a common server C. If machine A sends a single packet, then machine B sends two packets, which is interpreted as value 0. Conversely, if machine A sends two packets, then machine B sends one, which is interpreted as value 1. The study further showed that, when complex detection techniques like neural network or SVM are used, the approach gives fewer false positives.

Another model-based timing covert channel [112] mimics the statistical properties and contains a filter, an analyser, an encoder and a transmitter. It takes real traffic behaviour into account through the filter and passes the IPDs to the next module for analysis. Then an appropriate distribution function is chosen for encoding. Finally, covert packets are transferred. In [113, 114], detection by fragmentation is introduced in which the different statistical numbers of fragments, fragments sizes, rates or unrelated presences are used for testbeds. It also uses a known method for the elimination of original IP packets by re-fragmenting. PMTUD (Path MTU Discovery) and PLPMTUD (Packetization Layer Path MTU Discovery) are also used. Experiments proved that the proposed method works well for the covert communication containing a 100MB file with 1MB of hidden data. If the communication of the hidden data takes 13 minutes, then the risk of being detected is very low.

The researchers in [115] showed a novel reordering approach that is robust and resilient in nature to external reordering. It also has built-in error detection and correction capabilities that are achieved by assigning random symbols to each different permutation of consecutive packets and making subsets of permutations. To make code words, an innate reordering scheme is used. This method shows an error rate of only 0.1% because of its external reordering effects. Coding efficiency is checked by sending 2 bits per packet to increase the capacity level. In [116], a new class strategy of mitigating timing channels is introduced. The key point in this approach is the future availability of the packets which are predicted. So, a mitigator is able to detect the number of packets leaked. It uses an adaptive mechanism to generate epoch-based mitigators. Black box mitigation is used for timing channel changes; it is capable of finding out timing channel bounds. The main contribution of this approach is that it gives the information of control flow through storage channels. While in [117], Huffman coding is used to improve timing channel performance by taking robustness, accuracy, capacity and covertness into account, as the performance is an important part of covert communication no matter whether it is considered by the attacker side or the victim side. In this approach, to increase the capacity and covertness, the redundancy of letters is used. Also certain parameters are investigated for experimental results. If covert communication takes place at a rate of 58.35 bps while the RTT is estimated at 35ms, the accuracy rate increases by 250%. Even if N-ary is kept as low as 2, the rate can be raised by 50.7% to 25.12bps.

Potential risks of covert communication are practically proven in [118] showing how the high bandwidth of memory buses and redesigning of timing channels help in the conduction of covert communication. Cloud computing is considered as a robust data transmission domain. To prove that the proposed solution in [118] works well, a testbed is created in the Amazon EC2 cloud and attacks are sent from a host machine. Even high bandwidth with the reliable transmission of covert channel attacks is shown as possible intake. In addition, the author in [119] described a new packet length based approach in which a covert channel uses a secret sharing scheme for exploiting a covert medium of communication. Different security analysis mechanisms are used to overcome the downsides of the previous approaches. For example, a new covert path is discovered on VoLTE (Voice over LTE) that is used for covert communication by postponing or extending its silence period [36]. The silence period is considered as a normal phenomenon for voice communication and its suspicion is quite undetectable. To increase the robustness, the Gray code along with the silence periods is employed against all noise channels regardless of whether they are intended or unintended. Experiments were made on the VoLTE traffic to find out the effectiveness of covert channels in terms of voice quality, robustness and detectability. This approach outperforms in robustness against channel jitter.

In [120], the authors proposed a model based on generic detection to prevent covert channel communication in networks. This model is based on a supervised machine learning approach where data sets are generated and applied on IP, TCP and DNS protocols. After various experiments, this study revealed that SVM is optimal and effective for detecting the covert channels in the TCP/IP protocol suite. Authors also pointed out that SVM has not got sufficient results against the DNS protocol. It is observed that a decision tree works well on DNS, but it is not feasible to detect new attack vectors, especially when attacks are vector masked

Table 3.3 provides a summary of IP based covert channels strategies.

Author	**Year**	**Strategy**
Zander, et al. [106]	2006	Active wardens are used to change TTL (Time to live) values. Covert communication is held by varying the TTL field values in IP headers in the traffic normalization process.
Shah, et al. [107]	2006	A keyboard device JitterBug is introduced. This device is placed between a victim and an attacker’s

		communication server, and for every key stroke a secret message is delivered.
Cabuk, et al. [121]	2006	A timing channel based on replay attack is improved by dividing a hidden text in two groups of IPDs 0 and 1. One group of IPDs sends binary 0, while the other sends binary 1.
Trabelsi, et al. [109, 110]	2007	A covert channel is proposed and implemented, in which the use of the "traceroute" command and IP header's option of Record routes are taken into account to perform hidden communication.
Allix [111]	2007	The count of replies while having a conversation between two endpoints on the Internet is kept as a threshold and considered as 0 or 1 to deliver covert communication.
Gianvecchio, et al. [112]	2008	This approach uses 4 modules to generate covert communication, named as filter, analyser, encoder and transmitter that work step by step to bend an original message with a hidden message.
Mazurczyk and Szczypiorski [113], [114]	2009	A new potential covert communication method is introduced for dealing with oversized IP packets: IP fragmentation, PMTUD (Path MTU Discovery) and PLPMTUD (Packetization Layer Path MTU Discovery). It is proposed for both IPv4 and IPv6.
El-Atawy and Al-Shaer [115]	2009	Reordering messages in some special manner is used to send out hidden information.
Askarov, et al. [116]	2010	Black box mitigation is used for timing channel changes, which is capable of finding out timing channel bounds.
Wu, et al. [117]	2012	Huffman coding is used to improve timing channel performance by taking robustness, accuracy, capacity and covertness into account.
Wu, et al. [118]	2015	Potential risks of covert communication in cloud computing are practically proven, showing how the high bandwidth of memory buses and redesigning of timing channels help in the conduction of covert communication.
Lu, et al. [119]	2016	A packet length-based approach is proposed in which a covert channel uses a secret sharing scheme to exploit a covert medium for communication.
Zhang, et al. [36]	2018	A new covert path is discovered on VoLTE (Voice over LTE) where covert communication is modulated by postponing or extending silence periods.
Ayub, et al. [120]	2019	This genetic approach is used to detect network storage covert channels by supervised machine learning classification techniques. SVM is used for covert channel detection in transport and network

		layers and achieved better results in the application layer.

Table 3.3: Summary of IP based Covert Channel Approaches

3.2.2 Covert Channels in Transport Control Protocol

TCP provides an end-to-end connection-oriented service. Chakinala, et al. [122] introduced a reordering scheme in which a timing covert channel is created by reordering TCP segments and making use of the Sequence Number field. The mathematical model used performs an information-theoretic formula to prove that the Nash equilibrium exists. New distance metrics are created using a permutation strategy to perform error correction through codes. Their results show a good correlation between simulated and theoretical findings. The effects of these combinatorial bounds are checked on various topologies to prove the best results as compared to the other approaches. In [123], the Cloak named software is developed and implemented to help in mimicking TCP-based flow packets by ten different encoding and decoding techniques for the development of covert communication. It uses Enumerative Combinatorics for encoding a hidden message into different TCP flows while only a fixed number of TCP packets are used. The former faces decoding problems which occur due to inherent channels timing. Cloak uses a two-step detection algorithm and is considered the best solution but in a controlled environment. Moreover, Luo, et al. [124] proposed a storage covert channel strategy by modifying the Acknowledge Sequence Number field. It does not make amendments directly to the IP or TCP header files unlike the previous approaches. A new covert channel which encodes covert messages into the TCP acknowledgments (ACKs) is called CLACK. It uses a partial encoding technology to hide covert messages. Its variable network delay between packets and packet reordering technique make it resilient in nature. CLACK is implemented and validated on a testbed.

In [125], the authors introduced ACKLeaks as a combinatorial approach for a covert channel in which covert messages are embedded with TCP ACK type packets from TCP connections (single or multiple). ACKLeaks uses a combinatorial approach to evade content-based detection. It is possible for the same ACKLeaks to exploit the existing connections of TCP. ACKLeaks is also augmented with another approach for WebLeaks. Both approaches are considered the best for information leakage. This is confirmed through extensive experiments for the evaluation of their performance. In [126], another approach is implemented for a timing covert channel using TCP-Script, which embeds a secret message in a normal TCP packet flow, and then TCP feedbacks are exploited. To increase the decoding accuracy, TCP reliability and feedback services are used. The secret message is coded as an array of integers, where each integer is pre-agreed between the encoders and decoders. All the integers are encoded to a burst with TCP data segments one by one, with an encoding period. The two adjacent data packets are separated with an appropriate time interval. This timing covert channel is robust in nature and works against network jitter, packet loss and reordering, but its capacity is very low and easy to get deducted from normal traffic, which makes this approach weaker.

In [127, 128], the authors applied a RSTEG (Retransmission Steganography) method to TCP packets in which all TCP retransmission mechanisms are used: RTO (Retransmission Timeout), FR/R (Fast Retransmit/Recovery) and SACK (Selective ACK). The main idea behind this is not that it invokes the retransmission intentionally on receiving successful TCP segments but that the steganogram is sent as a retransmitted segment in place of user data over the payload field. To avoid detection, RSTEG is kept near the normal traffic to make the retransmission look intentional. Hash functions are used to make covert segments. The only limitation this method faces is that normal networks are not used for the retransmissions.

The authors in [129] showed that 10 fold techniques are used for encoding messages covertly into TCP-based communication; N packets of X flows allow different combinations to be checked to ensure the accuracy. It is an active detection method. It helps in mimicking TCP-based flow packets by issuing ten different encoding decoding techniques for developing covert communication. It uses Enumerative Combinatorics for encoding a hidden message into different TCP flows while only a fixed number of TCP packets are used. The authors in [130] investigated the IP Identification header fields of IP. They examined various IP header fields and their potential for being exploited in the context of the transmission of secret information. The paper presents a software implementation that performs detailed network data packet analysis. It summarises potential detection and prevention techniques discussed in the aforementioned sections.

Table 3.4 illustrates the covert channels in TCP and their strategies.

Author	**Year**	**Strategy**
Chakinala, et al. [122]	2006	A TCP sequencing packets ordering channel strategy is used for covert communication based on mathematical techniques like eigenvalues, permuters and a distance bounded model to achieve message hiding.
Luo, et al. [123]	2007	Cloak named software is developed and implemented that helps in mimicking TCP-based flow packets by issuing ten different encoding decoding techniques for developing covert communication.
Luo, et al. [125]	2008	An approach is used for a TCP-based timing channel, where TCPScript is used to address the shortcomings of encoding data. TCPScript helps in embedding messages to be shared into TCP data bursts while TCPpsilas feedback shows that the chosen technique works well to enhance accuracy.
Luo, et al. [126]	2009	A technique for covertly sending messages between two communication endpoints is presented. The TCP packet ACK flag is used for the purpose. A software tool named CLACK is developed to perform the ACK-based encoding before the covert communication happens.
Mazurczyk, et al. [127], [128]	2010	RESTG (Retransmission Steganography) for TCP is used where a retransmission packet does not reply with the original packet but carries the steganogram.
Luo, et al. [129]	2011	A combinatorial approach is designed in which WebLeaks and ACKLeak are used as two covert channels that are capable of leaking the information of a web session through the acknowledgment traffic. 10 fold techniques are used for encoding messages covertly into TCP-based communication; N packets of X flows allow different combinations to be checked to ensure the accuracy.

Table 3.4: Summary of related work with regard to covert channels in TCP

3.2.3 Covert Channels in UDP

An approach to covert channels in UDP creates a covert message space by looking into the datagram for the presence or absence of the Checksum field. This field is trusted as an option in UDP. The TCP based approaches introduced earlier can possibly be applied to the UDP Checksum field. UDP has 3 fields [130] where secret data can be covertly hidden, which are Source address, Length and Checksum.

The authors in [131] proposed covert channel tunnelling based on format transforming encryption for TCP traffic by using the protected static protocol. This allows TCP traffic to be converted into UDP traffic using a hidden Markov model. This model uses a protocol proxy to avoid side-channel analysis. The proposed framework includes a facility for the conversion from TCP traffic into UDP traffic, observation-based format transforming encryption (FTE) and a detection mechanism.

Table 3.5 shows the UDP based covert channels and their strategies.

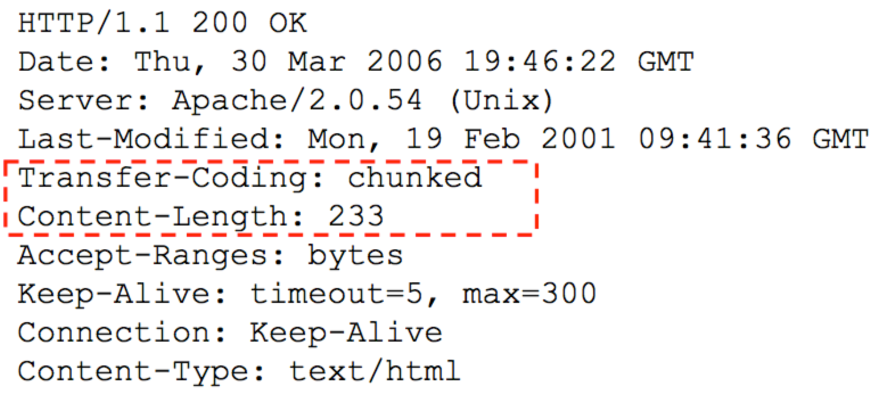
Author	Year	Strategy
Thyer [130]	2008	UDP packets allow embedding or updating to 3 options in UPD headers for covert communication, which are known as Source Address, Length and Checksum.
Oakley, et al. [131]	2020	This technique is based on tunnelling and FTE to covert TCP traffic into UDP traffic.

Table 3.5: Summary of UDP based Covert channels

3.2.4 Covert Channels in HTTP

In [90], the authors introduced an approach to hiding data in HTTP packets. HTTP treats all resultant linear or white spaces or characters (optional line feed [CLRF], spaces [SP] and tabs [HT]) in the header in exactly the same way as it treats a single space character. For example, [HT] is considered as binary 1 and [SP] as binary 0. As HTTP headers never come in any specific order it is possible to embed hidden data in the header's ordering. Even the header's case insensitivity has the advantage of using any capitalization as a header value, which could be used to make a covert channel. Van Horenbeeck [132] presented a tool named Wondjina used to create a tunnel by using the HTTP Entity Tag (ETag) that allows one client to check whether it is locally in a cached current copy. Even a Content-MD5 header is able to transfer up to 128 bits of packet data per HTTP message. Similarly, the approach used in [133] uses the last approach along with an Access-Control-Allow-Origin approach and Content-Location header information. According to [87, 134], the detection for this type of approach checks network communication for compliance with a standard specification used. Any abnormalities and variations to the standard are considered as leakage. For example, Figure 3.1 shows that transfer-coding and content-length fields cannot be sent together. In HTTP, the header chunked transfer encoding is a data transfer mechanism in which data is sent in chunks from source to destination. In such cases, the transfer encoding is used instead of the content-length as the total size of the content is dynamically set and it is unknown.

Table 3.6 shows the HTTP based covert channel detection solutions.



```
HTTP/1.1 200 OK
Date: Thu, 30 Mar 2006 19:46:22 GMT
Server: Apache/2.0.54 (Unix)
Last-Modified: Mon, 19 Feb 2001 09:41:36 GMT
Transfer-Coding: chunked
Content-Length: 233
Accept-Ranges: bytes
Keep-Alive: timeout=5, max=300
Connection: Keep-Alive
Content-Type: text/html
```

Figure 3.1: Transfer-coding and Content-Length fields cannot appear together (Source: [90])

Author	**Year**	**Strategy**
Kwecka [90]	2006	The capitalization of words is used in the HTTP header as a covert channel to hide information.
Van Horenbeeck [132]	2006	An approach is introduced in which covert data is hidden in HTTP/2.
Duncan and	2010	HTTP header fields are used to hide information to cover the HTTP network traffic by making selected arbitrary

Martina [133]		files compressed and encrypted using a stream cipher.

Table 3.6: Summary of HTTP based covert channel detection solutions

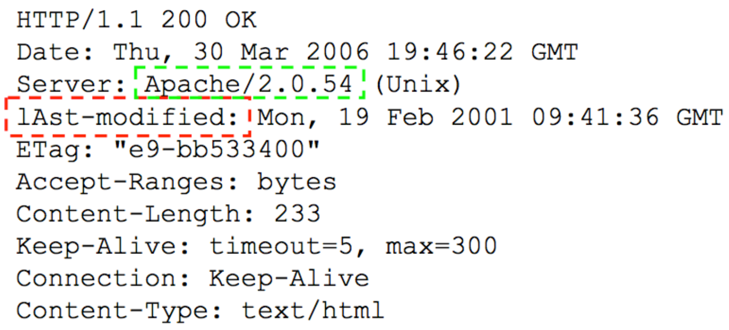
3.2.5 Covert Channels in RTP and RTCP

Paper [135] shows how to create new covert channels using RTP (Real-Time Protocol) and RTCP (Real Transmission Control Protocol) along with delayed audio packet steganography, where 8-bits Padding fields are used. The RTP extension header of a variable length generates random initial values of 16-bits Sequence Numbers along with the 32-bits Timestamp field in the very first RTP network packet. The experiment result shows that covert data can be sent unidirectionally at the speed of 1.3mbs with 96% of the covert bandwidth value.

In [136], another approach is introduced, which uses a SIP (Session Initiation Protocol)/SDP (Session Description Protocol) strategy to create a covert channel during the signalling phase of a VoIP (Voice over IP) call. SIP/SDP has fewer utilized or free fields that always use the network-based steganography. This approach allows 2,000 unidirectional bits to transfer over the covert channel. The VoIP exchange of information is very low as compared to other covert channels in different protocols.

Paper [137] shows a technique used to illustrate how timing channels use timing features for covert communication. It is introduced using the same timing channels but unlike the previous approaches that are constructed on a single transport layer protocol, this technique is based on both RTP and RTCP protocols at the application layer. The features like Run Length Code and Multi Zero Code are used to enhance the robustness and unpredictability features of covert channels.

According to [138], the detection for this type of approach to covert channels compares messages with signatures of known covert channels, which are stored in a database. For example, Figure 3.2 shows the detection of a header as the Apache server only allows the generation of message headers with the upper case. Table 3.7 shows the covert channels in RTP and RTCP and their strategies.



```
HTTP/1.1 200 OK
Date: Thu, 30 Mar 2006 19:46:22 GMT
Server: Apache/2.0.54 (Unix)
lAst-modified: Mon, 19 Feb 2001 09:41:36 GMT
ETag: "e9-bb533400"
Accept-Ranges: bytes
Content-Length: 233
Keep-Alive: timeout=5, max=300
Connection: Keep-Alive
Content-Type: text/html
```

Figure 3.2: Header detection of message headers with the upper case (Source: [90])

Author	Year	Strategy
Mazurczyk and Szczypiorski [135]	2008	Two strategies are used. The first includes applying all the IP, TCP and UDP approaches to hiding data with RTP (Real-Time Protocol) and RTCP (Real Time Control Protocol). The second uses LACK (Lost Audio Packets Steganography) to achieve the capability of hybrid storage timing covert channels for data hiding.
Mazurczyk and Szczypiorski [136]	2008	An approach is introduced in which a SIP (Session Initiation Protocol) strategy is used to create covert channels during the signalling phase of a VoIP (Voice over IP) call.
Lizhi, et al. [137]	2012	This technique illustrates how timing channels use timing features for covert communication. It is introduced using the same timing channels but unlike the previous approaches constructed on a single transport layer protocol, this technique is based on both RTP and RTCP protocols at the application layer. The features like Run Length Code and Multi Zero Code are used to enhance the robustness and unpredictability features of covert channels.

Table 3.7: Summary of covert channels in RTP and RTCP

3.2.6 Data Leakage Analysis

To determine from forensic logs if any unauthorised data is leaked, the following analysis methods are used [139-141]:

- *Univariate Analysis*: In this type of analysis, measurements are obtained for an individual variable or an attribute to measure the overall irregularity. Such analysis methods are often utilised when the value of a specific variable or one variable is involved. They are also used as a prerequisite to ensure that the collected data series is stationary so that correlated patterns within the data can be obtained.

- *Multivariate Analysis*: In this case, measurements are obtained for more than one variable. Techniques such as Principal Component Analysis, Multiple Regression Analysis or Partial Least Squares are used to identify prevailing patterns in the data that are categorised as trends and hence depict outliers that are being monitored. This type of analysis has known advantages to reduce the probability of the Type I error and the capability of decomposing correlated measurements into new sets of uncorrelated measurements that are beneficial for pattern recognition. Additionally, they provide better accuracy in the context of monitoring as leakages can be simultaneous across header fields.

- *Time-Based Reference*: In aggregation with other statistical analysis techniques, a time-based reference analysis method is used to observe the chronology of events. It is an important form of analysis where the header of a packet has no noticeable leakage; however the leakage is based on the delay of packets being delivered to their destination [121, 142].

3.2.6.1 Covert Storage Channel Detection Techniques

Misuse detection is relevant in the context of covert storage channels and their detection is normally performed at the boundary of the network, e.g. firewalls or by intrusion detection systems. Usually, these systems are fully capable of eliminating any covert data on storage channels by the use of traffic normalising devices [143]. For instance, a firewall on the perimeter network could be configured to clear optional header fields of protocols when it is aware that those fields/features are not supported in the infrastructure (for example, the Type of Service (TOS) field in IP headers). Other such cases include the Identifier (ID) field on an IP packet that is used to identify the packet and re-assemble the original message at a destination. The field can be manipulated in such a way that the first byte represents the ID legibly, while the second byte acts as a covert bit carrier [144]. Similarly, it is possible to detect a covert storage channel that uses Do-Not-Fragment (DF) and More

Fragment (MF) to transmit covert bits. If a detection mechanism is in place, then the presence of covert communication can be informed by the MF field set to 1 even when the last fragment of the last message (original message) is less than the packet size and hence should be set to zero.

To detect an ISN value being used as a covert storage channel, an ISN generation model is proposed that predicts an ISN value in a TCP session based on the previous value of ISN generated by the operating system used [85, 145, 146]. It is difficult to generate unpredictable numbers using a computer, as pseudo random number generation functions are fixed algorithms that repeat a number and produce the same result on a different machine that can be used to predict output values. Even if a random number generator function is not known, the generation of the same sequence of numbers based on limited system internal states can be used to create a specific algorithm. Through the use of a neural network, a model can be prepared that can compare a predicted ISN value to the actual value and compute a difference. A large difference can suggest that the value is not set by the original stack and hence is a covert value [147]. As discussed earlier, a Markov model is a technique that is proposed for checking TCP protocol packets to ensure that a packet follows the given rules, and in case it does not do so, the presence of a covert communication is indicated [148]. The known TCP states are defined as states of the Markov model. For various protocols, different models can be generated and used together with the Kullback-Leiblei (KL) [149] method for checking if a data set presents a divergence from valid transmission. KL divergence helps to measure the loss of information when approximation is made.

Zhai, et al. [148] analysed various redundancies in the TCP protocol and proposed a model called TCP Markov Model (TMM) for various application scenarios. The method is different from SVM or Neural Network for the detection of covert channels. This method does not get trained using abnormal data and instead uses actual normal data for discovering potential covertness. Their experiments have shown a good detection performance, but it has been found that if the embedding probability is less than 0.2, the model becomes unstable, i.e. the detection probability decreases as the divergence of normal and abnormal data becomes low.

As mentioned earlier, covert channels can be created using a 1-bit-per-packet noisy covert channel using the Time to Live (TTL) field as suggested in [106, 150]. Zander, et al. [106] proposed improved 1-bit-per-packet covert channel encoding in the TTL field by analysing the initial TTL values and normal TTL occurring in the network. The use of two different starting values of TTL in packets has been suggested in [106], where a high value is deemed as binary 1 and a low value as binary 0. Their technique offers high detection using normal initial TTL and needs to be deployed as an active warden on the edge or perimeter of the network similar to the case in [150], but this technique suffers from low performance about detection. One of various techniques that have been used for compromising active wardens to continue scanning and normalizing the network traffic flow is that of protocol switching. Protocol Switching Covert Channels (PSCCs) transfer covert information by sending network packets with different selected protocols. Wendzel and Zander [151] offered the first detection method for detecting PSCCs. The algorithm distinguishes between normal and covert traffic by studying the maximum rate of PSCC traffic. It is observed that using C4.5 (Decision Tree) low and high rates of PSCCs can be detected. The detection accuracy is 98 to 99% for channels with >= 4 bits/second, but with channels of only 1bit/second the accuracy is 96%. However, with 99% accuracy, the false positive rate of 1% could be problematic as most traffic is marked as normal traffic, i.e. covert traffic goes through undetected.

For covert storage channel detection, another path outside rule-based techniques is AI-based. AI techniques offer the ability to learn the behaviours of various attack vectors and consequently build rules for detecting the attacks or even similar attacks. One of the first data mining techniques is applied to IPv6 in [152] and it uses Fuzzy Logic. The fuzzy logic-based detection is applied to the Network Time Protocol (NTP) protocol and the results show that the system can accurately detect covert data with high accuracy and a low false positive rate of about 2%. However, the experimental data sources are different which may have biased data sets. An a-priori algorithm is used to detect router header misuse attacks with a detection rate of 72% in [153]. The low detection rate could be attributed to the choice of features selected for creating the initial model. These features include host IP, Host Port, TCP Flag and State, Protocol and Requested Service. Saad, et al. [154] also used a fuzzy Logic-based approach to detect ICMPv6 Flooding Attack with high accuracy and a low root mean square error (0.26). The approach evaluated 2,000 records of normal and abnormal traffic using 1,000 to 1,500 pings as flooding ICMPv6 rates. However, the method is unable to detect DoS or DDoS attacks.

Zulkiflee, et al. [155] introduced SVM to detect IPv6 attacks where the features selected for creating the model included Time Interval, Source IP, Source Port, Destination Port and Protocol. For experimentation [155] authors used real network traffic with the average accuracy of 99.95%. Salih, et al. [156] used the Naïve Bayes algorithm to detect IPv6 covert channels. Their research work used ten features for the classification stage that included Traffic Class, Flow Label, Hop Limit, Source Addresses, Hop Limit, Next Header, Payload Length, ICMPv6 Type, UCMPv6 Code, ICMPv6 Payload and Reserve Bit. The detection accuracy reported is 94.55% proving the effectiveness of the Naïve Bayes algorithm. The algorithm is suitable for linear data and is unable to extract patterns from random/pseudo random data sets as they tend to be non-linear/chaotic. The problem with this type of approach is that it identifies a set of features that include non-qualified ones to be used in attack detection. One of these features is packet capturing time that leads to the misclassification of the packet. During model creation, the classifier considers time intervals between packets as a relevant feature to indicate an attack that effectively means if the intervals are outside the specified intervals, the packets are considered as being used for covert communication. This leads to a lot of false positives.

Another method of covert data communication using storage channels is by the use of a packet length. In [157] the authors proposed a scheme for embedding covert data by modulating the length of link layer frames. The method is applied to 256 frame lengths to embed covert bytes. The frame length distribution is random and does not imitate the normal network traffic flow. Another scheme is proposed by Yao and Zhang [158] where the authors used the length of a packet to transfer a covert message. The authors created a covert channel in which a sender and a receiver shared a periodically updated matrix with elements representing unique unsorted packet lengths. The sender, using the bits hidden in the message, determines the matrix row and randomly chooses a packet length from that row. The receiver finds the gained packet length in the matrix and again constructs the bit of the message according to the row number. In both schemes, the sender and receiver share a secret matrix with the cells representing unique lengths. However, as the packet length distribution of the covert channel is not equal to the packet length distribution of normal traffic, this type of covert channel is discoverable. Nair, et al. [159] proposed a detector that analyses the length distribution of datagrams for an application. The data is collected for a sufficiently long time to study the packet length distribution. It is observed that in the time series of IP datagrams, the nature of packet lengths is randomly distributed. As a result of the random distribution, they claim that none of the existing techniques can be used, hence they proposed the creation of a two-dimensional feature space that is used to train a steganalytic classifier, where a packet length vector is determined for a datagram stream to train the classifier. The Fischer Liner Discriminant (FLD) is used to reduce the dimensional space to a single dimension followed by a Linear Discriminant Analysis (LDA) classifier to train a single dimension feature. The experiments conducted with a moderate payload have shown very high accuracy.

Chow, et al. [160] used Kullback-Leibler divergence also known as relative entropy. Unlike the previous work on the computation of an irregularity index based on the Mahalanobis distance, they used the difference between two sets of probability distributions: a theoretical model for the expectation of normal traffic and a real model using the frequency distribution of TCP flags. During detection, the TCP flag frequency distribution of network traffic is computed for each unique IP pair. For the evaluation of accuracy and efficiency, they used real regular data traffic and covert storage channel messages using various encoding schemes. The relative entropy revealed the dissimilarity of a different frequency distribution as compared to a normal profile. It is noticed that when TCP flags are uniformly distributed, the detection of the covert storage channel is simpler but the vice versa can be challenging. Also, the normal profile requires adaptation and possible better granularity for various behaviours.

The approach to detecting payload tunnels, e.g. HTTP tunnelling using behaviour profile, was first introduced by Pack, et al. [161]. The behaviour profile is created using several metrics that include the average packet size, ratio of large and small packets, pattern of changes in the packet size, number of packets sent and received, and duration of connection between a host and a destination. Similarly to [160], if the behaviour diverges from normal HTTP behaviour, it is attributed as an HTTP tunnel, indicating that a covert storage channel is used. A tool is developed by Borders and Prakash [162] for detecting outbound HTTP tunnels using a similar approach to [161]. The tool created a training model by analysing HTTP traffic over some time. Once the training is complete, it is then able to detect any deviations from the normal profile and present results in metrics such as the request size, request irregularity, time between requests, time of the data and outbound bandwidth usage.

Tumoian and Anikeev [147] provided methods that can automatically create ISN generation models for an operating system. They created and trained the model based on nightly runs in order to maintain the system with the ever changing ISN generation principle. Their investigations indicate the detection of covert channels with acyclic connections. They used the NUSHU source code for the creation of a covert channel. NUSHU creates a covert channel by sensing the establishment of a connection and then replacing the kernel generated ISN with a covert ISN. The delta value between the kernel generated ISN and NUSHSU’s covert ISN is saved before sending the packet. When a packet containing ACK returns, NUSHU calculates the original ACK by getting the covert ACK and adding delta to it.

Along with the above, a different dimension about the detection of anomalies is the sheer amount of anomalous data or the amount of data being generated in the network. The existing detection systems are not able to process the data in a perfect way. Furthermore, as traditional methods are solely based on either anomaly detection or deep packet inspection, it is often subjected to raising false alarms such as in [163, 164] where the authors suggested a collaborative method for mitigation by employing alerting correlation and mitigation in multistage workflow that is supported by software defined networking (SDN). SDN is responsible for decoupling the data plane from the control plane and hence able to support access to data on different network layers and change data flows in the network. In [164], the authors presented a multistage covert storage channel detection and mitigation model that takes advantage of SDN. Various elements of SDN, e.g. a monitor, a correlator and a controller, play separate yet complementary roles that help in processing of big data. Their experiment in a simulated network indicated a low false positive rate. However, the experiment did not accommodate or simulate an increase in sequence numbers. Yao and Zhang [158] also used SDN to access critical information about network flows and improved the efficiency and accuracy of detection and mitigation. In this case, the effectiveness is evaluated on a realistic testbed GENI (Global Environment of Network Innovations) as opposed to its predecessors. The overall results are not different from its counterparts.

3.2.6.2 Covert Storage Channel Detection in Cloud Storage

Recent work is extended from the covert storage channel realms to cloud storage services. Hovhannisyan, et al. [165] exposed some vulnerability in de-duplication services offered on cloud based storage, which can be used as a covert channel. The data de-duplication is a mechanism for reducing storage cost by the elimination of redundant data, hence each data chunk is identified with a unique value that is used for comparing and locating duplicate files on the server [166]. Rather than storing multiple copies of a file, the server only stores the original file. To detect de-duplication, there are tools such as transmission time detection in which files already on the server are smaller than the ones not on the server, so the bandwidth consumption would be higher for the files that need to be synchronised with those on the server. In a single-bit covert channel, the sender program resides on a victim’s machine and the receiver program on an attacking machine. If the sender has two files denoted as 0 and 1 and wants to send file 1 to the receiver, then it will send file 1 to the cloud and the receiver will receive it. If the sender wants to send file 0 to the receiver, it uploads file 0 to the cloud. If the receiver finds out that file 1 is being uploaded faster than file 0, then it is possible to infer that de-duplication has occurred and file 1 sent by the sender, i.e. indicating that 1 is transmitted. Similarly, the receiver can infer the sending of 0. The authors also proposed a multiple bit per file method in which a single file can be represented by multiple bits. The coding scheme is tested extensively on different cloud storage systems. Caviglione, et al. [167] analysed various schemes that can be used as covert channels in personal cloud storage services which include volumes-based, throttle-based, timing and storage-based, type-based, timestamp-based, uploading of new files, renaming files, movement of files, size modulation, alteration of file, type of device and modification of folder. As with typical protocol-based covert messaging, in this context headers are changed too. They implemented two methods and tested their performance in terms of bandwidth and robustness. The results indicate the possibility of the communication channel being used as a covert channel. To mitigate the risk, they proposed energy-based and design-based techniques. The research is still in early stages and promises countermeasures in the future.

3.2.6.3 Covert Storage Channel Detection in Wearables

Denney, et al. [168] worked on a new covert storage channel in the context of wearables. According to the authors, a covert sender application can create a notification using a varied range of notification ID numbers as a coded message. A covert receiver application can read an incoming notification and interpret the encoded covert message. Using the notification ID, the actual text can be disguised to increase secrecy. The authors demonstrated the functionality and feasibility of the covert channel and are working on methods to block such a covert message sending technique.

On the other side of the spectrum is the usage of covert storage channels for sending secure data. Singh at al. proposed an approach that uses covert communication to enhance Vehicular Ad-hoc Networks (VANETs) security [169]. The model proposed uses a covert storage channel to communicate secure data while unimportant data is transmitted via an overt channel. It is noteworthy, however, that the suggestion of using the covert channel for useful purposes does not change the fact that the covert channels are dangerous ongoing threats that will need to be closed.

3.2.6.4 Covert Timing Channel Detection Techniques

In [170, 171], the authors discussed the use of delay time between packets for data encoding as a channel for sending covert information. Figure 3.3 shows an example of a high-level design diagram of a covert timing channel. Effectively, this means that attackers do not need to create a new set of data packets and use the time between packets to encode data. In this method, there is no secret information embedded in a header or payload and it is covert timing that is used for encoding messages. The authors proposed a methodology for detecting covert timing channels based on how closely a source comes to achieve a given channel capacity. They used the statistical analysis of inter-packet delays for classifying between regular traffic and the traffic that communicates through the modulation of inter-packet delays [172].

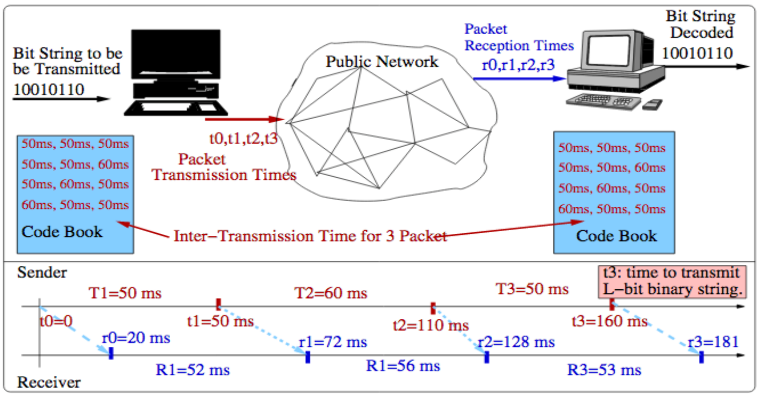


Figure 3.3: High level design diagram of covert timing channel (Source: [173])

As mentioned earlier, the disruptions of covert timing channels add random delays to network traffic, and reduce the channel time capacity and system performance. To disrupt covert timing channels, the authors in [174-176] dedicated their effort towards the design of systems. In [177, 178], the authors eliminated covert timing channels in their system design. More recent research works investigated the design, detection and countermeasures of covert timing channels [121, 179, 180].

Several design issues related to covert timing channels are discussed in [121], including one of the first developments to defend against IP covert timing channels. In their scenario, a machine is compromised and the defensive perimeter (represented by a firewall or intrusion detection system) monitors the communications outside the corporate network. It is noteworthy here that a covert timing channel can be used to pass information through the defensive edge perimeter undetected. The authors developed a detection mechanism by the identification of the regularity of inter-transmission times. In their design, a message sender and a receiver agree upon the length of a transmission interval as *ω* milliseconds (ms). To signal bit value '1', the sender transmits a packet in the middle of the time interval, and to signal bit value '0', the sender remains silent during the interval. This scheme achieves a rate of 16.67 bit/s with a 2%-character decoding error (ω = 60 ms) in an experimental setting with the average Round Trip Time (RTT) between the sender and receiver being 31.5 ms. Their coding scheme limits the data rate achievable for the timing channel and assumes that the sender and receiver have synchronised time. A constant shift in delay and jitter can have a cascade effect that causes subsequent bits to be decoded incorrectly. This method is also used in [126], where the authors have expanded the scheme specifically for TCP packets. The major advantage of the expanded scheme is that when a packet is lost, a bit is flipped but the synchronisation remains unaffected. Further, it is noted that when the pattern of the bits is uniform, the distribution of inter-packet delays is close to the Geometric distribution. This will enhance the channel capacity and decoding accuracy.

In [107], the authors built a device called Keyboard JitterBug that slowly leaks information being typed over the network. JitterBug is an example of a passive covert timing channel, so no new traffic is created to transmit information. Figure 3.4 shows a scenario where JitterBug can be used.

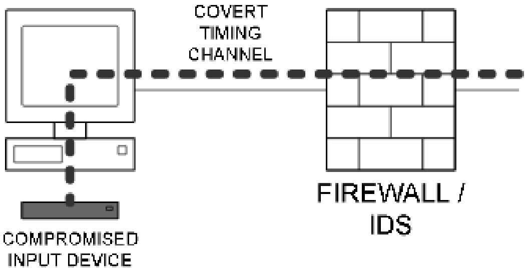


Figure 3.4: Keyboard JitterBug scenario (Source: [180])

In this scenario, JitterBug creates small delays in key-presses to affect the inter-packet delays of a networked application. It transmits a 1 bit by increasing an inter-packet delay or a value that is modulus of ω milliseconds (ms) and transmits a 0 bit by increasing an inter-packet delay to a value that is modulus of $[\frac{\omega}{2}]$ ms. It is noted that when ω is small, the distribution of JitterBug traffic is very similar to normal (legitimate) network traffic. Also, for avoiding patterns in inter-packet delays being multiples of ω and $[\frac{\omega}{2}]$, a random sequence s_i is subtracted from original inter-packet delays before the modulus operation can be carried out. To improve the performance, they used a 4-bit to 1-keypress encoding scheme. As the timing channel rate is tightly bound to human input and is very low, it can be detected only by a sophisticated detection algorithm such as the one in [180].

Broadly, there are two classes of detection tests that are used in the implementation of countermeasures for protection against covert timing channels, i.e. shape tests and regularity tests. In shape tests, the shape of traffic is described using its mean, variance and distribution (first-order statistics). In regularity tests, the regularity of traffic is described using correlations in the data (second-order statistics) and other such statistics measures.

Peng, et al. [181] presented that the Kolmogorov–Smirnov test (KS-Test) is effective for watermarked inter-packet delays in the form of a covert timing channel, which is also discussed in [182]. The watermarked inter-packet delays are shown to have the sum of normal and uniform distributions. The KS-Test can be used to determine if a certain sample comes from an appropriate distribution. In the KS-Test, the distance between the test sample and the training set represents legitimate or illegitimate behaviour. If the test score is small, it shows that the sample is close to legitimate behaviour. If the test score is large, then it indicates the possible occurrence of a covert timing channel. The KS-Test is not dependent upon a specific distribution (it is non-parametric and distribution free), thus it applies to various types of traffic (with different distributions).

In [121, 183], the authors presented a method for detecting a covert timing channel through regularity. This detection determines whether or not the variance of inter-packet delays is relatively constant. The premise is that in most network traffic cases, the variance of inter-packet delays changes over time, whereas in the case of covert timing channels, the variance of inter-packet delays remains relatively constant, assuming that the encoding scheme does not change.

In the regularity test, the standard deviation of inter-packet delays is measured. If the sample has a low regularity score, then it indicates that the sample is highly regular, signalling the possibility of a covert timing channel. In the regularity test, a sample is separated into sets of ω inter-packet delays and then for each set, its standard deviation is computed. This method is used to detect the network covert timing channels and increase the bandwidth of the covert channel.

In [180], the authors introduced a new entropy-based approach that can be used for various covert timing channels. The authors observed that the creation of a covert timing channel has an effect on the entropy of the original process, which means a change in the process entropy provides a clue for detection. By entropy, they mean a measure of complexity or regularity that can be measured using the average entropy per random variable. The authors used this observation, made use of entropy and corrected conditional entropy for the detection of covert timing channels. In their experiments they can detect the covert timing channels with abnormal regularity using the corrected conditional entropy. Their experiments also show that using entropy they can detect various covert timing channels including those with a distribution close to that of legitimate traffic (for example JitterBug), which could not be detected using the previous detection methods.

The method used in [121] for protection against covert timing channels is referred to as ε-similarity. In ε-similarity, the measure of the proportion of similar inter-packet delays is monitored. Clusters are prepared based on the centroid of similar packet delays at multiple time intervals. Although this technique has the advantage of limited access to an application where inter-packet delays are expected to be large (namely in [37]), it would not serve as a technique for generic purposes as offered by

[180, 183]. Another statistical measure is introduced in [179], where a mean-max ratio is computed to test bimodal or multimodal distributions that could be introduced by binary or multi-symbol covert channels. The assumption in [179] is that traffic with inter-packet delays follows a normal distribution and hence mean-max ratio ≈ 1, while legitimate traffic does not follow a normal distribution.

In [184], the authors proposed the usage of multiple signal processing techniques to distinguish adversaries from benign workloads. They suggested that when two processes are communicating and creating conflict misses, they are taking cache spaces from each other and thus can be negatively correlated. By filtering the non-negative correlated part of these processes, the repetitiveness of cache occupancy can be traced, leading to cache timing channel detection.

Various detection techniques for covert channels have various costs in terms of processing power and sensitivity in their execution. Protocol-based detection is simple to implement and has a low processing power requirement. However, it can only detect badly written implementations. Signature-based detection is more sensitive and will raise notifications when protocol implementation signatures do not match. The processing requirement for signature-based detection is higher than protocol-based detection. However, it is not as high as the behaviour-based detection of malicious packets, which is most sensitive and efficient but has a very high false positive rate.

According to [185, 186], the detection techniques are standardised and a mandatory field is used to hold a payload. For instance, the TCP header’s sequence number field gets a random ISN and the SYN flag is set to 1 to initiate communication with a server. Upon the completion of a three-way handshake, the sequence number is incremented by the length of data and so on until the FIN flag is set to close the connection. To use an ISN as a message, its sender can create the ISN to be the message that needs transmitting. Such a case of using a message as an ISN is impossible to detect. It can be prevented by the use of proxy servers. However, most corporate networks do not use proxy servers and instead reside behind a firewall. This is a problem that needs addressing.

3.3 Comparison of Existing Techniques

Based on the existing literature [30-32], covert channel detection techniques are categorised into pattern, machine learning and statistics-based approaches. The pattern-based approaches have a common denominator of matching a captured packet header against a known pattern to identify if the packet has covert data in the header [33]. These techniques provide the high accuracy and speed of detection but suffer inability to cope with new threat vectors, incomplete data sets and high maintenance requirements. The machine learning-based approaches [30-32, 34] use a set of rules created from complex data sets for classification. These methods overcome some shortcomings of the pattern-based approaches, improve the ability to cope with incomplete data sets and further offer insights into latent information within the data sets. However, the benefits are offset by the shortcomings of large system footprints, low speed of detection and high maintenance requirements. Finally, the statistical approaches [35] measure deviations from expected stochastic behaviour, which are capable of handling incomplete data sets. Additionally, these approaches offer the speed of detection and ability to detect new threat vectors. The main shortcoming of these approaches is the possibility of false positive results.

Table 3.8 provides an analysis summary of these covert channel detection techniques together with their benefits and limitations.

Techniques	**Concept Definition**	**Sub-techniques**	**Benefits**	**Limitations**
Pattern-Based	Scoring of captured data using matching patterns	N/A	• High accuracy • High speed of detection • Easy implementation	• Unable to cope with new threats • Partially autonomous • High maintenance requirement
Machine Learning-Based	Classification of complex data sets using training models based on partially	1. Partially observable Markov decision process 2. Clustering	• Tolerant of incomplete data sets • Ability to handle complex	• Low speed of detection • Have a large system footprint • High

	complete data sets	and outlier detection 3. 3. Bayesian technique	multivariate data sets • Ability to assess and learn from data • Offer latent insights	maintenance requirement
Statistical	Methods that measure deviations from expected stochastic behavior, i.e. they are suitable for the cases where observations are not exactly reproducible	1. Guilt model 2. Univariate methods 3. Multivariate methods 4. Time based reference methods	• High accuracy • High speed of detection • Ability to detect new threats • No initial requirement for a highly trained stochastic model for defining system boundaries • Offer deviation-based scoring	• Difficult to create thresholds • Possibility for the Type I and Type II errors

Table 3.8: Summary of data leakage detection techniques

3.4 Tools and Technologies

Data Leakage Prevention (DLP) solutions are designed to detect data breach incidents/cases in a timely fashion [187]. They involve the identification, monitoring and protection of three specific groups of data:

1. Data At Rest (also sometimes referred to as Risk) (DAR) – DAR refers to the data that is recorded and stored in filing systems, persistent storages (databases) or other mediums [97]. The data can be considered secure only if the following conditions are met:

   a. Encryption has been applied to the medium where data is stored. This encryption is required to be strong so that any brute force attack can be averted.
   b. The key that is used to decrypt the data, is not stored on the same medium and not present on other nodes associated with the medium. The key is random and of a sufficient length required by the strong encryption algorithm used.

      (Failing the above conditions, the data in question is susceptible to leakage and hence vulnerable).

2. Data in Use (DIU) – DIU refers to the data that is not in the DAR state, i.e. it is being used on a node and hence could be in memory, cache, etc. [97]. Such data can be considered secure only if the following conditions are met:
   a. Access to the memory/cache is controlled. For instance, the process that adds the data to a cache is the only process that can access that part of the cache.
   b. No matter how the process ends, e.g. termination of the process tree or even shutdown of the computer, the data cannot be retrieved from the memory or cache other than by going back to the DAR state.
3. Data In Motion (DIM) – DIM refers to the data that is transferred between two nodes on a network [97, 98]. Such data can be considered secure only if the following conditions are met:
   a. Both receiver and source nodes are capable of protecting the data to the level as mentioned in the above two data groups.
   b. The communication between the nodes is identified (the user is known), authenticated (the user’s identity is confirmed using a password, etc.), authorised (the user is allowed to access the data in question) and private (eavesdropping is not possible).

Considering the enormity of the threat of data leakage, some researchers have considered the matter of detection and leakage. The existing research is mainly categorised into content and behaviour-based methods.

The content-based approach includes rule and classifier-based methods. A rule-based method offers various rules that are defined as the words and terms that can appear in data. When used to protect against information leakage, these rules determine the confidentiality level of the terms appearing in the data to inform whether an alert need to be raised. Various research papers [188-191] have discussed this approach, including commercial products from Symantec (https://www.symantec.com/products).

The classifier based methods consist of various classification and machine learning techniques such as Support Vector Machine (SVM) [192, 193] and naïve Bayes [190, 194]. In this case, the data is represented as vectors with the terms and their frequencies being their features. These vectors are used to create a learning set/training model for the probabilistic classification of the data in the network to determine if it is being leaked or not [34].

The behaviour-based approach for the detection and prevention of data leakage emphasises anomalies in behaviour. The anomalies can be in certain communications or throughout their organisation. Existing studies have proposed the use of decision trees to categorise illegitimate traffic.

However, the aforementioned approaches fail to block leakage in the case where part of the data being shared is already published and only a small part of the data is marked for confidentiality. The reason for the failure to detect such leakage could be the following:

- Rule-based systems are binary in nature and hence too rigid. They tend to suffer from high false positive rates. In statistics, when performing multiple comparisons, a false positive ratio is the probability of falsely rejecting the null hypothesis for a particular test. The false positive rate is used to measure accuracy for a test where the result of packets containing covert data is presented with a degree of threshold probability. The false positive rate is calculated as FP/FP+TN, where FP is the number of false positives and TN is the number of true negatives. Organisations set rule-based systems with high thresholds to reduce the false positive detection. It is due to these high false positives that rule-based systems are not very common commercially.

- Classifiers such as SVM and naïve Bayes also fail to detect the leakage, as these are statistical approaches. All statistical systems work on most significant features when creating training models.
- Behaviour-based models are not used in such cases as information is sent to an intended recipient and the majority of information is published.

In the past 19 years, research and development in the area of network steganography have expanded, resulting in many techniques and tools being produced. Table 3.9 shows a list of the most cited tools [195], followed by their brief descriptions.

Tool	Year	Description
hCovert	2005	Covert channel using HTTP GET requests between webservers
VoVoIP	2007	Embedding data in PCM voice traffic exchanges
SteganRTP	2007	Uses RTP of VoIP as payload medium
Gary-World Team's	2008	Covert channel projects using TCP and IP headers [88]
Steganography Studio	2009	Training suite on network steganography tradecraft
NetCross	2010	Utilizes the DNS protocol to establish covert comm.
OpenPuff	2011	Multiprotocol embedding toolkit
SoCat	2013	Network relay transfer between two independent data channels

Table 3.9: Network Steganography Tools and Techniques [88]

hCOVERT: It is a steganography communication tool used to create a covert channel using an HTTP GET request to convey its message to a webserver and webserver log parsing to retrieve the message. This tool can send and receive messages [http://druid.caughq.org/]

VoVoIP: The Voice over Voice over IP tool is a proof of concept attack which demonstrates a new type of VoIP threat, a VoIP covert channel. With VoVoIP, a hidden conversation can be embedded and further compressed into regular PCM-based voice traffic (i.e. G.711 codec) [http://voipcc.gtisc.gatech.edu/].

SteganRTP: It is a steganography tool able to establish a full-duplex steganography data transfer protocol utilizing Real-time Transfer Protocol (RTP) packet payloads as the covert medium [196].

Steganography Studio: This is a tool to learn, use and analyse key steganography algorithms. It implements several algorithms highly configurable with a variety of filters. It can be downloaded from: http://stegstudio.sourceforge.net/.

NetCross: It is a tunnelling software tool particularly useful in restricted (read firewalled) network environments, which is able to establish IP tunnels exploiting Domain Name Resolution requests/responses. The current implementation is quite unstable and mostly intended for testing and research purposes [https://sourceforge.net/projects/netcross/].

OpenPuff: This is a professional new steganography tool that supports many carrier formats for images, audio, video, flash-Adobe and Windows executables authored by Cosimo Oliboni [https://embeddedsw.net/OpenPuff_Steganography_Home.html]:

- Images (BMP, JPG, PCX, PNG, TGA)
- Audio (AIFF, MP3, NEXT/SUN, WAV)
- Video (3GP, MP4, MPG, VOB)
- Flash-Adobe (FLV, SWF, PD

SOCAT: It is a multipurpose relay (SOcket CAT) and command-line based utility that establishes two bidirectional byte streams and transfers data between them. The life cycle of a socat instance typically consists of four phases.

- In the **init phase**, the command-line options are parsed and logging is initialized.

- During the **open phase**, socat opens the first address and then the second address.
- In the **transfer phase**, socat watches both streams' read and write file descriptors via select (). When data is available on one side and can be written to the other side, socat reads it, performs newline character conversions if required, and writes the data to the write file descriptor of the other stream. It then continues waiting for more data in both directions.
- When one of the streams reaches the end, the closing phase begins.

Based on the literature review in the previous sections, it can be concluded that existing solutions do not meet new threat vectors and incomplete data sets and incur high maintenance requirements of a covert data leakage monitoring framework. Furthermore, it is identified that the most common weakness of existing approaches is incapability to detect new attack vectors, particularly when the attack vectors are masked, i.e. the data leakage is via some fields in network packet headers where data can be non-linear [107, 120, 175].

Pattern and machine learning-based methods have failed in the cases where there is no fixed pattern and classification of non-linear data as discussed in [107, 176]. Statistical approaches do offer some degree of effectiveness, but again similar to machine learning-based approaches, they have failed due to the chaotic nature of non-linear data leaked via network packet headers [78, 177]. Overall, this is difficult for existing techniques to differentiate between normal data and the data masked to appear as normal data. Additionally, it is observed that various techniques proposed in the literature fail to cope with the diversity of fields in the TCP header. Due to this inability, they are not suitable for managing new unknown threats [176].

3.5 Summary

This chapter has presented the related work on deep packet inspection and deep content inspection. Existing data leakage approaches and available detection techniques for network protocols have been discussed based on the existing literature to establish the effectiveness of the current data leakage detection and prevention. More specifically, covert channels have been analysed in terms of their involvement in the transport control protocols including UDP, HTTP, RTP and RTCP. Existing detection techniques have been evaluated with regard to their strengths and weaknesses. The findings from the review confirm that a novel framework is required in order to rectify the weaknesses of the existing techniques to more effectively detect data leakage through covert channels in TCP packet headers.

Chapter 4 Covert Data Monitoring Framework and Statistical Detection Model

4.1 Overview

This chapter presents the detail of a proposed covert data monitoring framework for TCP header fields involving network steganography. This includes a high-level overview of the framework and its components. The monitoring and detection phases of the framework are then described, including a process used to overcome the limitations of existing approaches, an effective and efficient outlier monitoring method for covert data detection, and an algorithm used to create and maintain thresholds for the quantification of outliers.

4.2 Proposed Covert Data Monitoring and Detection Framework

It is clear from the previous chapters that the existing approaches lack the capability to guard against covert data leakage within the realms of network-based communication. Therefore, there is a need for a more capable covert data monitoring solution that can cope with the challenges and uncertainty presented by the chaotic nature of data in a complex network communication environment. In particular, a novel scheme is required to maintain thresholds for non-linear chaotic random natured data, and a viable approach is needed to analyse irregularity in an accurate and durable way to handle a tolerable timeframe.

To address the above need, a novel covert data monitoring framework is proposed here to specifically monitor complex non-linear chaotic random dynamic data with a degree of uncertainty. It focuses mainly on protection against data leakage over covert channels. The proposed framework also focuses on the detection of both deliberate and covert data leakage and endeavours to notify the detected incidents to relevant authorised parties. The proposed framework uses statistical measures to detect covert data leakages more efficiently with adaptable threshold values.

The framework is a hybrid based on various components that operate and reside on the hosts protected against data leakage. There is a degree of collaboration between various host components in order to ensure that the framework's threshold profiler remains updated autonomously. Furthermore, it is designed as a self-contained solution with a small system footprint, while ensuring that its detection speed remains high and the results remain effective. The framework uses a statistical approach for monitoring, detection and decision-making. The framework resides on the top of an operating system and has the capability to obtain the data from the network layer and utilise operating system functions such as notifications as shown in Figure 4.1.

The proposed framework operates by monitoring incoming network data in real-time and comparing it with the threshold profiles managed by the threshold profiler component. The threshold profiles are designed specifically to cope with dynamic non-linear chaotic random data and they are calculated using a novel threshold calculation/adaptation algorithm. It is by comparing the real-time data against the threshold values that a small modal change in header field values can indicate a leakage, which also takes a small amount of time. The framework is designed to refine the threshold profiles periodically using a threshold adaptation algorithm. This adaptation allows the framework to evolve and also ensures that unknown data leakage strategies and attacks can be combated.

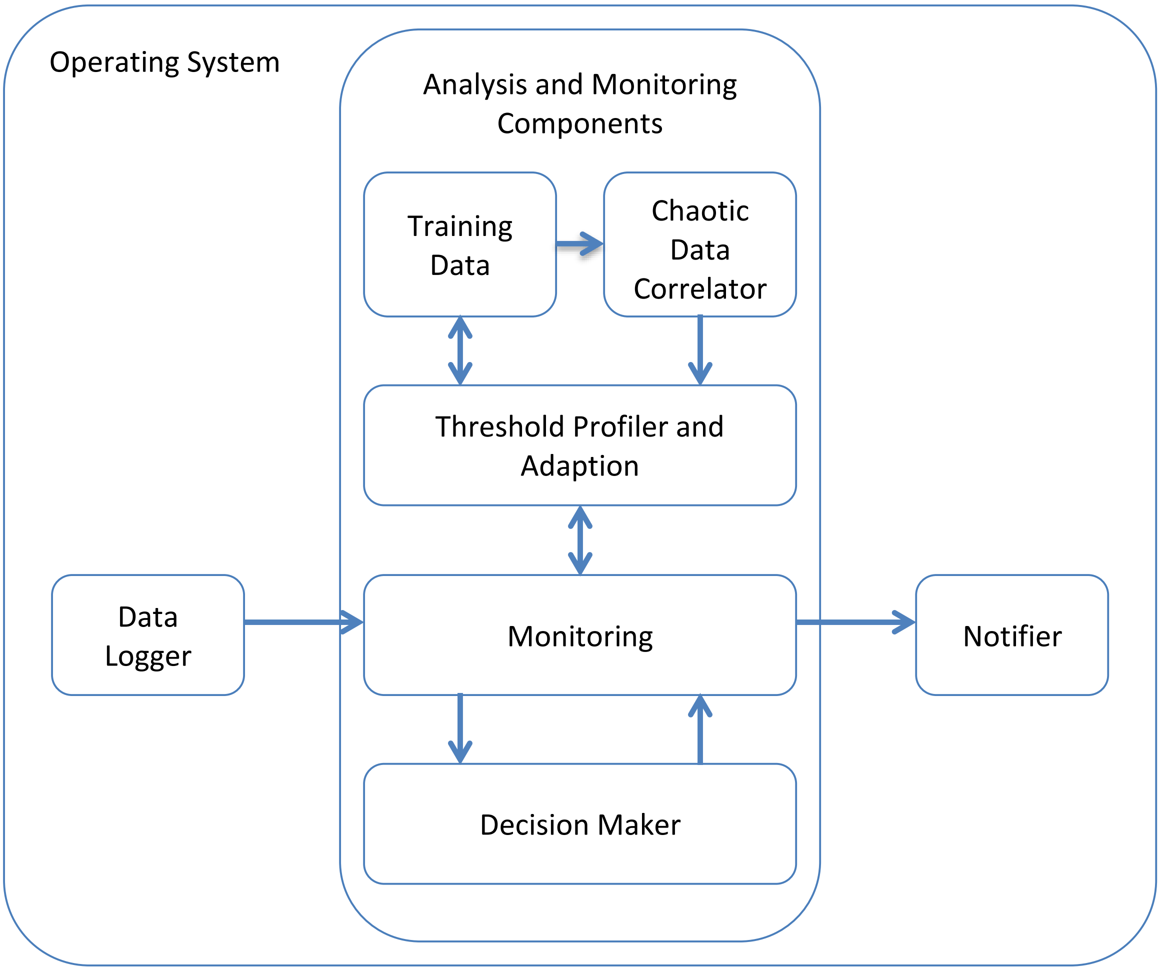


Figure 4.1: High Level architecture of the proposed covert data monitoring framework

The framework is so distinct where not all deviation scores derived by the decision-maker component are flagged as data leakage incidents but instead, these are further analysed using a novel approach to the quantification of skewed results, which informs the probability of data leakage. The process uses an approach to moving average for decision making about data leakage, which helps to increase the level of accuracy and reduce the likelihood of the Type I and Type II errors.

As part of the proposed framework, a simple scenario is assumed where a target/victim machine or any single machine in the network of any size is infected with malware which establishes an external covert link using TCP, collects confidential files, divides them into small pieces and blends them into normal network traffic for transmission to a malicious recipient. It is also assumed that the proposed framework is installed at the external link of the network.

4.3 Covert Data Detection Framework Design

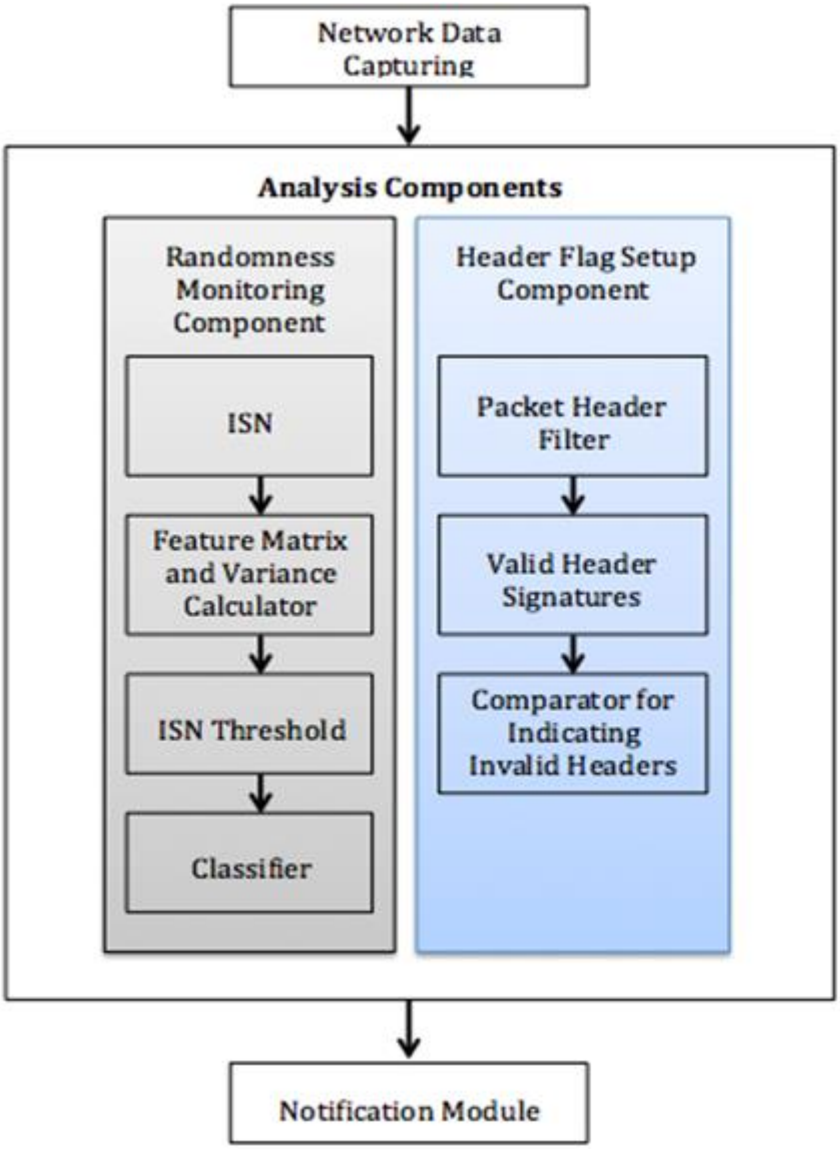

Network Data Capturing
Analysis Components
Randomness Monitoring Component
ISN
Feature Matrix and Variance Calculator
ISN Threshold
Classifier
Header Flag Setup Component
Packet Header Filter
Valid Header Signatures
Comparator for Indicating Invalid Headers
Notification Module

Figure 4.2: Framework components and interoperations

Before explaining the details of various techniques involved in the proposed covert data-monitoring framework, it is essential to understand various components of the framework itself. This section explains the overall design of the framework by referring to Figure 4.2 that illustrates its components and interoperations for covert data leakage detection. These components are explained below:

1. ***Network Data Capturing***: This component is responsible for collecting different protocol data, e.g. TCP/IP packets. It collects all packets irrespective of the protocol used and then filters them to TCP ones. Once the packets are collected and filtered, their header values are selected, including *Initial Sequence Number (ISN)*, *Reserved, Options, Padding, Type of Service, Flags* (e.g. *URG, ECE* and *ECN*) and *Fragment Offset* with their details given in Table 4.1.

Field Name	**Description of Field**
Sequence Number	This field is used to set a number for each TCP packet so that the TCP stream can be sequenced properly.
Acknowledgement Number	Upon receiving a packet, this field is set in a reply to indicate that the packet is received successfully.
Data Offset	This field indicates the length of a TCP header and points to where the data part of the TCP packet begins.
Flags	The following describes the function of the flags when the field is set to 1: CWR: Sending host responded in the congestion control mechanism. ECE: ECN – Echo URG: Urgent Pointer field is used. ACK: Acknowledgement field is valid. PSH: Segment requests a push. RST: Reset of the connection. SYN: Synchronization. FIN: No more data from the sender.
Window	Size of the sender's receive window.
Checksum	Indication to check if the header is damaged.
Urgent Pointer	First urgent data byte in the packet.
Options	Various TCP options.
Data	Upper layer information.

Table 4.1: TCP header value illustration

The data is collected using tools such as WinDump [197] that provides functionality for filtering out various fields of the header. For example, extracting*ISN* provides packets with ISNs indicated by the *SYN* flag set to 1. The captured packets provide all the data needed by the Analysis Components, e.g. unused header bit fields, modification of header fields and header fields where random numbers are required.

2. ***Analysis Components***: The purpose for this set of components is to offer different types of analysis that can be applied to captured-network data. The components are enumerated as follows:
    a. *Randomness Monitoring Component*: In TCP/IP communication, TCP sequence numbers are used to coordinate the transmission and receiving of data reliably. The SYN flag when set to 1 indicates that a new ISN has to be created. When SYN is clear, the accumulated sequence number of the first data bytes of the packet for a given session is indicated. The sequence number in the TCP header comprises a random initial number that is used for the communication between two network nodes. It is mandatory and has a maximum value of 2^{32} bits. To ensure the integrity of the TCP/IP connection, every stream is assigned a unique random sequence number. This is done so that attackers may not be able to perform blind spoofing (a practice where ISN can be guessed) and change the data integrity of a packet. However, these random unique sequence numbers can be generated out of certain data, which can be used as ISNs to make the sequence number field in the TCP header as a covert channel to cause data leakage through unconventional methods. The purpose of this component is to provide features for checking the randomness of the TCP header ISN field to prevent ISN-spoofing attacks using the following subcomponents:
        i. *ISN*: The purpose of this subcomponent is to extract ISNs from the packets collected. Its embedding dimension is equal to the number of dimensions used to create an ISN threshold. A delayed coordinate method is used to compute those dimensions and a vector representing an ISN. This component has been implemented using two different methods, first-order differential equation and discrete legendary polynomial.
        ii. *Feature and Variance Calculator*: The purpose of this component is to create a feature matrix by calculating the distances between a given vector and every other vector. This is followed by a statistical calculation of variance within the population of training vectors.

   iii. *ISN Threshold*: A threshold value is computed in order to create a boundary for anomaly detection.
   iv. *Classifier*: The purpose of this subcomponent is to use the ISN threshold and a given vector to determine whether the vector is an outlier or within the normal population.

b. ***Header Flag Set-up Component*:** Several TCP/IP header fields are reserved for the future use or remain unused during the whole communication process, e.g. *Reserved*, *Options* and *Type of Service*. These fields can be used as holders for encoded secret data. Overall there are 64 possible combinations for a 6-bit flag field out of which 29 are valid and others remain invalid [198]. For example, an *Urgent Pointer* field (a 16-bit) in the TCP header becomes redundant when the *URG* flag is not set [199]. Similarly, Fisk, et al. [200] identified an *RST* flag for data hiding. Zhao et al. [45] indicated that when IP packets are fragmented, *Fragment Offset* informs a receiving machine wherein a particular fragment of the overall message can be located. However, when *DF* (Do not Fragment) Flag is set, *Fragment Offset* can be used for sending secret data. The purpose of this component is to provide features so that invalid header flags can be identified using the following subcomponents:

   i. *Packet Header Filter*: The purpose of this subcomponent is to filter the header part of a packet for analysis. The packet consists of a header and a payload. The solution proposed in this thesis only focuses on headers for covert communication.
   ii. *Valid Header Signatures*: This subcomponent is a persistent store for valid header signatures against which the filtered header packets are compared.
   iii. *Comparator for Indicating Invalid Headers*: The purpose of this subcomponent is to compare collected headers against the valid header signature database.

The leakage analysis using the two different components is conducted at the packet level, i.e. using a time-delay method (where it employs KS-Test or

Regularity Test techniques for measuring dispersion), an initial sequence number randomness method (where it employs a phase space reconstruction technique for finding random number outliers) and a flag setting method (where it employs a technique for finding valid flag settings).

3. ***Notification Module***: The purpose of this module is to raise an alert(s) when a data leakage case is detected. The alerts are logged with a responsible set of users notified using a mail plugin.

The framework design described above is a generic representation of how the various components fit together to form a complete unit for data leakage detection.

4.4 Covert Data Monitoring Framework Operation at Runtime

This section provides a detailed explanation of the various components of the proposed framework at runtime. Their operations are categorised into six different phases that are illustrated with different colours in Figure 4.3. Note that the flowchart attempts to present a success time scenario and potential consequences but it does not depict cases of failure at runtime.

Trainer Phase (Green)

It is part of the training phase. During the initial trainer stage, the framework initiates the computation of an optimal number of dimensions to find a correlation between various points of the data series. Note that this phase is only triggered once for a data type in the ISN field and no subsequent computation is required unless a data type changes. The training data is computed from a single set of series that is transformed into multiple dimensions using the delayed coordinate mechanism.

Threshold Calculator (Blue)

It is also part of the training phase. During the threshold calculation stage, the system collects the network data and stores them in the persistent storage. Note that it is presumed that the data collected at this stage is free of any covert data leakage activity. To calculate a threshold, an iterative process is required that would consume the trainer data and compute the threshold value.

Threshold Value (Grey)

The threshold value is stored in the persistent storage. Note that the threshold value can be computed by Threshold Calculator using the trainer data or adapted via the Adapt Threshold phase as a result of outlier detection.

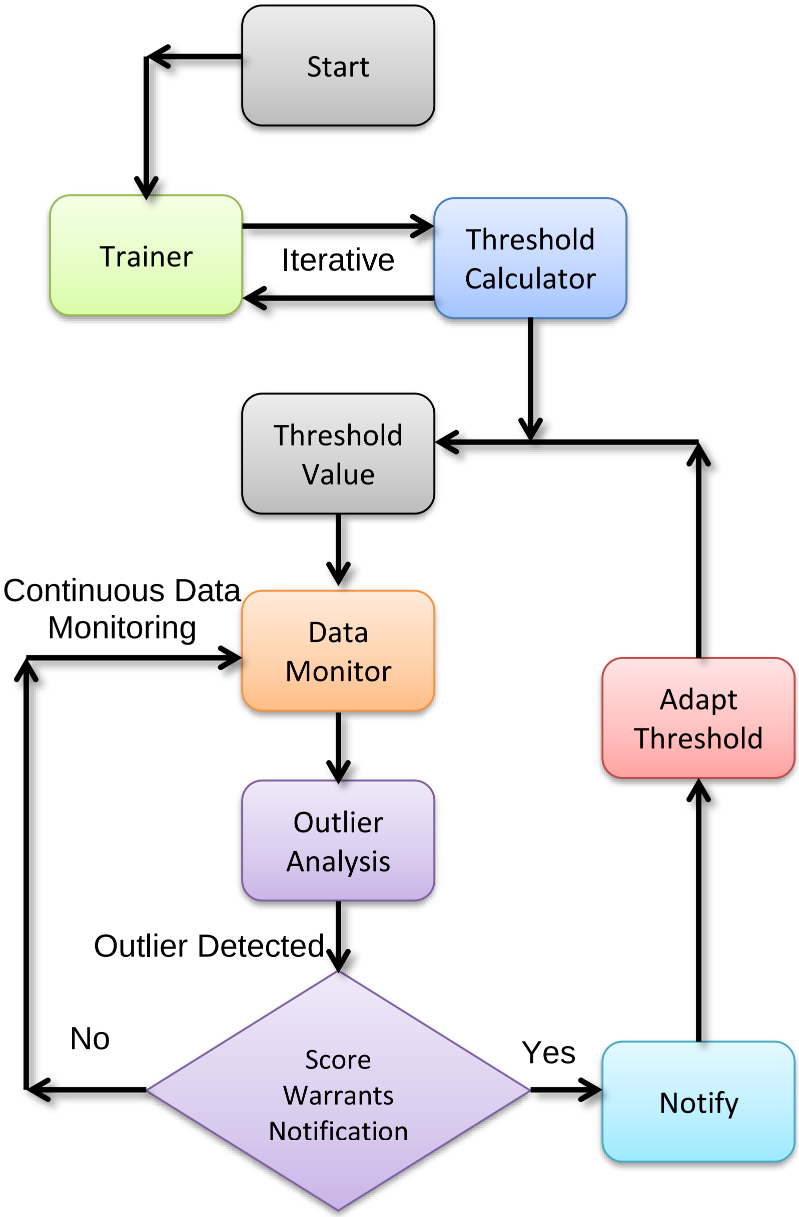


Figure 4.3: Covert data monitoring framework at runtime

Data Monitoring Phase (Orange)

This phase of the proposed framework is where the real-time monitoring of TCP packets is conducted. The monitoring includes the comparison of a computed third-order feature variance value to the threshold value from the previous phase. Note that during runtime this phase does not require lots of hardware resources as the comparison is conducted with a small buffered set of incoming TCP packets.

Outlier Analysis Phase (Purple)

Once the data monitoring phase raises an alarm regarding a breach of the threshold value, the statistical outlier analysis phase is started. This phase uses deviation to calculate the amount of movement in the third-order feature variance value and computes a moving average value by comparing it to historic values computed over the last four periods. Only if the average value is skewed towards a breach, then the notification phase begins, and otherwise, the alarm is recorded but no notification event is raised. This is to reduce the number of the Type I and Type II errors.

Notification Phase (Cyan)

The notification phase is an alert system that proactively raises an alarm and also records the data leakage breach. It is one of the requirements for the covert data-monitoring framework, which is not available in the related work. Notification methods include calls to low-level functions of the operating system to take remedial actions and disable the network interface as well as functions to send alert emails to the system administrator.

Threshold Adaptation (Red)

The threshold adaptation phase includes threshold recalculation if the amount of skew found during the outlier analysis is larger than the expected. The adaptation forms a closed-loop whereby the threshold value can be recomputed proactively to increase the likelihood of detecting outliers and improve the overall accuracy of the framework. The newly adapted threshold value is stored in Threshold Value's persistent storage.

4.5 Proposed Algorithm for Folding Non-Linear Chaotic Random Data

As described in [85], the ISNs generated by operating systems by using Linux 2.0 ISN generator. Which are pseudo-random numbers that have initial relation to ISNs generated in the past. To forecast a future value in a series, several methods are available, e.g. trend curve based prediction for linear systems as well as logarithmic regression and Auto Regressive Integrated Moving Average (ARIMA) [85] for non-linear data. ARIMA is used to compute the number of initial values (dimensions) that are used to compute a future value. In this case, ARIMA is employed to compute the embedding dimension that is the number of past values used to predict future values so that the next ISN value can be calculated and compared to the ISN value within the packet. The comparison is critical because if the values are different beyond the threshold value, the packet is considered as an outlier.

The coefficients of the ARIMA model are used as an embedding dimension for the conversion of non-linear random chaotic data to a vector. The summary of the procedure is presented in Figure 4.4 with its details given in the sub-sections below.

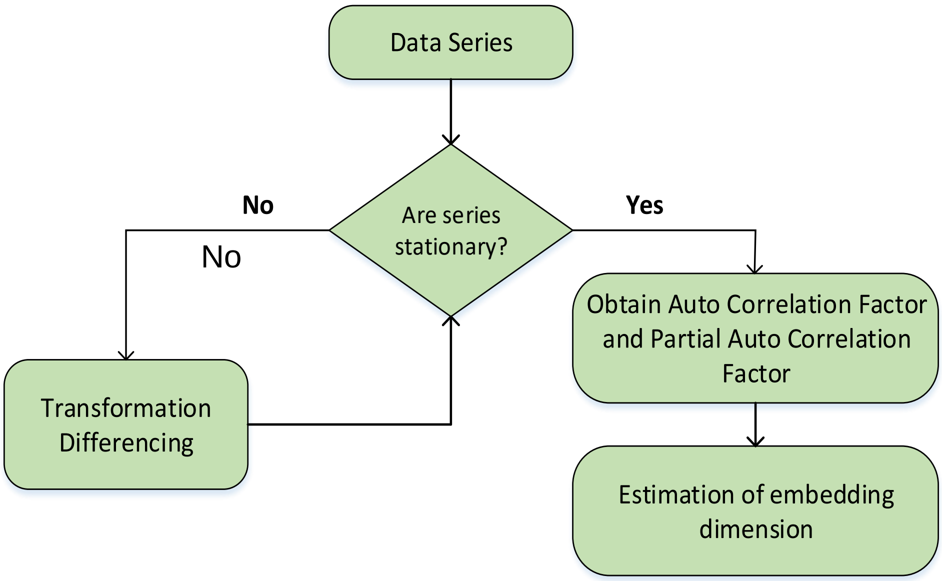

Data Series
Are series stationary?
No
No
Yes
Obtain Auto Correlation Factor and Partial Auto Correlation Factor
Transformation Differencing
Estimation of embedding dimension

Figure 4.4: Flowchart for determining the number of folds / parameters for correlating non-linear chaotic random data using ARIMA – Training Process

4.5.1 Conversion of data series into stationary series

To compute the ARIMA model [201] parameters and the embedding dimension, it is essential to ensure that the data series concerned are stationary. Note that ISNs appear to be independent, but dependent on past values [85] and the stationarity of the series is the way to model the dependence. Stationary series mean that statistical properties such as mean, variance and autocorrelation are constant from period to period. Note that in non-linear random chaotic data, it requires several passes before the series becomes stationary. Generally, there are two types of non-stationary series: one with a non-constant mean and one with non-constant variance. If the mean is not constant then the series is made stationary using a different process, whereas if the variance is not constant, then power transformation is applied to the series. Applying power transformation removes a shift from data distribution, making the distribution normal while applying differencing removes a systematic time-dependent structure. The proposed solution does not consider applying power transformation to retain the data distribution without removing the time-dependent patterns from the data, thus retaining the data width (not transforming it into a normal distribution). That’s why Figure 4.4 only shows the application of the differencing method.

4.5.2 Obtaining the Auto Correlation and Partial Auto Correlation Factors

Auto Correlation Factor (ACF) means similarity in a series based on time lags, while Partial Autocorrelation Factor (PACF) means conditional correlation. Both ACF and PACF are used for predicting parameters of the ARIMA model. If ACF or PACF shows a damped sinusoidal feature, then no order is given to parameters. Otherwise, if ACF and PACF cut off at a certain time lag, then the cutting value forms the value of the parameter. By looking at the ACF and PACF plots of the different series, it tentatively identifies the numbers of AR and/or MA terms that are needed. Autocorrelation and partial autocorrelation are measures of association between current and past series values and indicate which past series values are most useful in predicting future values [202]. With this knowledge, it is possible to determine the order of processes in an ARIMA model. More specifically,

- ACF: At lag k, this is the correlation between series values that are k intervals apart.
- PACF: At lag k, this is the correlation between series values that are k intervals apart, accounting for the values of the intervals between.

4.5.3 Estimation of Embedding Dimension

As mentioned in Section 4.5.2, the cut-off at a certain time lag forms the embedding dimension that is used as the number of dimensions required for each vector representing a non-linear random chaotic number.

4.6 Statistical Algorithm for Maintaining Thresholds

This section discusses the components required for the statistical algorithm that is used for maintaining thresholds.

Figure 4.5 illustrates the process of maintaining thresholds. This is a highly critical case as it allows corporate data to be leaked without even raising alerts. Furthermore, it is a noiseless storage covert channel for data leakage and hence it can very quickly allow the receiver side to collate data. Indeed, this flaw needs to be patched as it has far-reaching implications.

According to [203, 204], it is difficult to generate an unpredictable number using a computer. The reason for this is that computers are designed to strictly execute a defined set of commands in a repeatable and accurate way. A fixed algorithm is used to produce the same result on a different computer that can hence effectively predict output values (provided that the internal state of a remote system is accurately reconstructed) [203].

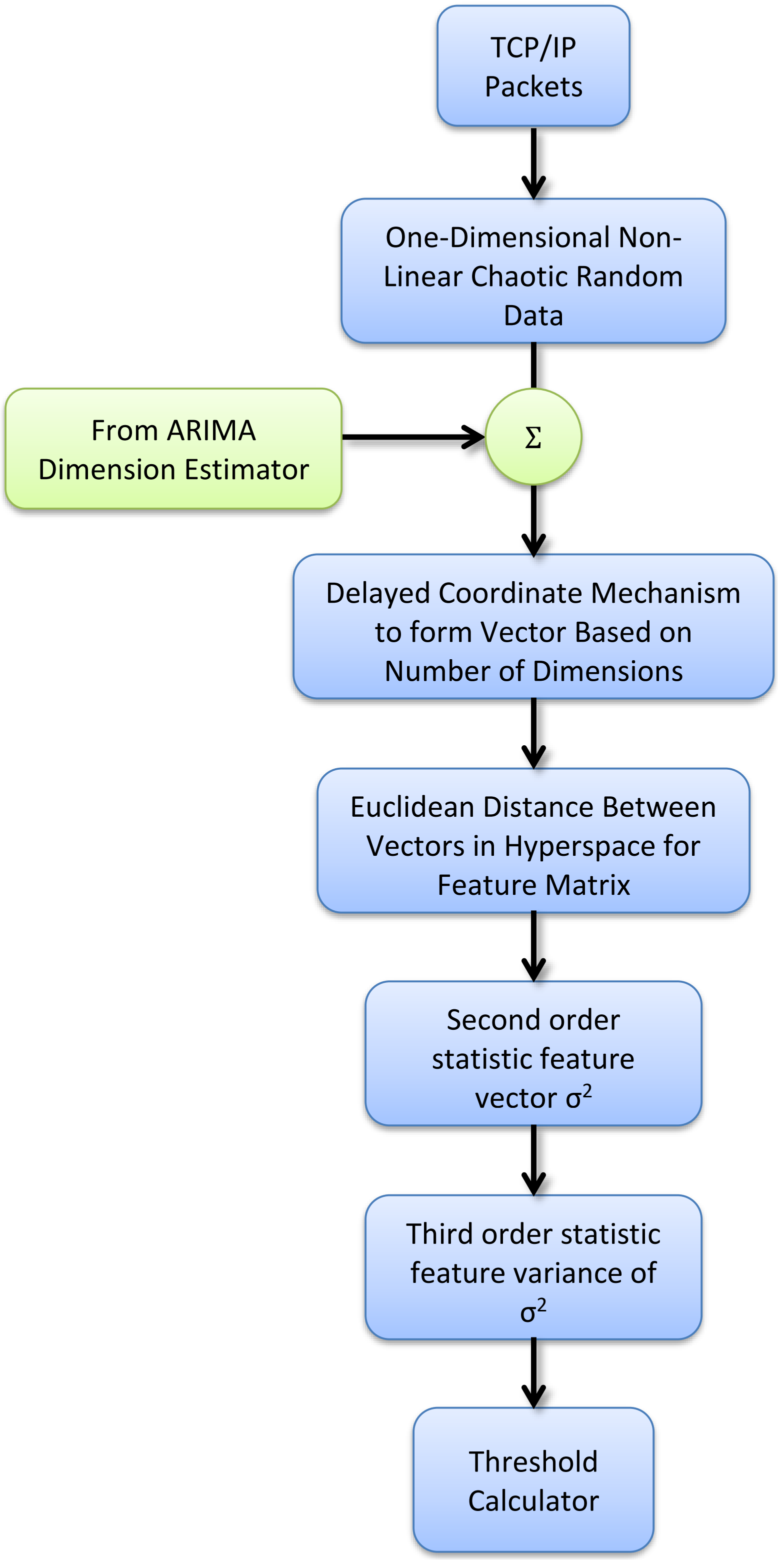

TCP/IP Packets
One-Dimensional Non-Linear Chaotic Random Data
From ARIMA Dimension Estimator
Σ
Delayed Coordinate Mechanism to form Vector Based on Number of Dimensions
Euclidean Distance Between Vectors in Hyperspace for Feature Matrix
Second order statistic feature vector σ²
Third order statistic feature variance of σ²
Threshold Calculator

Figure 4.5: Flowchart for creation of features and threshold calculation for a profile - Training Process

Figure 4.5 shows the process followed for the maintenance of thresholds, where the one-dimensional non-linear data is expanded in terms of previous values to compute a threshold value.

It is observed through a Pseudo-Random Number Generator (PRNG) that the algorithm starts generating the same set of sequences over again because of the limited number of internal states that can be used by the algorithm [205]. It is also observed that the PRNG used for sequence numbers in TCP headers follows patterns (based on different implementations) in operating systems [206]. In addition, this is realised through Phase Space Analysis as well [203], showing that a correlation between subsequent results is followed when generating random numbers.

The aforementioned property between generated random numbers can be used to find out if a subsequent random number is correlated in a specific way.

Chaotic systems can be found widely in the nonlinear field. A chaotic system is usually characterised by the dynamical invariants such as correlation dimension, Lyapunov exponent, etc. These invariants can reflect objectively the internal characteristics of a chaotic system. One of the highly popular methods for the solution of these invariants is phase space reconstruction. Its purpose is to construct a system state by using its historical entries and viewing it in a higher dimensional space so that any recognisable patterns can be observed for chaotic data.

For the purpose of creating TCP/IP packets, PRNGs are used to generate a sequence of numbers that approximates to properties of random numbers. It is the randomness of ISNs that makes it difficult for attackers to predict them. However, random numbers generated using an algorithm are actually pseudo-random, not truly random. This makes the communication open to vulnerability. It is the uniqueness of ISNs within a given time frame that ensures that fragments of different packets are not assembled into one packet on the receiving end. A PRNG in operating systems is modelled as a function with a short random seed as its input and the output being indistinguishable from truly random bits. Various implements for PRNGs exist that implement a deterministic function [207]. Herring and Palmore [208] reported that pseudo-random number generators are derived from deterministic chaotic dynamic systems, thus making a connection between chaos and pseudo-random number generators.

Chaos can be defined as a random and non-uniform phenomenon in the deterministic non-linear system that can be revealed using chaos theory. With conventional methods such as the Fourier transformation, chaos looks like noise, while within the realm of phase space, chaos has structure [38].

Phase space reconstruction is a very useful nonlinear or chaotic signal processing technique to find out about dynamic systems. Topologically, the phase space and original system are equivalent and hence it is possible to recover the non-linear dynamics of the generating system [209]. The implication is that the complete dynamics of the system are accessible in this space.

4.7 Proposed Statistical Algorithm for Quantification of Outliers

This section details the quantification of outliers when detected. Note that the actual quantification algorithm is discussed in Section 4.9.

With header fields that can take random data, it becomes highly dynamic and difficult to monitor for covert data leakage. Therefore, when a threshold breach is reported it does not automatically indicate a data leakage event. Instead, each such event where a deviation from the threshold is noted is subjected to further analysis to quantify the outlier. As mentioned in Section 3.3 that indicates the shortcomings of related techniques, the framework presented in this thesis proposes a state-of-the-art quantification algorithm that is aligned with the requirements.

The proposed quantification algorithm provides a score to each outlier detected, thus forming a dual-stage analysis process. In the first stage, the actual event of outlier detection is reported, and the second stage scores the deviation in relation to the threshold to measure the extent of the breach. This improves the accuracy of detection and reduces the likelihood of the Type I and Type II errors.

The illustration of the outlier quantification process is presented in Figure 4.6. It can be observed that the initial stage of outlier detection is aligned with threshold computation as shown in Figure 4.5, while the latter parts of the process delve into reasoning about whether the detected outlier warrants a notification to be triggered.

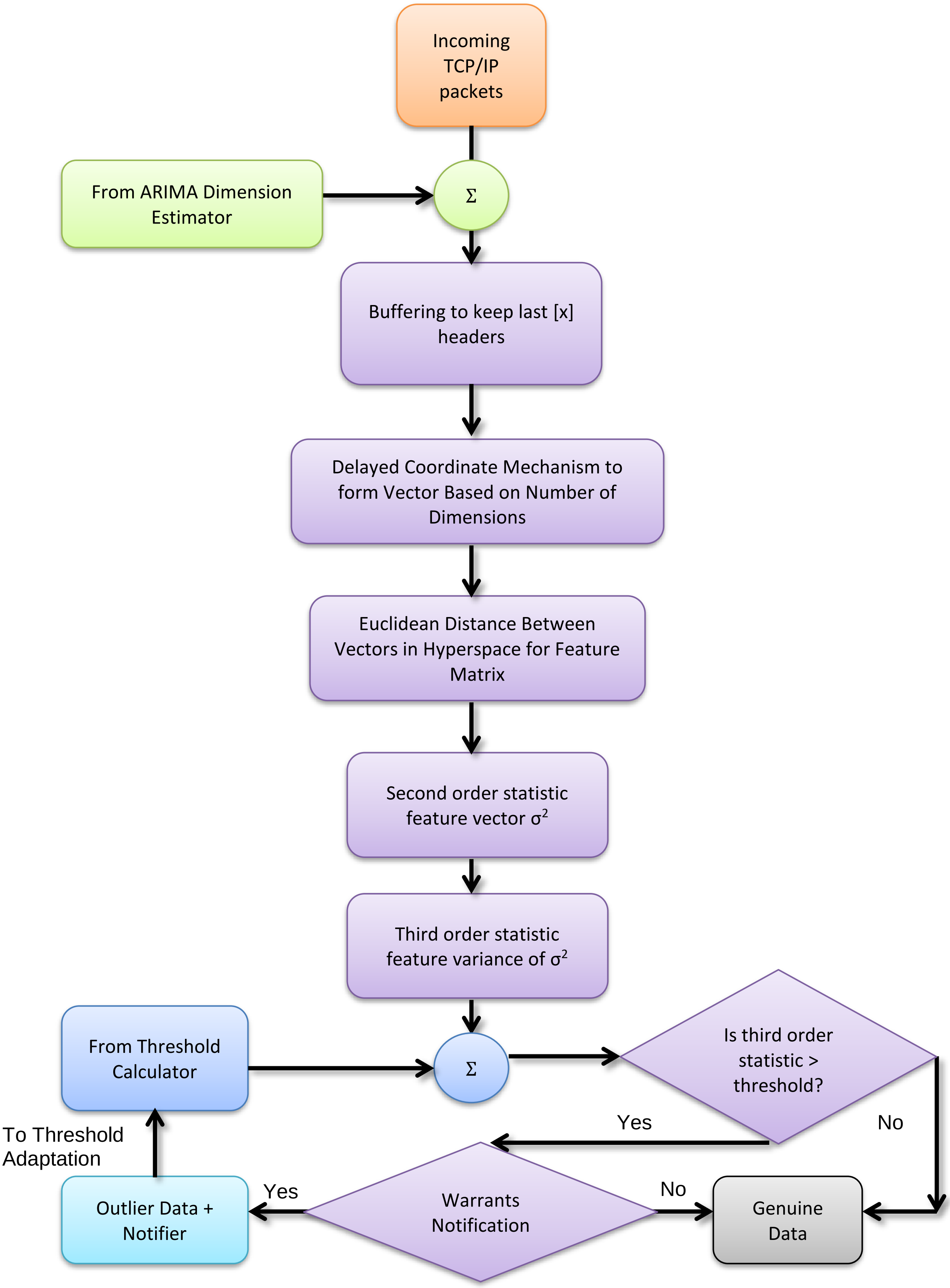

Incoming TCP/IP packets
From ARIMA Dimension Estimator
Σ
Buffering to keep last [x] headers
Delayed Coordinate Mechanism to form Vector Based on Number of Dimensions
Euclidean Distance Between Vectors in Hyperspace for Feature Matrix
Second order statistic feature vector σ²
Third order statistic feature variance of σ²
From Threshold Calculator
Σ
Is third order statistic > threshold?
Yes
No
To Threshold Adaptation
Outlier Data + Notifier
Yes
Warrants Notification
No
Genuine Data

Figure 4.6: Flowchart for quantification of outliers – the process to decide if a certain packet is marked as malicious or normal

4.8 Proposed Statistical Threshold Creation Algorithm

The starting point for the analysis of any non-linear data set (for instance, initial sequence numbers generated for packet creation in TCP/IP communication) is the construction of a phase space or creation of a portrait of its phase space. The state of the system is described as the state of its variables, and *n* state variables observed at time t form a vector in an *n* dimensional space called phase space. The state of the system typically changes with time and hence the vector in the phase space describes the trajectory of the system or evolution/dynamics of the system. It is this shape of trajectory that hints/indicates about the system. Chaotic or periodic systems are characteristics in the phase space.

To reveal the hidden structure of a random number phase space, its construction using "delay coordinates" is widely used. For given *ISN(i)* numbers, the phase space (data sets) is constructed as follows:

$$Y_i = (ISN(i), ISN(i-1), \ldots, ISN(i-(m-1))$$

Equation 4.1: ISNs in phase space

where *i = 1, 2, ... n – m +1*, *n* is the number of ISNs, and *m* is the dimension. Vector Y_i is the new phase space (data set) that is formed from time-delayed values of the initial ISN value scalar measurements. This has been calculated using two different methods: the first-order differential equation and discrete Legendre polynomials with their details given below.

First-order Difference Method

The first-order difference (as shown in Equation 4.2) is constructed as follows [85]:

$$\begin{aligned} x(i) &= Y1 - Y2 \\ y(i) &= Y2 - Y3 \\ z(i) &= Y3 - Y4 \\ w(i) &= Y4 - Y5 \end{aligned}$$

Equation 4.2: ISNs in 4-dimensional phase space

where *i = n, n-1, n-2,…,* 5. *x(i), y(i), z(i)* and *w(i)* are called point coordinates that are used to create a phase space data set. The collection and subtraction of five ISN values help to generate four different streams of data from a single data series, i.e. it helps to construct different coordinates which are actually the delay coordinates.

Establishing these vectors collectively in the realm of R^m forms a phase space. Chaos theory dictates that phase space vectors are fully representative of the non-linear dynamics of the original data set when the embedding dimension m is large enough.

For optimal embedding m, according to [209], the observation of a real process generally does not yield all of its state variables. This is generally due to the reason that either not all state variables are known or not all of them can be measured.

It is clear that the numbers generated are the result of unknown regression with dependent components. As they are unknown, it can be assumed that they are formed of independent variables $y_1, y_2, y_3.., y_t$ where y_{t+1} is dependent upon the value of y_t. This type of time series is known as a univariate time series (the series with single observations recorded over regular intervals). In a univariate time series, the past value of an independent variable is used to calculate the value of a new independent variable as depicted in the following equation:

$$y_t = \beta y_{t-1} + \varepsilon$$

Equation 4.3: Generic representation of univariate time series

Here, β is a constant and ε is an error value. It is clear from Equation 4.3 that sometimes when the set of explanatory variables required by a regression model is unavailable (true for random numbers generated as a function of time with unknown other variates), then it becomes a beneficial choice to use only a single variable to forecast future values.

Among the models available for modelling univariate series are:

1. AutoRegressive Model (AR)
2. Moving Average Model (MA)
3. AutoRegressive Moving Average Model (ARMA)
4. AutoRegressive Integrating Moving Average Model (ARIMA)

In the AR model, y_t depends only upon its own past values y_{t-1}, y_{t-2}, y_{t-3}, etc. Thus $y_t = f(y_{t-1}, y_{t-2}, y_{t-3}, \ldots, \varepsilon_t)$, where ε_t is a noise or error term. A common representation of an autoregressive model depending upon p past values is called an *AR(p)* model and represented as:

$$y_t = c + \sum_{i=1}^{p} \varphi_i y_{t-i} + \varepsilon_t$$

Equation 4.4: Generic representation of AR model

Here, *c* is a constant and φ_i is used for coefficient. For Equation 4.4, it is important to know that the value of *p* is about how far back in time the value of y should be picked to estimate y_t. Generally, in a real-life phenomenon, it is observed that past values up to 3 steps (i.e. forming an *AR(3)* model) are sufficient [210]. In this context, it is critical to computing p so that accuracy can be obtained.

In the MA model, y_t depends only on its error terms, ε_{t-1}, ε_{t-2}, ε_{t-3}, etc. and μ is the mean. A common representation of a moving average model where it depends on q past values is called a *MA(q)* model and defined as:

$$y_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \varphi_i \varepsilon_{t-i}$$

Equation 4.5: Generic representation of MA model

The ARMA model refers to a combined usage of the AR and MA models, denoted as ARMA(p,q). The time series data has a mean μ zero which is already subtracted. Hence, it is represented as:

$$y = \varepsilon_t + \ c + \sum_{i=1}^{q} \varphi_i \varepsilon_{t-i} + \sum_{i=1}^{p} \varphi_i y_{t-i}$$

Equation 4.6: Generic representation of ARMA model

ARIMA is a forecasting technique applied to many real-time series, which is based on an autoregressive part and a contribution from a moving average. It operates on time series that are stationary and helps to project the future values of a series based entirely on its past values. As the inherent property of univariate time series is a correlation on past p / q independent variables, it is critical to make them stationary. The first difference of a time series is the series of changes from one period to the next. If y_t denotes the value of the time series *y* at period *t,* then the first difference of *y* at period *t* is equal to y_t-y_{t-1}. If the first difference of *y* is stationary and also completely random (not autocorrelated), then a random walk model can describe *y*: each value is a random step away from the previous value. If the first difference of *y* is stationary but not completely random, i.e. its value at period t is auto correlated with its value at earlier periods, then a more sophisticated forecasting model such as exponential smoothing or ARIMA may be appropriate. Note that non-stationary series regressions may result in spurious regressions, i.e. the cases where the regression equation shows a significant relationship between two variables when there is none. Box-Jenkins methodology is used for the estimation of univariate series that are stationary:

1. ACF – It refers to the way that the observations in a series are related to each other and is measured by simple correlation between the current observation (y_t) and the observation in the p^{th} period from the current one (i.e. y_{t-p}). It is defined as:

$$\rho_k = Corr\left(y_t, y_{t-p}\right) = \frac{Cov(y_t, y_{t-p})}{\sqrt{var(y_t)}\sqrt{var(y_{t-p})}}$$

Equation 4.7: Autocorrelation (ACF)

Here, *Cov* is covariance and *var* is variance. p_k informs about how many periods back one should look into for creating the stationary series.

2. PACF – It refers to the degree of correlation between y_t and y_{t-p}, which is defined as:

$$\rho_k = Corr\left(y_t, y_{t-p}\right) = \frac{Cov(y_t, y_{t-p}|y_{t-p-1}, y_{y-p-2})}{\sqrt{var(y_t|y_{t-p-1}}, y_{t-p-2})\sqrt{var(y_{t-p}|y_{t-p-1}, y_{t-p-2})}}$$

Equation 4.8: Partial Correlation (PACF)

The ACF or PACF is available for various values of lags of autoregressive and moving average components, i.e. *p* and *q*.

Discrete Legendre polynomials

A refined approach to estimating derivatives uses discrete Legendre polynomials suggested by [211]. This concept can be generalized to irregular sampling and relates the *jth* derivative at x_i to a weighted sum of the *p* nearest points to each side of x_i as:

$$\frac{\boldsymbol{d^j}}{\boldsymbol{dt^j}}\boldsymbol{x_i} \approx \frac{\boldsymbol{j}!}{\boldsymbol{c_{j,p}}(\boldsymbol{\Delta t_{i,n}})} \sum_{\boldsymbol{n=-p}} \boldsymbol{r^{(i)}_{j,p,n}}\, \boldsymbol{x_{i+n}}$$

Equation 4.9: Discrete Legendre polynomials

Here, *r* is the weight being assigned, and *n* is a total number of points. In this case, equal weights to all data points are tried. Once a data point is suspected, its weight can be increased or decreased to highlight it. $c_{j,p}$ is a normalization constant. *t is* the time interval and *Δt* is the time difference.

$$\boldsymbol{c_{j,p}}(\boldsymbol{\Delta t_{i,n}}) = \sum_{\boldsymbol{n=-p}}^{\boldsymbol{p}} (\boldsymbol{\Delta t_{i,n}})^{\boldsymbol{j}}\, \boldsymbol{r^{(i)}_{j,p,n}}$$

Equation 4.10: Equal weightage to all data points

In Equation 4.10, $\boldsymbol{\Delta t_{i,n} = t_{i+n} - t_i}$ and, the weights are given by the discrete Legendre polynomials $\boldsymbol{r^{(i)}_{j,p,n} = r_{j,p}(\Delta t_{i,n})}$ that can be calculated recursively by the relation:

$$\boldsymbol{r^{(i)}_{j,p,n}} = \frac{\boldsymbol{1}}{\boldsymbol{c_j p^j}}[\boldsymbol{\Delta t^j_{i,n}} - \sum\nolimits_{\boldsymbol{k=0}}^{\boldsymbol{j-1}} \boldsymbol{r^{(i)}_{k,p,n}} \sum\nolimits_{\boldsymbol{k=0}}^{\boldsymbol{j-1}} \boldsymbol{\Delta t^j_{i,l^T\, k^{(i)},p,l}}\,]$$

Equation 4.11: Recursively calculation by the relation

for $\boldsymbol{2p \geq j}$ with $\boldsymbol{r^{(i)}_{0,p,n} = 1/c_0}$. The normalization constants $\boldsymbol{c_j}$ can be determined by the condition:

$$\sum_{n=-p}^{p} (\boldsymbol{r}_{j,p,n}^{(i)})^2 = \mathbf{1}$$

Equation 4.12: Normalization

It should be noted that the discrete Legendre polynomials are not a discretization of the common Legendre polynomials. Instead, for *p*→∞, they converge to the latter. By changing the parameter *p* (the number of points forward and backward to each side that is included for estimating the derivatives) it is possible to control the smoothing of the data. That is, some noise can be averaged out which makes this procedure more robust for noise than other methods. For estimating the optimal value of *p, a* procedure is discussed in [211, 212]. In general, it is recommended to choose a *p* value of the order of the embedding dimension, i.e. of the order of the highest derivative that needs to be estimated. In the limited case of choosing *p* as small as possible, this approach reduces to a central difference quotient.

Finite-differencing is generally not the best method for estimating derivatives of discretely sampled, noisy functions.

In this research project, a chosen data set was used and then the aforementioned procedure was applied along with an ISN legality algorithm.

Dimension Calculation

Based on Equation 4.2 & 4.9, the four-dimensional phase vector r_i is constructed as:

$$r_i = [x(i), y(i), z(i), w(i)], i = 1, 2, \dots, j, j = N - 4$$

Equation 4.13: Four-dimensional phase vector

Let *R* represent the phase space data set formed from Equation 4.13 and Equation 4.14, where the number of phase space vectors in *R* is *N* – 4:

$$R = \left[r_1, r_2, \dots, r_j\right]$$

Equation 4.14: Representation of phase space data set

In order to extract features / patterns from *R*, the distance between two vectors r_i and r_j is calculated in the phase space as follows:

$$d_{i,j} = \sqrt{\left(x(i)-x(j)\right)^2+\left(y(i)-y(j)\right)^2+\left(z(i)-z(j)\right)^2+\left(w(i)-w(j)\right)^2}$$

Equation 4.15: Euclidean distance between two vectors r_i and r_j

Computing the distances between two vectors in *R* forms a 2-dimensional matrix as shown below:

$$D = \begin{pmatrix} d_{1,1} & d_{1,2}\cdots & d_{1,j} \\ d_{2,1} & d_{2,2}\cdots & d_{2,j} \\ \vdots & \vdots\cdots & \vdots \\ d_{k,1} & d_{k,2}\cdots & d_{k,j} \\ \vdots & \vdots\cdots & \vdots \\ d_{j,1} & d_{j,2}\cdots & d_{j,j} \end{pmatrix}$$

Equation 4.16: 2 dimensional distance matrix formed by Euclidean distances between two vectors in *R*

Here, *k = 1, 2, … j,* A row vector represents the distances between a specified vector and all the vectors in *R*, hence the row size is j. As the distance between a vector and itself is always zero, it makes the diagonal entries of zeros.

$$d(k) = \left[d_{k,1}, d_{k,2}, \dots, d_{k,j}\ \right], k = 1, 2, \dots, j$$

Equation 4.17: Row vector *d(k)*

Note that in Equation 4.17, $d_{k,k} = 0$. The variance of row vector *d*(*k*) is calculated as follows:

$$\sigma_k^2 = \frac{1}{j-1}\sum_{i=1}^{j}(d_{k,i}-\mu_k)^2, i, k = 1,2,\dots,j$$

Equation 4.18: Variance of row vector *d(k)*

Here, μ_k is the mean value of row vector *d(k).* σ_k^2 represents the variance of Euclidean distances between the specified vector r_k and all other vectors in *R*. Upon calculation of the variance, the variance vector σ^2 is obtained as shown in Equation 4.19.

$$\sigma^2 = [\sigma_1^2, \sigma_2^2, \dots, \sigma_j^2]$$

Equation 4.19: Variance vector *σ*^{2}

Furthermore, Equation 4.20 below calculates the variance of vector *σ*^{2}. In this research project, the computed variance is based on 1,000 ISNs that can be obtained by an ISN training sequence. Here μ_σ represents the mean value of those in σ^2. It is expected that the variation in distance will be of the same order, so 10% of the variance is selected as a threshold as shown in Equation 4.21. The justification of 10% as the threshold is to ensure that the false positive rate remains low.

$$var_\sigma = \frac{1}{j-1}\sum_{i=1}^{j}(\sigma_i^2 - \mu_\sigma)^2, i = 1,2,\dots,j$$

Equation 4.20: Variance of each variance vector *σ*^{2}

$$Threshold_{ISN} = \frac{var_\sigma}{10}$$

Equation 4.21: Threshold for normal ISN

4.9 Proposed Algorithm for Quantification of Outliers

The statistical model created in Section 4.8 provides threshold values that can be used for checking if covert non-linear random data is normal. However, to do so, the realm of the collected data set is transformed into a phase space so that a comparison can be made.

To identify that an incoming ISN is malicious or non-malicious based on the proposed detection model, an algorithm is presented in Table 4.2 based on the computations detailed in Section 4.8. For any incoming ISN, the coordinates of the multiple-dimensional phase vector are created based on the Step 2 of the algorithm in the table. This ensures that the coordinates are brought into the phase space using the same delayed coordinate technique that is used for the creation of the detection model. This is followed by the calculation of the distances between the vector *p(a, b, c, d)* and all the vectors r_i in the training data set *R* to obtain Euclidean distance vector $d = [d_{j,1}, d_{j,2},\ldots,d_{j,j}]$. The second-order variance σ^2_j of this distance vector is then calculated. This procedure is reapplied to ISN(j-1) and ISN (j-2) to get σ^2_{j-1} and σ^2_{j-2}. These results are used to calculate third order feature var_σ that is then compared with threshold $Threshold_{ISN}$. ISN is considered normal if the third-order feature is greater than the threshold and covert if it is smaller than the threshold.

1. *Capture incoming ISN (j).*
2. *Compute p(a, b, c, d) using follows:*

$$a = ISN(j) - ISN(j-1)$$
$$b = ISN(j-1) - ISN(j-2)$$
$$c = ISN(j-2) - ISN(j-3)$$
$$d = ISN(j-3) - ISN(j-4)$$

OR

$$\frac{d^j}{dt^j}x_i \approx \frac{j!}{c_{j,p}(\Delta t_{i,n})} \sum_{n=-p} r^{(i)}_{j,p,n}\, x_{i+n}$$

3. *Calculate distances between the vector p(a, b, c, d) and the vectors r_i in the training data set R to obtain the distance vector $d = [d_{j,1}, d_{j,2},\ldots,d_{j,j}]$.*
4. *Get the second-order statistics σ^2_j = variance(d).*
5. *Repeat Step 1 to 4 for ISN(j-1) and ISN(j-2) to obtain σ^2_{j-1} and σ^2_{j-2}.*
6. *Calculate the third-order statistics by calculating variance of all obtained variances using $var_\sigma = var(\sigma^2_j, \sigma^2_{j-1}, \sigma^2_{j-2})$.*
7. *Compare var_σ with $Threshold_{ISN}$.*
8. *If var_σ is greater than $Threshold_{ISN}$ then ISN is non malicious.*
9. *If var_σ is less than $Threshold_{ISN}$ then ISN is malicious.*
10. *Score var_σ of malicious ISN to determine the degree to which the outlier resides outside the expected score. The score is based on the weighted moving average of historical data.*
11. *If the expected score warrants a notification trigger then ISN is considered malicious.*

Table 4.2: Proposed algorithm for quantification of outliers

Framework (Figure 4.1)	**Framework Component (Figure 4.3)**	**Threshold Maintaining Algorithm Flowchart (Figure 4.5)**	**Outlier Quantification Algorithm Steps (Table 4.3)**	**Relevant Equations (Section 4.8)**	**Section**
Training Data	Trainer	• Delayed Coordinate Mechanism • Euclidean Distance Calculator • Second-order variance • Third-order variance		4.1, 4.2, 4.3	
Chaotic Data Correlator	Trainer	• ARIMA Dimension Calculator		4.4, 4.5, 4.6	4.5
Threshold Profiler and Adaptation	Threshold Calculator	• Threshold Calculator		4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15	4.6
Data Logging and Monitoring	Data Monitor		1, 2, 3, 4, 5, 6	4.1, 4.2, 4.3, 4.4, 4.5, 4.6	
Decision Maker	Outlier Analysis and Score Warrants Notification		7, 8, 9, 10		4.7

Table 4.3: References to framework elements, relevant sections and equations used in the proposed algorithm in Table 4.2

As the components and equations used for the framework specification are introduced in the different sections of this chapter, Table 4.3 is provided for their easy references to link them together.

4.10 Summary

It is realised that the phase space reconstruction can be used to find temporal patterns as long as the embedding dimension is large enough that it is realised using the auto-correlation model. The threshold value for the identification of covert non-linear chaotic random data is successfully created. This value will be used for the experiments to be detailed in the next chapter.

Chapter 5 Implementation & Evaluation

5.1 Overview

This chapter presents the implementation and evaluation of the proposed framework. The implementation includes outlier detection. The evaluation is based on a set of experiments to demonstrate the framework performance. The experiment results are also used for a comparison with related work.

5.2 Testing Strategy

A testing strategy used for the evaluation of the framework presented in Chapter 4 has different phases for conducting experiments, including data capturing, pre-processing and threshold calculation. More specifically, the following steps are adopted for the testing strategy:

- To test the framework with real traffic and hidden data, ISN values are collected from a live network data stream.
- Tools such as TCPDump, Wireshark, Capsa, Soft Perfect Network Protocol Analyser and Ethreape are used for capturing the network traffic.
- The data leakage detection algorithm in Table 4.2 is implemented and tested based on the collected network traffic to gather necessary data for the algorithm evaluation.
- Experiments on an independent data set are conducted to determine $Threshold_{ISN}$ used in the algorithm.

5.3 Implementation of Statistical Threshold Creation Algorithm

Auto Correlation Function (ACF) is one of the tools used to find patterns in the data [202]. Specifically, the ACF informs about the correlation between points separated by various time lags. So, the ACF informs about how correlated past data points are to future data points, for different values of the time separation.

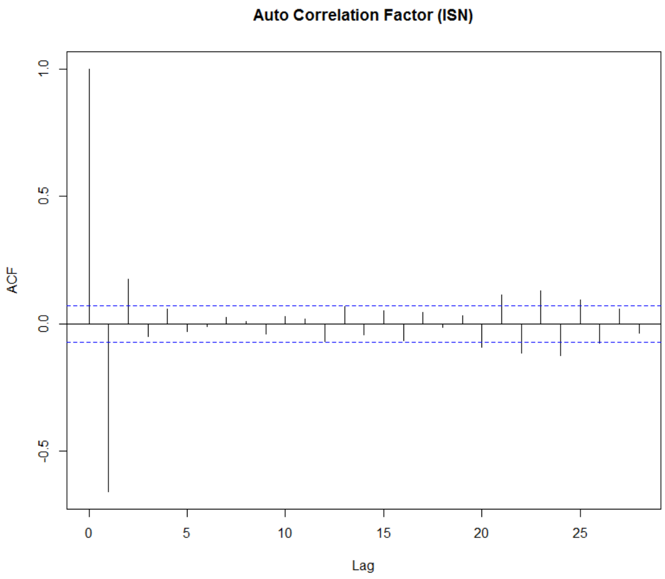


Figure 5.1: ACF plot for stationary ISNs with delayed coordinate at m =4, where data set is collection of 1000 ISNs

Partial Auto Correlation Function (PACF) is the correlation between two points that are separated by some number of periods, *n,* but with the effect of the intervening correlations removed. Hence, better correlation effectively means better temporal data patterns which are useful for the analysis of covert data within ISNs.

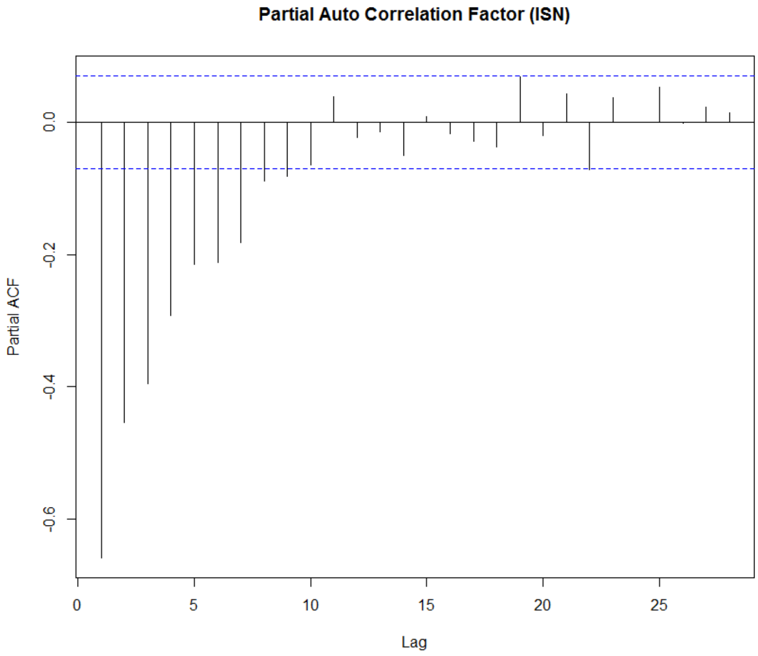
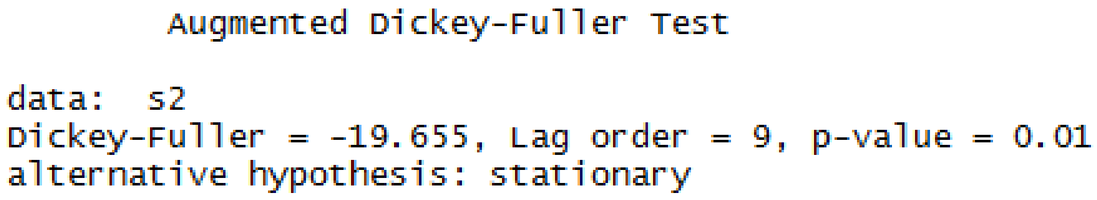


Figure 5.2: PACF plot for stationary ISNs with delayed coordinate data at m=4, where data set is collection of 1000 ISNs



```
        Augmented Dickey-Fuller Test

data:  s2
Dickey-Fuller = -19.655, Lag order = 9, p-value = 0.01
alternative hypothesis: stationary
```

Figure 5.3: Dickey-Fuller test for proving that ISN series have been made stationary at m= 4 after applying delayed coordinate method for phase space reconstruction based statistical analysis

Figure 5.2 shows the ACF plot for stationary ISNs with a delayed coordinate at m =4, where the dataset is collection of 1000 ISNs. Figure 5.2 shows the PACF plot for stationary ISNs with delayed coordinate data at m=4, where the dataset is collection of 1000 ISNs [202]. It is noticed that ACF and PACF are residing outside thresholds, indicating the auto regression of some order will be required. This is tested using the Dickey-Fuller Test (See Figure 5.3) to prove that the series is made stationary, which extracts a larger set of temporal patterns useful for the analysis here.

The experiments in this study are conducted on Ubuntu Linux VM that is mounted using Oracle Virtual Box on a Windows 7 computer. The choice of the operating system is dictated by the specification of a covert data creation code called Covert_TCP.

As mentioned earlier, the Covert_TCP code is used for creating TCP packets. The code written in C and is compiled on an Ubuntu Linux terminal. The code requires the passing of a switch value that informs the generation of TCP packets where ISN is the choice for covert channel communication.

Between a server and a client, there is also a TCP Dump program (that is shipped with Ubuntu Linux) used to capture the TCP packets sent from the server to the client. This program only requires an IP address and a port number that it needs to bind itself to, as well as a location where it should write captured TCP packets. The captured TCP packets are written in a *.pcap file.

The amount of data collected has a direct relation to the accuracy of threshold values. For the purpose of computing a threshold value, the following strategies are applied to ensure that the threshold value obtained has the least skewness and the subset of collected data for processing is not too large. To evaluate this, the following data sizes 100, 350, 700, 1,000, 2,000 and 3,000 are used and the amount of percentage change in the threshold value is measured to ensure durability and precision. Note that experiments are conducted over increments in the data size of 100. During the experiments, it is noticed that the duration of computation values and the average difference in the maximum and minimum values of the threshold is significant.

Data Size (Number of Packets)	**Duration for Threshold Computation (hours)**	**Average Maximum Difference in Threshold Values (%)**	**Average Minimum Difference in Threshold Values (%)**
100	0.5	81.41	53.56
350	1.25	69.62	39.15
700	1.75	21.96	7.82
1000	2	3.38	1.15
2000	5.25	2.91	0.97
3000	9	2.33	0.69

Table 5.1: Results indicating amount of time required for training with data sizes and percentage movement in threshold value

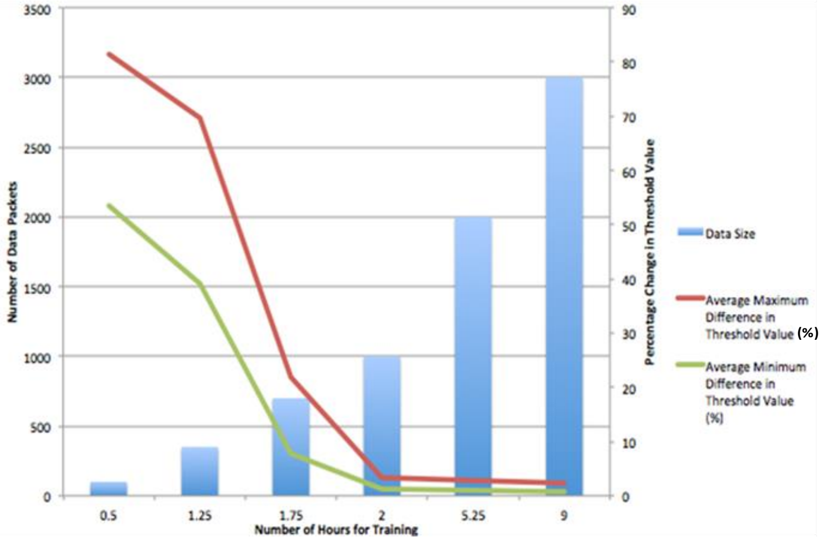


Figure 5.4: Average changes in threshold value with respect to data sizes and time required for creation of training model

The results from the computation of threshold values using these data sizes are presented in Table 5.1 and shown in Figure 5.4. The results from this experiment indicate that using 1,000 packets as the data size provides us with the most balanced value with respect to the training period as well as the asymptotic point where the amount of gain that can be achieved by using any higher data size approaches 0. Hence, for the testbed configuration in this study, the 1,000 data size and its corresponding value for the threshold are chosen. Note that this is not constant globally and can change based on hardware metrics available while computing the threshold value.

In order to test the proposed solution on real traffic and hidden data, ISN values are collected from live network data streams from various applications such as web browser, messenger and email client. The network packets contain the source IP address that shows the origin of network traffic. Figure 5.5 shows a high-level overview of how network packets are captured. Technically, link-layer network access allows this activity after packets bypass through the protocol stack which also includes kernel-level packet filtering.

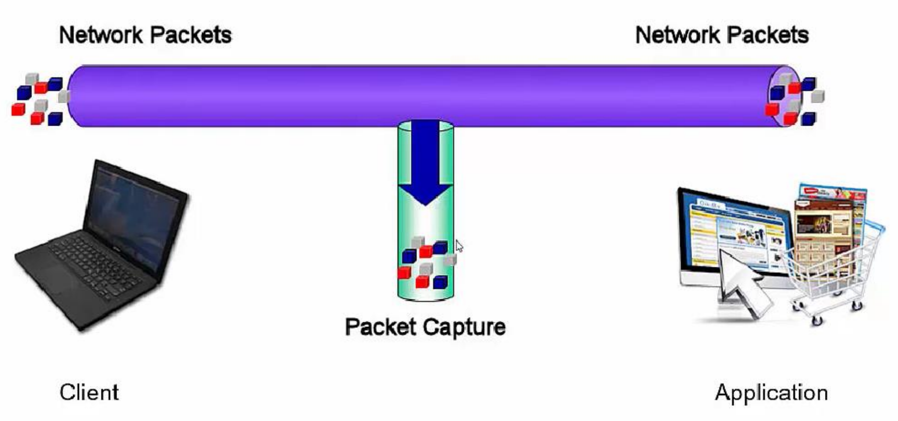
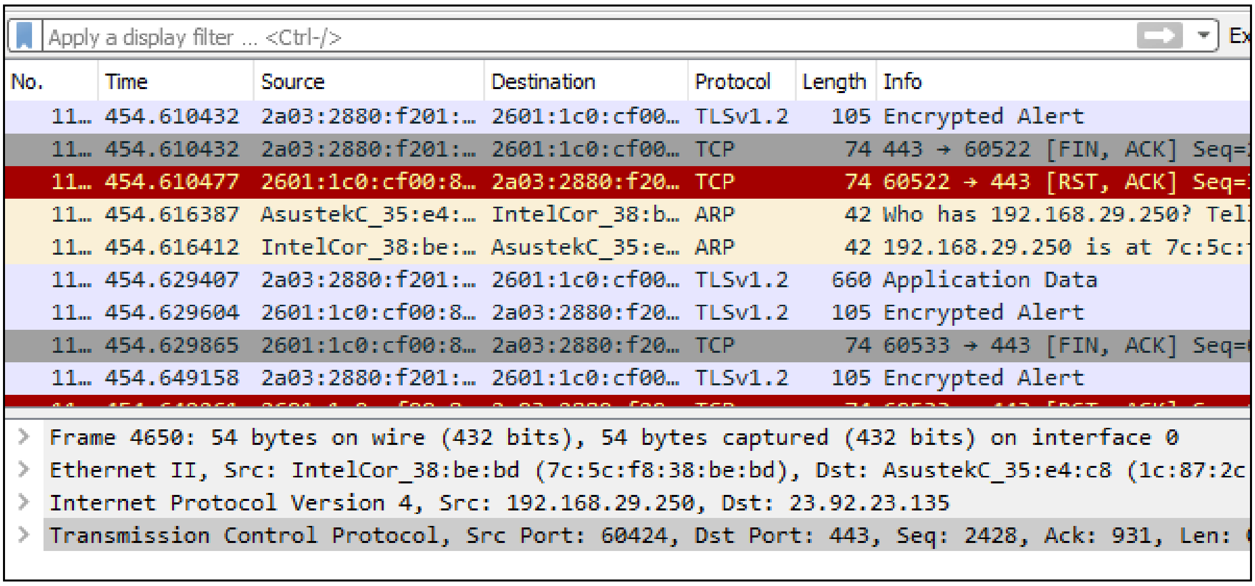


Figure 5.5: Network data packet capturing

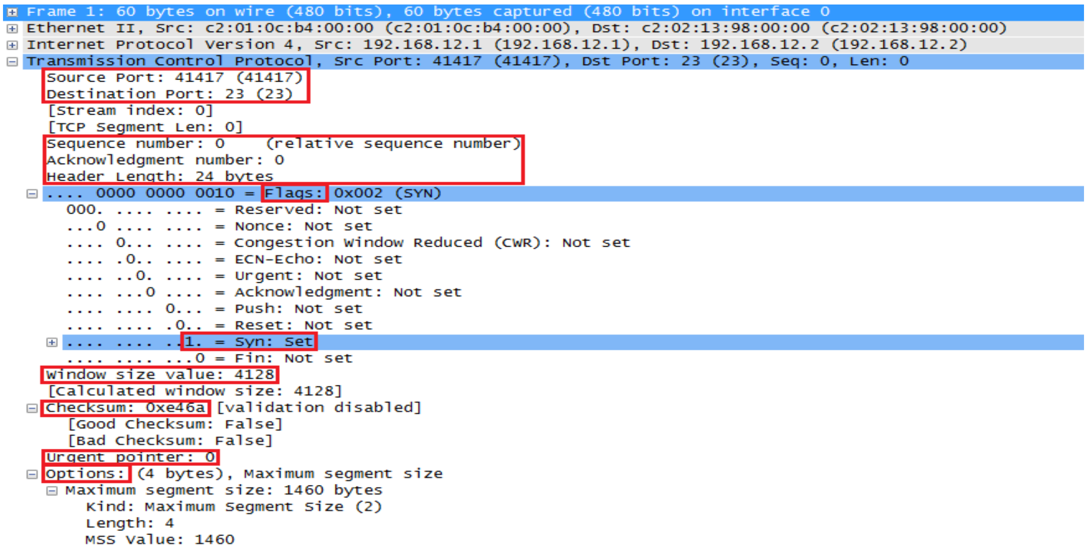
Wireshark is used for capturing network traffic. Note that the captured packets are actually just a copy of the original network packets, so these applications allow us to capture the network flow using the backend library WinPCap. WinPCap is a well-known driver that helps to gain low-level access to network layers. Figure 5. shows an example of a network packet that has been captured by Wireshark. It shows live traffic with detailed information about network packets, protocol types, source and destination IP addresses, and general information about packets. Dumpcap is one of the tools within Wireshark for the purpose of capturing network packets from the live network and also saving the stream of packets into a file (*.pcap). The pcap file can then be exported as a *csv file that contains a selected feature set essential for the detection of covert data. The main purpose is to gain such information from the network packets and Wireshark provides the needed information of TCP as shown in Figure 5..



No.	Time	Source	Destination	Protocol	Length	Info
11…	454.610432	2a03:2880:f201:…	2601:1c0:cf00…	TLSv1.2	105	Encrypted Alert
11…	454.610432	2a03:2880:f201:…	2601:1c0:cf00…	TCP	74	443 → 60522 [FIN, ACK] Seq=
11…	454.610477	2601:1c0:cf00:8…	2a03:2880:f20…	TCP	74	60522 → 443 [RST, ACK] Seq=
11…	454.616387	AsustekC_35:e4:…	IntelCor_38:b…	ARP	42	Who has 192.168.29.250? Tel
11…	454.616412	IntelCor_38:be:…	AsustekC_35:e…	ARP	42	192.168.29.250 is at 7c:5c:
11…	454.629407	2a03:2880:f201:…	2601:1c0:cf00…	TLSv1.2	660	Application Data
11…	454.629604	2601:1c0:cf00:8…	2a03:2880:f20…	TLSv1.2	105	Encrypted Alert
11…	454.629865	2601:1c0:cf00:8…	2a03:2880:f20…	TCP	74	60533 → 443 [FIN, ACK] Seq=
11…	454.649158	2a03:2880:f201:…	2601:1c0:cf00…	TLSv1.2	105	Encrypted Alert

```
> Frame 4650: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: IntelCor_38:be:bd (7c:5c:f8:38:be:bd), Dst: AsustekC_35:e4:c8 (1c:87:2c
> Internet Protocol Version 4, Src: 192.168.29.250, Dst: 23.92.23.135
> Transmission Control Protocol, Src Port: 60424, Dst Port: 443, Seq: 2428, Ack: 931, Len:
```

Figure 5.6: Network data packet capturing using Wireshark tool



```
⊞ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: c2:01:0c:b4:00:00 (c2:01:0c:b4:00:00), Dst: c2:02:13:98:00:00 (c2:02:13:98:00:00)
⊞ Internet Protocol Version 4, Src: 192.168.12.1 (192.168.12.1), Dst: 192.168.12.2 (192.168.12.2)
⊟ Transmission Control Protocol, Src Port: 41417 (41417), Dst Port: 23 (23), Seq: 0, Len: 0
    Source Port: 41417 (41417)
    Destination Port: 23 (23)
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgment number: 0
    Header Length: 24 bytes
  ⊟ .... 0000 0000 0010 = Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ⊞ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
    Window size value: 4128
    [Calculated window size: 4128]
  ⊟ Checksum: 0xe46a [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
    Urgent pointer: 0
  ⊟ Options: (4 bytes), Maximum segment size
    ⊟ Maximum segment size: 1460 bytes
        Kind: Maximum Segment Size (2)
        Length: 4
        MSS Value: 1460
```

Figure 5.7: Detailed information of network data packet

Figure 5. shows the ISN values collected through the data collection phase described above. It can be observed that these series appear to be completely random and do not show any dependencies among successive data coordinates. However, Figure 5. shows some form of emergent patterns because it is the result of a systematic interaction of its component parts. It also indicates that phase space reconstruction provides a useful result with regard to the dynamics of ISN values.

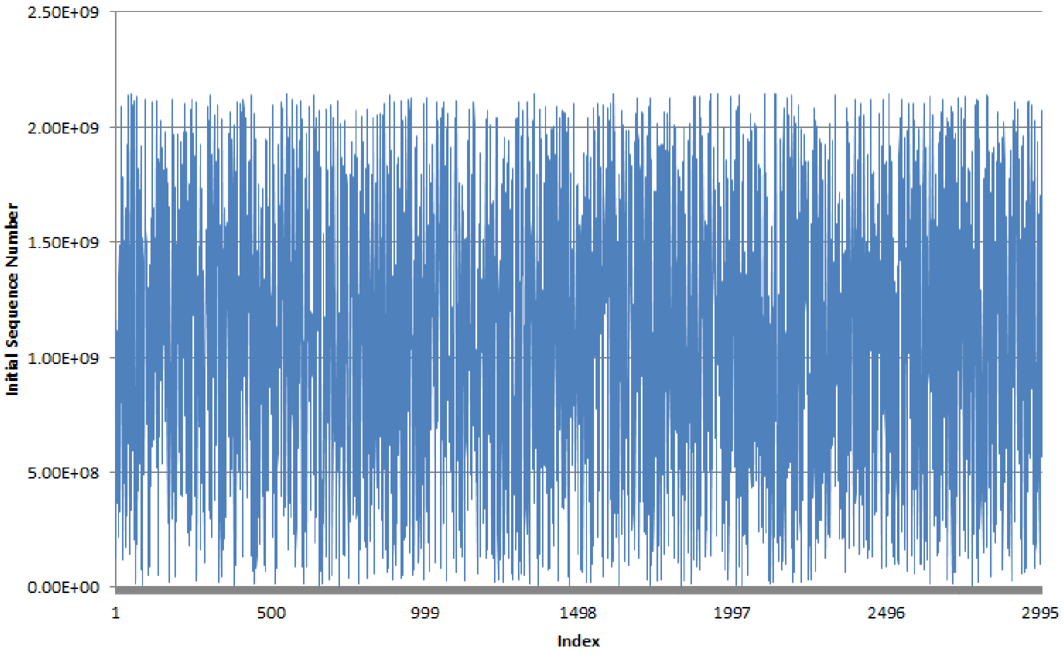


Figure 5.8: One-dimensional ISN data

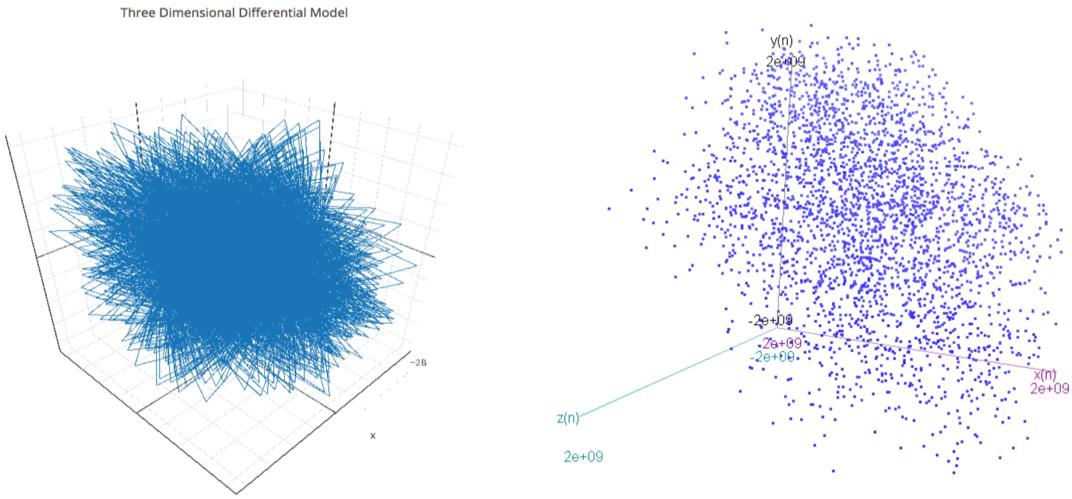


Figure 5.9: Three-dimensional differential model plot (Line Format and Scatter Format)

Establishing these vectors collectively in the realm of R^m forms a phase space. Chaos theory dictates that phase space vectors are fully representative of the non-linear dynamics of the original data set when the embedding dimension *m* is large enough.

As defined in Chapter 4, Equation 4.13 can be used to construct four-dimensional phase vectors r_i to form phase space data set *R* defined in Equation 4.14. To extract features / patterns from *R*, the distance between any two vectors in *R* needs to be calculated based on Equation 4.15. These distances then form a 2-dimensional matrix illustrated in Equation 4.16. For each row of the matrix, Equation 4.18 is applied to calculate a variance. These variances are represented a vector shown in Equation 4.19. Finally, this variance vector is used to compute overall variance var_σ based on Equation 4.20. Figure 5. illustrates the second-order statistic feature vector (variance) results and shows the results of the computed overall variance.

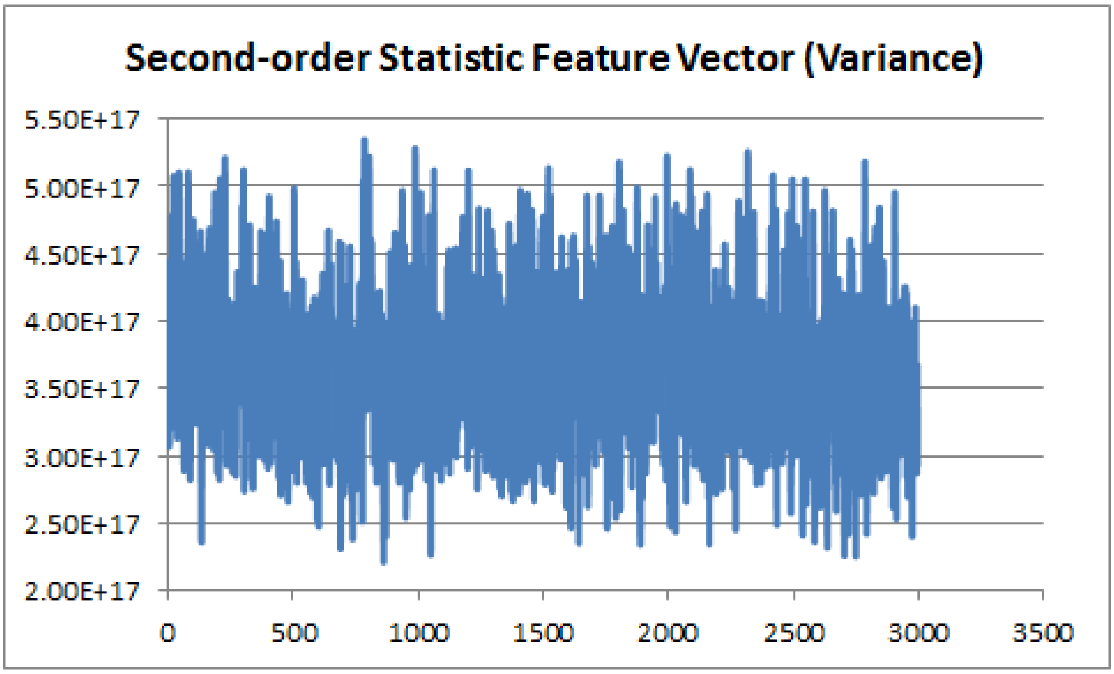


Figure 5.10: Second order statistic feature vector – variance between vectors in *R in the presence of threshold values*

For the experiment conducted in this study, var_σ is computed based on ISNs extracted from 1,000 collected TCP packets as proved by the experimentation shown in Figure 5.4, which produces var_σ = 2.62E+33. 10% of the var_σ value is then chosen as $Threshold_{ISN}$ according to Equation 4.21, i.e. $Threshold_{ISN}$ = 2.62E+32.

So far, the implementation of the threshold computation has been presented as well as the embedding dimension for the delayed coordinate system. In the next sections, the threshold value is used to quantify outliers where the data in ISN fields are injected using a number of strategies. This helps with a complete evaluation of the proposed algorithm in Chapter 4. These sections begin by creating an experimentation whereby the new quantification algorithm is tested by creating a covert message. The performance of the algorithm is tested together with state-of-the-art existing work to prove that the use of *Threshold$_{ISN}$* is more successful in terms of locating malicious ISNs in a given data set.

5.4 Experiments

The methodology for testing the proposed data leakage detection algorithm was used earlier in similar research works that include Support Vector Machine based algorithms [213] for the detection of covert data and covert ISNs [145]. The experiments in this study are explained stepwise through the algorithm and presented alongside the mathematical outcome for the reader to understand the workings clearly. The workings are depicted for shorter covert tests while for larger data sets, the results have been appended.

5.4.1 Experiment 1 – Demo Data (User Created)

This experiment is conducted in accordance with the steps of the proposed algorithm in Table 4.2. The implementation detail of each step is presented separately below.

Algorithm step 1: Capture incoming ISN

Keep the most recent 4 ISNs where Excel is used to create the ISN numbers. As an example, encode text “OLA QUE PASA” to ASCII numbers and then to ASCII CHARACTER * 256 * 65536, as shown in the table below:

Index	Character	ASCII	Covert ISN
1	O	79	1.33E+09
2	L	76	1.28E+09
3	A	65	1.09E+09

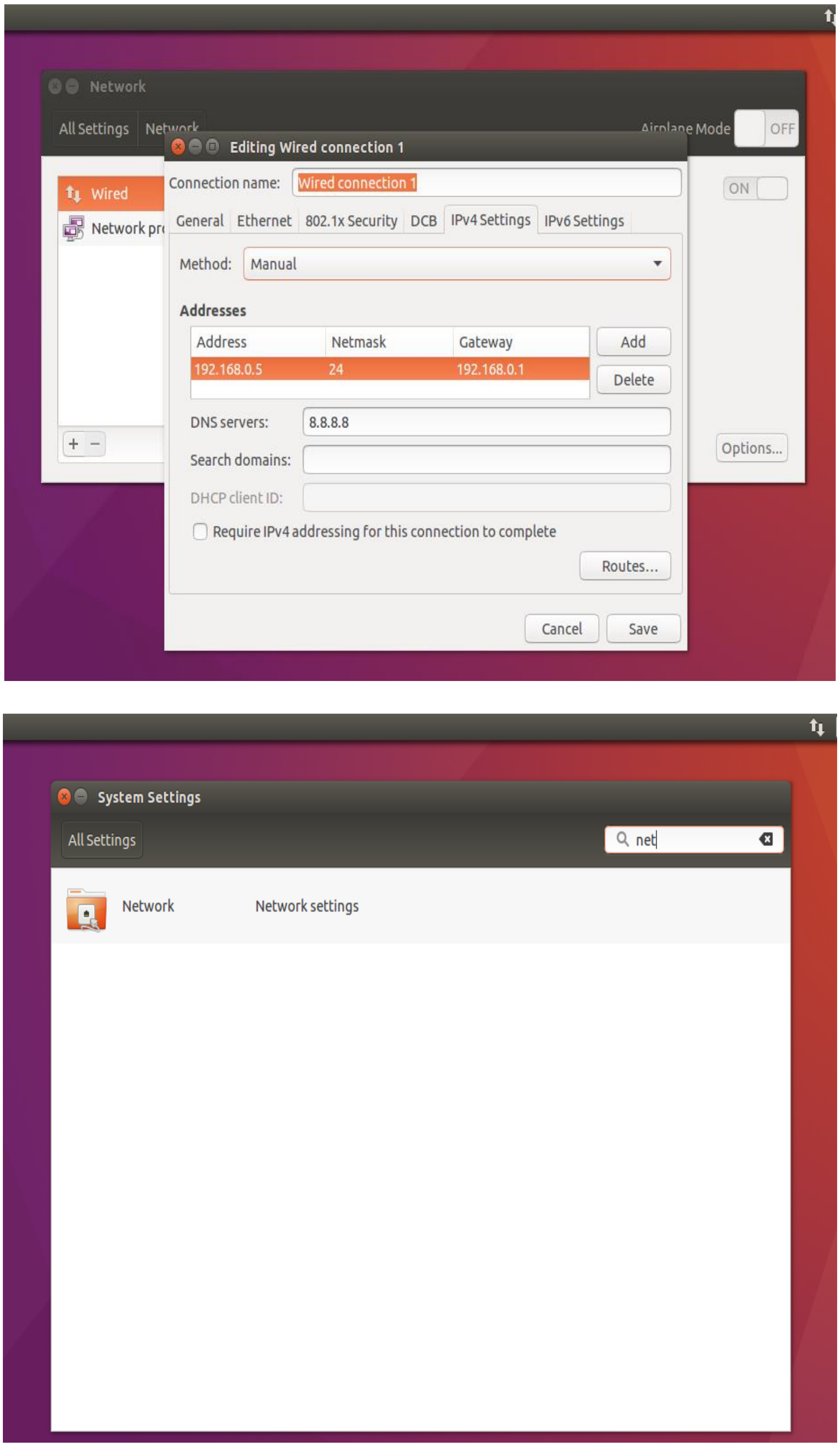
4	Q	81	1.36E+09
5	U	85	1.43E+09
6	E	69	1.16E+09
7	P	80	1.34E+09
8	A	65	1.09E+09
9	S	83	1.39E+09
10	A	65	1.09E+09

The encoding of ASCII is performed by the <ASCII CODE> * 256 * 65536. Here, the ASCII code for character O is 79 (Hex). This encoding enables the transformation of ‘O’ into the realm of ISN numbers. Using the above method, packets are sent to the destination host that expects to receive the information from the server. The capturing of the ISN field of each incoming packet is used to reconstruct a message. This has been practically implemented using two virtual machines and a covert channel code.

In the testbed proposed in this study, two Linux-based machines are installed in a virtual box. Here, Ubuntu 16.04 and Kali are used. One of the virtual machines (VMs) acts as a sender and the other as a receiver. Here, the Ubuntu machine is considered as a sender machine, while Kali Linux acts as a receiver machine.

For sending the data from one machine to another, an internal network is created between the two machines. First, change the adapter settings of the virtual box manager of each machine from Settings>Network>Adapter 1> Attached to: >Internal Network.

In the next step, the IP addresses of the VMs are changed by selecting Ubuntu Linux IP Settings to System Settings… > Network > Wired > Options > IPv4 Settings > Method: Manual and adding the information shown in Figure 5.11. Note that any IP address can be used.



Editing Wired connection 1

Connection name: Wired connection 1

Method: Manual

Addresses

Address	Netmask	Gateway
192.168.0.5	24	192.168.0.1

DNS servers: 8.8.8.8

Search domains:

DHCP client ID:

□ Require IPv4 addressing for this connection to complete

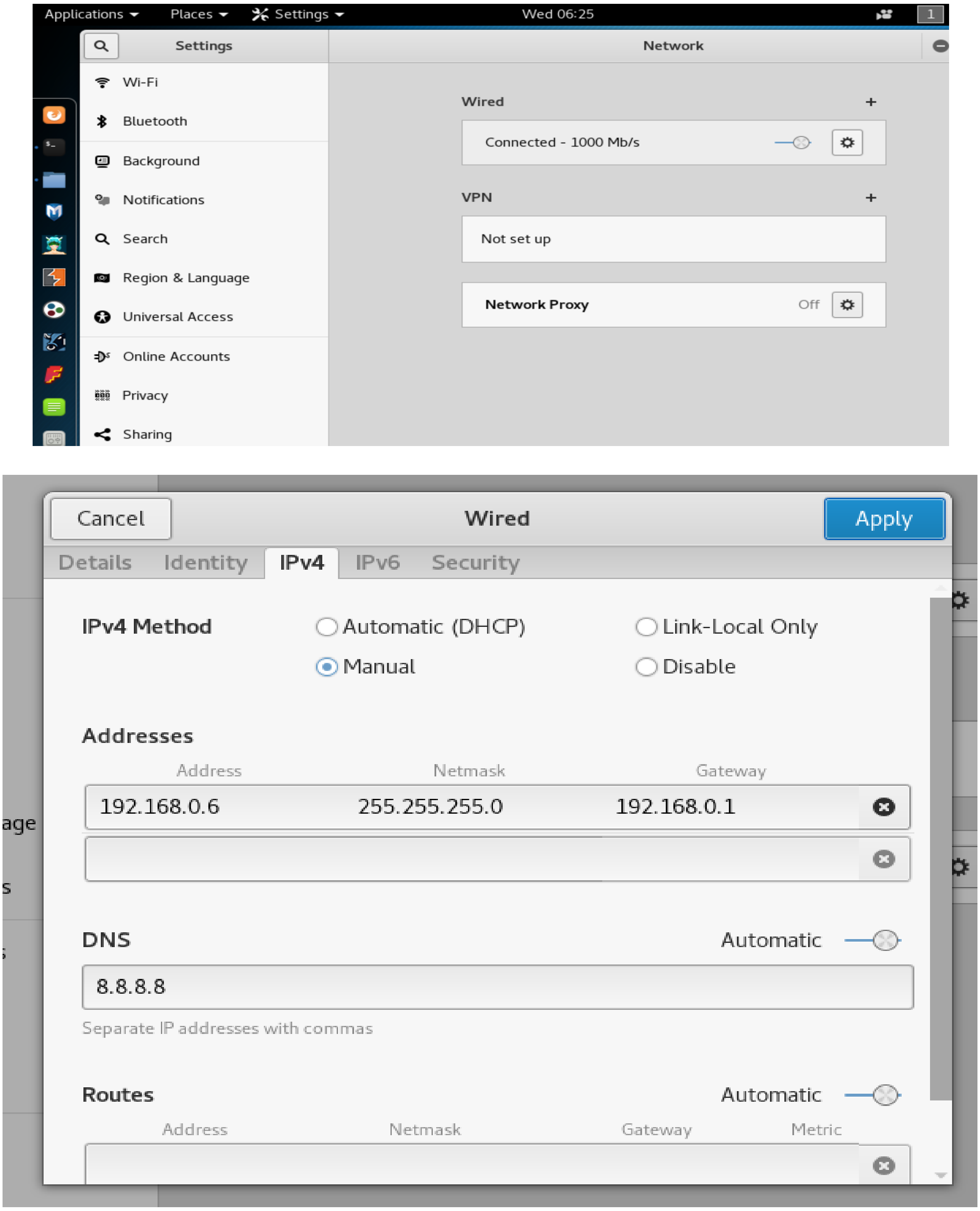
System Settings

net

Network Network settings

Figure 5.11: Network configuration within Ubuntu

Similarly, these steps are followed on Kali Linux by setting Wired connection > Wired settings > Settings > IPv4 Select IPv4 method as Manual and editing the addresses as illustrated in Figure 5.12.



Wired: Connected - 1000 Mb/s

VPN: Not set up

Network Proxy: Off

IPv4 Method: ○ Automatic (DHCP) ○ Link-Local Only ● Manual ○ Disable

Addresses

Address	Netmask	Gateway
192.168.0.6	255.255.255.0	192.168.0.1

DNS Automatic

8.8.8.8

Separate IP addresses with commas

Routes Automatic

Address	Netmask	Gateway	Metric

Figure 5.12: Network configuration within Kali

Now both machines are on the same network. In the next step, covert channels are implemented using the Covert TCP tool in [214]. The tool is executed on both machines and its flow diagram is depicted in Figure 5.13.

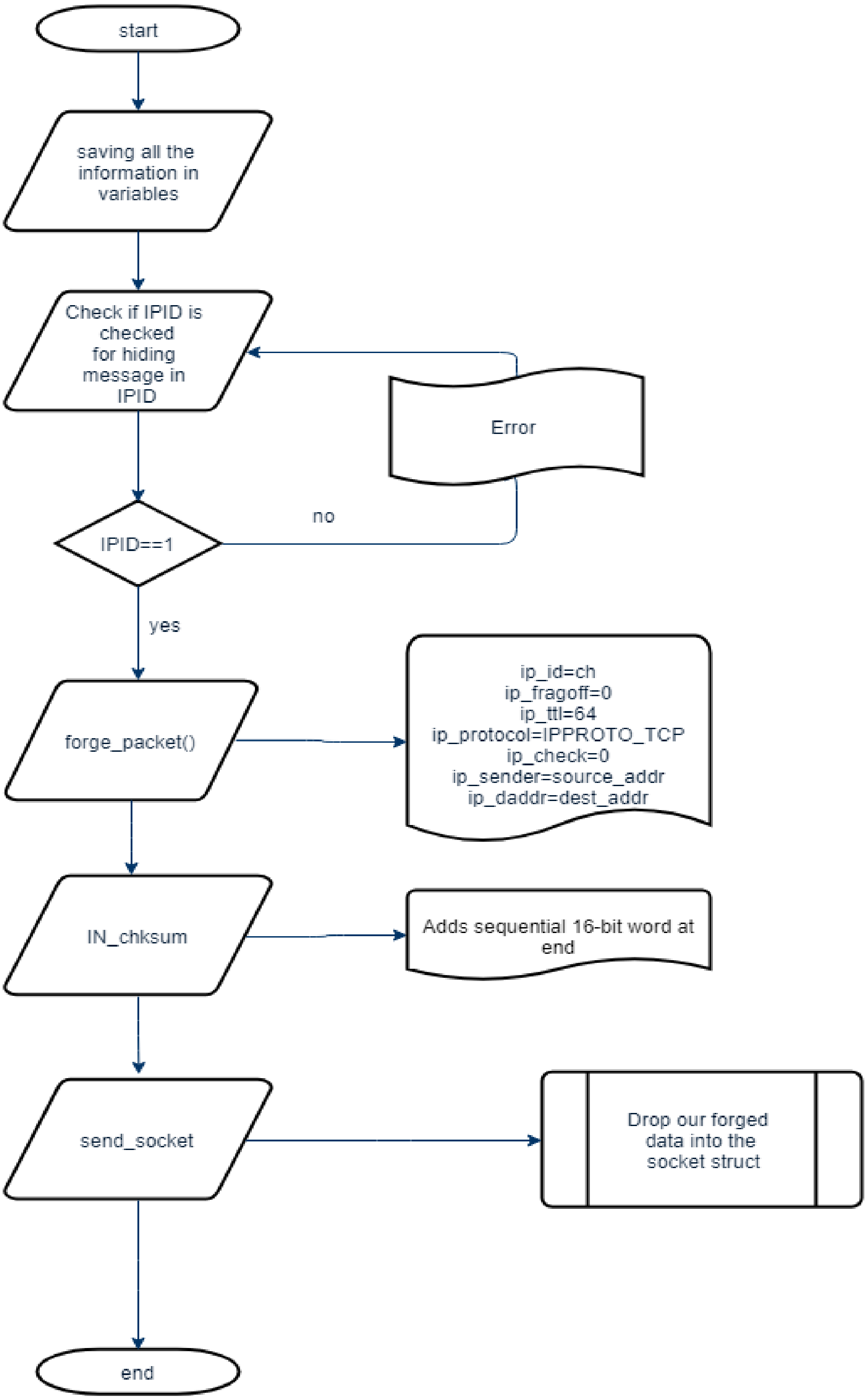

start
saving all the information in variables
Check if IPID is checked for hiding message in IPID
Error
no
IPID==1
yes
forge_packet()
ip_id=ch
ip_fragoff=0
ip_ttl=64
ip_protocol=IPPROTO_TCP
ip_check=0
ip_sender=source_addr
ip_daddr=dest_addr
IN_chksum
Adds sequential 16-bit word at end
send_socket
Drop our forged data into the socket struct
end

Figure 5.13: Flowchart for forging network packet with covert data used by Covert_TCP.c

For hiding data in IPv4 packets and making it a successful covert channel for communication between two ends over the network, the following steps are followed by the Covert TCP tool as illustrated in Figure 5.13:

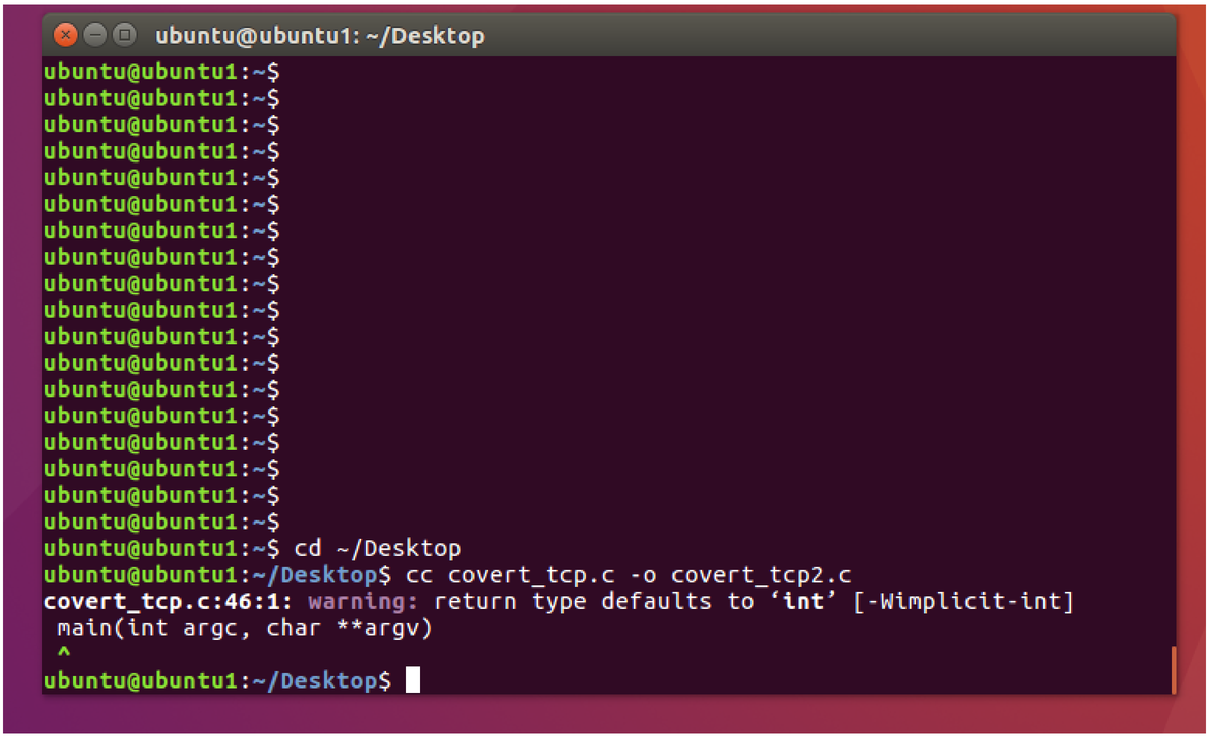
1. At first, all the important data is captured and represented by the following variables and structures:
   - source_address
   - dest_address
   - placeholder
   - protocol
   - tcp_length
   - tcphdr tcp
   - send_socket
   - recv_socket
2. Once all the data is stored, decide on the basis of different checks whether to choose the IPID (IP Identification) based encoding or hide the message within the sequence numbers.
3. Continue if IPID is checked for its Boolean value to be true.
4. To perform the actual encoding, a forge_packets function is called to execute an actual data hiding process. The following entities are used for hiding a message in the IP header:
   - ip_id
   - ip_fragoff
   - ip_ttl
   - ip_protocol
   - ip_check
   - ip_sender
   - ip_daddr
5. Afterwards, an IN_chksum function is called that encodes each alphabet of the hidden message with every packet by sequentially adding a 16-bit word at the end of IPID.

6. Finally, all the forged packets are dropped into the send structure by calling a send_socket function.

To configure the sender machine (Ubuntu), the following process is implemented:

i. We compiled the code file from the terminal using command **cc covert_tcp.c –o covert_tcp2.c** (note that your file should be in the same directory as the one you are compiling from, and otherwise use command **cd ~/Desktop** to change the directory if your code file is at Desktop).

ii. The above command creates a compiled file named as **covert_tcp2.c** in the same directory as shown in Figure 5.14.



```
ubuntu@ubuntu1: ~/Desktop
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$
ubuntu@ubuntu1:~$ cd ~/Desktop
ubuntu@ubuntu1:~/Desktop$ cc covert_tcp.c -o covert_tcp2.c
covert_tcp.c:46:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main(int argc, char **argv)
 ^
ubuntu@ubuntu1:~/Desktop$ 
```

Figure 5.14: Ubuntu terminal window showing successful compiling of Covert_TCP.c

iii. Now enter **cd.** to change the directory back to the root

iv. Manually place the compiled form of the code (**covert_tcp2.c**) in Home > Ubuntu > test (make a new folder named ‘test’ here) as illustrated in Figure 5.15.

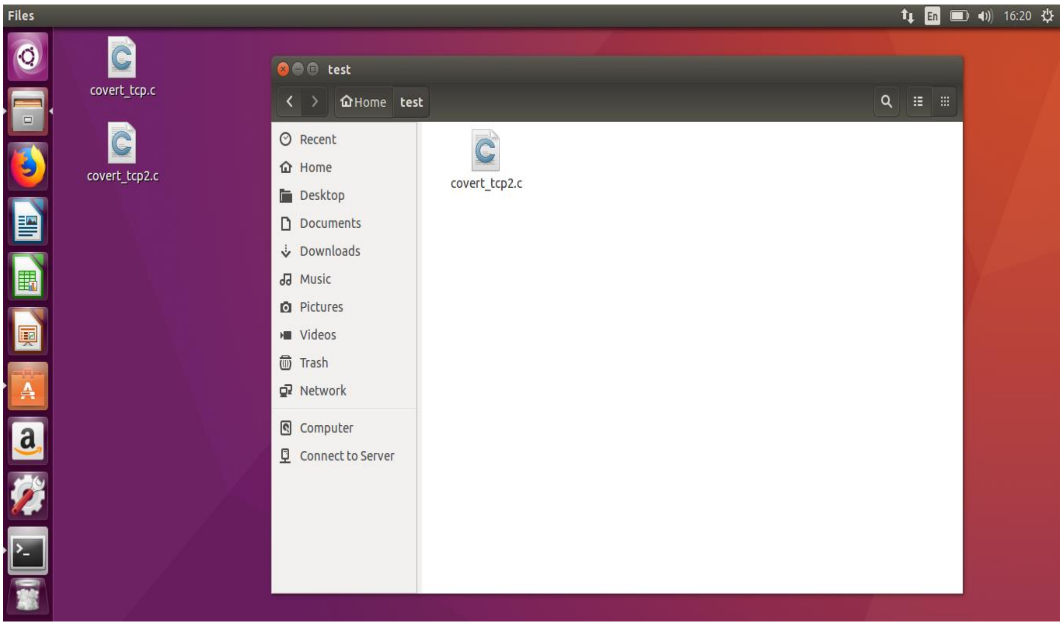


Figure 5.15: Compiled file Covert_TCP2.c

v. Place the text file (name as **send.txt**), which the sender wants to send out, in Computer > tmp as illustrated in Figure 5.16.

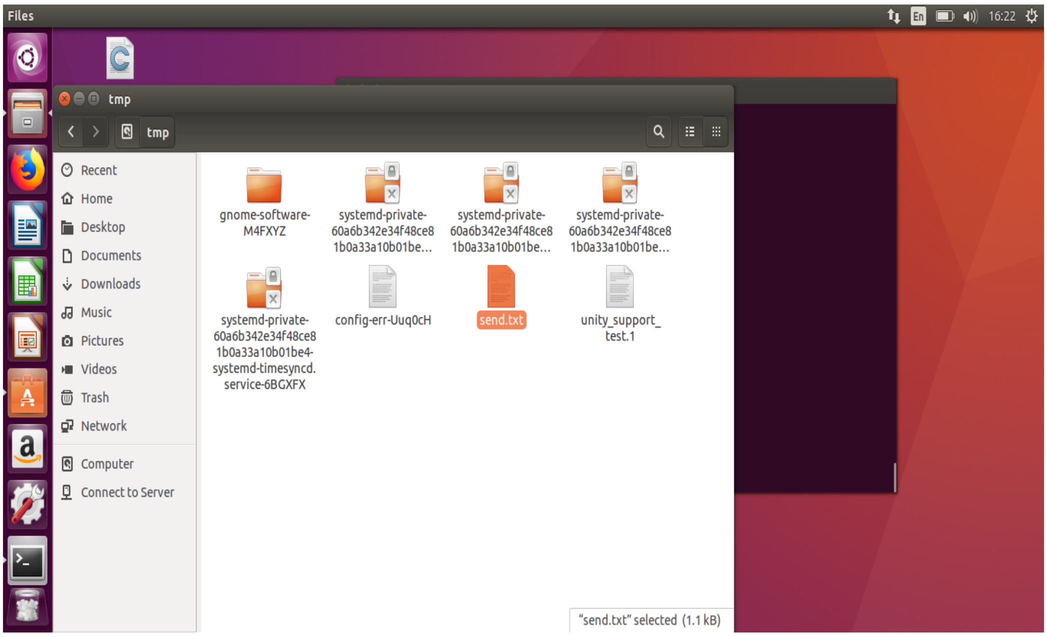
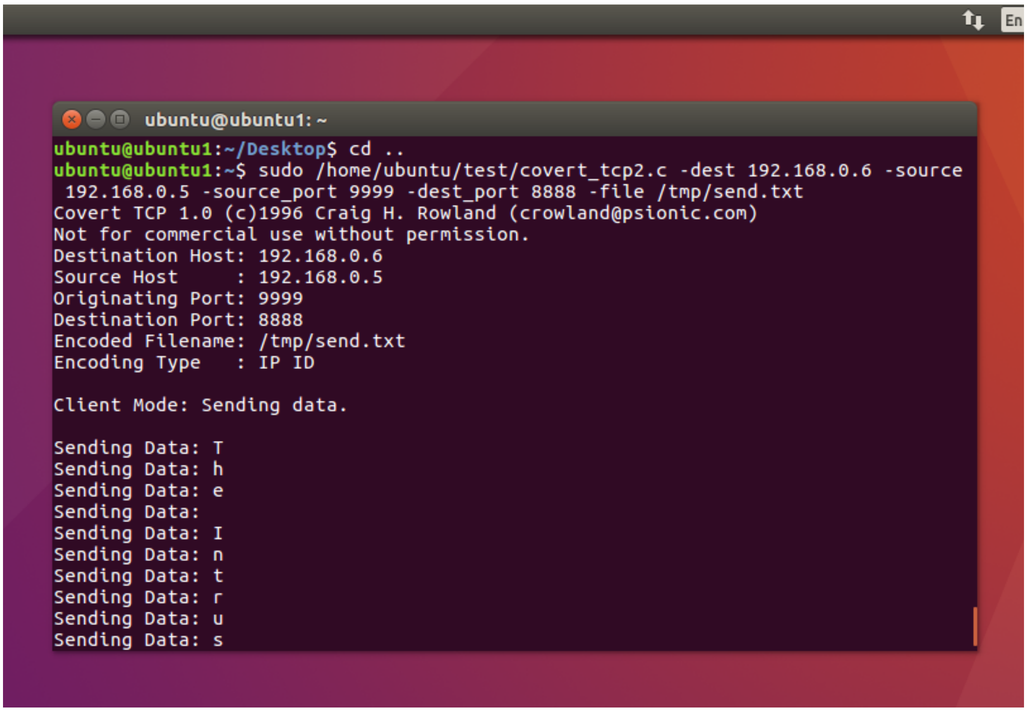

Files
16:22
tmp
tmp
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
Connect to Server
gnome-software-M4FXYZ
systemd-private-60a6b342e34f48ce81b0a33a10b01be...
systemd-private-60a6b342e34f48ce81b0a33a10b01be...
systemd-private-60a6b342e34f48ce81b0a33a10b01be...
systemd-private-60a6b342e34f48ce81b0a33a10b01be4-systemd-timesyncd.service-6BGXFX
config-err-Uuq0cH
send.txt
unity_support_test.1
"send.txt" selected (1.1 kB)

Figure 5.16: Temporary folder (tmp) showing the send.txt file that contains covert data to be send from source 192.168.0.5 to destination 192.168.0.6

vi. The following command is run from the terminal by the sender, which is also shown in Figure 5.17:

sudo /home/ubuntu/test/covert_tcp2.c –dest 192.168.0.6 –source 192.168.0.5 –source_port 9999 –dest_port 8888 –file /tmp/send.txt



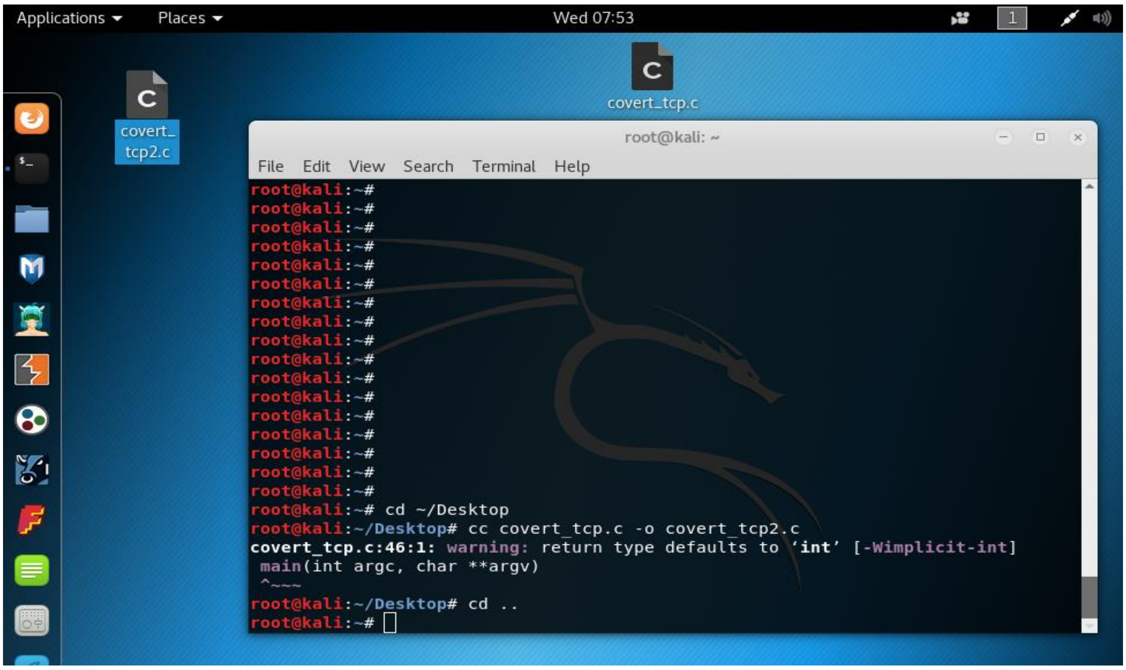
```
ubuntu@ubuntu1:~/Desktop$ cd ..
ubuntu@ubuntu1:~$ sudo /home/ubuntu/test/covert_tcp2.c -dest 192.168.0.6 -source
 192.168.0.5 -source_port 9999 -dest_port 8888 -file /tmp/send.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Destination Host: 192.168.0.6
Source Host      : 192.168.0.5
Originating Port: 9999
Destination Port: 8888
Encoded Filename: /tmp/send.txt
Encoding Type    : IP ID

Client Mode: Sending data.

Sending Data: T
Sending Data: h
Sending Data: e
Sending Data:
Sending Data: I
Sending Data: n
Sending Data: t
Sending Data: r
Sending Data: u
Sending Data: s
```

Figure 5.17: Ubuntu terminal showing the command used to send the file (send.txt) from source 192.168.0.5 to destination 192.168.0.6

Similarly, the receiver machine (Kali) is set up as follows:

vii. Compile the code file from the terminal using command **cc covert_tcp.c –o covert_tcp2.c.**

viii. This creates a compiled file named as **covert_tcp2.c,** which is shown in Figure 5.18.

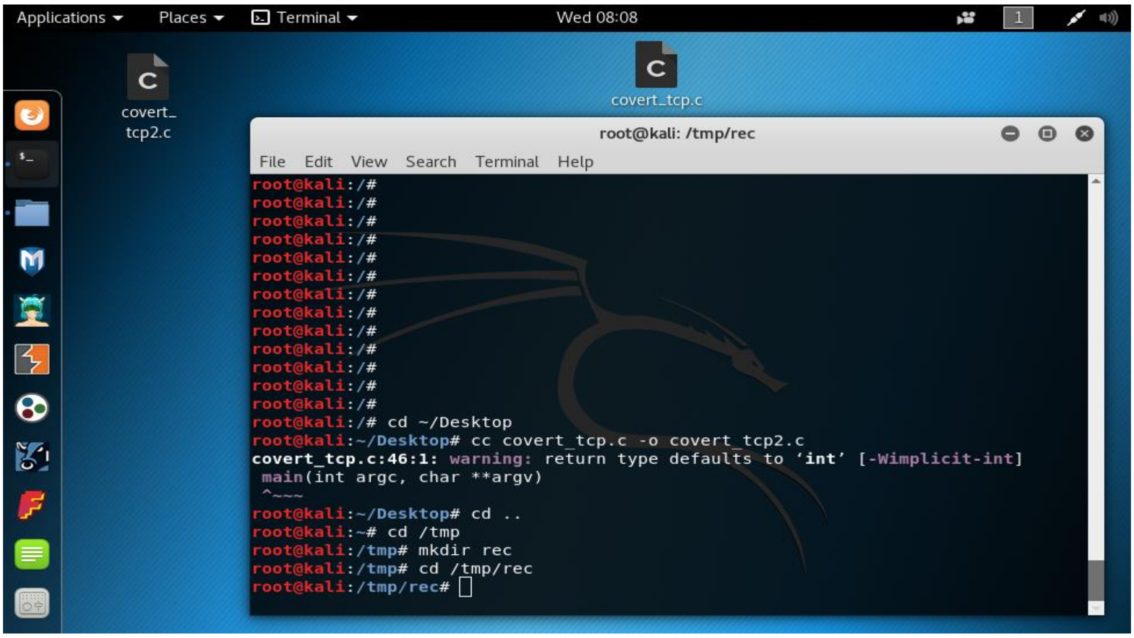


```
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~#
root@kali:~# cd ~/Desktop
root@kali:~/Desktop# cc covert_tcp.c -o covert_tcp2.c
covert_tcp.c:46:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main(int argc, char **argv)
 ^~~~
root@kali:~/Desktop# cd ..
root@kali:~# 
```

Figure 5.18: Kali terminal window showing successful compiling of Covert_TCP.c

ix. Enter **cd ..** to change the directory back to the root.

x. Open a 'tmp' folder from the terminal using command **cd ~/tmp** and then create a new sub-folder 'rec' using command **mkdir rec**, as shown in Figure 5.19.



```
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/#
root@kali:/# cd ~/Desktop
root@kali:~/Desktop# cc covert_tcp.c -o covert_tcp2.c
covert_tcp.c:46:1: warning: return type defaults to 'int' [-Wimplicit-int]
 main(int argc, char **argv)
 ^~~~
root@kali:~/Desktop# cd ..
root@kali:~# cd /tmp
root@kali:/tmp# mkdir rec
root@kali:/tmp# cd /tmp/rec
root@kali:/tmp/rec# 
```

Figure 5.19: Creation of "rec" folder within temporary folder (tmp) where data will be received

xi. Manually place the compiled form of the code (**covert_tcp2.c**) in Computer > tmp > rec (see Figure 5.20).

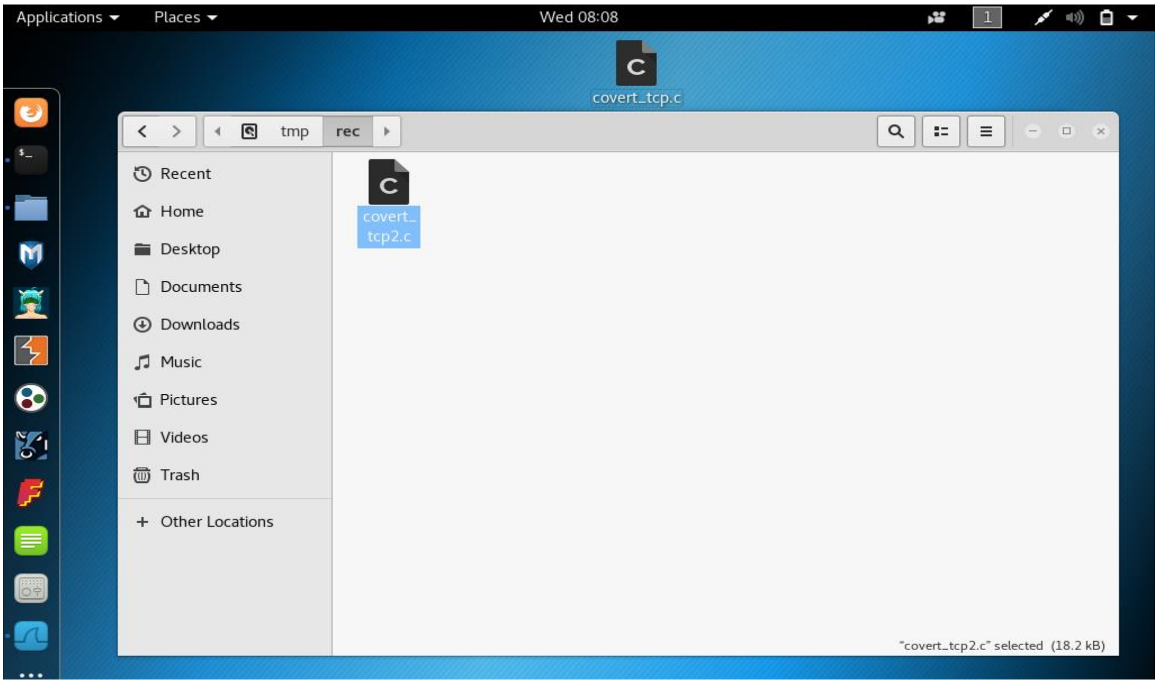
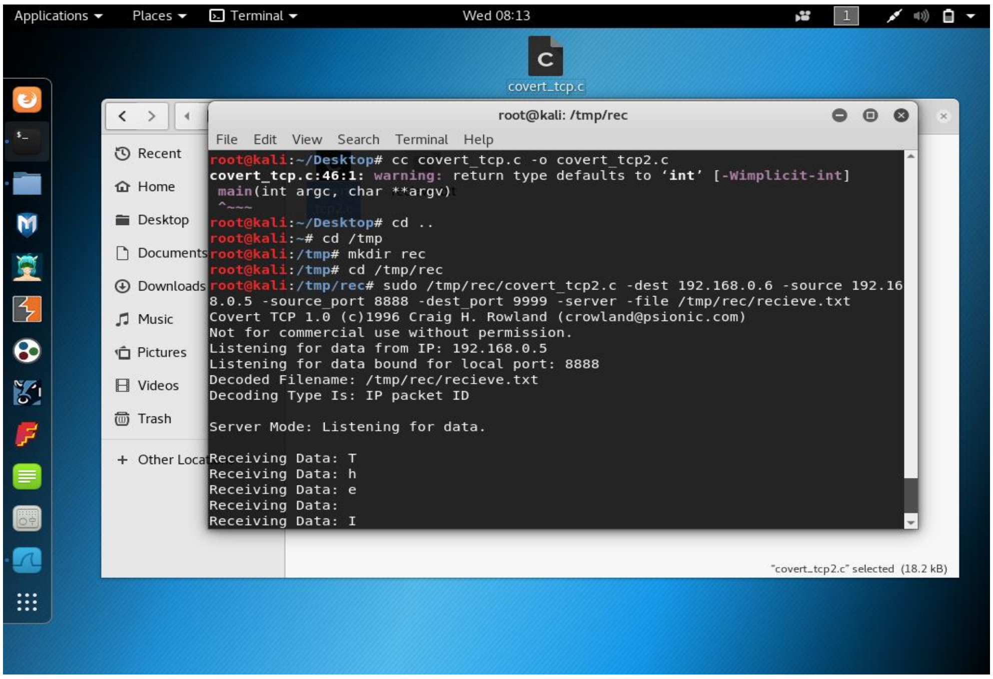


Figure 5.20: Location for compiled covert_tcp2 file within "rec" folder in tmp

xii. Run the following command by the receiver as shown in Figure 5.21:

sudo tmp/rec/covert_tcp2.c –dest 192.168.0.6 –source 192.168.0.5 –source_port 9999 –dest_port 8888 –server –file /tmp/rec/receive.txt



```
root@kali:~/Desktop# cc covert_tcp.c -o covert_tcp2.c
covert_tcp.c:46:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main(int argc, char **argv)
 ^~~~
root@kali:~/Desktop# cd ..
root@kali:~# cd /tmp
root@kali:/tmp# mkdir rec
root@kali:/tmp# cd /tmp/rec
root@kali:/tmp/rec# sudo /tmp/rec/covert_tcp2.c -dest 192.168.0.6 -source 192.16
8.0.5 -source_port 8888 -dest_port 9999 -server -file /tmp/rec/recieve.txt
Covert TCP 1.0 (c)1996 Craig H. Rowland (crowland@psionic.com)
Not for commercial use without permission.
Listening for data from IP: 192.168.0.5
Listening for data bound for local port: 8888
Decoded Filename: /tmp/rec/recieve.txt
Decoding Type Is: IP packet ID

Server Mode: Listening for data.

Receiving Data: T
Receiving Data: h
Receiving Data: e
Receiving Data: 
Receiving Data: I
```

Figure 5.21: Kali terminal showing execution of covert_tcp2 file where it is listening for data from source IP 192.168.0.5 and saves received data in file 'receive.txt'

xiii. The above command creates a new text file with name **“receive.txt”** defined in the command (see Figure 5.22).

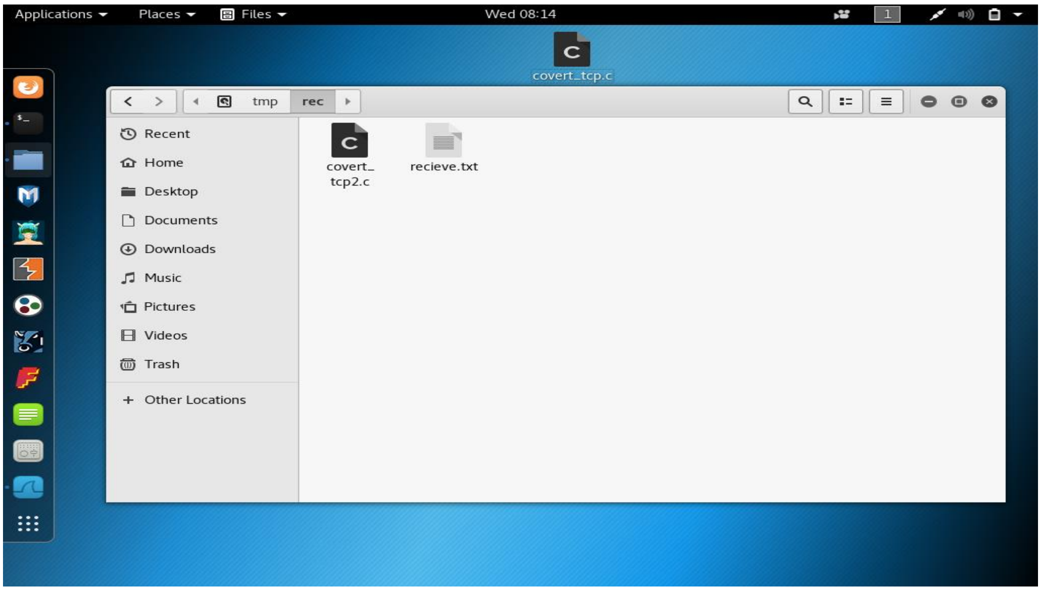
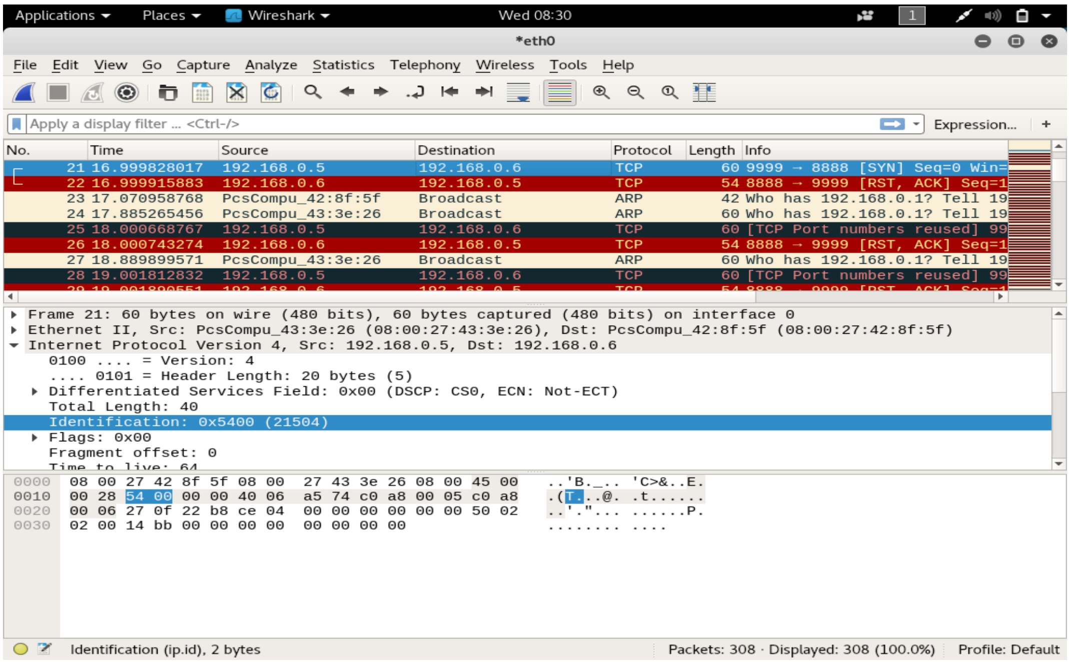


Figure 5.22: /tmp/rec folder showing receive.txt file

Wireshark is a packet capture and analysis tool widely used by both the research and development community. The Wireshark screen in Figure 5.23 shows that the covert data sent from the sender machine to the receiver machine is added in the IPIDs of the receiving packets (e.g. the IPID of the first receiving packet contains ‘T’).



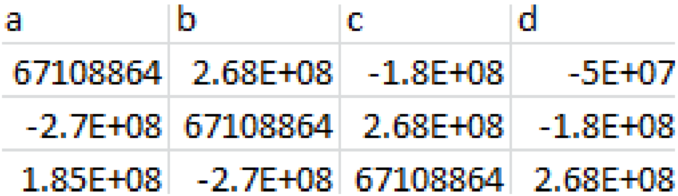
```
No.  Time          Source              Destination   Protocol Length Info
  21 16.999828017  192.168.0.5         192.168.0.6   TCP          60 9999 → 8888 [SYN] Seq=0 Win=
  22 16.999915883  192.168.0.6         192.168.0.5   TCP          54 8888 → 9999 [RST, ACK] Seq=1
  23 17.070958768  PcsCompu_42:8f:5f   Broadcast     ARP          42 Who has 192.168.0.1? Tell 19
  24 17.885265456  PcsCompu_43:3e:26   Broadcast     ARP          60 Who has 192.168.0.1? Tell 19
  25 18.000668767  192.168.0.5         192.168.0.6   TCP          60 [TCP Port numbers reused] 99
  26 18.000743274  192.168.0.6         192.168.0.5   TCP          54 8888 → 9999 [RST, ACK] Seq=1
  27 18.889899571  PcsCompu_43:3e:26   Broadcast     ARP          60 Who has 192.168.0.1? Tell 19
  28 19.001812832  192.168.0.5         192.168.0.6   TCP          60 [TCP Port numbers reused] 99

▸ Frame 21: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_43:3e:26 (08:00:27:43:3e:26), Dst: PcsCompu_42:8f:5f (08:00:27:42:8f:5f)
▾ Internet Protocol Version 4, Src: 192.168.0.5, Dst: 192.168.0.6
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0x5400 (21504)
  ▸ Flags: 0x00
    Fragment offset: 0

0000  08 00 27 42 8f 5f 08 00  27 43 3e 26 08 00 45 00   ..'B._.. 'C>&..E.
0010  00 28 54 00 00 00 40 06  a5 74 c0 a8 00 05 c0 a8   .(T...@. .t......
0020  00 06 27 0f 22 b8 ce 04  00 00 00 00 00 00 50 02   ..'."... ......P.
0030  02 00 14 bb 00 00 00 00  00 00 00 00              ........ ....

Identification (ip.id), 2 bytes          Packets: 308 · Displayed: 308 (100.0%)   Profile: Default
```

Figure 5.23: Wireshark showing data packets

The above method for the generation of covert network packets has been used in [215, 216] for detection algorithm testing. It has also been employed for the evaluation of a Support Vector Machine based detection model in [213].

Algorithm step 2: Compute p(a, b, c, d)

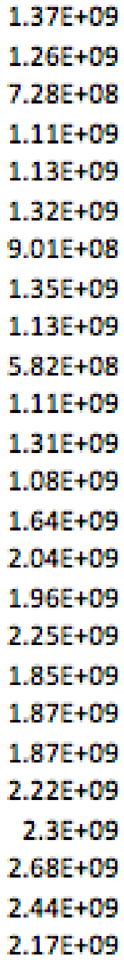
This computation has produced the following results by using the distance between vectors:



a	b	c	d
67108864	2.68E+08	-1.8E+08	-5E+07
-2.7E+08	67108864	2.68E+08	-1.8E+08
1.85E+08	-2.7E+08	67108864	2.68E+08

Algorithm step 3: Calculate distance between the vectors

This step is to obtain the distance vector d = [$d_{j,1}$, $d_{j,2},\ldots,d_{j,j}$]. A snippet of the distance calculation is shown below:



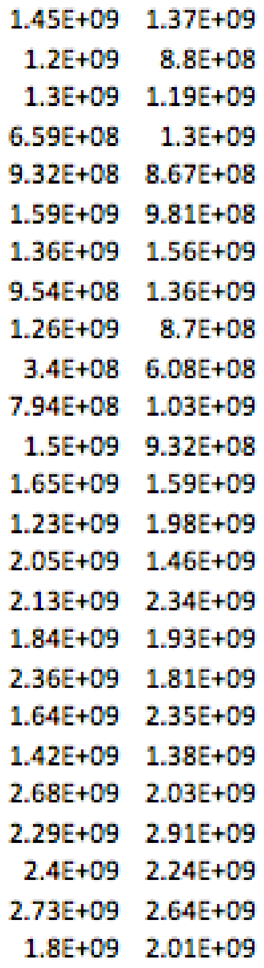
```
1.37E+09
1.26E+09
7.28E+08
1.11E+09
1.13E+09
1.32E+09
9.01E+08
1.35E+09
1.13E+09
5.82E+08
1.11E+09
1.31E+09
1.08E+09
1.64E+09
2.04E+09
1.96E+09
2.25E+09
1.85E+09
1.87E+09
1.87E+09
2.22E+09
 2.3E+09
2.68E+09
2.44E+09
2.17E+09
```

Algorithm Step 4: Get the second-order statistics

This step is to computeσ^2_j = *variance*(*d*) for *ISN*(*j*), which has resulted in σ^2_j = 3.56E+17.

Algorithm Step 5: Obtain σ^2_{j-1} and σ^2_{j-2}

Repeat step 1 to 4 for *ISN*(*j-1*) and *ISN*(*j-2*) with the distance calculation results shown below:



1.45E+09	1.37E+09
1.2E+09	8.8E+08
1.3E+09	1.19E+09
6.59E+08	1.3E+09
9.32E+08	8.67E+08
1.59E+09	9.81E+08
1.36E+09	1.56E+09
9.54E+08	1.36E+09
1.26E+09	8.7E+08
3.4E+08	6.08E+08
7.94E+08	1.03E+09
1.5E+09	9.32E+08
1.65E+09	1.59E+09
1.23E+09	1.98E+09
2.05E+09	1.46E+09
2.13E+09	2.34E+09
1.84E+09	1.93E+09
2.36E+09	1.81E+09
1.64E+09	2.35E+09
1.42E+09	1.38E+09
2.68E+09	2.03E+09
2.29E+09	2.91E+09
2.4E+09	2.24E+09
2.73E+09	2.64E+09
1.8E+09	2.01E+09

Based on the above results, σ^2_{j-1} and σ^2_{j-2} for *ISN(j-1)* and *ISN(j-2)* have been calculated as σ^2_{j-1} = 3.76E+17 and σ^2_{j-2} = 3.74E+17, respectively.

Algorithm Step 6: Calculate the third-order statistics

Based on the computed results forσ^2_{j}, σ^2_{j-1} and σ^2_{j-2}, the variance calculation has produced $var_\sigma = var(\sigma^2_j, \sigma^2_{j-1}, \sigma^2_{j-2})$ = 1.14E+32.

Algorithm steps 7, 8 and 9: Compare var$_\sigma$ with $Threshold_{ISN}$

If var_σ is greater than $Threshold_{ISN}$ then the ISNs are non-malicious, and otherwise they are malicious. As $Threshold_{ISN}$ was computed earlier as 2.62E+32 and $var_\sigma$is equal to 1.14E+32, covert ISNs are confirmed due to $var_\sigma < Threshold_{ISN}$.

5.4.2 Experiment 2 – Demo Data (Random)

The objective of this experiment is to reinforce the *Threshold$_{ISN}$* model by using it as a benchmark value for testing independent TCP packets in the context of sending information as covert ISN values in the TCP headers. The difference form the previous experiment is the *Threshold$_{ISN}$* value produced by the proposed data leakage detection algorithm, which is constant.

5.4.2.1 Set-up for Experiment

The following explains the set-up of an experiment that is conducted for testing the *Threshold$_{ISN}$* using an independent data set.

5.4.2.2 Operating System

The set-up of the experiment was conducted on Ubuntu Linux VM that is mounted using Oracle Virtual Box on a Windows 7 computer. The choice of the operating system is dictated by the specification of the covert data creation code called Covert_TCP (https://dunnesec.com/category/tools/covert_tcp/).

5.4.2.3 Code for Covert Data Creation

As mentioned above, the Covert_TCP code is used for creating TCP packets. The code written in C is compiled on an Ubuntu Linux terminal. The code requires the passing of a switch value that informs the creation of TCP packets where ISNs are the choice for covert channel communication.

5.4.2.4 Source of Data

The source of plaintext data that is sent via Covert_TCP from the server-side to the client side is created by Lorem Ipsum Generator (https://www.lipsum.com/). The generator is instructed to generate a number of random words without the consideration of some characteristics such as grammar usage. This has been the best source of data as it is truly random and hence fulfilling one of the critical requirements of the experiment conducted in this study.

5.4.2.5 Server and Client

The experiment requires two instances of Covert_TCP, which run on the same Ubuntu Linux VM.

The server instance of Covert_TCP requires:

- A folder containing a file 'send.txt' that contains random text generated using Lorem Ipsum Generator.
- Source IP address and port number for socket creation.
- Destination IP address and port number to send the data to or where the client is listening on.
- Switch value to inform that the passed data must be transformed and sent using the ISN field in the TCP header.

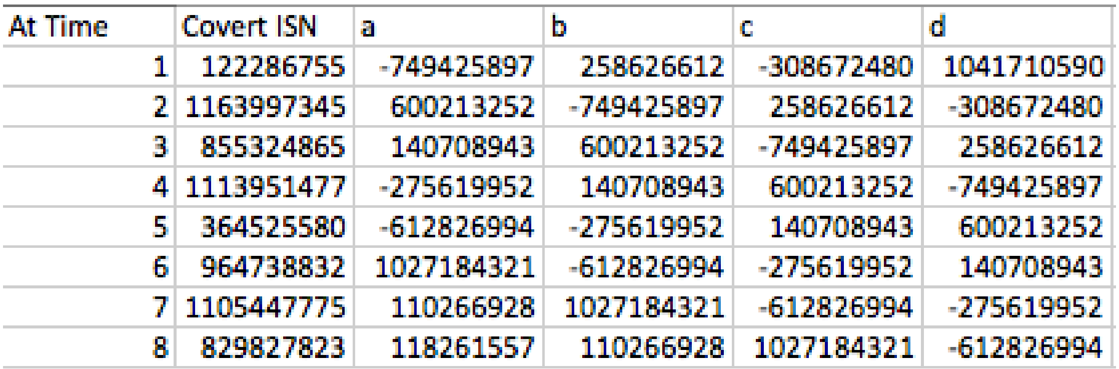
The client instance of Covert_TCP requires:

- A folder where the received data will be placed.
- Source IP address and port number for socket creation.
- Destination IP address and port number it is listening on.
- Switch value to inform that incoming TCP packets require the ISN field to be transformed back to plaintext.

5.4.2.6 Data Capture

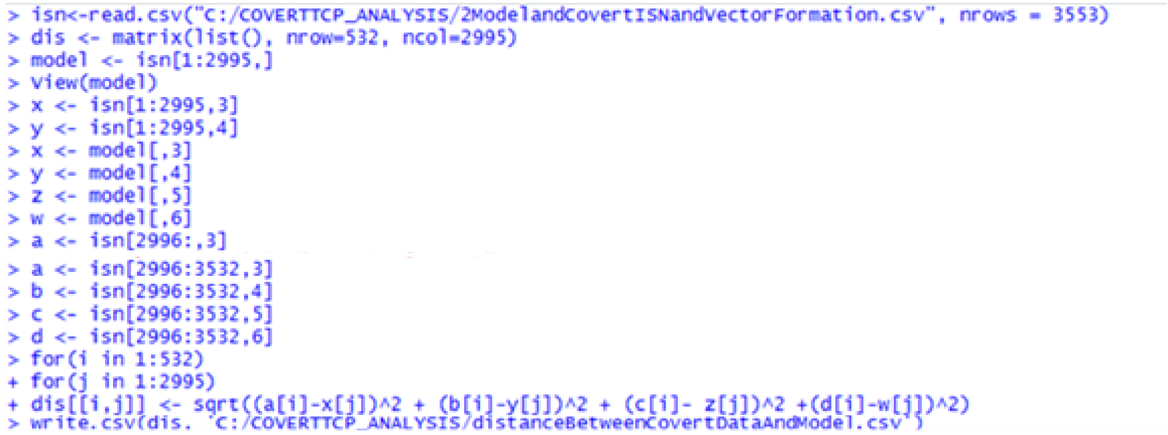
Between the server and the client, there is also a TCP Dump program (that is shipped with Ubuntu Linux), that is used to capture the TCP packets sent from the server to the client. This program only requires the IP address and port number that it needs to bind itself to and the location where it should write captured TCP packets. The captured TCP packets are written in a *.pcap file. The captured data in the *.pcap file is processed using Wireshark to export the ISN field from TCP packets to a *.csv file.Data Leakage Detection Algorithm

Having captured the TCP packets, the step 2 of the algorithm in Table 4.2 is executed to calculate *p*(*a, b, c, d*) with a snippet of the calculated results shown below:



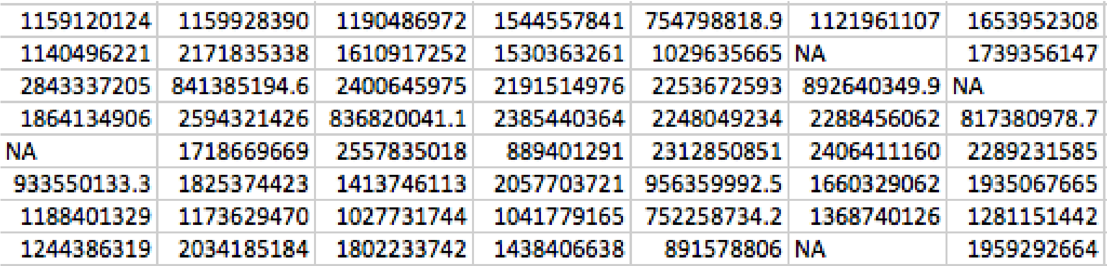
At Time	Covert ISN	a	b	c	d
1	122286755	-749425897	258626612	-308672480	1041710590
2	1163997345	600213252	-749425897	258626612	-308672480
3	855324865	140708943	600213252	-749425897	258626612
4	1113951477	-275619952	140708943	600213252	-749425897
5	364525580	-612826994	-275619952	140708943	600213252
6	964738832	1027184321	-612826994	-275619952	140708943
7	1105447775	110266928	1027184321	-612826994	-275619952
8	829827823	118261557	110266928	1027184321	-612826994

The step 3 of the algorithm is then activated to generate distance vector *d* = [$d_{j,1}$, $d_{j,2}$,…,$d_{j,j}$]. The main purpose of this step is generating the distance vector.



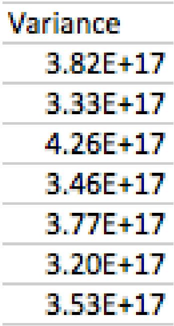
```
> isn<-read.csv("C:/COVERTTCP_ANALYSIS/2ModelandCovertISNandVectorFormation.csv", nrows = 3553)
> dis <- matrix(list(), nrow=532, ncol=2995)
> model <- isn[1:2995,]
> View(model)
> x <- isn[1:2995,3]
> y <- isn[1:2995,4]
> x <- model[,3]
> y <- model[,4]
> z <- model[,5]
> w <- model[,6]
> a <- isn[2996:,3]
> a <- isn[2996:3532,3]
> b <- isn[2996:3532,4]
> c <- isn[2996:3532,5]
> d <- isn[2996:3532,6]
> for(i in 1:532)
+ for(j in 1:2995)
+ dis[[i,j]] <- sqrt((a[i]-x[j])^2 + (b[i]-y[j])^2 + (c[i]- z[j])^2 +(d[i]-w[j])^2)
> write.csv(dis, 'C:/COVERTTCP_ANALYSIS/distanceBetweenCovertDataAndModel.csv')
```

The following shows a snippet of the results from the Euclidean distance calculation between a test vector and the training vectors.



1159120124	1159928390	1190486972	1544557841	754798818.9	1121961107	1653952308
1140496221	2171835338	1610917252	1530363261	1029635665	NA	1739356147
2843337205	841385194.6	2400645975	2191514976	2253672593	892640349.9	NA
1864134906	2594321426	836820041.1	2385440364	2248049234	2288456062	817380978.7
NA	1718669669	2557835018	889401291	2312850851	2406411160	2289231585
933550133.3	1825374423	1413746113	2057703721	956359992.5	1660329062	1935067665
1188401329	1173629470	1027731744	1041779165	752258734.2	1368740126	1281151442
1244386319	2034185184	1802233742	1438406638	891578806	NA	1959292664

The above calculated results allow the step 4 of the algorithm to computeσ^2_j = *variance(d)*. The following shows a snippet of the results from the variance computation, which denote the amount of change to each distance calculation. The different values are showing the amount of change from each distance.



Variance
3.82E+17
3.33E+17
4.26E+17
3.46E+17
3.77E+17
3.20E+17
3.53E+17

Figure 5.24: Second order variance computation snippet

Afterwards repeat the above steps for *ISN(j-1)* and *ISN(j-2)* to obtain σ^2_{j-1} and σ^2_{j-2}, as defined in the step 5 of the algorithm.

Based on the above computed variances, the step 6 of the algorithm calculates $var_\sigma = var(\sigma^2_j, \sigma^2_{j-1}, \sigma^2_{j-2})$, resulting in var_σ = 1.02E+32.

By comparing $var_\sigma$with $Threshold_{ISN}$, the decision on whether the ISNs are malicious can be made, as specified in the steps 7, 8 and 9 of the algorithm. Similar to the first experiment, covert ISNs are found due to var_σ(= 1.02E+32) < $Threshold_{ISN}$ (= 2.62E+32).

Again, the steps 10 and 11 of the algorithm are not included in the experiment.

The results of this experiment reiterate that the $Threshold_{ISN}$ value produced by the proposed data leakage detection algorithm in Table 4.2 is constant. Here the constant value refers to the fact that the test data is not a subset of original modelling as observed in the previous experiment.

5.4.3 Experiments 3, 4 and 5

The scenarios of the experiments are explained below:

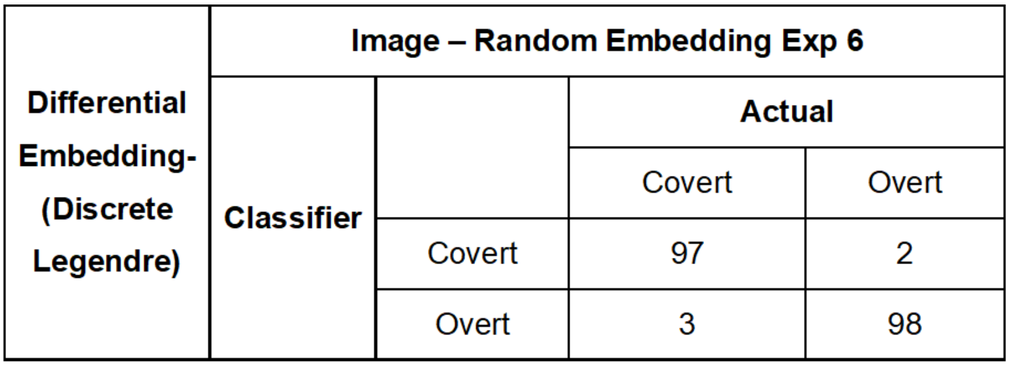
- *Experiment 3* - The data for this experiment has been strategically created in order to have covert sequence numbers in an alternate pattern, whereby, the first TCP packet has a true ISN, while the second TCP packet has a covert ISN inserted into the sequence number field.
- *Experiment 4* – The data for this experiment has been created in order to have covert sequence numbers appearing together in a bulk of six consecutively placed TCP packets followed by a true sequence number.

- *Experiment 5* – The data for this experiment has no covert entries and all sequence numbers are true.

The set-up of the experiments is the same as the one for the second experiment introduced in sections 5.4.2.2 – 5.4.2.7.

This experiment has been done with the help of a Discrete Legendre polynomial instead of the first-order difference. The data for this experiment was recorded using the same testbed described before. 2,000 ISN samples have been gathered from the network packets collected on the testbed.

The 2,000 ISN samples were divided into sets of 1,400 and 600. All the initial training phase steps (see section 5.2) were performed to find a phase space and the variance for 1,400 ISNs. The remaining 600 ISN values were used for covert embedding with 200 ISNs chosen randomly. The threshold is then calculated. An image has been used to perform covert embedding in this experiment. The image was converted to a 32 bit. The grey level values were then embedded using randomly picked ISN numbers, sometimes adding to the ISN numbers and sometimes simply replacing them, while making sure that the overall value does not exceed the limit for a 32-bit number (4,294,967,295). The result of applying the algorithm in Table 4.2 to this experiment is given in Figure 5.25. The results indicate the covert and overt classifier results.

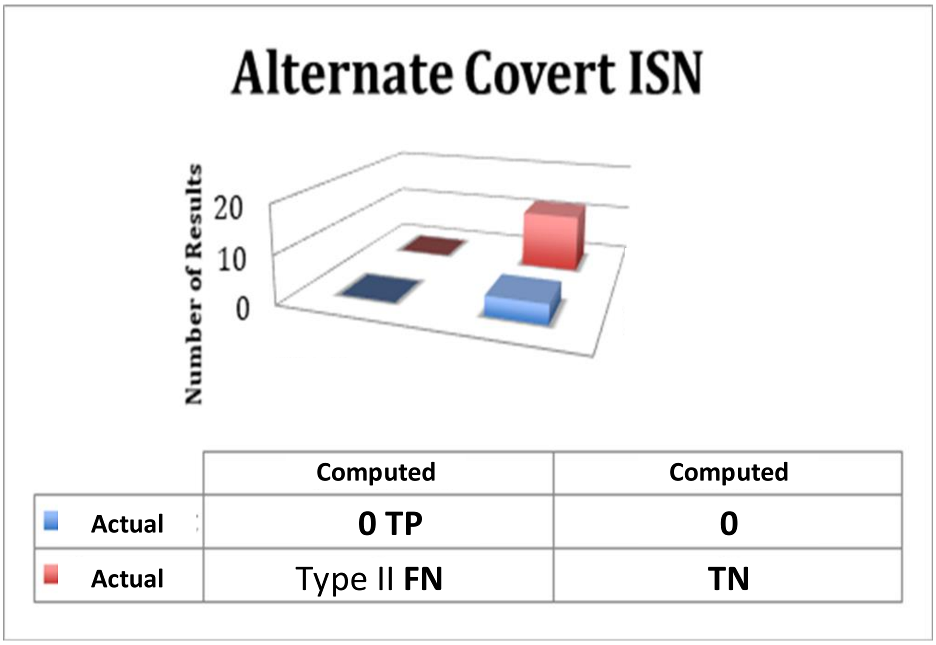


Differential Embedding- (Discrete Legendre)	**Image – Random Embedding Exp 6**			
	Classifier		**Actual**	
			Covert	Overt
		Covert	97	2
		Overt	3	98

Figure 5.25: Discrete Legendre based method (covert embedding)

5.5 Outcomes of Experiments

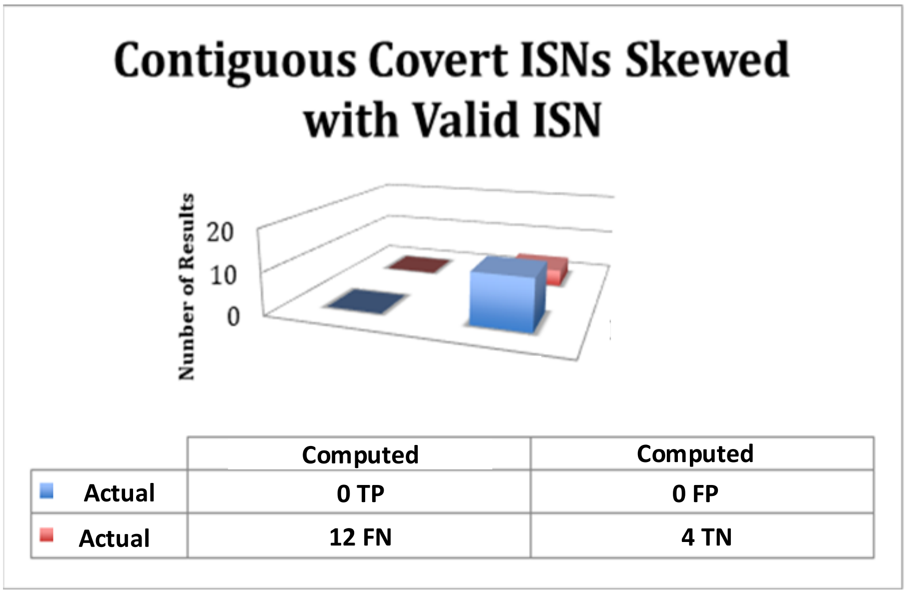
In experiment 3, it is found that with an alternate pattern of covert ISNs injected into TCP packets, the proposed algorithm in Table 4.2 is successful in locating malicious entries of the covert data in TCP packets. The number of true negatives is 3 out of 4 while the number of false negatives is 1 out of 4, causing a Type II error. Figure 5.26 shows the algorithm accuracy for alternate covert ISNs with ½ of packets being covert.



	Computed	Computed
🟦 Actual	0 TP	0
🟥 Actual	Type II **FN**	**TN**

Figure 5.26: Showing algorithm accuracy for alternate covert ISNs with 1/2 of packets being covert

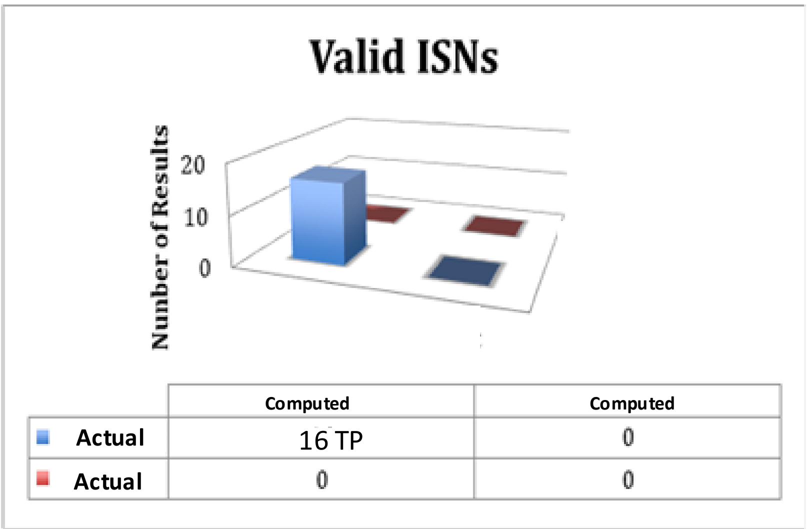
In experiment 4 with six consecutive entries of covert ISNs injected into TCP packets, the algorithm is not very successful in locating malicious entries of the covert data. The number of true negatives is 1 out of 4 while the number of false negatives is 3 out of 4, causing a Type II error. In the case where all entries are found to be covert, the algorithm is found to be very effective. Figure 5.27 shows the accuracy for contiguous covert ISNs.



	Computed	Computed
🟦 Actual	0 TP	0 FP
🟥 Actual	12 FN	4 TN

Figure 5.27: Showing algorithm accuracy for contiguous covert ISNs

In experiment 5 with all true entries of ISNs in TCP packets, the algorithm is very successful in identifying normal entries of the data because all entries are found to be covert. The number of true positive is 4 out of 4. Figure 2.28 shows the algorithm accuracy for a normal stream of data.



	Computed	Computed
🟦 Actual	16 TP	0
🟥 Actual	0	0

Figure 5.28: Showing algorithm accuracy for a normal stream of data

So far, the evaluation of the outlier quantification algorithm is presented. The covert data is injected into the ISN field on a TCP packet using a strategy for ASCII conversion, while ensuring that covert data is sent in the contiguous stream, alternate method, or with a known gap of valid sequence numbers. It was observed that Type II errors are found low to the amount of 20% for 2 out of 5 experiments, while a Type I error is found for making decisions about data leakage, thereby increasing the level of accuracy as well as reducing the likelihood of Type I and Type II errors. It is worthwhile noting that these experiments helped with establishing the boundaries of the algorithm without an initial requirement for a highly trained stochastic model for defining system boundaries. Furthermore, the quantification algorithm is also tested on valid sequence number data in isolation to evaluate errors raised. The algorithm performed as expected in four instances and did not report any Type I and Type II errors.

In summary, the results from the experiments show that the proposed statistical algorithm is effective in detecting non-linear chaotic data that is communicated in a covert manner using TCP/IP packet headers. The results also demonstrate that the negative Type II errors of the algorithm are minimal. Moreover, differently from the existing systems that operate in offline mode, the proposed system offers online covert channel detection, making it more efficient in providing notification in real-time rather than raising alerts passively later.

In the following sections, the proposed work in Chapter 4 is compared with related works.

5.6 Comparison and Performance Discussion

The basis of data embedding is proposed in [145] and used in several works from the literature including [213] and [215]. In [213] and [215], ISN or IP ID fields are used to leak data as covert channels. Furthermore, the research work proposed in [213] and [215] uses Support Vector Machine (SVM), that is proved to be highly effective to detect covert channels, for other techniques. As SVM is the best currently used approach for covert data channel detection, it becomes an obvious choice to compare the proposed algorithm in Table 4.2 against this detection methodology.

The comparison uses ASCII values ranging from 0 to 127 and embeds them in ISN or IP ID fields. Using this methodology, it is possible to pass data between hosts during the initial connection request phase. As mentioned in the last section, the experiments have generated and passed data using the same methodology. To collect data packets, the Wireshark application is used [217] together with WinDump [197] tools. A simple filter is added to the tools to get all initial packets with SYN set to 1, i.e. request for connection. After collecting these TCP/IP packets, the ISNs are extracted from them.

Method	**Support Vector Machine**	**Support Vector Machine**	**Proposed Detection Model**
Number of Features	3	1	1
Training ISN Data	10,000	10,000	1000
Accuracy of Result	99%	92%	98%
Detection Method	Offline	Offline	Online
Average Timing to detect (based on encoded data)	35.3s	15.3s	3.03s

Table 5.2: Performance comparison between Support Vector Machine and proposed detection model

Table 5.2 shows the results of the proposed detection algorithm against two different SVM-based systems. The results are produced based on experiments conducted on the Windows 7 operating system where for all the cases the training data is created using the Covert_TCP tool. The training and covert data is stored in the operating system for analysis in SVM-cases 1 and 2. For the proposed model, the covert data is sent over the network and analysed by the model on the fly.

In case 1, the SVM system used three features: Sequence Number, TCP Control Flag and TCP Checksum fields to create a training model. The training model used 10,000 packets to create pattern classification. This model was then used in offline mode where data analysis was not done in real-time to detect covert packets. It took an average of 35.3s to indicate if a certain packet is covert or otherwise. The accuracy of the system was found to be 99%. In case 2, the SVM system used only one feature, Sequence Number, to create a training model. This model was also created using 10,000 packets and used in offline mode to detect a covert packet at an average of 15.3s, but its accuracy decreased to 92%. Finally, the proposed model was trained online where only 1,000 TCP/IP packet headers were used in real-time to create a training model. It is observed that the model detected covert data at an average of 3.03s with 100% accuracy.

Table 5.2 shows that the proposed detection algorithm outperforms the SVM-based approach in each performance metric apart from the number of features in the second SVM case. As mentioned, the algorithm exploits the facts that ISNs are pseudo-random numbers and they appear to be chaotic and can be predicted with certain accuracy. This greatly reduces the complexity of the proposed algorithm, which makes it simpler to implement. Note that Table 5.2 indicates that the accuracy of SVM decreases if non-linear data is used instead of linear chaotic data. SVM is unable to detect non-linear chaotic data. In order to boost the accuracy of SVM, the number of features to be considered in a TCP packet needs be increased to three, i.e. the Sequence Number, TCP Control Flag and TCP Checksum fields. The purpose of increasing the number of features was to improve the accuracy of SVM. Hence, the computational complexity of SVM for the purpose of training is evaluated as O (number of samples x number of features) that includes solving convex optimisation. However, the proposed method only uses a single feature and does not contain any convex optimization issue. This leads to the computational complexity of the method being O (3 x number of samples). This effectively means the proposed method is able to not only provide higher accuracy with a lower number of features, but also offer those results in a fraction of the time. By using third-order statistical features, the method has identified normal and stego ISNs with an accuracy rate of 100%. For the Windows 7 operating system, the proposed model needs to be created once with 1,000 normal ISNs, while SVM uses 10,000 ISNs to train its model, including 5,000 normal and 5,000 stego-ISNs. Here, the stego-ISNs are used in order to simulate existing network conditions where data leakage is carried in chunks, not in a continuous stream. Finally, the proposed method is deployed online for steganalysis, whereas SVM works only in offline mode. It is hence possible that a covert message passes through before the completion of the SVM model training or steganalysis.

Finally, the proposed framework has the capabilities of notification, continuous data monitoring and threshold adaptation. The threshold adaptation offers the improvement of a threshold value based on active data monitoring and outlier detection, making the framework self-improvable over time.

5.7 Achievements

The following enumerates the original research objectives and the achievement of those objectives through this study:

1. To do a detailed background study of different types of data leakage attacks, use of steganography for data leakage, covert channels, and different packet parameters which can be used for covert communication.

   This objective is successfully completed by studying a range of existing research literature as well as by keeping abreast with known vulnerabilities. The outcome of this objective is presented in Chapter 2.

2. To perform detailed literature research in different covert channel detection methods and identify research gaps.

   The scope of covert channel detection on protocols such as TCP and UDP is very large, which is defined in the header fields and various combinations of header fields that are used for the purpose of covert communication. Existing research into the detection of covert channels on the header fields of those protocols is assimilated in order to understand the weaknesses of existing techniques. This objective is successfully presented in Chapter 3.

3. To design a novel component-based monitoring framework to detect data leakage at the transport layer of networks in real-time. Furthermore, the framework should have the following abilities:

   a. To analyse real-time TCP traffic for covert data and quantify data leakage without the usage of data signature patterns.

   b. To establish thresholds that can be used to identify outliers while performing data leakage detection.

The novel covert data detection framework is proposed in Chapter 4 and the various components of the framework are realised through the proposed algorithms for the purposes of threshold maintenance and outlier quantification. A statistical approach is used for the detection of covert non-linear data as opposed to a data signature-based approach. Also, a strict classifier-based approach built on the support vector machine or a weighted approach used by neural networks is discounted due to their incapability to work with non-linear data as well as the high cost of training model creation. The details of the framework are specified in Chapter 4.

4. To implement and evaluate the proposed solution using a data set.

   The proposed algorithms are implemented using the R-programming language and tested with various tools. The data set used for the testing is a random set of covert ISNs. The experiments conducted include the use of the Covert_TCP tool for server client-based network packet ISN spiking on a single machine as well as the extension of the above onto virtual machines using Kali and Ubuntu flavours. The full details are presented in Chapter 5.

   The objective is completed by the creation of a simulation test where a socket-based client-server system was used to send covert data in the header field (specifically Initial Sequence Number) of the TCP protocol. The environment is later extended to virtual machines where a client and a server are over a virtual network. Every received packet is passed through the proposed covert channel detection algorithm in order to indicate if the packet contains covert data with an associated probability. The further detail of the testbed is provided in Chapter 5.

5.8 Summary

This chapter has presented the experiment set-up and results for the performance evaluation of the proposed framework. A comparison with the related work in the field of covert data detection has also been provided to demonstrate the advantages of the framework over the related work. The stego-ISN and Covert_ISN tools are used based on various metrics to check the framework performance in terms of its detection accuracy. Overall, the experiment results have proved that the proposed detection and data leakage detection algorithms are suitable for finding stego-ISNs.

Chapter 6 Conclusions and Future Work

This chapter summarises the main findings of this study, including the research work conducted and the experiment evaluation performed. The chapter also discusses future directions for further developing the proposed framework.

6.1 Conclusions

A statistical technique helps with the quantification of the level of covert communication by computing the deviation of observed data from an expected threshold value. The technique has a two-stage analysis process where a deviation score is calculated in real-time over a buffer of collected TCP headers as well as over a larger collection of TCP headers. It offers an insight into temporal-based data leakage detection for larger data collections informing an analyst of time-based indicative analysis as well as on-the-fly outlier analysis which are both improvements on existing detection techniques. Furthermore, the proposed framework is novel in that not all deviation scores computed are flagged as data leakage incidents but instead the framework offers further analysis using a novel approach to the quantification of skewed results which informs about the probability of data leakage. The process uses a moving average approach for making decisions about data leakage, thereby increasing the level of accuracy as well as reducing the likelihood of Type I and Type II errors.

TCP/IP used for data transmission over the Internet is an ideal medium for steganography, and attacks based on covert channels make Internet-based communication vulnerable. As mentioned in Chapter 2, covert communication takes full advantage of reserved fields in the TCP/IP header and modifies or inserts values in those header fields for covert communication which is difficult to detect as conventional intrusion detection tools do not scan for vulnerabilities in the header part of a data packet. In Chapter 3, it is found from the literature review that there are not many research works that are actively working in this area of covert channel communication and those that are active are using techniques that are computationally complex, have lower accuracy, require a large amount of training data, and more importantly work only offline. Such approaches are not suitable for practical implementation.

This study has analysed a number of covert communication channels and existing methods for the detection of covert communication via a number of header fields. It has identified the ISN header field that requires further research due to its random / chaotic nature. The proposed work in this thesis is able to analyse ISNs and prove that though they appear random, their dynamic non-linear features have a structure when their realm is changed to phase space reconstruction. A statistical model is thus created in this study for computing a threshold used by the proposed data leakage detection algorithm to determine malicious and non-malicious ISNs. This state-of-the-art algorithm works in tandem with a live network (online) to offer low computational complexity, high accuracy and greater speed, in addition to better true negative and true positive rates.

The following summarises the characteristics of the proposed framework and contributions thereof:

1. Pseudo Random Number Generators are hypothesised that they are a deterministic chaotic system and are able to apply chaos theory for analysing collected ISNs. Chapter 2 has shown through the literature review that random numbers used within the ISN field of IP-based communication are generated on various operating systems using their own algorithms that are presented as pseudo-random numbers. Furthermore, it is shown that these numbers (although appearing to be random) do follow a pattern in hyperspace. These characteristics are then used to do further research work where a PRNG is considered to be deterministic. A delayed coordinate-based mechanism has been used for the study of non-linear data. This reconstruction preserves the properties of the non-linear dynamic system that does not change under smooth coordinate changes. The autocorrelation factor informs us about the embedding dimension that is later used in the delayed coordinate computation. The above helps us with making a concrete

premise that the initial sequence numbers used on operating systems are in fact pseudo-random and hence deterministic. The literature study did not reveal machine learning, pattern or statistical methods for such scenarios other than the support vector machine or neural networks that however are dealing with ASCII data but not non-linear data. The author believes that this study contributes towards a better understanding of pseudo-random numbers and a smoothing process within the realms of network data packets.

2. There is not any framework found in the existing literature that could be used to develop various components for monitoring network traffic for the detection of outliers. This thesis has proposed a novel framework where its distinct components with their purposes are described in Chapter 4, including an outlier detection component, a notification component and a threshold re-computation component in case of skewness. This framework is then used as a blueprint to create effective algorithms for the components in Chapter 4.
3. A statistical model is proposed in this thesis for the quantification of outliers. The process of computing thresholds requires a small initial training data footprint, thus making thresholds operating system specific and computationally less complex. Furthermore, in order to calculate a threshold, there is no requirement to pass malicious ISNs to the model in order to train it.
4. A data leakage detection algorithm has also been proposed that can reliably detect malicious and non-malicious ISNs. This algorithm is based on the above statistical model for the threshold computation. The computational complexity of the proposed algorithm is found to be lower than the Support Vector Machine system, where the algorithm has a complexity of O (3 x number of samples), while the other system operates with the complexity of O (number of samples x number of features). Note that the number of samples used for the algorithm is much smaller than that for the other system (see Table 5.2).

In summary, the new framework offers continuous data monitoring, threshold maintenance and alert notification. Its main advantage is the capability of providing more efficient results with tolerance/threshold values than the existing work.

6.2 Future Work

6.2.1 Worm Detection Using Phase Space Reconstruction

It is interesting to see that conventionally, the detection of specific worms relies on their signatures. However, extending the proposed approach in this thesis should be able to show that specific worms occupy specific regions of a phase space. The invariant subspace of a worm would separate it from normal traffic, thus detecting the traffic that is malicious. This would provide better solutions to worm detection.

6.2.2 Anomalous Behaviour Detection in Heart Beat Pulses

Electrocardiogram data is time-series data where patterns such as low amplitude and high frequency can possibly indicate heart problems. The patterns can be further studied using statistical models where thresholds can be established before further actions are taken. There is a lack of a framework catering for needs such as raising alerts as well as data monitoring and updating/adapting of a threshold value used for checking packets for covert data. The proposed monitoring and detection framework is useful for continuous data monitoring, threshold maintenance and alert notification.

6.2.3 Sensor Data on Wireless Networks or Internet of Things (IoT)

There is a lot of sensor data that is collected from wireless networks or IoTs. The data like the case addressed in this thesis may appear extremely random but would have hidden structures that can be studied using chaos theory and phase space reconstruction. Issues such as deviation from typical system behaviour may indicate problems with the network or anomalous activity on the network. The proposed monitoring and detection framework is useful for continuous data monitoring, threshold maintenance and alert notification suitable for resource limited sensor nodes and things.

6.2.4 Distributed Covert Channel Detection Using Data Mining Approaches

When considering a more scalable and flexible detection solution, a distributed covert channel detection methodology comes to the forefront. This methodology has not been considered in this thesis as its objectives are different. However, it is something that is becoming more prevalent especially within the context of machine learning where training models need online and offline creation more often. In machine learning-based covert channel detection cases, this could be taxing on a single processor and hence paper [218] has proposed the usage of distributed covert channel detection.

The phase space-based transformation motivated from the work for studying non-linear dynamic systems has essentially helped with the transformation of single dimension ISN data to multi-dimensional vectors. It is this process of the transformation that puts things in perspective for this research work. The proposed framework is used for continuous data monitoring, threshold maintenance and alert notification and to detect covert data leakages especially with regard to non-linear chaotic data. The main advantage of the proposed framework is its capability of providing more efficient results with tolerance/threshold values. The proposed mathematical model can also be used for on-the-fly detection of covert data as opposed to offline processing methods.

As part of my future work, I will investigate the effectiveness of my model for other operating systems such as Windows.

References

[1] C. Wueest, "Internet Security Threat Report (ISTR): Financial Threats Review 2017," *Symantec, July,* vol. 18, p. 2017, 2017.

[2] C. C. Magzine. (2017). *Cybersecurity Ventures Official Annual Cybercrime Report*. Available: https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/

[3] IBM. (2018). *2018 Cost of Data Breach Study, Global Overview*. Available: https://www.ibm.com/security/data-breach

[4] H. M. s. Government. (2015). *Information security breaches survey 2015* Available: https://www.gov.uk/government/publications/information-security-breaches-survey-2015

[5] N. Nissim, A. Cohen, C. Glezer, and Y. Elovici, "Detection of malicious PDF files and directions for enhancements: a state-of-the art survey," *Computers & Security,* vol. 48, pp. 246-266, 2015.

[6] Y. F. Nia and A. Nowroozi, "Behavior and system based backdoor detection focusing on CMD phase," in *2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*, 2015, pp. 128-133: IEEE.

[7] S. H. Sellke, C.-C. Wang, S. Bagchi, and N. Shroff, "TCP/IP timing channels: Theory to implementation," in *IEEE INFOCOM 2009*, 2009, pp. 2204-2212: IEEE.

[8] D. Chiu, S.-H. Weng, and J. Chiu, "Backdoor use in targeted attacks," *A Trend Micro Research Paper,* 2017.

[9] (17-Feb). *Stealing data using steganography*. Available: ttps://www.c-sharpcorner.com/news/stealing-data-using-steganography

[10] (17-Feb). *Steganographic Hacking - How it Works.* Available: https://adamlevin.com/2018/02/22/steganographic-hacking-works/

[11] (17-Feb). *Magento Malware Uses Steganography to Steal Payment Card Data*. Available: https://www.securitynewspaper.com/2016/10/18/magento-malware-uses-steganography-steal-payment-card-data/

[12] (17-Feb). *Insider uses steganography to steal trade secrets for China ⋆ Security On Demand*. Available: https://www.securityondemand.com/news-posts/insider-uses-steganography-steal-trade-secrets-china/

[13] (17-Feb). *Hackers' latest weapon: Steganography*. Available: https://publications.computer.org/computer-magazine/2018/11/15/how-steganography-works/

[14] L. M. Cameron. *With Cryptography Easier To Detect, Cybercriminals Now Hide Malware In Plain Sight. Call It Steganography. Here's How It Works*.

Available: https://publications.computer.org/computer-magazine/2018/11/15/how-steganography-works/

[15] Y.-H. Jin *et al.*, "A rapid advice guideline for the diagnosis and treatment of 2019 novel coronavirus (2019-nCoV) infected pneumonia (standard version)," *Military Medical Research,* vol. 7, no. 1, p. 4, 2020.

[16] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys & Tutorials,* vol. 9, no. 3, pp. 44-57, 2007.

[17] B. Panajotov and A. Mileva, "Covert Channels in TCP/IP Protocol Stack," *ICT innovations web proceedings,* pp. 190-199, 2013.

[18] K. Cabaj, L. Caviglione, W. Mazurczyk, S. Wendzel, A. Woodward, and S. Zander, "The new threats of information hiding: the road ahead," *IT Professional,* vol. 20, no. 3, pp. 31-39, 2018.

[19] W. Mazurczyk, K. Szczypiorski, and B. Jankowski, "Towards steganography detection through network traffic visualisation," in *2012 IV International Congress on Ultra Modern Telecommunications and Control Systems*, 2012, pp. 947-954: IEEE.

[20] E. Brown, B. Yuan, D. Johnson, and P. Lutz, "Covert channels in the HTTP network protocol: Channel characterization and detecting Man-in-the-Middle attacks," *Journal of Information Warfare,* vol. 9, no. 3, pp. 26-38, 2010.

[21] B. Panajotov and A. Mileva, "Covert Channels in TCP/IP Protocol Stack," in *ICT Innovations 2013 Web Proceedings*, 2013.

[22] S. J. Murdoch, "Covert channel vulnerabilities in anonymity systems," University of Cambridge, Computer Laboratory2007.

[23] R. Heady, G. F. Luger, A. Maccabe, and M. Servilla, *The architecture of a network level intrusion detection system*. University of New Mexico. Department of Computer Science. College of Engineering, 1990.

[24] R. Bace and P. Mell, "NIST special publication on intrusion detection systems," BOOZ-ALLEN AND HAMILTON INC MCLEAN VA2001.

[25] K. K. Abhaya, R. Jha, and S. Afroz, "Data mining techniques for intrusion detection: A review," *International Journal of Advanced Research in Computer and Communication Engineering,* vol. 3, no. 6, pp. 6938-6942, 2014.

[26] Z. Dewa and L. A. Maglaras, "Data mining and intrusion detection systems," *International Journal of Advanced Computer Science and Applications,* vol. 7, no. 1, pp. 62-71, 2016.

[27] J. Cao *et al.*, "Covert Channels in SDN: Leaking Out Information from Controllers to End Hosts," in *International Conference on Security and Privacy in Communication Systems*, 2019, pp. 429-449: Springer.

[28] S. Zander, G. Armitage, and P. Branch, "Covert channels and countermeasures in computer network protocols [reprinted from ieee

communications surveys and tutorials]," *IEEE Communications Magazine,* vol. 45, no. 12, pp. 136-142, 2007.

[29] S. Zander, G. Armitage, and P. Branch, "Covert channels and countermeasures in computer network protocols [reprinted from ieee communications surveys and tutorials]," *IEEE Communications Magazine,* vol. 45, no. 12, pp. 136-142, 2007.

[30] M. Alam and S. Sethi, "Covert Channel Detection framework for cloud using distributed machine learning," *CoRR-Computing Research Repository-arXiv,* 2015.

[31] T. Sohn, J. Seo, and J. Moon, "A study on the covert channel detection of TCP/IP header using support vector machine," in *International Conference on Information and Communications Security*, 2003, pp. 313-324: Springer.

[32] K. Cabaj, W. Mazurczyk, P. Nowakowski, and P. Żórawski, "Towards Distributed Network Covert Channels Detection Using Data Mining-based Approach," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1-10.

[33] C. Zhiyong and Z. Yong, "Entropy based taxonomy of network convert channels," in *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, 2009, vol. 1, pp. 451-455: IEEE.

[34] W. De Mulder, "Optimal clustering in the context of overlapping cluster analysis," *Information Sciences,* vol. 223, pp. 56-74, 2013.

[35] A. S. Coronado, "Computer Security: Principles and Practice," *Journal of Information Privacy and Security,* vol. 9, no. 2, pp. 62-65, 2013.

[36] X. Zhang, Y.-A. Tan, C. Liang, Y. Li, and J. Li, "A covert channel over volte via adjusting silence periods," *IEEE Access,* vol. 6, pp. 9292-9302, 2018.

[37] S. Cabuk, "Network covert channels: Design, analysis, detection, and elimination," Purdue University, 2006.

[38] Y. Kang, X. Li, Y. Lu, and C. Yang, "Application of chaotic phase space reconstruction into nonlinear time series prediction in deep rock mass," in *Fuzzy Systems and Knowledge Discovery, 2008. FSKD'08. Fifth International Conference on*, 2008, vol. 5, pp. 593-597: IEEE.

[39] D. Llamas, C. Allison, and A. Miller, "Covert channels in internet protocols: A survey," in *Proceedings of the 6th Annual Postgraduate Symposium about the Convergence of Telecommunications, Networking and Broadcasting, PGNET*, 2005, vol. 2005.

[40] M. Ławniczak, S. Bauch, and L. Sirko, "Handbook of Applications of Chaos Theory," ed: CRC Press, Boca Raton USA eds. Christos Skiadas and Charilaos Skiadas, 2016.

[41] C. Frazier and K. M. Kockelman, "Chaos theory and transportation systems: Instructive example," *Transportation Research Record,* vol. 1897, no. 1, pp. 9-17, 2004.

[42] L. Sardonini, "Invariants estimation in nonlinear time series," alma, 2007.

[43] Q. Lü, G. Shen, and R. Yu, "A chaotic approach to maintain the population diversity of genetic algorithm in network training," *Computational Biology Chemistry,* vol. 27, no. 3, pp. 363-371, 2003.

[44] X. Haibo and Z. Laibin, "Development actualities of pipeline leak-detection technologies at home and abroad," *Oil Gas Storage Transportation research part F: traffic psychology and behaviour,* vol. 20, no. 1, pp. 1-15, 2001.

[45] H. Zhao and Y.-Q. Shi, "Detecting covert channels in computer networks based on chaos theory," *Information Forensics and Security, IEEE Transactions on,* vol. 8, no. 2, pp. 273-282, 2013.

[46] A. Shabtai, Y. Elovici, and L. Rokach, *A survey of data leakage detection and prevention solutions*. Springer Science & Business Media, 2012.

[47] P. Papadimitriou and H. Garcia-Molina, "Data leakage detection," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 23, no. 1, pp. 51-63, 2011.

[48] G. Clayton. (2015, 14 May). *Data Loss Prevention and Monitoring in the Workplace: Best Practice Guide for Europe*. Available: http://www.symantec.com/connect/sites/default/files/21263455_EMEA_WP_DLP_Monitoring_workplace_09_12.pdf

[49] Kaspersky. (2015, 14 May). *Kaspersky Security 9.0 for Microsoft Exchange Servers*. Available: https://support.kaspersky.co.uk/11619

[50] R. B. Security. (2016, 16 April). *Bad Luck Over The Upcoming Badlock Vulnerability?* Available: www.riskbasedsecurity.com/2016/03/bad-luck-over-the-upcoming-badlock-vulnerability/

[51] K. Thomas *et al.*, "Data breaches, phishing, or malware? Understanding the risks of stolen credentials," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1421-1434.

[52] K. Thomas *et al.*, "Data breaches, phishing, or malware?: Understanding the risks of stolen credentials," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1421-1434: ACM.

[53] J. P. Singh, "Analysis of SQL Injection Detection Techniques," *arXiv preprint arXiv:1605.02796,* 2016.

[54] Z. S. Alwan and M. F. Younis, "Detection and prevention of SQL Injection attack: A survey," *vol,* vol. 6, pp. 5-17, 2017.

[55] A. Joshi and V. Geetha, "SQL Injection detection using machine learning," in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2014, pp. 1111-1115: IEEE.

[56] P. R. McWhirter, K. Kifayat, Q. Shi, and B. Askwith, "SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel," *Journal of information security and applications,* vol. 40, pp. 199-216, 2018.

[57] S. W. Boyd and A. D. Keromytis, "SQLrand: Preventing SQL injection attacks," in *International Conference on Applied Cryptography and Network Security*, 2004, pp. 292-302: Springer.

[58] T. Micro, "Data-stealing malware on the rise: Solutions to keep businesses and consumers safe," *Retrived from http://us. trendmicro. com/imperia/md/content/us/pdf/threats/securitylibrary/data_stealing_malware_focus_report_-_june_2009. pdf,* 2009.

[59] B. Research, "New Data-Stealing, Cryptomining Malware Campaign on the Rise," 2018.

[60] F. Labs. (2018, 2 May). *How Malware can Steal your Data and What you can do to Stop It*. Available: https://interact.f5.com/rs/653-SMC-783/images/EBOOK_How-Malware-Can-Steal-Your-Data.pdf

[61] W. Ashford, "Rise in data-stealing Betabot malware," ed: Computer Weekly, 2018.

[62] J. A. Chaudhry, S. A. Chaudhry, and R. G. Rittenhouse, "Phishing attacks and defenses," *International Journal of Security and Its Applications,* vol. 10, no. 1, pp. 247-256, 2016.

[63] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Communications of the ACM,* vol. 50, no. 10, pp. 94-100, 2007.

[64] C. Hoffman, "What is DNS Cache Poisoning?," ed: How to Geek, 2017.

[65] Techopedia. (1-Feb). *Direct Memory Access (DMA)*. Available: https://www.techopedia.com/definition/2767/direct-memory-access-dma

[66] B. Morgan, E. Alata, V. Nicomette, and M. Kaâniche, "Bypassing IOMMU protection against I/O attacks," in *2016 Seventh Latin-American Symposium on Dependable Computing (LADC)*, 2016, pp. 145-150: IEEE.

[67] P. Stewin and I. Bystrov, "Understanding DMA malware," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, 2012, pp. 21-41: Springer.

[68] K. Zetter, "Acker Lexicon: What is a Backdoor?," ed: Wired, 2014.

[69] J. Clements and Y. Lao, "Backdoor Attacks on Neural Network Operations," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2018, pp. 1154-1158: IEEE.

[70] S. L. Thomas and A. Francillon, "Backdoors: Definition, Deniability and Detection," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, 2018, pp. 92-113: Springer.

[71] Techopedia. (2016, 1-Jan). *Keylogger*. Available: https://www.techopedia.com/definition/4000/keylogger

[72] M. Hussain *et al.*, "The rise of keyloggers on smartphones: A survey and insight into motion-based tap inference attacks," *Pervasive and Mobile Computing,* vol. 25, pp. 1-25, 2016.

[73] A. Solairaj, S. Prabanand, J. Mathalairaj, C. Prathap, and L. Vignesh, "Keyloggers software detection techniques," in *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 2016, pp. 1-6: IEEE.

[74] A. Emigh, "The crimeware landscape: Malware, phishing, identity theft and beyond," *Journal of Digital Forensic Practice,* vol. 1, no. 3, pp. 245-260, 2006.

[75] S. Z. Goher, B. Javed, and N. A. Saqib, "Covert channel detection: A survey based analysis," in *High Capacity Optical Networks and Emerging/Enabling Technologies*, 2012, pp. 057-065: IEEE.

[76] U. Budhia and D. Kundur, "Digital video steganalysis exploiting collusion sensitivity," in *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense III*, 2004, vol. 5403, pp. 210-222: International Society for Optics and Photonics.

[77] M. Jain and S. K. Lenka, "A Review on Data Leakage Prevention using Image Steganography," *Department of Computer Science and Engineering, Mody University of Science and Technology ISSN,* vol. 23197323, 2016.

[78] T. G. Handel and M. T. Sandford, "Hiding data in the OSI network model," in *International Workshop on Information Hiding*, 1996, pp. 23-38: Springer.

[79] S. Wendzel, W. Mazurczyk, L. Caviglione, and M. Meier, "Hidden and uncontrolled–on the emergence of network steganographic threats," in *ISSE 2014 Securing Electronic Business Processes*: Springer, 2014, pp. 123-133.

[80] S. Das, S. Das, B. Bandyopadhyay, and S. Sanyal, "Steganography and Steganalysis: different approaches," *arXiv preprint arXiv:.02211,* 2011.

[81] K. Ahsan and D. Kundur, "Practical data hiding in TCP/IP," in *Proc. Workshop on Multimedia Security at ACM Multimedia*, 2002, vol. 2, no. 7.

[82] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert messaging through TCP timestamps," in *International Workshop on Privacy Enhancing Technologies*, 2002, pp. 194-208: Springer.

[83] S. Bellovin, "Defending against sequence number attacks RFC 1948," 1996.

[84] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," vol. 2, no. 5, 1997.

[85] M. Zalewski. (2001, 17 April). *Strange attractors and tcp/ip sequence number analysis*. Available: http://lcamtuf.coredump.cx/newtcp/

[86] M. A. N. Mahajan and I. Shaikh, "Detecting Covert Channels in TCP/IP Header with the Use of Naive Bayes Classifier," 2015.

[87] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-based survey and categorization of network covert channel techniques," *ACM Computing Surveys (CSUR),* vol. 47, no. 3, p. 50, 2015.

[88] T. G. Handel and M. T. Sandford II, "Hiding data in the OSI network model," in *Information Hiding*, 1996, pp. 23-38: Springer.

[89] J. Classen, M. Schulz, and M. Hollick, "Practical covert channels for WiFi systems," in *Communications and Network Security (CNS), 2015 IEEE Conference on*, 2015, pp. 209-217: IEEE.

[90] Z. Kwecka, "Application layer covert channel analysis and detection," Edinburgh Napier University, 2006.

[91] J. P. Black, "Techniques of network steganography and covert channels," San Diego State University, 2014.

[92] W. Stallings, *Cryptography and Network Security*, 5th Edition ed. New York: Prentice Hall, 2008.

[93] S. Z. Goher, B. Javed, and N. A. Saqib, "Covert channel detection: A survey based analysis," in *High Capacity Optical Networks and Emerging/Enabling Technologies*, 2012, pp. 057-065: IEEE.

[94] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys Tutorials,* vol. 9, no. 3, pp. 44-57, 2007.

[95] X. Shu and D. D. Yao, "Data leak detection as a service," in *Security and Privacy in Communication Networks*: Springer, 2012, pp. 222-240.

[96] P.-C. Lin, Y.-D. Lin, Y.-C. Lai, and T.-H. Lee, "Using string matching for deep packet inspection," *Computer,* vol. 41, no. 4, pp. 23-28, 2008.

[97] S. Nawafleh, M. Hasan, Y. Nawafleh, and S. Fakhouri, "Protection and defense against sensitive data leakage problem within organizations," *European Journal of Business and Management,* vol. 5, no. 23, pp. 87-95, 2013.

[98] H. Sethuraman and M. Abdul Haseeb, "Data loss/leakage prevention," ed, 2013.

[99] J. Marecki, M. Srivatsa, and P. Varakantham, "A Decision Theoretic Approach to Data Leakage Prevention," in *Social Computing (SocialCom), 2010 IEEE Second International Conference on*, 2010, pp. 776-784: IEEE.

[100] P. Papadimitriou and H. Garcia-Molina, "A model for data leakage detection," in *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, 2009, pp. 1307-1310: IEEE.

[101] A. Kundu. (2006, 16 January). *Information Leaks and Safe Web Services*. Available: http://www.techrepublic.com/resource-library/whitepapers/information-leaks-and-safe-web-services/

[102] R. Yokomori, F. Ohata, Y. Takata, H. Seki, and K. Inoue, "Analysis and implementation method of program to detect inappropriate information leak," in *Quality Software, 2001. Proceedings. Second Asia-Pacific Conference on*, 2001, pp. 5-12: IEEE.

[103] S. Kuninobu, Y. Takata, H. Seki, and K. Inoue, "An information flow analysis of programs based on a lattice model," Technical report of IEICE SS 20002000.

[104] M. McCormick, "Data theft: a prototypical insider threat," in *Insider Attack and Cyber Security*: Springer, 2008, pp. 53-68.

[105] R. Capizzi, A. Longo, V. Venkatakrishnan, and A. P. Sistla, "Preventing information leaks through shadow executions," in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, 2008, pp. 322-331: IEEE.

[106] S. Zander, G. Armitage, and P. Branch, "Covert channels in the IP time to live field," 2006.

[107] G. Shah, A. Molina, and M. Blaze, "Keyboards and Covert Channels," in *Usenix security*, 2006, vol. 6, pp. 59-75.

[108] S. Zander, G. Armitage, and P. Branch, "Stealthier inter-packet timing covert channels," in *International Conference on Research in Networking*, 2011, pp. 458-470: Springer.

[109] Z. Trabelsi, H. El-Sayed, L. Frikha, and T. Rabie, "Traceroute based IP channel for sending hidden short messages," in *International Workshop on Security*, 2006, pp. 421-436: Springer.

[110] Z. Trabelsi, H. El-Sayed, L. Frikha, and T. Rabie, "A novel covert channel based on the IP header record route option," *International Journal of Advanced Media and Communication,* vol. 1, no. 4, pp. 328-350, 2007.

[111] P. Allix, "Covert channels analysis in TCP/IP networks," IFIPS School of Engineering, University of Paris-Sud XI, Orsay, France2007.

[112] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-based covert timing channels: Automated modeling and evasion," in *International Workshop on Recent Advances in Intrusion Detection*, 2008, pp. 211-230: Springer.

[113] W. Mazurczyk and K. Szczypiorski, "Steganography in handling oversized IP packets," in *2009 International Conference on Multimedia Information Networking and Security*, 2009, vol. 1, pp. 559-564: IEEE.

[114] W. Mazurczyk and K. Szczypiorski, "Evaluation of steganographic methods for oversized IP packets," *Telecommunication Systems,* vol. 49, no. 2, pp. 207-217, 2012.

[115] A. El-Atawy and E. Al-Shaer, "Building covert channels over the packet reordering phenomenon," in *IEEE INFOCOM 2009*, 2009, pp. 2186-2194: IEEE.

[116] A. Askarov, D. Zhang, and A. C. Myers, "Predictive black-box mitigation of timing channels," in *Proceedings of the 17th ACM conference on Computer and communications security*, 2010, pp. 297-307: ACM.

[117] J. Wu, Y. Wang, L. Ding, and X. Liao, "Improving performance of network covert timing channel through Huffman coding," *Mathematical and Computer Modelling,* vol. 55, no. 1-2, pp. 69-79, 2012.

[118] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: high-bandwidth and reliable covert channel attacks inside the cloud," *IEEE/ACM Transactions on Networking,* vol. 23, no. 2, pp. 603-615, 2015.

[119] X. Lu, Y. Wang, L. Huang, W. Yang, and Y. Shen, "A Secure and Robust Covert Channel Based on Secret Sharing Scheme," in *Asia-Pacific Web Conference*, 2016, pp. 276-288: Springer.

[120] M. A. Ayub, S. Smith, and A. Siraj, "A Protocol Independent Approach in Network Covert Channel Detection," in *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, 2019, pp. 165-170: IEEE.

[121] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: design and detection," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 178-187: ACM.

[122] R. Chakinala, A. Kumarasubramanian, R. Manokaran, G. Noubir, C. P. Rangan, and R. Sundaram, "Steganographic communication in ordered channels," in *International Workshop on Information Hiding*, 2006, pp. 42-57: Springer.

[123] X. Luo, E. W. Chan, and R. K. Chang, "Cloak: A ten-fold way for reliable covert communications," in *European Symposium on Research in Computer Security*, 2007, pp. 283-298: Springer.

[124] X. Luo, E. W. Chan, and R. K. Chang, "CLACK: A network covert channel based on partial acknowledgment encoding," in *2009 IEEE International Conference on Communications*, 2009, pp. 1-5: IEEE.

[125] X. Luo, P. Zhou, E. W. Chan, R. K. Chang, and W. Lee, "A combinatorial approach to network covert communications with applications in web leaks," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, 2011, pp. 474-485: IEEE.

[126] X. Luo, E. W. Chan, and R. K. Chang, "TCP covert timing channels: Design and detection," in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, 2008, pp. 420-429: IEEE.

[127] W. Mazurczyk, M. Smolarczyk, and K. Szczypiorski, "Retransmission steganography applied," in *2010 International Conference on Multimedia Information Networking and Security*, 2010, pp. 846-850: IEEE.

[128] W. Mazurczyk, M. Smolarczyk, and K. Szczypiorski, "On information hiding in retransmissions," *Telecommunication Systems,* vol. 52, no. 2, pp. 1113-1121, 2013.

[129] X. Luo, E. W. Chan, P. Zhou, and R. K. Chang, "Robust network covert communications based on TCP and enumerative combinatorics," *IEEE Transactions on Dependable and Secure Computing,* vol. 9, no. 6, pp. 890-902, 2012.

[130] J. Thyer, "Covert data storage channel using ip packet headers," SANS Institute, 2008.

[131] J. Oakley, L. Yu, X. Zhong, G. K. Venayagamoorthy, and R. Brooks, "Protocol Proxy: An FTE-based Covert Channel," *Computers Security,* p. 101777, 2020.

[132] M. Van Horenbeeck, "Deception on the network: thinking differently about covert channels," 2006.

[133] R. Duncan and J. E. Martina, "Steganographic Message Broadcasting using Web Protocols," in *proceedings of: Simposio Brasilerio de Seguranca (SBSeg 2010), Fortaleza, Brasil*, 2010.

[134] J. Jaskolka, R. Khedri, and K. E. Sabri, "Investigative support for information confidentiality part I: Detecting confidential information leakage via protocol-based covert channels," *Procedia Computer Science,* vol. 34, pp. 276-285, 2014.

[135] W. Mazurczyk and K. Szczypiorski, "Steganography of VoIP streams," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2008, pp. 1001-1018: Springer.

[136] W. Mazurczyk and K. Szczypiorski, "Covert Channels in SIP for VoIP signalling," in *International Conference on Global e-Security*, 2008, pp. 65-72: Springer.

[137] Y. Lizhi, H. Yongfeng, Y. Jian, and L. Bai, "A novel covert timing channel based on RTP/RTCP," *Chinese Journal of Electronics,* vol. 21, no. 4, pp. 711-714, 2012.

[138] R. Goudar and P. More, "Hybrid Covert Channel an Obliterate for Information Hiding," in *Proceedings of the Third International Conference on Trends in Information, Telecommunication and Computing*, 2013, pp. 609-613: Springer.

[139] C. J. Huberty and J. D. Morris, "Multivariate analysis versus multiple univariate analyses," *Psychological bulletin,* vol. 105, no. 2, p. 302, 1989.

[140] K. Clarke and R. Warwick, "An approach to statistical analysis and interpretation," *Change in marine communities,* vol. 2, pp. 117-143, 1994.

[141] A. Liu, J. Chen, and L. Yang, "Real-time detection of covert channels in highly virtualized environments," in *International Conference on Critical Infrastructure Protection*, 2011, pp. 151-164: Springer.

[142] V. Berk, A. Giani, G. Cybenko, and N. Hanover, "Detection of covert channel encoding in network packet delays," *Rapport technique TR536, de lUniversit√© de Dartmouth,* vol. 19, 2005.

[143] C. Kreibich, M. Handley, and V. Paxson, "Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics," in *Proc. USENIX Security Symposium*, 2001, vol. 2001.

[144] C. Wang, C. Zhang, B. Wu, Y. a. Tan, and Y. Wang, "A novel anti-detection criterion for covert storage channel threat estimation," *Science China Information Sciences,* vol. 61, no. 4, p. 048101, 2018.

[145] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," *First Monday,* vol. 2, no. 5, 1997.

[146] S. M. Bellovin and F. Gont, "Defending against sequence number attacks," 2012.

[147] E. Tumoian and M. Anikeev, "Network based detection of passive covert channels in TCP/IP," in *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, 2005, pp. 802-809: IEEE.

[148] J. Zhai, G. Liu, and Y. Dai, "A covert channel detection algorithm based on TCP Markov model," in *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, 2010, pp. 893-897: IEEE.

[149] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics,* vol. 22, no. 1, pp. 79-86, 1951.

[150] H. Qu, P. Su, and D. Feng, "A typical noisy covert channel in the IP protocol," in *Security Technology, 2004. 38th Annual 2004 International Carnahan Conference on*, 2004, pp. 189-192: IEEE.

[151] S. Wendzel and S. Zander, "Detecting protocol switching covert channels," in *Local Computer Networks (LCN), 2012 IEEE 37th Conference on*, 2012, pp. 280-283: IEEE.

[152] L. Y. L. Yao, L. Z. L. ZhiTang, and L. S. L. Shuyu, "A fuzzy anomaly detection algorithm for ipv6," in *Semantics, Knowledge and Grid, 2006. SKG'06. Second International Conference on*, 2006, pp. 67-67: IEEE.

[153] Z. Liu and Y. Lai, "A data mining framework for building intrusion detection models based on IPv6," in *International Conference on Information Security and Assurance*, 2009, pp. 608-618: Springer.

[154] R. M. Saad, S. Ramadass, and S. Manickam, "A study on detecting ICMPv6 flooding attack based on IDS," *Australian Journal of Basic and Applied Sciences,* vol. 7, no. 2, pp. 175-181, 2013.

[155] M. Zulkiflee, M. Azmi, S. Ahmad, S. Sahib, and M. Ghani, "A framework of features selection for ipv6 network attacks detection," *WSEAS Trans Commun,* vol. 14, no. 46, pp. 399-408, 2015.

[156] A. Salih, X. Ma, and E. Peytchev, "Detection and classification of covert channels in IPv6 using enhanced machine learning," 2015.

[157] M. Padlipsky, D. Snow, and P. Karger, "Limitations of end-to-end encryption in secure computer networks," MITRE CORP BEDFORD MA1978.

[158] Q.-z. Yao and P. Zhang, "Coverting channel based on packet length," *Computer engineering,* vol. 34, no. 3, pp. 183-185, 2008.

[159] A. S. Nair, A. Sur, and S. Nandi, "Detection of packet length based network steganography," in *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, 2010, pp. 574-578: IEEE.

[160] J. Chow, X. Li, and X. Mountrouidou, "Raising flags: Detecting covert storage channels using relative entropy," in *Intelligence and Security Informatics (ISI), 2017 IEEE International Conference on*, 2017, pp. 25-30: IEEE.

[161] D. J. Pack, W. Streilein, S. Webster, and R. Cunningham, "Detecting HTTP tunneling activities," MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB2002.

[162] K. Borders and A. Prakash, "Web tap: detecting covert web traffic," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 110-120: ACM.

[163] Y. Sun, L. Zhang, and C. Zhao, "A Study of Network Covert Channel Detection Based on Deep Learning," in *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2018, pp. 637-641: IEEE.

[164] H. Song and X. Li, "Collaborative detection of covert storage channels," in *Military Communications Conference, MILCOM 2016-2016 IEEE*, 2016, pp. 515-520: IEEE.

[165] H. Hovhannisyan, K. Lu, R. Yang, W. Qi, J. Wang, and M. Wen, "A novel deduplication-based covert channel in cloud storage service," in *Global Communications Conference (GLOBECOM), 2015 IEEE*, 2015, pp. 1-6: IEEE.

[166] L. C. Han, "Multi-bit data de-duplication-based cloud storage channel covert," *Measurement,* vol. 144, pp. 52-57, 2019.

[167] L. Caviglione, M. Podolski, W. Mazurczyk, and M. Ianigro, "Covert channels in personal cloud storage services: The case of Dropbox," *IEEE Transactions on Industrial Informatics,* vol. 13, no. 4, pp. 1921-1931, 2017.

[168] K. Denney, A. S. Uluagac, K. Akkaya, and S. Bhansali, "A novel storage covert channel on wearable devices using status bar notifications," in *Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual*, 2016, pp. 845-848: IEEE.

[169] A. Singh and K. Manchanda, "Establishment of bit selective mode storage covert channel in VANETS," in *Computational Intelligence and Computing Research (ICCIC), 2015 IEEE International Conference on*, 2015, pp. 1-4: IEEE.

[170] Y. Shoukry, J. Araujo, P. Tabuada, M. Srivastava, and K. H. Johansson, "Minimax control for cyber-physical systems under network packet scheduling attacks," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*, 2013, pp. 93-100: ACM.

[171] V. Berk, A. Giani, G. Cybenko, and N. Hanover, "Detection of covert channel encoding in network packet delays," *Rapport technique TR536, de lUniversite de Dartmouth,* p. 19, 2005.

[172] R. Soltani, D. Goeckel, D. Towsley, and A. Houmansadr, "Fundamental Limits of Covert Packet Insertion," *arXiv preprint arXiv:1903.11640,* 2019.

[173] S. H. Sellke, C.-C. Wang, S. Bagchi, and N. Shroff, "TCP/IP timing channels: Theory to implementation," in *INFOCOM 2009, IEEE*, 2009, pp. 2204-2212: IEEE.

[174] J. Giles and B. Hajek, "An information-theoretic and game-theoretic study of timing channels," *Information Theory, IEEE Transactions on,* vol. 48, no. 9, pp. 2455-2477, 2002.

[175] W.-M. Hu, "Reducing timing channels with fuzzy time," *Journal of computer security,* vol. 1, no. 3-4, pp. 233-254, 1992.

[176] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The pump: A decade of covert fun," in *Computer Security Applications Conference, 21st Annual*, 2005, pp. 7 pp.-360: IEEE.

[177] J. Agat, "Transforming out timing leaks," in *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 2000, pp. 40-53: ACM.

[178] R. A. Kemmerer, "A practical approach to identifying storage and timing channels: Twenty years later," in *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, 2002, pp. 109-118: IEEE.

[179] V. Berk, A. Giani, and G. Cybenko, "Covert channel detection using process query systems," *Proceedings of FLOCON 2005,* 2005.

[180] S. Gianvecchio and H. Wang, "Detecting covert timing channels: an entropy-based approach," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 307-316: ACM.

[181] P. Peng, P. Ning, and D. S. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Security and Privacy, 2006 IEEE Symposium on*, 2006, pp. 15 pp.-349: IEEE.

[182] P. Yang, H. Zhao, and Z. Bao, "A probability-model-based approach to detect covert timing channel," in *Information and Automation, 2015 IEEE International Conference on*, 2015, pp. 1043-1047: IEEE.

[183] K. S. Lee, H. Wang, and H. Weatherspoon, "PHY covert channels: Can you see the idles?," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, 2014, pp. 173-185.

[184] H. Fang, F. Yao, M. Doroslovački, and G. Venkataramani, "Negative Correlation, Non-linear Filtering, and Discovering of Repetitiveness for Cache Timing Channel Detection," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 2522-2526: IEEE.

[185] D. M. Dakhane and P. R. Deshmukh, "Active warden for TCP sequence number base covert channel," in *Pervasive Computing (ICPC), 2015 International Conference on*, 2015, pp. 1-5: IEEE.

[186] A. Sur, A. S. Nair, A. Kumar, A. Jain, and S. Nandi, "Steganalysis of network packet length based data hiding," *Circuits, Systems, and Signal Processing,* vol. 32, no. 3, pp. 1239-1256, 2013.

[187] R. Tahboub and Y. Saleh, "Data leakage/loss prevention systems (DLP)," in *Computer Applications and Information Systems (WCCAIS), 2014 World Congress on*, 2014, pp. 1-6: IEEE.

[188] J. I. Helfman and C. L. Isbell, "Ishmail: Immediate identification of important information," in *AT&T Labs*, 1995: Citeseer.

[189] W. W. Cohen, "Learning rules that classify e-mail," in *AAAI spring symposium on machine learning in information access*, 1996, vol. 18, p. 25: California.

[190] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A Bayesian approach to filtering junk e-mail," in *Learning for Text Categorization: Papers from the 1998 workshop*, 1998, vol. 62, pp. 98-105: Madison, Wisconsin.

[191] J. Staddon, P. Golle, M. Gagn√©, and P. Rasmussen, "A content-driven access control system," in *Proceedings of the 7th symposium on Identity and trust on the Internet*, 2008, pp. 26-35: ACM.

[192] W. W. Cohen and Y. Singer, "Context-sensitive learning methods for text categorization," *ACM Transactions on Information Systems (TOIS),* vol. 17, no. 2, pp. 141-173, 1999.

[193] H. Drucker, D. Wu, and V. N. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural networks,* vol. 10, no. 5, pp. 1048-1054, 1999.

[194] J. Hovold, "Naive Bayes Spam Filtering Using Word-Position-Based Attributes," in *CEAS*, 2005, pp. 41-48.

[195] J. Collins and S. Agaian, "Trends toward real-time network data steganography," *arXiv preprint arXiv:1604.02778,* 2016.

[196] *SteganRTP - RTP Covert Channel*. Available: https://sourceforge.net/projects/steganrtp/

[197] WinDump. (2006, 17 May). *WinDump - tcpdump for Windows using WinPcap*. Available: www.winpcap.org/windump/

[198] D. Llamas, "Covert channel analysis and data hiding in tcp/ip," Napier University, 2004.

[199] A. Hintz, "Covert channels in TCP and IP headers," *Presentation at DEFCON,* vol. 10, 2002.

[200] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil, "Eliminating steganography in Internet traffic with active wardens," in *Information Hiding*, 2002, pp. 18-35: Springer.

[201] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[202] (3-May). *Autocorrelation and Partial Autocorrelation Functions*. Available: https://www.ibm.com/support/knowledgecenter/en/SS3RA7_15.0.0/com.ibm.spss.modeler.help/timeseries_acf_pacf.htm

[203] H. Abarbanel, *Analysis of observed chaotic data*. Springer Science & Business Media, 2012.

[204] P. Berge, Y. Pomeau, and C. Vidal, *Order within chaos*. Wiley and Sons, 1984.

[205] L. Kocarev, "Chaos-based cryptography: a brief overview," *Circuits and Systems Magazine, IEEE,* vol. 1, no. 3, pp. 6-21, 2001.

[206] S. J. Murdoch and S. Lewis, "Embedding covert channels into TCP/IP," in *Information hiding*, 2005, pp. 247-261: Springer.

[207] L. Dorrendorf, Z. Gutterman, and B. Pinkas, "Cryptanalysis of the windows random number generator," in *Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 476-485: ACM.

[208] C. Herring and J. I. Palmore, "Random number generators are chaotic," *Communications of the ACM,* vol. 38, no. 1, pp. 121-122, 1995.

[209] F. Takens, "Detecting strange attractors in turbulence," *Lecture notes in mathematics,* vol. 898, no. 1, pp. 366-381, 1981.

[210] H. F. Lopes. (2016, 23rd September). *AR, MA and ARMA models*. Available: http://hedibert.org/wp-content/uploads/2016/04/ar-ma.pdf

[211] J. F. Gibson, J. Doyne Farmer, M. Casdagli, and S. Eubank, "An analytic approach to practical state space reconstruction," *Physica. D, Nonlinear phenomena,* vol. 57, no. 1-2, pp. 1-30, 1992.

[212] J. Lekscha and R. V. Donner, "Phase space reconstruction for non-uniformly sampled noisy time series," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 28, no. 8, p. 085702, 2018.

[213] T. Sohn, J. Seo, and J. Moon, "A study on the covert channel detection of TCP/IP header using support vector machine," in *International Conference on Information and Communications Security*, 2003, pp. 313-324: Springer.

[214] D. Morgan. (4 May). *CS530L - Security Systems*. Available: http://www-scf.usc.edu/~csci530l/downloads/

[215] K. Ahsan, "Covert channel analysis and data hiding in TCP/IP," *Canada, University of Toronto,* 2002.

[216] E. Zielinska, W. Mazurczyk, and K. Szczypiorski, "Trends in steganography," *Communications of the ACM,* vol. 57, no. 3, pp. 86-95, 2014.

[217] Wireshark, "Wireshark Go Deep," ed: Online, 2010.

[218] K. Cabaj, W. Mazurczyk, P. Nowakowski, and P. Żórawski, "Towards Distributed Network Covert Channels Detection Using Data Mining-based Approach," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, p. 12: ACM.