Approaches and Techniques for Fingerprinting and Attributing Probing Activities by Observing Network Telescopes

Elias Bou-Harb

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

April 2015

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Elias Bou-Harb**

Entitled: **Approaches and Techniques for Fingerprinting and Attributing Probing Activities by Observing Network Telescopes**

and submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

C. Mulligan	Chair
N. Zincir-Heywood	External Examiner
D. Qiu	External to Program
T. Eavis	Examiner
J. Bentahar	Examiner
M. Debbabi	Thesis Supervisor
C. Assi	Co-supervisor

Approved by

Chair of Department or Graduate Program Director

Dean of Faculty

ABSTRACT

Approaches and Techniques for Fingerprinting and Attributing Probing Activities by Observing Network Telescopes

Elias Bou-Harb, Ph.D.
Concordia University, 2015

The explosive growth, complexity, adoption and dynamism of cyberspace over the last decade has radically altered the globe. A plethora of nations have been at the very forefront of this change, fully embracing the opportunities provided by the advancements in science and technology in order to fortify the economy and to increase the productivity of everyday's life. However, the significant dependence on cyberspace has indeed brought new risks that often compromise, exploit and damage invaluable data and systems. Thus, the capability to proactively infer malicious activities is of paramount importance. In this context, generating cyber threat intelligence related to probing or scanning activities render an effective tactic to achieve the latter.

In this thesis, we investigate such malicious activities, which are typically the precursors of various amplified, debilitating and disrupting cyber attacks. To achieve this task, we analyze real Internet-scale traffic targeting network telescopes or darknets, which are defined by routable, allocated yet unused Internet Protocol addresses.

First, we present a comprehensive survey of the entire probing topic. Specifically, we categorize this topic by elaborating on the nature, strategies and approaches of such probing activities. Additionally, we provide the reader with a classification and an exhaustive review of various techniques that could be employed in such malicious activities. Finally, we depict a taxonomy of the current literature by focusing on distributed probing detection methods.

Second, we focus on the problem of fingerprinting probing activities. To this end, we design, develop and validate approaches that can identify such activities targeting enterprise networks as well as those targeting the Internet-space. On one hand, the corporate probing detection approach uniquely exploits the information that could be leaked to the scanner, inferred from the internal network topology, to perform the detection. On the other hand, the more darknet tailored probing fingerprinting approach adopts a statistical approach to not only detect the probing activities but also identify the exact technique that was employed in the such activities.

Third, for attribution purposes, we propose a correlation approach that fuses probing activities with malware samples. The approach aims at detecting whether Internet-scale machines are infected or not as well as pinpointing the exact malware type/family, if the machines were found to be compromised. To achieve the intended goals, the proposed approach initially devises a probabilistic model to filter out darknet misconfiguration traffic. Consequently, probing activities are correlated with malware samples by leveraging fuzzy hashing and entropy based techniques. To this end, we also investigate and report a rare Internet-scale probing event by proposing a multifaceted approach that correlates darknet, malware and passive dns traffic.

Fourth, we focus on the problem of identifying and attributing large-scale probing campaigns, which render a new era of probing events. These are distinguished from previous probing incidents as (1) the population of the participating bots is several orders of magnitude larger, (2) the target scope is generally the entire Internet Protocol (IP) address space, and (3) the bots adopt well-orchestrated, often botmaster coordinated, stealth scan strategies that maximize targets' coverage while minimizing redundancy and overlap. To this end, we propose and validate three approaches. On one hand, two of the approaches rely on a set of behavioral analytics that aim at scrutinizing the generated traffic by the probing sources. Subsequently, they employ data mining and graph theoretic techniques to systematically cluster the probing sources into well-defined campaigns possessing similar behavioral similarity. The third approach, on the other hand, exploit time series interpolation and prediction to pinpoint orchestrated probing campaigns and to filter out non-coordinated probing flows.

We conclude this thesis by highlighting some research gaps that pave the way for future work.

DEDICATION

I dedicate this thesis to my family. Without their unconditional love and ever-lasting support, nothing of this would have been possible.

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere and utmost gratitude to my supervisors, professors Mourad Debbabi and Chadi Assi. Without their crucial guidance, genuine care and paramount support, I would have not been where I am today. As I move forward in my career, I promise them to abide with what they have taught me from the academic as well as the personal perspectives.

I would also like to thank my brothers and colleagues Claude Fachkha, Agop Koulakezian and Mohamad Mehdi for always being there throughout the course of our doctorate degrees. Their continuous support, collaborative characters and authentic care were decisive to complete this degree.

My gratitude also goes to my colleagues, Amine Boukhtouta, Hamad Binsalleeh, Nour-Eddine Lakhdari, Farkhund Iqbal, Djedjiga Mouheb, Elise Epaillard, Ribal Atallah, Bassam Moussa, Nicolas El Khoury, Son Dinh and Taher Azeb. I thank them for their collaborations and their friendship, wishing them the best of luck in their future endeavors.

I would also like to express my appreciation towards the staff of the Concordia Institute for Information Systems Engineering. I thank them for providing crucial aid and constant support throughout the 6 years that I have spent at Concordia University.

Finally, I would like to thank my partner Camille Perrin for her love and care and my family for their ever-lasting support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Cyberspace is the electronic world created by interconnected networks of information technology and the information on those networks. It can be defined as the interdependent network of information technology infrastructure, including the Internet, telecommunication networks, computer systems, and embedded industrial processors and controllers. Cyberspace is a global commons where more than 1.7 billion people are linked together to exchange ideas and services [8]. Moreover, it underpins almost every facet of a modern society and provides critical support for the economy, civil infrastructure, public safety, and national security. Cyberspace is controlled and operated using information and communication technologies. The latter could be considered as the nervous system of our today's world, as critical infrastructure such as telecommunication, transportation, financial services, agriculture, electric grids and public health services profoundly depend on it for their successful operations.

It is evident that individuals, industry and governments are embracing the many advantages that cyberspace offers. According to two recent reports [8, 9], 87% of North American corporations used the cyberspace to conduct business, where the online sales revenue due to that were estimated at $62.7 billion. Moreover in 2013, 74% of households had paid Internet service, 59% of personal tax filings were completed electronically and 67% of North Americans had banked online. Furthermore, governments have also become increasingly dependent on the Internet. The Canadian federal Government alone offers more than 130 commonly used services online, including tax returns, employment insurance forms and student loan applications [10]. Thus nowadays, the success of cyberspace is an essential asset which demands protection against malicious misuse and other destructive attacks. This task is indispensable yet very challenging.

Recent events have indeed demonstrated that cyberspace could be subjected, at the speed of light and in full anonymity, to severe attacks with drastic consequences. One particular research revealed that 90% of corporations have been the target of a cyber attack, with 80% suffering a significant financial loss [11]. In addition, the cyber security report [8] elaborated that in a recent one year period, 86% of large North American organizations had suffered a cyber attack where the loss of intellectual property as a result of these attacks doubled between 2011 and 2013. Moreover, the report alarmed that more than 60% of all the malicious code ever detected, originating from more than 190 countries, was introduced into cyberspace solely in 2013. In general, cyberspace could facilitate the following cyber attacks:

- Distributed Denial of Service (DDoS) [12]: It is an attempt to make a computer or network resources unavailable. It consists of attacks that are deployed to temporarily or indefinitely shutdown services. The timing of such attacks can be coordinated to exploit the availability of critical organization infrastructure by directing enormous flood of Internet traffic towards a small set of targeted Internet Protocol (IP) addresses. By flooding the available bandwidth with intensive traffic, DDoS can effectively bring down a service with potential loss of financial revenue.
- Advanced Persistent Threats (APTs) [13]: These cyber attacks employ high stealthy techniques and are typically target-specific threats. They are advanced since their operators have a full spectrum of intelligence-gathering techniques at their disposal. APTs assign priorities to specific tasks rather than opportunistically seek information for financial or other gain. The attack is typically conducted through continuous monitoring and interaction in order to achieve the defined objectives. Further, such

attacks are executed by coordinated human actions rather than just relying on automated pieces of code. Their operators are often very skilled, motivated, organized and well funded by external organizations.

- Zero-day Attacks [14]: These attacks exploit the observation of newly discovered yet un-patched vulnerabilities to achieve their malicious tasks. While a number of detection mechanisms have been proposed to protect against these attacks, including, access control lists on the edge network, port-knocking and application white-listing [15], these cyber attacks are still very dominant and pose serious issues and challenges.

- Cyber Terrorism: With the increase of cyber warfare incidents such as Stuxnet [16] and the Russian-Georgian War [17], cyber attacks can shift from targeting corporations to targeting governments and military facilities. With the ever escalating political tension between various world parties, these cyber attacks are becoming a fourth dimension of warfare [18] and a leading advantage to their operators.

Further, numerous incidents demonstrated that cyberspace has been exposed to amplified, debilitating and disrupting cyber attacks leading to drastic impacts on provided network and Internet services. For instance, Google has recently been targeted by a cyber attack in which 7 of its services, including, maps, news and translator, were hacked and defaced [19]. Further, a leading North American university has lately announced that a critical online database containing students and professors sensitive information, including social insurance numbers, addresses, and salaries, has been leaked to an anonymous party [20]. Another academic institute disclosed that several high-ranked employees' direct-deposit earnings have been hijacked and redirected to unknown and untraceable bank accounts [21]. Moreover, the Petroleum Producers' Associate, an oil and gaz organization, suffered from a devastating cyber attack that hit its website and its operations [22]. Another example would be numerous governmental websites of the United States, Russia, Finland, Pakistan, and Armenia that were also recently deemed as victims of cyber crime [23, 24, 25]. Indeed, despite efforts to protect the cyberspace, the latest reports from senior government officials [26] highlighted that only limited progress has been made in improving the cyber security of crucial networks.

1.1 Context and Motivation

Probing, the task of scanning enterprise networks or Internet wide services, searching for vulnerabilities or ways to infiltrate IT assets, is a significant cyber security concern. The latter is due to the fact that probing is commonly the primary stage of an intrusion attempt that enables an attacker to remotely locate, target, and subsequently exploit vulnerable systems. It is basically a core technique and a facilitating factor of the above mentioned and others cyber attacks. For instance, hackers have employed probing techniques to identify numerous misconfigured proxy servers at the New York Times to access a sensitive database that disclosed more than 3,000 social security numbers [27]. Further, the United States Computer Emergency Readiness Team (US-CERT) revealed that attackers had performed coordinated probing activities to fingerprint WordPress sites and consequently launched their targeted attacks [28]. Moreover, it was disclosed that hackers had leveraged sophisticated scanning events to orchestrate multiple breaches of Sony's PlayStation Network taking it offline for 24 days and costing the company an estimated $171 million [29]. More alarming, a recent incident reported that attackers had escalated a series of "surveillance missions" against cyber-physical infrastructure operating various US energy firms that permitted the hackers to infiltrate the control-system software and subsequently manipulate oil and gas pipelines [30]. Thus, it is not surprising that Panjwani et al. [31] concluded that a momentous 70% of attacks against cyber systems are preceded by some form of probing activity.

1.2 Objectives and Contributions

The main objective of this thesis is to generate cyber threat intelligence related to the inference and attribution of probing activities. Indeed, the capability to infer and attribute probing activities is a very important task to achieve, as this will aid in preventing cyber attacks from occurring or vulnerabilities from being exploited. Specifically, the mentioned aim could be clarified by the following three complementary objectives:

- Provide inferences and insights to enterprise networks related to their perceived probing activities in addition to generating global cyber intelligence related to Internet-wide malicious activities.
- Investigate probing activities in an attempt to attribute such activities to certain malware infections as well as to certain Internet-scale malicious events.
- Design approaches that can infer and attribute large-scale probing campaigns, which refer to a very recent phenomenon of such activities.

This thesis attempts to tackle the above mentioned objectives. To this end, our contributions can be summarized as follows:

- Design and implementation of approaches for inferring probing activities.
- Design and implementation of correlation mechanisms between probing activities, malware samples and passive dns traffic for attribution purposes.
- Design and implementation of approaches and techniques for inferring and attributing large-scale probing campaigns.

It is worthy to mention that the outcome of this thesis has been published in [32, 33, 34, 35, 36, 7, 37, 38]. Moreover, other work, which was executed during the course of the Ph.D. but is not included in the thesis, has appeared in [39, 40, 41, 42, 43].

1.3 Organization

The road-map of this thesis is as follows. In the next Chapter, we provide a survey related to the probing topic. Further, we provide necessary background information related to network telescopes and show how it can be exploited to generate various cyber threat intelligence. In Chapter 3, we elaborate on the work related to the design, implementation and validation of approaches for inferring enterprise and Internet-scale probing activities. In Chapter 4, we describe the design and implementation of a correlation mechanism between probing and malware activities for attribution purposes. To this end, we also report a rare Internet-scale malicious event by executing a multifaceted approach that correlates three types of real cyber security data. In Chapter 5, we tackle the problem of inferring and attributing large-scale probing campaigns. In this context, we present, validate, compare and contrast three devised approaches. Finally, Chapter 6 concludes this thesis, summarizes its contributions and highlights some research gaps that pave the way for future work.

Chapter 2

Background

In this chapter, we present a comprehensive survey of the entire probing topic. Further, we disclose necessary background information that aims at facilitating the highlighted concepts in this thesis.

2.1 Cyber Scanning: A Comprehensive Survey

This section presents a comprehensive survey of the entire probing topic. It categorizes cyber scanning by elaborating on its nature, strategies and approaches. It as well provides the reader with a classification and an exhaustive review of its techniques. Moreover, it taxonomies the current literature by focusing on distributed cyber scanning detection methods.

2.1.1 Related Surveys

Although probing or cyber scanning has been studied before, especially its detection techniques, the literature still lacks a survey that provides the readers, coming from different backgrounds, with a comprehensive coverage of the topic. For instance, Barnett et al. [44] solely focused on scanning techniques by providing a taxonomy. Their taxonomy analyzed three main scanning techniques, namely, TCP, UDP and ICMP scans. They presented the techniques using patterns and utilized scanning speed as an additional attribute. In another survey, Bhuyan et al. [45] elaborated on port scans and their detection methodologies. This survey highlighted on single-source and distributed detection techniques. Moreover, it provided some information on available detection data sets and evaluation metrics. The above two surveys are the closest to the work that we present in this section. However in this manuscript, we assert that we further contribute in the following points:

1. By primarily providing a categorization of the entire probing topic. We achieve this by discussing cyber scanning's nature, approaches and strategies. This offers the readers a strong, coherent and a clear entry point into the topic.

2. By providing a classification for 19 cyber scanning techniques. We thoroughly further discuss this exhaustive and comprehensive list of techniques, and provide their advantages and disadvantages. We as well present a complete summary of those techniques.

3. By withdrawing a unique literature taxonomy of distributed cyber scanning detection methodologies. This as well covers new material after 2010, the year of the latest related survey [45].

2.1.2 Cyber Scanning: Nature, Strategies & Approaches

In this section, we provide a categorization of the entire cyber scanning topic as depicted in Figure 2.1. We further present a discussion that elaborates on the nature, strategies and approaches of cyber scanning.

Nature of Cyber Scanning

The cyber scanning topic can be first classified based on its nature. The nature deals with whether the scanning or probing is performed actively or passively. In this section, we present those criteria and consequently discuss their advantages and disadvantages.

Active Scanning: Active scanning is the process of identifying network services by injecting certain packets known as the probe packets towards network hosts and devices and subsequently monitoring their responses. Active scanning is typically employed by

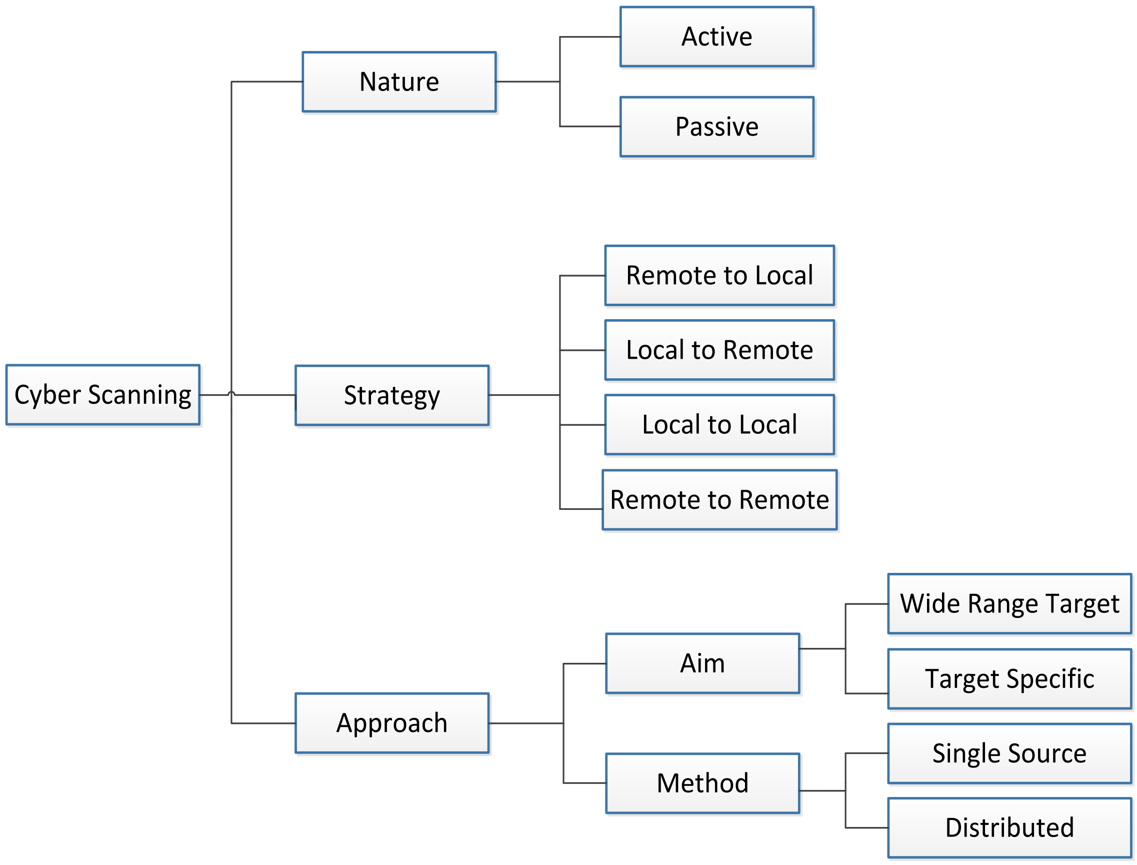


Figure 2.1: A Categorization of the Cyber Scanning Topic

malicious adversaries to probe a network for certain vulnerabilities. However, active scanning has a legitimate use as part of a robust network security policy. It allows a network operator to discover the open services in the network in an attempt to check those for known vulnerabilities. The probe packets could either be generic, targeting a specific protocol rather than a certain application, or they can be targeted, focusing on a precise host application. An instance of a generic probe packet could be the typical TCP handshaking procedure [46] for establishing a connection. The latter technique could be used to identify services operating on well-known ports. However, this technique is deficient in two cases. First, this method will only verify the readiness to open a TCP connection and not what service is supported by the connection. Thus, it tends to misinterpret services running on non standard ports. Second, it fails to classify services that have no standard ports, or those that use dynamic port assignment such as services utilizing the remote procedure call (RPC) protocol [47]. UDP probing is another employed approach for active scanning. Certain protocols, especially those running on well-known UDP ports, will successfully respond to a UDP probe packet. Moreover, it can be indirectly inferred the presence of a UDP service by lack of a negative response; many hosts automatically generate ICMP port unreachable messages [31] when no process is listening to a given UDP port. Although a lack of response is not definitive, but it might indicate the presence of a UDP service.

One example of active scanning is known as operating systems fingerprinting [48]. This procedure is rendered by the real-time attempt to remotely determine the operating system (type and version) of a particular host of interest. The idea is to send packets to a host so that any responses (or lack of responses) could be analyzed. The responses to these sequences of packets form a signature or a fingerprint for the remote operating system that can be compared against a signature database of known operating system versions. Operating system fingerprinting takes advantage of the observation that each operating system's network stack [46] (i.e., software that implements the TCP/IP protocol) has slight variations in the way it responds to certain packets. These variations offer the ability to determine the type of the remote host operating system. Another example of active scanning is application fingerprinting [49]. It is the real-time action of trying to remotely determine the applications or services running on a particular host of interest. Servers routinely send information about the applications they are running to client systems during normal connection activities. The initial text sent by servers during a connection attempt is known as a banner. The act of harvesting banners during an active identification of network systems and their applications [50] is an interesting and a beneficial concept. For instance, banner grabbing would be routinely performed during vulnerability testing (i.e., penetration testing) of the network. The software versions advertised in application banners can identify potential security issues if it is determined that the software version contains known vulnerabilities.

Passive Scanning: Passive scanning [51] identifies network services by observing traffic generated by servers and clients as it passes an observation point. Specialized hardware or software could be inserted and installed at the monitoring point to successfully establish passive monitoring. Many routers can ‘mirror’ ports, sending copies of packets out another interface to a monitoring host. Furthermore, hardware taps such as optical splitters place no additional burden on the router, but require a brief service interruption to install. Alternatively, Wireshark [52] is one of the most prominent passive software tools.

Detection of well-known services (both TCP and UDP) with passive monitoring is fairly straightforward. An exchange of traffic with a given host indicates an operational service. For TCP, the monitoring host only need to capture TCP connection setup messages (i.e., the SYN packets [46]); completion of the three way handshake clearly indicates that a service is available. Under normal operations, the presence of a positive response to a connection request (SYN/ACK) is a sufficient evidence of a TCP service. UDP services can also be identified by observing traffic; however, since UDP is a connectionless protocol, the concept of ‘server’ and ‘client’ is not sufficiently clear without application protocol information. In addition, while bi-directional traffic positively indicates a UDP service, unidirectional traffic may also indicate a service (since UDP does not mandate a response), but may as well indicate unsolicited probe traffic. As with active probing, passive scanning can not identify services that do not run on well-known ports.

One example of passive monitoring is Passive Asset Detection System (PADS) [53]. The system is a signature-based software used to passively detect network assets using application fingerprinting. It attempts to provide an accurate and current listing of the hosts and services offered on the network. It utilizes the TCP, ARP, and ICMP protocols [54] to perform its signature matching.

Discussion: Based on the aforementioned active and passive scanning descriptions, we consequently discuss their advantages and disadvantages.

In general, active scanning provides a comprehensive report of all open and unprotected ports at the time of the probing. However, it will not detect ports that are filtered by firewalls or obscured by mechanisms such as port knocking [55]. Active scanning typically performs very fast in achieving its task. The main disadvantage of active probing is that it is very intrusive. Active probes solicit a response that would not have been sent otherwise. This can be detected and logged by the host or intrusion detection systems, particularly if one systematically scans all hosts in a region. A second disadvantage of active scanning is that it does not identify hosts that may be temporarily unavailable at the time of the scan. This disadvantage can be mitigated with multiple active scans, although additional scans my draw further attention and hence increase the probability of being detected.

Passive monitoring has the advantage of being non-intrusive. In fact, it generally cannot be detected by either communication parties. A second advantage of passive monitoring is that it can better detect active services running on transient hosts. Thus, it is specifically effective against machines that are frequently powered off such as laptops, or hosts temporarily disconnected from the network. Third, passive monitoring can detect services that active probing misses because of firewall configurations. Fourth, passive monitoring can also provide insights into trends and other behaviors which active probing cannot. While monitoring servers, passive monitoring can also track clients, providing extra information such as server popularity and server load. Finally, since passive monitoring consumes no network resources (other than the monitoring host), it can be run on a long-term basis as part of normal network operations. The main disadvantage of passive monitoring is that it only detects services that are active. Therefore, silent servers go unmonitored, even though they may still pose vulnerabilities. Nevertheless, this disadvantage can be mitigated by long term monitoring.

Cyber Scanning Strategies

Cyber scanning activity can as well be defined by which strategy it adopts. Mainly, we can classify those strategies into four classes; remote to local scanning, local to remote scanning, local to local scanning and remote to remote scanning. The first three classes take into consideration the boundaries of a specific enterprise network and define the direction of the cyber scanning activity. Such activity can be generated by a diverse numbers of hosts, targeting any number of hosts, and using various cyber scanning methods and techniques.

The remote to local scanning refers to a remote host, outside the boundary of a specific network, performing some sort of cyber scanning on a host inside the enterprise network. This strategy is the most worrisome for enterprise network administrators as they attempt to protect their IT infrastructure from unknown external adversaries. Local to remote cyber scanning occurs when a host, within the administrative control of the enterprise network, scan systems outside the network boundary. In this context, the scanning host is performing network reconnaissance against external systems. This strategy may cause serious legal issues against the enterprise network since its infrastructure would be used for malicious purposes against Internet systems [56]. Moreover, local to local cyber scanning refers to a host that scans systems within the boundaries of the enterprise network in which it resides. Topological scanning worms [57] frequently employ this type of scanning strategy. Local to local scanning activity can occur within or between network subnets. Figures 2.2, 2.3 and 2.4 summarize the aforementioned discussed three strategies.

On the other hand, remote to remote scanning does not depend on certain boundaries. It can be defined as world wide cyber scanning campaigns. Rather than focusing on a specific enterprise network as a target, this strategy aims at probing and sequentially exploiting Internet's wide services. This strategy is often distributed, possesses sophisticated stealth capabilities and is typically highly coordinated. Chapter 5 of this thesis will tackle the latter problem.

Cyber Scanning Approaches

The third classification of the cyber scanning topic, as shown in Figure 2.1, is based on its approaches. Such approaches are composed of aims and methods. The aims specify

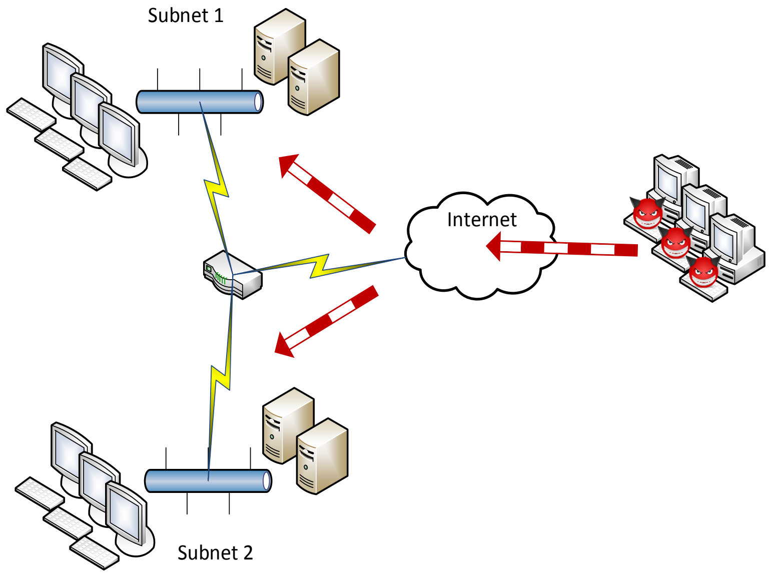


Figure 2.2: Remote to Local Probing

what is being targeted while the methods state how the cyber scanning is performed. The latter are discussed subsequently.

Wide Range Cyber Scanning: Wide-range reconnaissance can be defined as the rapid scanning of large blocks of Internet addresses in the search for a specific service or vulnerability. Typically, there is little human interaction in this type of reconnaissance. This is characteristic of auto-rooters [58] and worm propagation. Auto-rooters are composite tools that augment basic port scanning functionality by launching an attack as soon as an open port is located on a target system; they are often used for the rapid enrollment of vulnerable systems into botnets [59] compromised systems. Simple scanning worms propagate by indiscriminately probing the Internet as rapidly as possible to locate and infect vulnerable systems.

Target-Specific Cyber Scanning: In contrast, numerous sophisticated scanning techniques allow stealthy, focused scanning of a predetermined target host or network. The following techniques belong to this category:

- Indirect scanning occurs when an attacker uses some systems to scan a target and

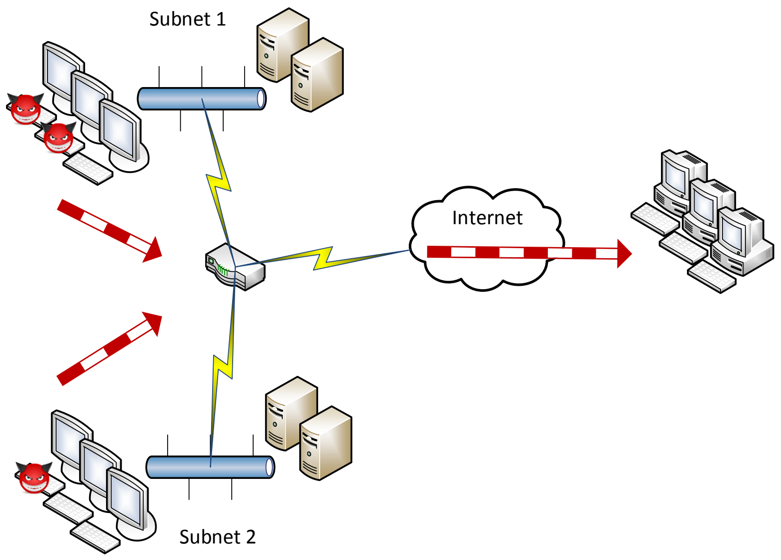


Figure 2.3: Local to Remote Probing

other systems to attack the same victim. If the scanning activity from the scanning system is detected, the attacker simply uses another scanning system. A slightly more sophisticated variation uses throw-away scanning systems; previously compromised systems are disposed by the attacker after executing the malicious tasks. In this case, any traced back scanning activity will be attributed to the owner of the compromised system and not to the real attacker.

- Botnet scanning [60] occurs when a collection of compromised systems (bots or zombies) are used to scan a target. The bots are not necessary on a contiguous set of IP address but rather could be very dispersed. For instance, consider a botnet that has an exploit capability against a network service. A botnet of just 254 systems would be able to scan an entire Class C network for that service by sending a single packet from each bot (each with a unique IP address). In this example, perhaps correlating the scanning campaign would be possible, however, it would not reveal the true adversary (the operator of the command and control center) since the bots are basically zombie members.

- Low and slow scanning [61] occurs when an attacker slowly scans a target host or

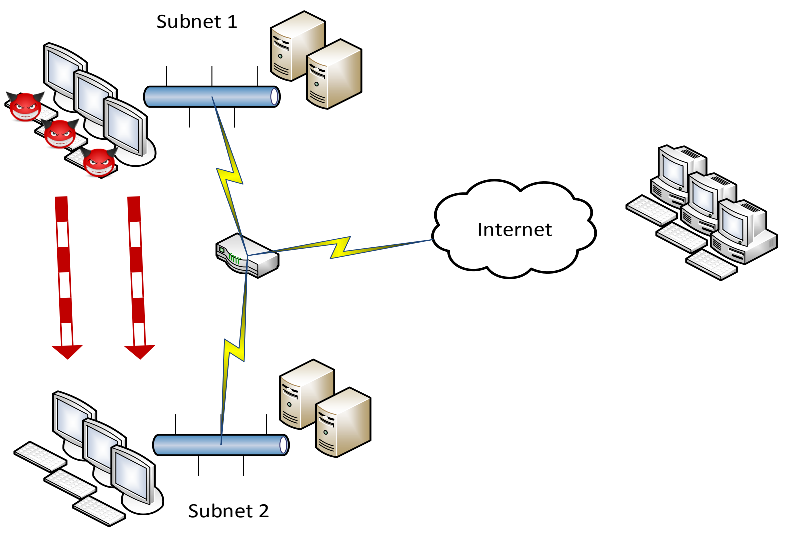


Figure 2.4: Local to Local Probing

network (i.e., a single scanning campaign may take days, weeks or months). Slow scans may blend into the network noise never exceeding detection thresholds or exhausting detection system state.

Single Source Cyber Scanning: A single source cyber scanning activity operates in a one (source) to many (targets) fashion. Single source cyber scanning could be classified as belonging to one of four types; vertical, horizontal, strobe and block scans [62]. A vertical scan consists of a port scan of some or all ports on a single computer. The other three types of scans are used over multiple IP addresses. A horizontal scan is a scan of a single port across multiple IP addresses. If the port scan is of multiple ports across multiple IP addresses, it is called a strobe scan. A block scan is a port scan against all ports on multiple IP addresses. Note that in general, a vertical scan can be defined as consisting of six or more ports on a single computer, while a horizontal scan as consisting of five or more IP addresses within a single subnet.

Distributed Cyber Scanning: Distributed scanning [63] occurs when multiple systems act in a union strategy to scan a network or host of interest. Typically, one system will act as a central node and collect the scanning results from all participating systems. Distributing the scanning activity reduces the scanning footprint from any single system and thus decreasing the likelihood of being detected.

Summary

In this section, we provided a categorization of the cyber scanning topic. Additionally, we presented a discussion that elaborated on the nature, strategies and approaches of cyber scanning. From the above, we can extract the following few points:

- Active scanning is efficient but is very intrusive.
- Passive scanning is less intrusive, works well in the presence of firewalls and is optimized to operate effectively with transient hosts.
- Cyber scanning strategies include remote to remote scanning also dubbed as cyber scanning campaigns. The latter possess sophisticated stealth capabilities and are typically highly coordinated.
- Botnet scanning is both target-specific and a distributed cyber scanning method.

2.1.3 Cyber Scanning Techniques

In this section, we introduce a classification of probing techniques as shown in Figure 2.5. We elaborate on this by presenting the techniques and their details in terms of exchanged messages and their scanning abilities. Moreover, we pinpoint and discuss, when applicable, their advantages, their disadvantages and the scenarios when the techniques are best used. Finally, we present a summary of the cyber scanning techniques that includes, but is not limited to, the transport protocol the technique aims to identify, their exchanged messages, and whether the technique is immune to firewall detection.

Open Scan

Open scan, also known as the vanilla scan, is the simplest scanning technique. It refers to the method that follows the same TCP handshake connection that every other TCP-based

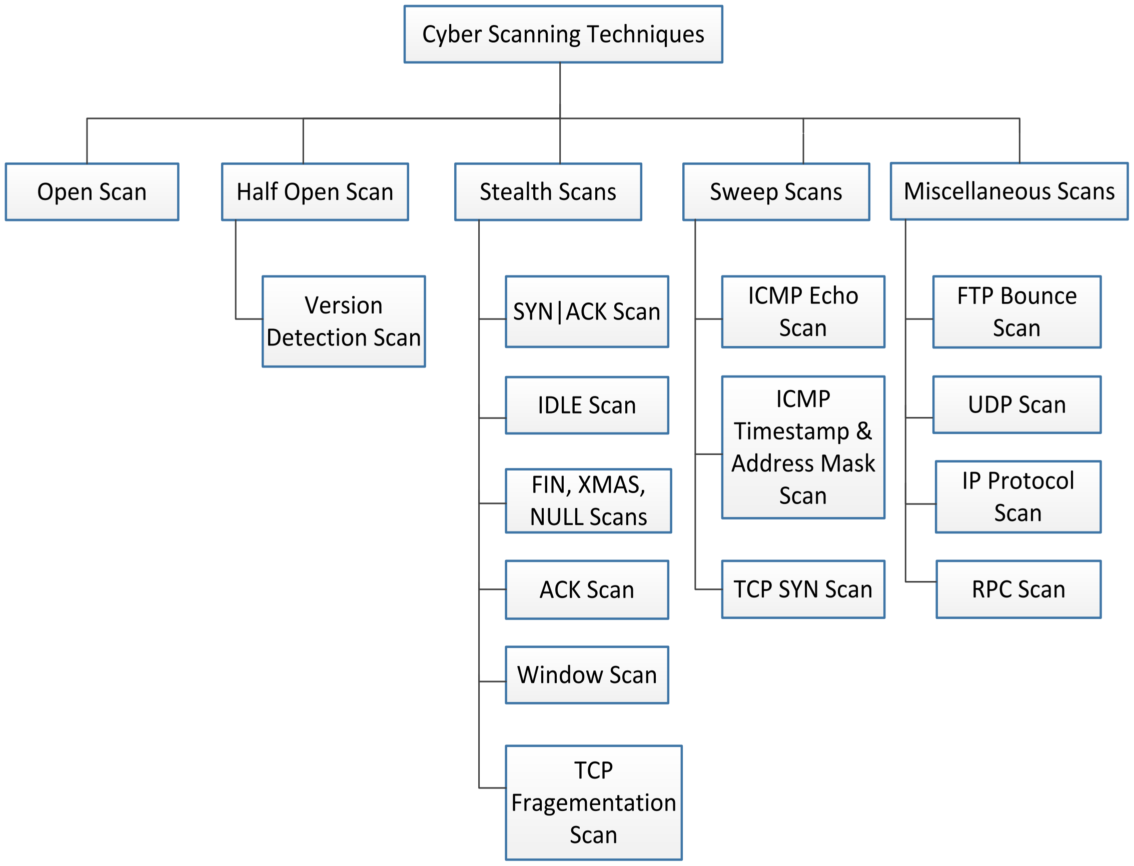
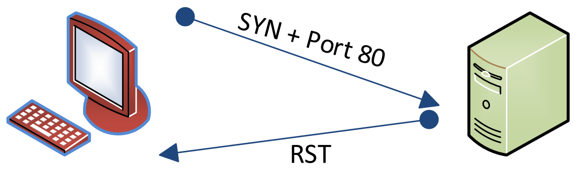


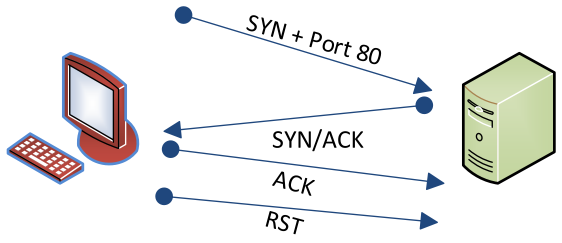
Figure 2.5: A Classification of Cyber Scanning Techniques

application uses. Hence, this scanning technique is considered 'Open' since it reacts as a normal TCP connection to determine if a port is available. It utilizes the connect() [64] call functionality that is used by the operating system to initiate a TCP connection to a remote device. The Open scan is illustrated in Figure 2.6.

This technique utilizes the TCP protocol and the SYN flag to detect TCP ports. When a closed port is targeted, the victim replies with a RST flag. On the other hand, when an open port is detected, the victim replies with an ACK flag. It is worthy to note that this simple technique is easily detected by a firewall. An advantage of this technique is that it can achieve its scan in a very simplistic way without requiring any other functionalities or privileges. The latter fact is because this technique utilizes systems' normal TCP-based methods when connecting to the target. A disadvantage of this scan is apparent when connection logs are examined. Since the Open scan technique requires



(a)



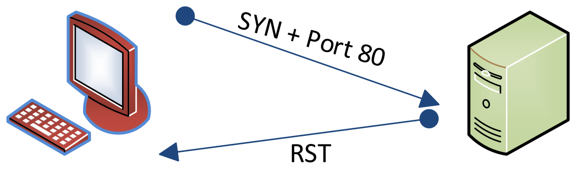
(b)

Figure 2.6: The Open Scan targeting a closed (2.6a) and an open port (2.6b)

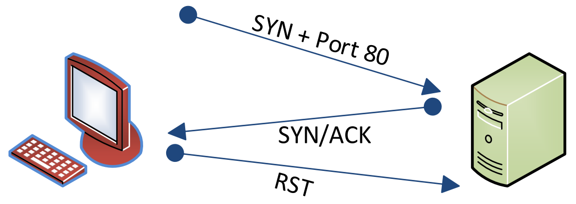
the completion of a TCP connection, normal application processes immediately follow. Although these applications are directly met with a RST packet, however, the application has already provided the appropriate login screen or service page. By the time the RST is received, the application initiation process is already well underway and additional system resources are used. Because this scan technique is evident and easily identified when browsing through applications event logs, it might be considered the TCP scan of last resort. If privileged access is not available and the determination of open TCP ports is absolutely necessary, this scan may be the only method that is available and can be used.

Half-Open Scan

The Half-Open scan, commonly dubbed as the TCP SYN scan, is a common method for port identification that allows the scanner to gather information about open ports without completing the TCP handshake process. When an open port is identified, the TCP handshake is reset before it can be completed. Similar to an Open Scan, a Half-Open scan targeting a closed port will receive a RST packet. However, if the source receives an acknowledgment to a SYN request, meaning a port is open, then the source directly sends a RST frame to reset the session, and therefore the handshake is never completed. This technique is shown in Figure 2.7.



(a)



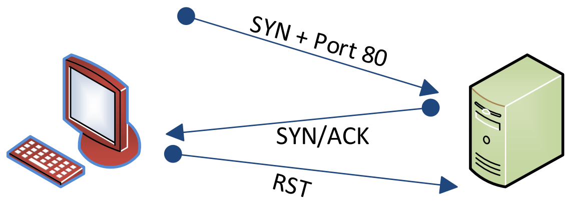
(b)

Figure 2.7: The Half-Open Scan targeting a closed (2.7a) and an open port (2.7b)

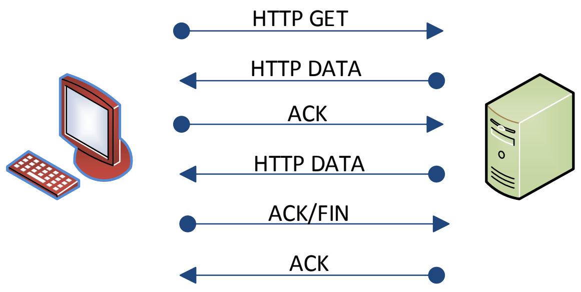
Since this scan technique never actually creates a TCP session, it is advantageous in two ways. First, it is not logged by destination applications. This point makes the Half-Open method somehow more stealthier than the Open-Scan method, as it is less visible in the destination systems' application logs. Second, it is less stressful to the application service because it does not force the application to initialize or for systems resources to be allocated. On the other hand, this method suffers from one disadvantage. Since there is a need to create new raw packets that do not completely abide by the TCP handshake, the half-open connection process requires some elevated systems privilege at the source to be successful, which is not always feasible. It is significant to mention that this method can operate on any operating platform and the fact that it only half-opens the TCP connections, makes it a very efficient and a streamline method. Nevertheless, since the Half-Open scan technique is structured and uses known TCP flags, it is effortlessly detected by an edge firewall.

Version Detection Scan

Although the Version Detection scan does not aim to detect open ports as the previous mentioned methods, however, it exploits them by probing the software applications running on remote devices. This method would typically utilize the Half-Open scan technique prior to executing its own scan method. If open ports are found, the Version Detection scan will begin the probing process by directly communicating with the remote applications on the open ports to uncover as much information as possible. Such information may include the type, the version and the status of that service, the underlying operating system and its version and other services that depend on that running service. This information can be of benefit to a network manager for proper and effective patching purposes or it can be analyzed by the scanner or attacker to exploit a certain known vulnerability of a specific running service. The Version Detection Scan is depicted in Figure 2.8. In



(a)



(b)

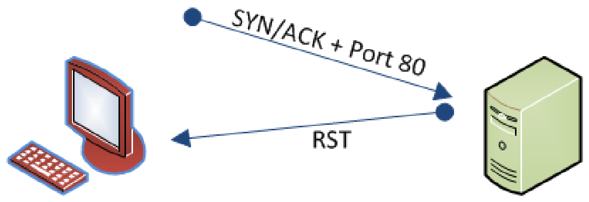
Figure 2.8: The Half-Open scan (2.8a) executing prior to the Version Detection Scan (2.8b)

this illustration, the Version Detection Scan technique first executed the TCP SYN scan. After detecting that port 80 is open, it ran its own scan to probe the service on that port. This method poses two advantages from a network management perspective. Its aids in network host applications' version management where hosts showing older software revisions are identified and further action is taken. Second, it assists in locating software that is not compliant with organizational standards. This is as well an effective method of verifying the licenses of application services. Nonetheless, this technique possesses two disadvantages. First, it requires significant processing power (less significant using today's machines but still a point to consider) and elevated networking bandwidth since it needs to probe all the services and consequently transmit all their information. Second, and since this technique will open numerous sessions with the remote applications, its activity is usually written in application logs which makes it a less stealthier technique.

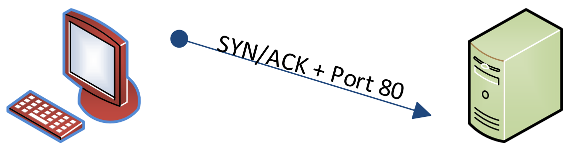
Stealth Scans

The aforementioned cyber scanning techniques only use the typical SYN flag to investigate open ports. Hence, they are easily detected and logged by intrusion detection systems. In this section, we present and discuss stealth scans. These techniques try to avoid filtering devices by employing certain set of flags other than SYN to appear as legitimate traffic. All these techniques resort to inverse mapping to determine open ports.

SYN|ACK Scan: The SYN|ACK scan is a slight modification of the Half-Open scan. Instead of just sending a SYN flag, the source sends a SYN in addition to an ACK flag to the target. For a closed port, the target will reply with a RST flag while a request to an open port will not generate a response. The latter is due to the fact that the TCP protocol requires a sole SYN flag to initiate a connection. The SYN|ACK scan is illustrated in Figure 2.9. This scan technique may generate a notable amount of false positives. For instance, packets dropped due to filtering devices, network traffic, timeouts, etc. can provide an incorrect inference of an open port while the port is in fact closed. However, this is a relatively fast scan method that avoids the three-way handshake and does not utilize a sole SYN flag.



(a)



(b)

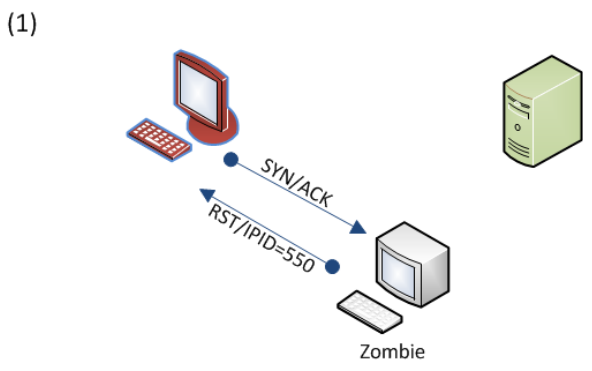
Figure 2.9: The SYN|ACK Scan targeting a closed (2.9a) and an open port (2.9b)

IDLE Scan: A more complex stealth technique that utilizes the previous SYN|ACK scan and the Half-Open scan is known as the IDLE scan. The technique aims at gathering port information using another station on the network (the zombie) where the scanning process appears as it has been initiated from the zombie IP address instead of the actual source station. This scanning method exploits IP fragmentation identification sequences and implements IP address spoofing. For the scanning process to be executed, the identified zombie machine should satisfy the following two requirements:

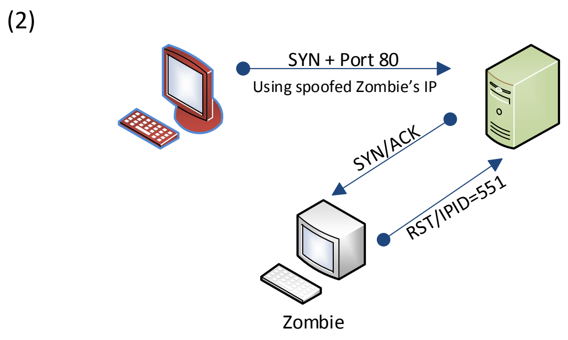
- The zombie host must be idle (hence the name 'IDLE' scan). This requirement ensures that the IP identification frames will remain consistent throughout the duration of the scan.
- The zombie host must provide consistent and predictable IP identification (IPID) values. Most operating systems satisfy this requirement.

This technique is clarified in Figure 2.10.

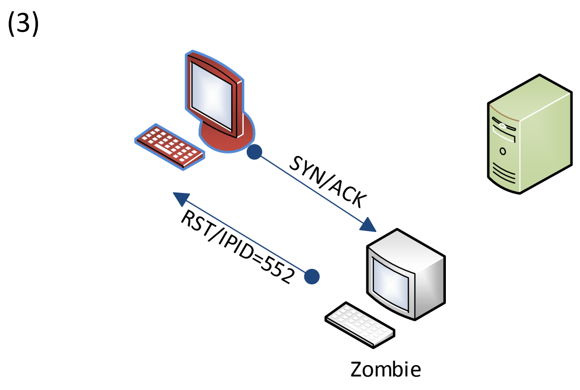
First, in Figure 2.10a, the source sends a SYN/ACK flags to the zombie host expecting a RST flag as a response. This RST packet contains the initial IPID. In Figure 2.10b, the source executes a Half-Open scan, using the spoofed IP address of the zombie, targeting the destination host. If that port is open, the destination will reply to the



(a)



(b)

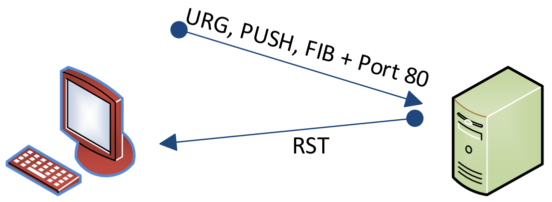


(c)

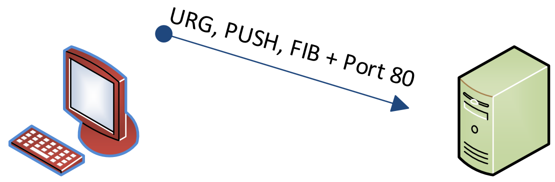
Figure 2.10: IDLE scan executing process

zombie with a SYN/ACK. The zombie, not expecting a SYN/ACK since he never sent a SYN, will reply by a RST packet. The latter response will indeed increment the zombie's IPID. Finally, in Figure 2.10c, the original host resends the initial SYN/ACK probe to the zombie station. If the IPID has been incremented, then the source will infer that the port that was spoofed in the original SYN frame is open on the destination target. If the IPID has not been incremented, then the source will conclude that the port is closed. IDLE scan technique of spoofing IP addresses and checking IPIDs allows the source to find open ports from a distance, even if packet filters are in place. The source simply requires any open port to a zombie host to complete the communication process. One of the core advantages of this technique is its stealth factor. A destination station will never identify the IP address of the scanning host. On the other hand, the disadvantages of this technique are three folds. First, there should be a satisfaction of the zombie workstation requirements prior to commencing the scanning process, especially the idle state requirement. Second, and although the technique implements source IP address spoofing, however, the source will still be identified if the technique is used on a local subnet. This last fact is legitimate since the source MAC address on that subnet is not spoofed and hence with some network investigation the source would be pinpointed. The third disadvantage is similar to the disadvantage of the Half-Open scan technique which is rendered by the inability to create raw packets that do comply completely with the TCP handshake procedure without elevation of privileges, which is not always achievable.

FIN, Xmas Tree, and Null Scans: These three cyber scanning techniques are grouped together since their individual functionality is very similar. They are members of the ‘stealth’ scans because they send a single frame to a TCP port without any TCP handshaking or any additional packet transfers. They operate identically to the SYN|ACK scan, but they differ by which flags they send. The FIN, the Xmas Tree and the Null scanning techniques respectively send packets with the FIN flag, URG, PUSH, and FIN flags, and packets with empty flags. In all cases, the closed ports are required to reply to the probe packet with RST, while open ports must ignore the packet in question [46, 65]. Note that, the Xmas Tree scan takes its name from the flags related to (00101001), which appear similar to the lights of a Christmas tree. The latter technique is depicted in Figure 5.2c, while the FIN and Null scans have similar illustrations as Figure 2.9 but with different flags as previously mentioned. Since no TCP sessions are created for any of these scans, they



(a)



(b)

Figure 2.11: The Xmas Scan targeting a closed (2.11a) and an open port (2.11b)

are remarkably quiet from the perspective of the remote device's applications. Therefore, none of these scans should appear in any of the application logs. These scans are as well some of the most minimal port-level scans that could execute. For a closed port, only two packets are transferred. On the other hand, only a single frame is necessary to identify an open port. However, these techniques have two drawbacks. They are ineffective against Microsoft machines as all ports will appear to be closed regardless of their actual state. Nonetheless, this provides a backhanded advantage, since any device showing open ports must not be a Windows-based device. The second drawback is related to generating raw packets, which as mentioned earlier in this section, requires elevation of systems' privileges.

ACK Scan: The ACK scan is not intended to identify an open port. This stealth cyber scanning technique will only provide a filtered (non-reachable) or an unfiltered (reachable) disposition because it never connects to an application to confirm an 'open' state. At a first glance, this appears to be rather limiting but in reality, the ACK scan can characterize the ability of a packet to traverse firewalls or packet filtered links. Another implementation of the ACK scan can take advantage of the IP routing function to deduce the state of the port from the time to live (TTL) value [66]. An ACK scan operates by sending a TCP ACK frame to a remote port. If there are no responses or an ICMP destination unreachable message is returned, then the port is considered to be filtered. If the remote port returns a RST packet, the connection between the source and the remote target is categorized as unfiltered. Figure 2.12 demonstrates the ACK scan process.

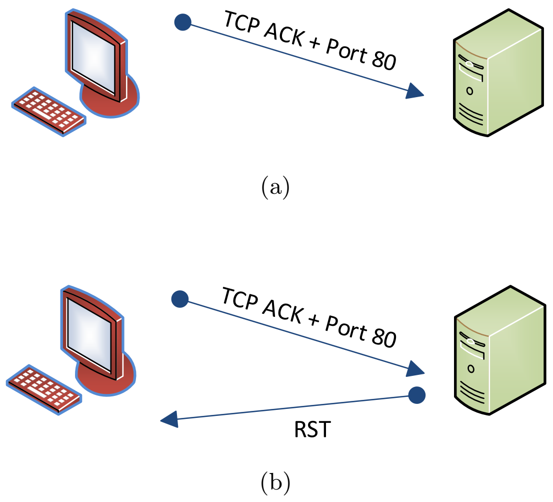
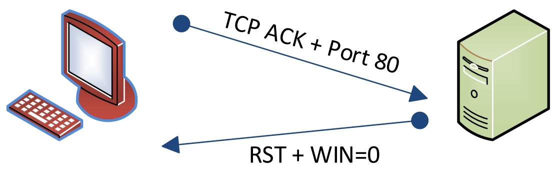


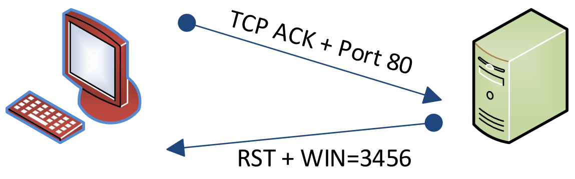
Figure 2.12: The Ack Scan targeting a non-reachable (2.12a) and a reachable target (2.12b)

On one hand, and since the ACK scan does not open any application sessions, the conversation between the source and the remote target is relatively simple. Thus, the scan of a single port is almost invisible, especially when combined with other network traffic. On the other hand, the ACK scans simplicity is also its largest disadvantage. Because it never tries to connect to a remote device, it can never definitively identify an open port. Although the ACK scan does not identify open ports, it does an impressive job of identifying ports that are filtered through a firewall. This list of filtered and unfiltered port numbers is extremely useful as a reconnaissance method for a future more detailed scan that focuses on specific port numbers and perhaps their vulnerabilities.

Window Scan: The Window scan, dubbed after the TCP sliding window [46], is a scanning technique used with certain TCP stacks. It is almost identical to the ACK scan, however, it has been found that certain TCP stacks return a window size number when responding to an ACK packet; a RST frame response from a closed port replies with a window size of zero and a RST frame response from an open port replies with a non-zero window size. Figure 3.5i shows the latter process.



(a)



(b)

Figure 2.13: The Window Scan targeting a closed (2.13a) and an open port (2.13b)

An advantage of the window scan is that it does not open a session, hence there exist no application log associated with the scanning operation. Unless there are additional firewalls or network limits at the operating system level, the scan should go unnoticed. However, the window scan does not operate on all devices, and the number of operating systems vulnerable to this unintended window size consistency is dwindling as operating systems are upgraded and patched. In general, the window scan is useful when looking for open ports while simultaneously maintaining a low level of network traffic. When vulnerable operating systems are identified, the window scan provides a low impact method of locating open ports.

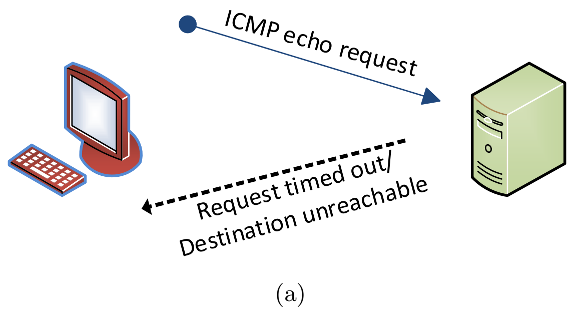
TCP Fragmentation Scan: This stealth scanning technique can be defined as a process of executing a scan rather than a scanning technique by itself. It employs either the Half-Open scan or the FIN scan techniques to carry out its scanning methodology. This technique exploits the idea of decomposing the packet header (the probe packet) into smaller packets in an attempt to evade packet filters. This technique almost always is effective since packet filters or intrusion detection systems do not buffer the entire set of packets due to performance issues. Rather, they process the packets individually causing the bypass of the assembled scanning packet. One possible drawback of this technique is rendered by the fact that some destinations do not have the ability to correctly merge the decomposed packets causing dropped probe packets and eventually the failure of this method.

Sweep Scans

At the beginning of this section, we have discussed certain cyber scanning techniques that solely adopt the SYN flag to carry out their procedure. Subsequently, we presented and elaborated on stealth scanning techniques that utilize a mixture of flags to achieve their goal. In this section, we highlight on Sweep scans. These techniques do not aim at identifying active ports but rather at identifying active hosts. They are characterized as performing sweeps, since their purpose is to identify the status of as many hosts as possible instead of focusing on an individual host. In fact, they typically utilize the network's subnet broadcast address as a destination address to target the majority of the hosts. They operate by generating any request that would prompt a remote station's response. They can be defined as cyber scanning facilitators because they pinpoint active hosts just before the actual scanning techniques of active hosts take place.

ICMP Echo Request Scan: This technique is one of the simplest and most known scanning technique to identify active hosts. It is abundantly used on Windows and Linux machines by invoking the 'ping' command. The idea is to send an ICMP echo request message to the target and wait for a reply. If there is an ICMP Echo Reply message, this indicates that the target is active. Otherwise, it means either the target is not active (request timed out reply) or the the original request never reached the target (destination unreachable reply). Figure 2.14 portrays this technique.

An advantage of the ICMP echo technique is that it does not depend on any particular application or open port to work. If a remote device communicates via TCP/IP,



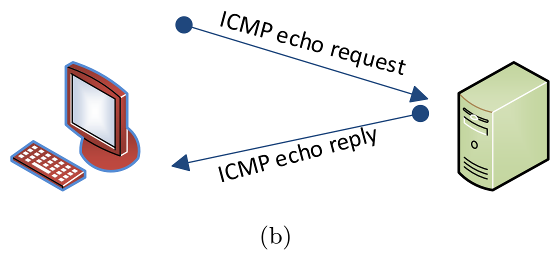
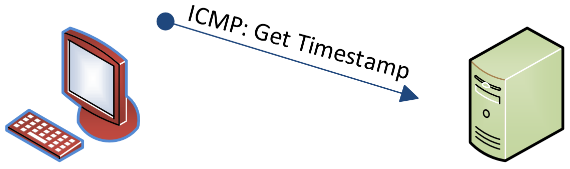


Figure 2.14: The ICMP Echo Request targeting a non-active (2.14a) and an active host (2.14b)

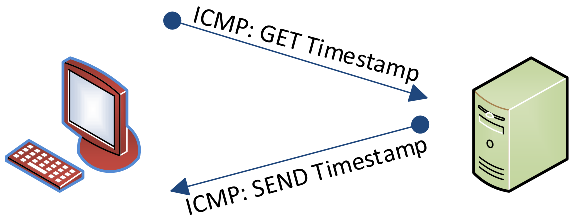
then it is most often a candidate for the ICMP echo request scan. A disadvantage of this technique is that ICMP is one of the most filtered protocols in enterprise networks. Since the ICMP protocol has the ability to redirect traffic, identify available workstations, and pinpoint closed ports on a target, when a firewall or packet filter is first installed, it is a common security guideline to restrict ICMP.

ICMP Timestamp & Address Mask Scans: These techniques take advantage of the seldom used ICMP messages (Timestamp & Address Mask) to determine if a remote target is active. They function similarly to other ICMP-based scans; the source sends an ICMP Get Timestamp or Get Address Mask messages and wait for an ICMP Send Timestamp or ICMP Send Mask responses. ICMP Timestamp scan is shown in Figure 2.15.

Both methods suffer from serious drawbacks. In both techniques, their corresponding probing messages are very rare to occur in a network and thus they can be very easily detected. Moreover, they both do not achieve promising results when targeting relatively



(a)



(b)

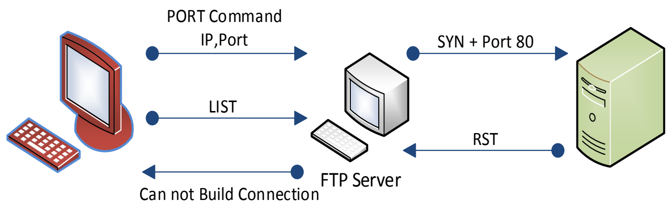
Figure 2.15: The ICMP Timestamp scan targeting a non-active (2.15a) and an active host (2.15b)

updated operating systems or networking hardware.

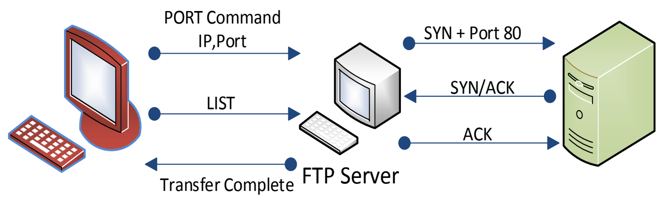
TCP SYN Scan: This cyber scanning facilitator technique is operationally identical to the Half-Open scan technique but the goal in this case is different. In this TCP SYN scan, the source scanner is awaiting a RST packet from a closed port or an ACK packet from an open port. The interesting and effective point of this technique is that either result from the scan will provide the source with a proof that an active system resides at that destination IP address. This technique is advantageous since it accomplishes its goal in just few packets. Such minimal amount of network traffic appears to be similar to a typical TCP handshake. As a result, this technique can appear as legitimate network traffic and would go undetected.

Miscellaneous Scans

This section aims at providing further cyber scanning insights by shedding the light on scans that deal with various protocols. These include the FTP bounce, UDP, IP protocol and RPC scans.



(a)



(b)

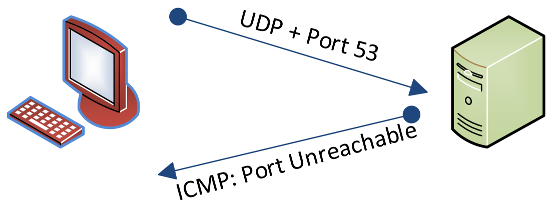
Figure 2.16: The FTP Bounce scan targeting a closed (2.16a) and an open port(2.16b)

FTP Bounce Scan: Similar to IDLE scan, the FTP bounce attack employs a third host (the FTP server) to act as a proxy between the source host (scanner) and the destination target. The FTP bounce attack takes advantage of the passive mode FTP [67]. This mode completely separate the command connections from the data connections. This allows the FTP server to be effective in the presence of firewalls since the FTP server would be responsible for building the outbound data connection with the remote host. Furthermore, it allows the source to send a PORT command [67] to the proxy FTP server where the latter would direct the data towards a completely different host (the target). The FTP Bounce scan is simplified in Figure 2.16. The first step of the FTP Bounce scan occurs when the source connects to the vulnerable FTP server. Sequentially, the source transmits a PORT command [67] coupled with the IP and port of the target. The FTP server forwards that request to the target. If the intended target's port is closed (Figure 2.16a), the FTP server responds to the source stating the it can not build the connection. On the other hand, if the port is open (Figure 2.16b), the FTP server responds with a message stating that a transfer has been successfully completed. Depending on the reply, the source will infer if the port is open or closed. The advantages of this technique are two folds. First, the technique uses the standard FTP communication to achieve its task. Since the FTP service is found in the majority of enterprise networks, the technique seems feasible almost all the time. Second, it possesses stealth features as the source is using a proxy to direct his scan. Conversely, the disadvantages of the FTP bounce scan comprise of the following: First, this scan technique is only successful on TCP ports. Since FTP does not connect to remote devices using UDP, it is not possible to retrieve any feedback on the availability of UDP ports. Second, the process of bouncing through an FTP server is slow when compared to other scanning methods. Additionally, the port scanning requests can only check a single port at a time. Third, and since this technique initiates an application session with the FTP server, the FTP servers will log the connection and all its commands making this method vulnerable to being detected.

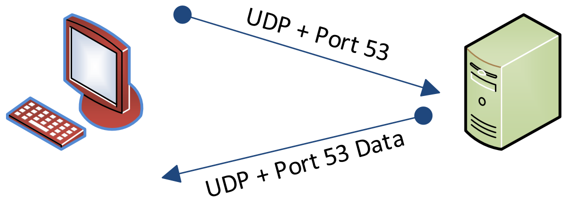
UDP Scan: UDP scan does not require any SYNs, FINs, or any other handshaking flags. The lack of formal communications process in UDP greatly amplifies the effectiveness of this scan technique. The UDP scan is demonstrated in Figure 2.17.

A closed port will reply with an ICMP port unreachable message while an open port responds with some UDP data. A critical advantage of this technique is that it operates very efficiently on Windows-based devices since they do not usually implement any type of ICMP rate limiting [68].

IP Protocol Scan: The goal of the IP protocol scan is to inquire about any additional IP protocols in use by the target station, including ICMP, TCP, and UDP. If a router would be scanned, additional IP protocols such as EGP or IGP may be identified. Figure 2.18 depicts this technique. An unavailable IP protocol does not respond to the scan while an available IP protocol provides a response specific to the protocol type. An IP protocol scan looks fairly obvious if packet traces are investigated; since most networking protocols are based on TCP or UDP, any deviation from those two protocol types is conspicuous.

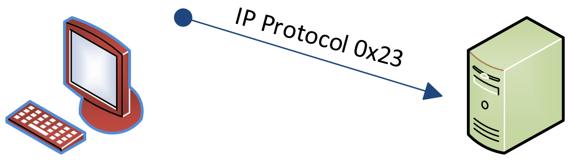


(a)

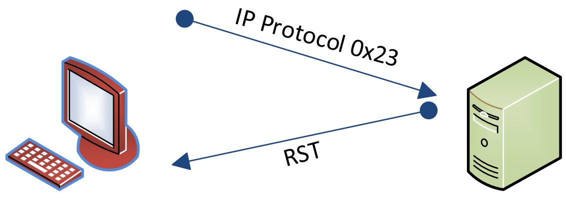


(b)

Figure 2.17: The UDP scan targeting a closed (2.17a) and an open port (2.17b)



(a)



(b)

Figure 2.18: Unavailable (2.18a) Vs. An Available IP Protocol (2.18b)

RPC Scan: In an attempt to disclose applications' information that operate using the remote procedure call (RPC) [47], the RPC scan sends RPC null messages to previously detected open ports. If any RPC application is running on the target, the reply will include

CST	EOP	ID. T	ID. U	PR	SM	RMC	RMO	IFD
Open scan	-	√	-	TCP	SYN	RST	ACK	-
Half-Open scan	√	√	-	TCP	SYN	RST	ACK	-
Version Detection scan	-	-	-	TCP	SYN	RST	ACK	-
SYN|ACK scan	√	√	-	TCP	SYN/ACK	RST	-	√
IDLE scan	-	√	-	TCP	SYN/ACK, SYN	RST/IPID	RST/IPID	√
FIN scan	√	√	-	TCP	FIN	RST	-	√
XMAS scan	√	√	-	TCP	URG, PUSH, FIN	RST	-	√
NULL scan	√	√	-	TCP	-	RST	-	√
ACK scan	√	√	-	TCP	ACK	-	RST	√
Window scan	√	√	-	TCP	ACK	RST+WIN	RST+WIN	√
TCP Fragm. scan	√	√	-	TCP	SYN or FIN	RST	-	√
ICMP Echo scan	-	-	-	ICMP	ICMP Echo	ICMP MSG	ICMP Reply	-
ICMP Timestamp scan	-	-	-	ICMP	ICMP Timestamp	-	Timestamp	-
ICMP Sub. Mask scan	-	-	-	ICMP	ICMP Sub.Mask	-	Sub. Mask	-
TCP SYN scan	-	-	-	TCP	SYN	RST	ACK	-
FTP Bounce scan	-	√	-	TCP	PORT	Error MSG	Conn. Est.	-
UDP scan	-	-	√	UDP	UDP Pkt	ICMP Unreach.	UDP Data	-
IP Protocol scan	-	-	-	IP	IP Prot. MSG	-	Protocols	-
RPC scan	-	-	-	RPC	RPC NULL	-	RPC App Info	-

Table 2.1: Summary of Probing Techniques

information such as the application's name, version and status. This technique possess the ability to detect RPC applications running on non-RPC default ports. On the contrary, and since the technique establishes application sessions, its transaction events will be written in application logs, and thus the technique is easily detected. Another drawback of this technique is that it relies on previously detected open ports to operate and does not detect open ports by itself.

Summary

The previously discussed cyber scanning techniques are summarized in Table 2.1. The summary includes the cyber scanning technique (CST), whether or not it requires elevation of systems's privileges at the source to operate (EOP), whether it identities TCP or

UDP ports (ID. T/ ID. U), the protocol it employs (PR), its messages that it sends (SM), its received messages when the target port is closed or the host is unreachable (RMC), its received messages when the target port is open or the host is reachable (RMO) and finally whether the technique is immune to firewall detection (IFD).

From the summary table, we can extract the following few points:

- TCP is the most employed cyber scanning protocol.
- Although stealth scanning techniques are immune to being detected by a firewall, however, almost all except the IDLE scan necessitate elevation of systems' privilege at the source to successfully operate.
- To identity UDP ports, only one cyber scanning technique could be utilized.
- The Half-Open scan technique could be used for port-identification as well as for detecting active network hosts.

2.1.4 Literature Review - Distributed Detection Techniques

In this section, we present a review of the recent literature on distributed cyber scanning detection techniques. Distributed cyber scanning, which is illustrated in Figure 2.19, refers to the task of decomposing and coordinating the scanning using various compromised systems or bots. Typically, the scanning is controlled by a main attacker dubbed as the scanning botmaster who operates the command and control center and the entire network of bots (or botnet) for coordinated communication, propagation, and other attack activities. Distributed cyber scanning is often thought of as operating in a many (sources) to one (target) fashion, where the target system is often a single entity or a limited number of systems. Moreover, this type of scanning possess stealth features and could be performed during a prolonged period of time (i.e., a slow scan).

The work presented in this section covers the period from 2001 up to November 2012. The studied literature solely focus on distributed detection techniques (many to one) and excludes single source detection techniques (one to one and one to many). For the latter,

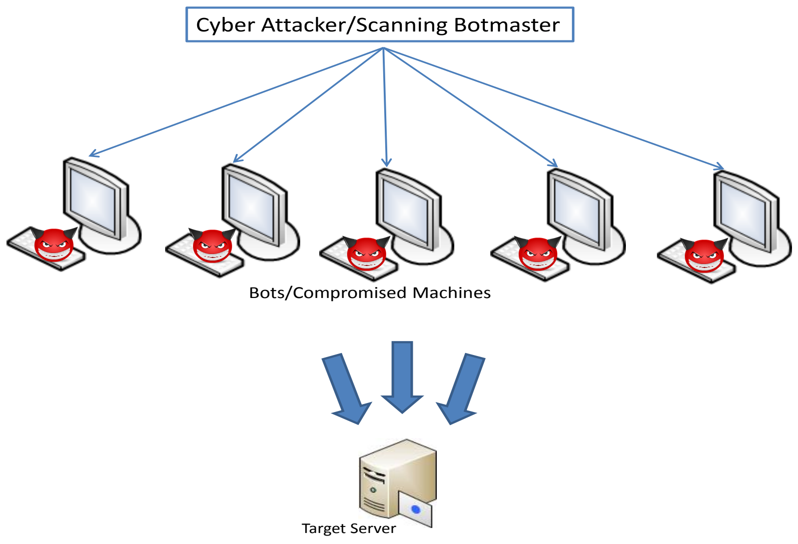


Figure 2.19: Executing Distributed Cyber Scanning

please refer to survey [45]. Many to many detection techniques, as briefly discussed in Section 2.1.2, are yet to be investigated in the literature. The generated taxonomy on distributed cyber scanning detection techniques is shown in Figure 2.20. It is based on the approaches taken by the authors to achieve their detection task. The approaches are decomposed into four categories, namely, statistical, algorithmic, mathematical and heuristical.

Statistical Approaches

These distributed cyber scanning detection approaches include techniques such as statistical characterization (features) of data samples, extrapolation or interpolation of data based on some best-fit, error estimates of observations, or spectral analysis of a data model.

Zhang et al. [1] proposed a scan detection method based on a distributed cooperative model. Their technique is composed of feature-based detection, scenario-based detection and statistic-based detection. Their proposed architecture, which is depicted in Figure 2.21, is decomposed into 5 layers; sensors, event generators, event detection agents, a fusion center, and a control center.

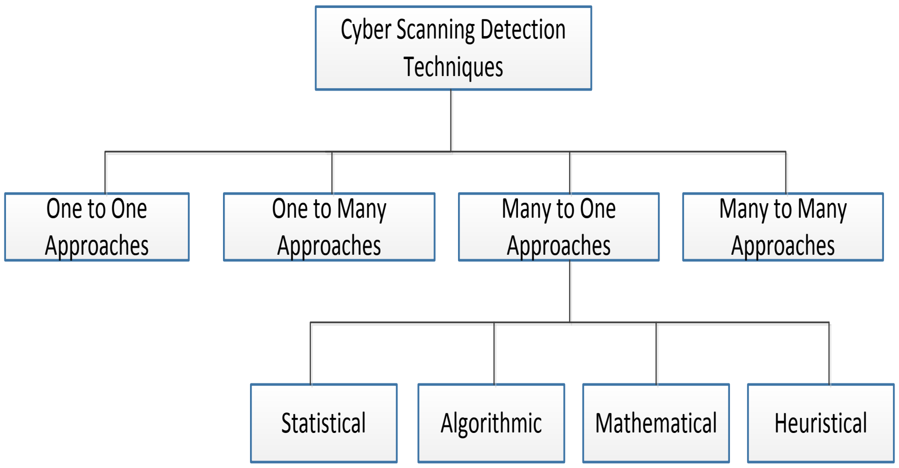


Figure 2.20: Taxonomy-Distributed Cyber Scanning Detection Techniques

The authors explained that the sensors collect data and system log information. Event generators check and filter data based on normal and abnormal information. Event detection agents detect the integrated data so as to decide whether the event is an intrusion behavior or not. The undetermined data is sequentially sent to a fusion center for further analysis. The Fusion center analyzes correlations and performs fusion analysis for the

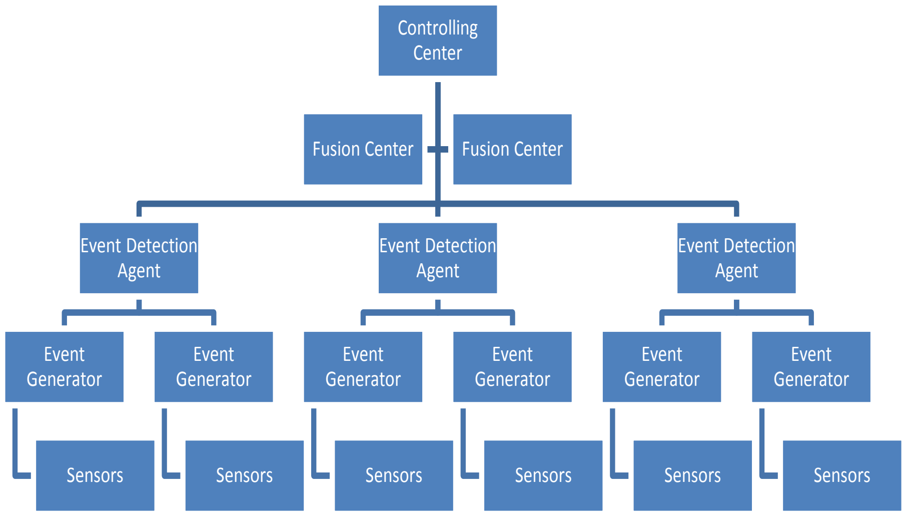


Figure 2.21: Distributed Architecture of Cooperative Intrusion Detection [1]

data submitted by event detection agents in order to increase the decision accuracy. Finally, the control center monitors, coordinates and adjusts each event detection agent and its corresponding load. The technique's statistic-based detection is based on predefined thresholds that allowed the detection of both scan attack and denial of service attacks. The authors claimed that their method not only can detect those scan attack with obvious features, but also it can detect the attack with stealth features and variants of the attack.

A positive point of this paper is that the proposed technique is well suited in a distributed large-scale environment. Moreover, the multi-layer architecture exploits the advantages of various approaches, including, statistical and scenario-based. However, few drawbacks could be extracted. First, the paper's experimental results were based on a simulated described scenario rather than real world data samples. Second, the authors did not test the accuracy of their technique against large-scale distributed scans.

In another work [2], Baig et al. proposed a time independent feature set model (IFSM) for the detection of slow, random and distributed cyber scanning activity. Their proposed technique is based on the observation that scanners, being unaware of systems and network topologies, send most of their probes to inactive hosts or closed ports resulting in many RST and ICMP packets. They designed a database that records information about that case and they took into consideration hosts that are behind a network address translation (NAT) [69] routers and those who use the dynamic host control protocol (DHCP) [70] server. The authors developed as well an algorithm that implemented that technique in addition to a pruning method used when system memory runs low. Finally, the authors empirically tested their technique using DARPA's data set [71]. The results demonstrated that their proposed IFSM performs well for detecting slow and fast scans. A snapshot of their technique detecting ports scans and IP sweeps is shown if Figure 2.22.

The work in [2] is a successful example of the usage of statistical feature-based elements in detecting cyber scanning. Nevertheless, this papers suffers from two core disadvantages. The technique presented in the paper is solely dependent on RST and ICMP packets. Thus, the technique can only detect scans that actually use or return those packets. Although the latter task can be a common behavior, a significant number of network

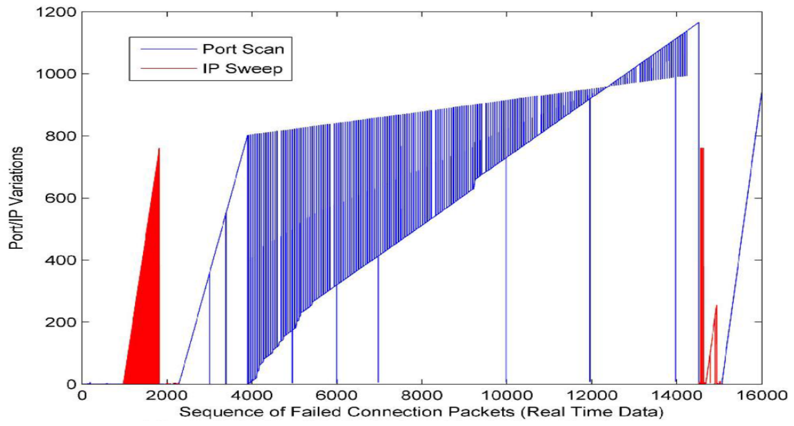


Figure 2.22: IFSM detecting Ports Scans and IP Sweeps [2]

devices do not allow the propagation of those packets back to the source. Second, although the authors claimed that their technique is effective in detecting slow and distributed cyber scanning, they did not demonstrate that empirically nor did they provide any scenarios to achieve that.

Furthermore, Staniford et al. [72] presented a method for the detection of stealth port scans. Their technique is divided into two layers; the Stealthy Probing and Intrusion Correlation Engine (SPICE) and the Statistical Packet Anomaly Detection Engine (SPADE). Using an entropy-based metric, SPADE determines if a packet is malicious and sequentially pass to SPICE. The latter engine inserts the packet into a correlation graph, where the nodes represent packets and the connections between nodes contain weights indicating the strength of the relationship between the packets. The weights are based on a combination of four feature characteristics, namely, equality, proximity, separation, and covariance. In the final graph, all edges with weights less than a certain threshold are dropped, and the remaining subgraphs represent interesting network events.

The work done by Staniford et al. [72] has the following weaknesses. Firstly, SPICE was not designed specifically to detect coordinated scanning activity, rather it just forms clusters based on similar properties using the correlation graph. Secondly, the authors do not report the true and false negative and positive rates for their approach. Thirdly, although the authors claimed that they detect distributed scanning, they have not provided sufficient details to replicate or proof their results.

Algorithmic Approaches

These distributed cyber scanning detection approaches employ step-by-step procedures for calculations, data processing, and formal automated reasoning.

Baldoni et al. [3] proposed a collaborative architecture where each target network deploys local sensors that send alarms to a collaborative layer. This, in turn, correlates this data with the aim of (1) identifying coordinated cyber scanning activity while (2) reducing false positive alarms and (3) correctly separating groups of attackers that act concurrently on overlapping targets. The proposed architecture is illustrated in Figure 2.23. Locally

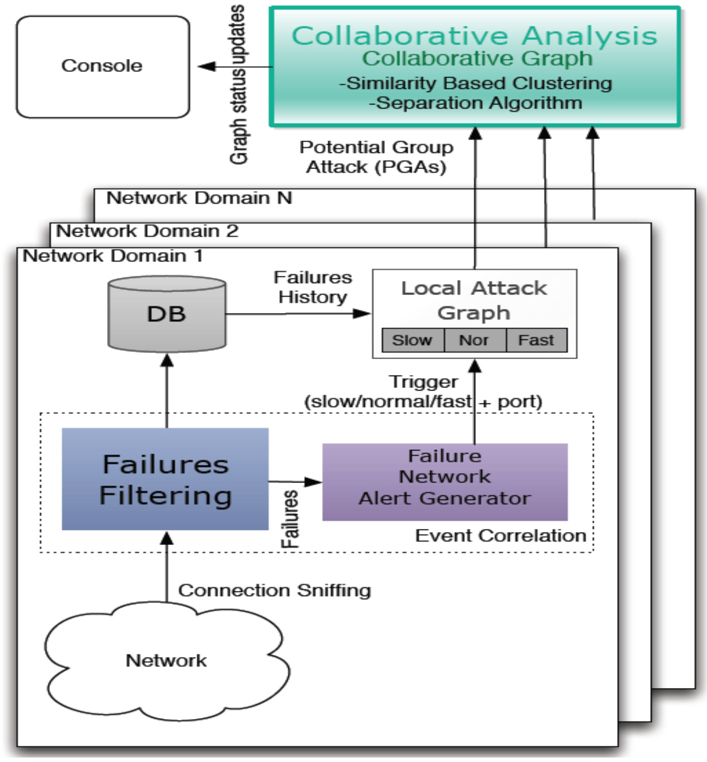


Figure 2.23: Proposed Collaborative Architecture [3]

deployed sensors adopt graph-based clustering algorithms over non-established TCP connections to generate alarms. The collaborative layer employed a similarity approach to aggregate alarms and approximated optimization algorithms to separate distinct group of attackers. The soundness of the proposed approach was tested on real network traces.

The above work however has the following limitations. First, their proposed system is designed to leverage information coming from various network domains to detect distributed scanning. Hence, the collaborative layer, in fact, is totally ineffective when the adversary is acting only against one network domain. Second, their system assumed that the target set of an attack contains contiguous IP addresses, which is not always true. Third, if the distributed scanning is being generated by a large number of nodes, where each node only sends one or few number packets, then the system would consider those as individual scans rather than correlating them.

In an another research work on distributed cyber scanning detection [73], the author presented an approach to detecting coordinated attacks that is based on adversary modeling of the desired information gain. In this research paper, a detection algorithm has as well been developed that is based on solutions to the set covering problem, where the aim was to recognize coordinated activity by combining events such that a large portion of the information space is covered with minimal overlap. The author demonstrated the approach by developing a coordinated scan detector, where the targets of a port scan are distributed amongst multiple coordinating sources. The author elaborated that in this case, the adversary wishes to gain information about the active hosts and ports on a particular network. Moreover, the paper provided an algorithm that is capable of detecting horizontal and strobe scans against contiguous address spaces. Finally, the paper presented experimental results of the proposed algorithm in a controlled environment, demonstrating that it has an acceptably low false positive rate.

A core limitation of the work in [73] is that the input for the proposed algorithm

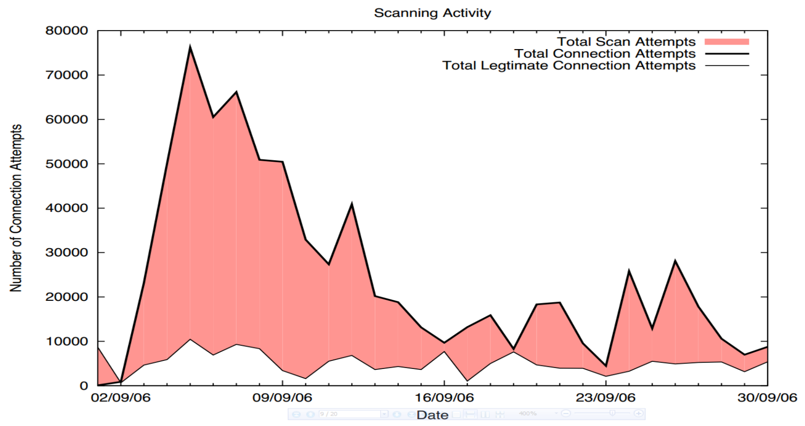


Figure 2.24: Darkports and Exposure Maps in detecting Scanning [4]

consists of single-source port scans. Thus, if an attacker can avoid detection by the single-source scan detector, then he as well would avoid detection by the developed coordinated scan detector.

In an alternative research article, Whyte et al. [74] discussed the notions of darkports and exposure maps. The former are unused ports on active systems while the latter is a technique rendered by passively characterizing the connectivity behavior of internal hosts in a network as they respond to both legitimate connection attempts and scanning attempts. Their proposed technique differs from other scanning detection techniques as they rely on identifying the services offered by the network instead of tracking external connection events. Moreover, the authors demonstrated how they could exploit darkports for defensive purposes as shown in Figure 2.24. Additionally, they presented some methods to detect advanced cyber scanning activity such as distributed scanning. Finally, they evaluated their approach using three different real data sets.

The above work has the following limitations. First, the proposed approach requires a prolonged training period (initializing time) to build the network map, decreasing its chances from being operationally feasible. Second, the actual network map populating process is based on observed TCP SYN ACK. Nowadays, there exist a significant number of stealth cyber scanning activity that never utilizes the TCP SYN ACK. Third, the authors' proposed heuristics to detect, attribute and match distributed cyber scanning is based on source IP grouping. They also considered clusters of three or more remote hosts that target the same destination ports as a distributed scan. This is ineffective if the sources are spoofed, change regularly due to DHCP usage, or target different ports.

Furthermore, Yegneswaran et al. [75] presented a broad, empirical analysis of Internet intrusion activity using a large set of NIDS and firewall logs. Their breakdown of scan types showed not only a large amount of worm activity but also a substantial amount of scanning activity. To gain insight into the global nature of intrusions, the authors used their data to project the activity across the global Internet. They also presented a high level information theoretic evaluation of the potential of using data shared between networks as a foundation for a distributed intrusion detection infrastructure. Their analysis indicated that small collections of logs from smaller networks may not be sufficient to identify either worst offenders or most popular port targets for attacks. Additionally, their research claimed to detect distributed scanning activity by defining a distributed scan as scans from multiple sources (five or more) aimed at a particular port of destinations in the same /24 subnet within a one hour window.

A main drawback, related to the authors' definition of coordination or distributed scans, could be withdrawn from the above work. The definition misses several possible coordinated/distributed scans, such as scans from fewer than five sources, or scans where each source scans in a different hour. Additionally, they did not consider the case where completely unrelated sources might scan the same port on the same /24 subnet within the same hour. Their technique will neither report nor detect that case as a distributed cyber scanning activity.

Mathematical Approaches

These distributed cyber scanning detection approaches utilize mathematical models, finite state machines and other algebraic and geometric techniques to achieve their detection task.

The author, Treurniet J., of [76] presented an approach that is based on the idea that anomalous scanning activity could be detected using a finite state machine model that reflects the progression of a TCP connection through a sequence of states via its control flags. By storing such anomalies and applying correlation mechanisms, the author claim that she could detect slow and distributed scans. A proof of concept prototype was implemented which used both DARPA's data [71] and operational data injected with crafted anomalies to test the system. The author reported zero false negative and very few false positives.

The system proposed in this work [76] is evidently advantageous by its space requirements which makes it operationally feasible. On the other hand, this research work is limited in the following: First, the experimental data was based on filtered data which can not accurately reflect the system's performance. Second, the system's implementation is based on MATLAB [77] which reduces its operational capabilities from an efficacy point of view. Third, the distributed correlation engine is based on simplistic criteria and operates erroneously when the scan is destined to overlapping targets or ports.

In another work by the same author [78], a new system was proposed that is capable of detecting slow scans and distributed scans. The work was built on previous work [76] by refining the TCP model and adding support for UDP and ICMP. The proposed method is composed of two stages. First, sessions are formed from packet data using simple state machine models of TCP, UDP, and ICMP traffic. Second, common activities are identified in terms of groups of sessions which are referred to as activity patterns. The author verified that the system correctly identifies crafted slow scans injected into real traffic and found that most scans are below current detection thresholds. By combining the detected scans with the session directionality, the author was able to give context to the scan alerts and identify the scans that require immediate attention.

The research work presented above possess the following advantages. First, it requires no training period and little knowledge of the local network configuration. Second, it successfully attempts to separate backscatter [79] from inverse mapping traffic. On the other hand, and although the author claimed the the system is able to detect distributed scans, the system inaccurately group targeted distributed scans with other similar untargeted scans.

Bhuyan et al. [80] presented the adaptive outlier based approach for coordinated scan detection (AOCD). Their proposed approach is based on two techniques. First, the principal component analysis based feature reduction technique was adopted to identify the relevant feature set. These feature sets are used during cluster formation. Second, a variant of the Fuzzy C-means clustering algorithm [81] was as well employed to cluster information. Their algorithm also adopts an outlier scoring mechanism for each feature traffic data object and sequentially report it as malicious or not. The authors tested their algorithm using different real-life datasets and against other available literature techniques.

The work in [80] has few limitations. Firstly, it requires a training period and hence 1) its accuracy can be significantly affected when dealing with other new data and 2) it requires some initialization time which is not always feasible in an operational environment. Secondly, the empirical results demonstrated in the work were tested with only four scan types and hence other scan types might either go undetected or have inaccurate clustering. Thirdly, their proposed approach assumed that the target of the scanning is a set of contiguous addresses, which is not always correct.

Heursitical Approaches

These distributed cyber scanning detection approaches utilize non-formal expert based analysis including, but not limited to, visualization techniques, filter-based heuristics, previous incident analysis, and multidisciplinary techniques.

Robertson et al. [5] introduced the System Detection surveillance techniques for enclave environments (ESD) and peering center environments (PSD). The system employed a cascading filter design, as depicted in Figure 2.25, which coordinated a series of specialized heuristics across connection records, individual probes, scans and coordinated scanning groups. Their proposed approach operates as follows. First, approximate

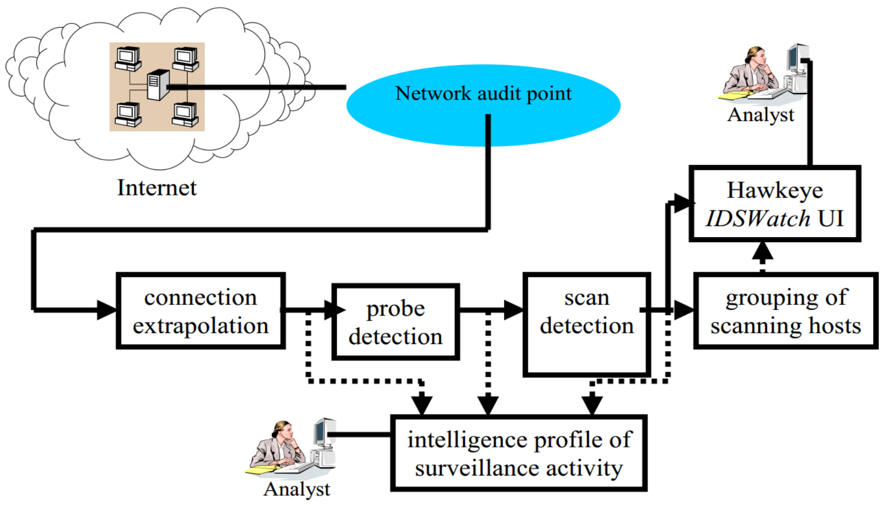


Figure 2.25: Architecture of Surveillance Detection [5]

sessions between source and destination IP pairs are extrapolated in accordance with a certain model. Second, each extrapolated session that represents a failed connection attempt is assumed to be a probe. Third, each probing IP is given a score based on the number of unique destination IP/port pairs probed. The IP is in turn considered a scanner if its score is greater than an empirically derived alert threshold. Their system was tested using real-time data and has shown to accurately discover great quantities of surveillance activities, including distributed scans.

The above system is advantageous in being scalable due to data reduction in the used filters and efficient in high bandwidth environments. However, on the other hand, their work assumed, with regards to distributed scanning activity, that a scanner is likely to use several IP addresses on the same subnet to carry on its probing act. This implies that if a particular IP address scans a network, IP addresses near this IP address, rather than those far away, are more likely to have also scanned the network. This assumption is not always valid, especially when dealing with botnet scanning. Another limitation is that their proposed algorithm could be susceptible to decoys intended to cause false positives.

In a different research work, Choi et al. [6] presented the parallel coordinate attack visualization (PCAV) as illustrated in Figure 2.26. PCAV displays network traffic

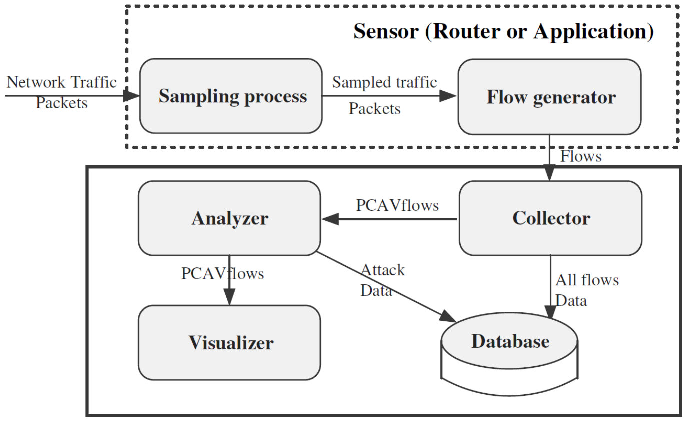


Figure 2.26: PCAV System's Design [6]

on the plane of parallel coordinates using the packet flow information such as the source IP address, destination IP address, destination port and the average packet length in a flow. The parameters are used to draw each flow as a connected line on the plane, where a group of polygonal lines form a particular shape in case of an attack. From the observation that each attack type possesses a unique pattern, the authors developed nine signatures coupled with their detection mechanisms based on an efficient hashing algorithm. The authors validated their proposed technique on three real network data samples and reported a very low false positive rate.

Although the authors asserted that their technique is able to detect and visualize distributed and coordinated scanning, they did not empirically validate that.

Stockinger et al., in their published research [82], presented a multidisciplinary high-performance query-driven visualization technique for the purpose of anomaly detection. They combined indexing mechanisms with a new approach to visual analytics to efficiently populate visual histograms. Additionally, the authors applied the histogramming technology in conjunction with a specialized visual analytics application for analyzing distributed scans. They tested their system using network connection data that was collected by Bro [83] at a governmental location.

The work presented by Stockinger et al. possesses few drawbacks. First, their system is passive in the sense that it might be effective in the analysis of distributed scans but not in real-time detection. Second, since the technique is based on visualization, it is hard to provide numerical analysis of the technique's false positive and negative rates. The authors did not provide any guidelines concerning that fact.

Last but not least, the authors of [84] discussed the application of visualization techniques to the problem of anomaly fingerprinting. This research work explored the application of several visualization techniques and their usefulness towards identification of attack tools and incidents. They used application, network and transport layer information to accomplish the visualization. The authors argued that their technique will aid other detection systems such as those using signature and statistical-based approaches to detect anomalies. Moreover, the authors briefly discussed the effectiveness of their technique in detecting distributed cyber scanning activity.

The work in [84] is interesting by allowing the identification of various scanning tools by visualizing their corresponding traffic. Indeed, this allows the detection of new attack tools and types without replying on signature based systems that would typically fail in such scenarios. Nevertheless, the work lacks the following. First, the authors did not provide any metrics on how their system would perform when operating on real-time data. Second, it is ambiguous how clusters of distributed scanners are formed using their technique.

Summary

This section presented a literature review by solely focusing on many to one cyber scanning detection techniques, commonly referred to as distributed approaches. From what has been previously discussed, we can extract the following few points:

- In general, limited work has been done targeting the problem of detecting distributed cyber scanning detection.
- Statistical methods are the least exploited to solve that problem.
- Algorithmic approaches, especially those utilizing clustering mechanisms, are the most effective techniques.
- There exist a lack of effective, accurate and efficient distributed source scanning clustering techniques.

2.2 Network Telescopes

In this section, we provide brief but relevant background information related to network telescopes in addition to clarifying how they can be exploited to generate various cyber threat intelligence.

A network telescope, also commonly referred to as a darknet or an Internet sink, is a set of routable and allocated yet unused Internet Protocol (IP) addresses [85]. It represents a partial view of the entire Internet address space. From a design perceptive, network telescopes are transparent and indistinguishable compared with the rest of the Internet space. From a deployment perspective, it is rendered by network sensors that are implemented and dispersed on numerous strategic points throughout the Internet. Such sensors are often distributed and are typically hosted by various global entities, including Internet Service Provides (ISPs), academic and research facilities and backbone networks. Figure 2.27 portrays a simple schema relating the dark space with the Internet space.

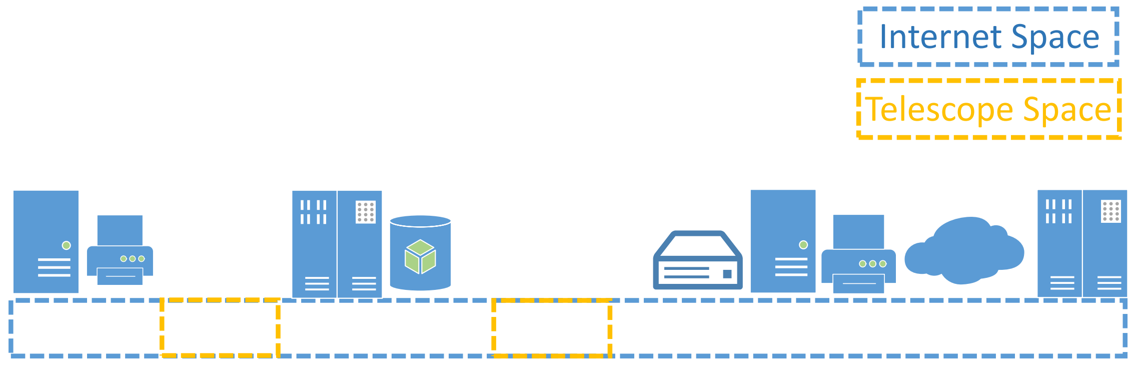


Figure 2.27: A Network Telescope as part of the Internet Space

The aim of network telescopes is to provide a lens on Internet-wide malicious traffic; since telescope IP addresses[1] are unused, any traffic targeting them represents a continuous view of anomalous unsolicited traffic. Such traffic, which is often dubbed as Internet Background Radiation (IBR) [86, 85], could be leveraged to generate various cyber threat intelligence, related, but not limited to, probing activities, denial of service attacks and reflective amplification attacks. In the sequel, we briefly clarify how the latter is achieved.

A network telescope is indeed an effective approach to infer various Internet-scale probing activities [87]. Figure 2.28 illustrates a simplistic example in which a probing machine is scanning the Internet space. Such machine could have been previously infected by a worm and is trying to propagate or perhaps is participating in an automated wide Internet-scale scanning task. Whatever is the reason behind its scanning activities, it can be seen that some of its network probe packets hit the network telescope and thus are subsequently captured. Recall, that the probing machine, while spraying its probes across the Internet space, can not avoid the network telescope as it does have any knowledge about its existence. Further, it has been shown in [88] that it is extremely rare if not impossible for a probing source to have any capability dedicated to such avoidance.

A network telescope is as well effective in pinpointing victims of denial of service (dos) attacks [89]. Figure 2.29 illustrates such scenario. The compromised machines are

[1]also frequently referred to as dark or darknet IP addresses

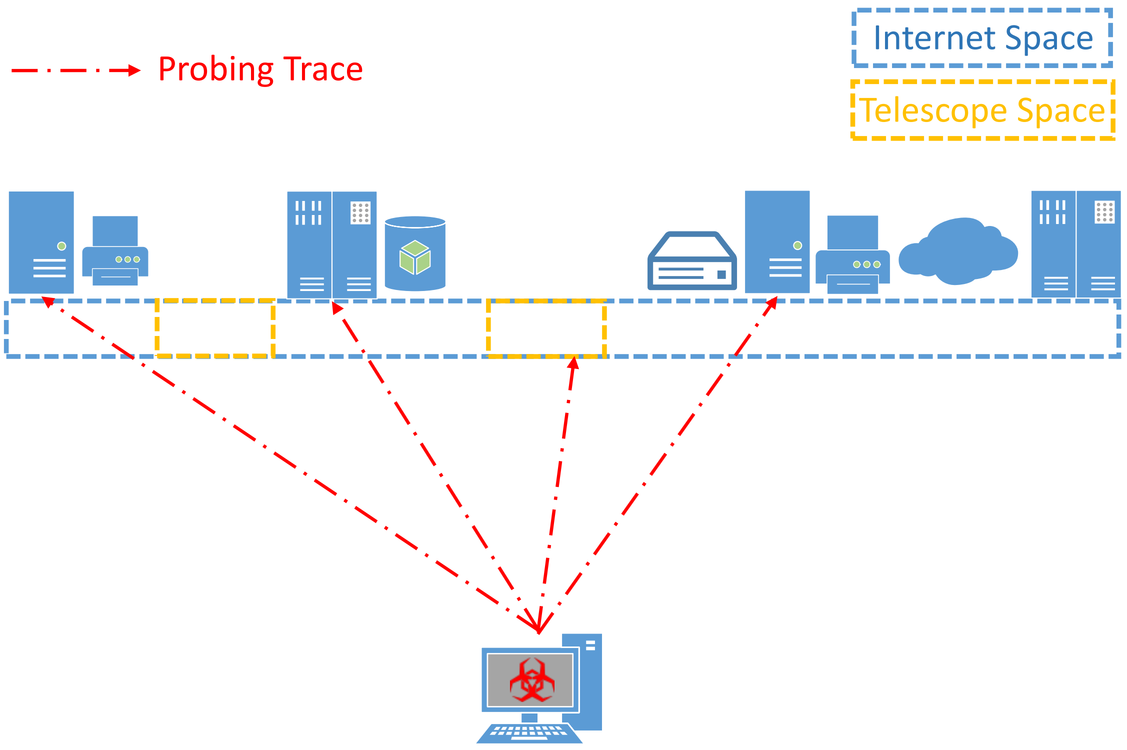


Figure 2.28: A Network Telescope capturing Probing Activities

directed to launch a distributed denial of service attack towards SRV1. To hide their identities, these machines spoof their addresses and replace them with random IP addresses. In this specific scenario, they replace their addresses with those of machines M1, M2 and other IP addresses. Such other addresses happen to be those of the network telescope. When the attack is launched, the reply packets will be directed towards M1 and M2 as well as to some dark IP addresses. Traces that hit the telescope are often dubbed as backscattered packets and could be effectively employed to infer that SRV1 has been the target of a dos attack.

In this last scenario, a network telescope is leveraged to infer reflective/amplified dos attacks [90]. Indeed, such attacks are an emerging form of distributed dos attacks that rely on the use of publicly accessible UDP servers[2], known as open resolvers, as well as

[2]Although TCP resolvers have been also shown to be vulnerable to such abuse [91]

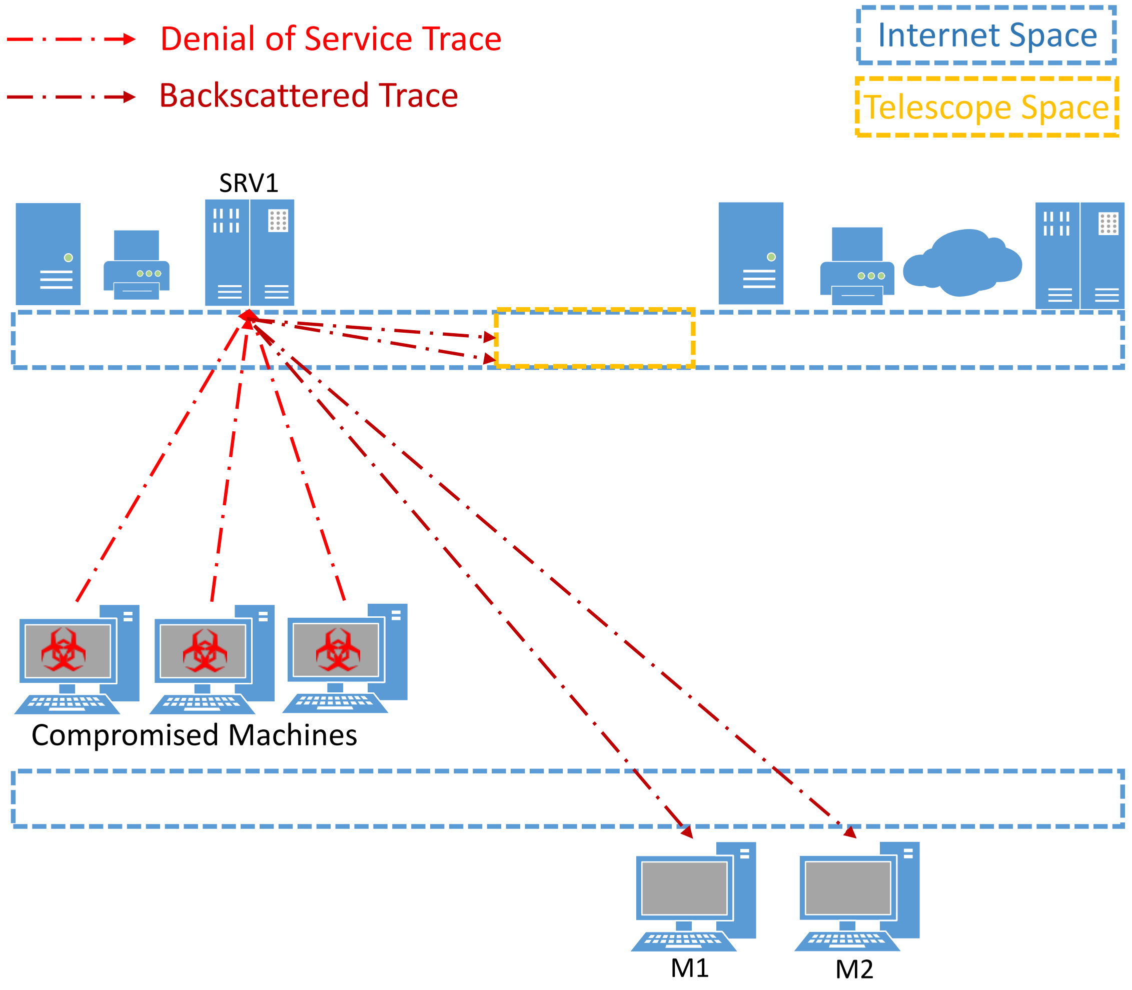


Figure 2.29: A Network Telescope pinpointing Victims of Denial of Service Attacks

bandwidth amplification factors to overwhelm a victim with a significant amount of traffic. The idea is to send simple queries to such resolvers in which the replies, that aim at flooding the victim, are orders of magnitude larger. Such approach is behind the notorious 300 and 400 Gbps attacks that hit the Internet in the last few years [92]. Figure 4.5 depicts a specific scenario where the resolvers are open Domain Name System (DNS) servers.

The compromised machines are directed to execute a reflective dos attack against Org1. To achieve that, they initially spoof their identities by using those of Org1 and subsequently send simple `ANY` queries to the DNS open resolvers. The latter DNS query

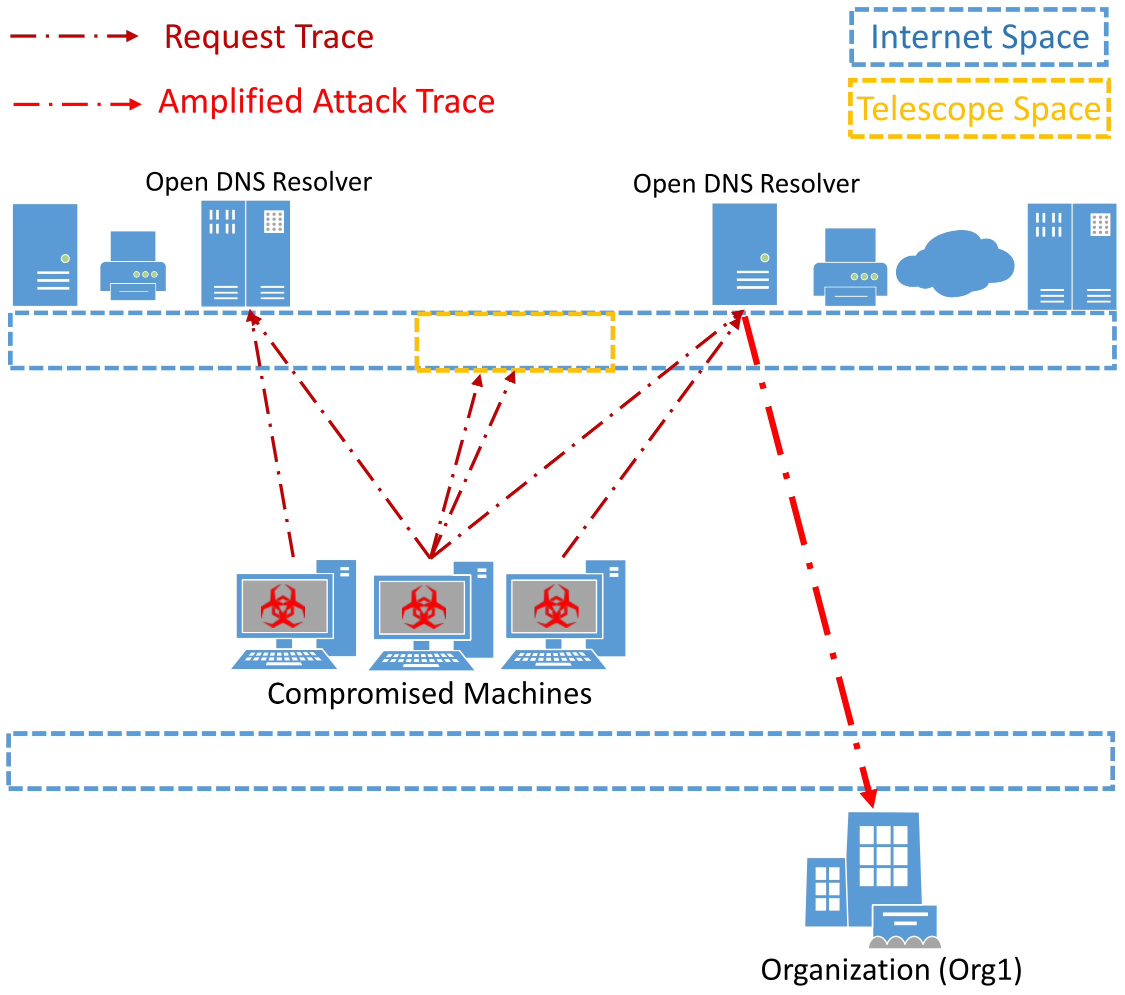


Figure 2.30: A Network Telescope pinpointing Sources of Reflective DoS Attacks

intends to pull all the available information from the DNS resolvers related to a requested domain. The domain in the request trace is often a noteworthy one that possess a significant amount of information. Commonly, the compromised machines will spray such queries on the Internet space in a hope to reach as many open resolvers as possible in order to increase the overall amplification factor. Intuitively, some of those requests will hit the network telescope and hence will be captured. Requests that actually reach open resolvers will be amplified and directed towards Org1.

It might be beneficial to mention at this point of the thesis that, for the past three years, we have been receiving, on a continuous real-time basis, raw darknet/telescope data from a trusted third party, namely, Farsight Security [93]. Such traffic originates from the Internet and is destined to numerous /13 network sensors. The data mostly consists of unsolicited TCP, UDP and ICMP traffic[3]. In this thesis, we will be leveraging various portions of such data to validate the numerous proposed models and approaches.

In the next chapter, we will be focusing on the problem of Inferring enterprise as well as Internet-scale probing activities.

[3]`https://archive.farsightsecurity.com/SIE_Channel_14/`

Chapter 3

Inferring Probing Activities

In this chapter, we elaborate on the design, implementation and validation of approaches for inferring enterprise and Internet-scale probing activities.

3.1 On the Inference of Enterprise Probing Activities

We present in this section an approach that tackles the issue of detecting corporate cyber scanning. The employed technique is based on a non-attribution anomaly detection approach that focuses on *what* is being scanned rather than *who* is performing the scanning. To empirically validate the proposed technique, we utilize and examine two real network traffic datasets and implement two experimental environments. The results show that for a class C network with 250 active hosts and 5 monitored servers, the training period of the proposed detection technique required a stabilization time of less than 1 second and a state memory of 80 bytes. Moreover, in comparison with Snort's sfPortscan technique, it was able to detect 4215 unique scans and yielded zero false negative.

3.1.1 The Non-Attribution Anomaly Detection Technique

In this section, we present the non-attribution anomaly detection technique and provide a discussion related to its training and detection periods.

Idea Rationale

The rationale behind the idea states that the available services that are provided by the hosts within an enterprise network represent the facade of that network; the offered services induce the possible leakage of information that could be retrieved by an attacker during a successful scan. Hence, the idea takes full advantage and solely of the network topology by constructing what we refer to as 'local host facade' (LHF) and 'enterprise network facade' (ENF). The former is the accessible services per host while the latter is the combination of all accessible services of all active hosts within the network.

ENF Management

In the training phase of our technique, we leverage the SNMP [94] to manage the ENF. SNMP is an Internet-standard protocol for managing devices on IP networks. It consists of components for network management, including an application layer protocol, a database schema, and a set of data objects. The protocol's information exchange is performed between a management station and an agent (embedded in the managed entity) in the form of SNMP messages. For an in-depth review of SNMP, including its inner workings, we refer the readers to [95].

The idea is to exploit specific de-facto SNMP procedures to manage the ENF. The latter task is divided into constructing the ENF by retrieving the list of listening ports on each host and maintaining (adding/deleting certain IPs/ports) the list in case of any change in accordance with a certain predefined update threshold. In the following, we briefly discuss the employed SNMP procedures and consequently elaborate on their roles in managing the ENF.

The `SNMP Receive-GetRequest` procedure [94] is issued by an SNMP management station in order to read or retrieve an object value from a managed entity. The managed SNMP entity responds to a `GetRequest` protocol data unit (PDU) with a `GetResponse` PDU. The `GetRequest` operation is atomic; either all the values are retrieved or none is.

If the responding entity is able to provide values for all the variables listed in the incoming `VariableBindings` list, then the `GetResponse` PDU includes the `VariableBindings` field coupled with a value supplied for each variable. If at least one of the variable values cannot be supplied, then no values are returned [94].

In this current work, the procedure, namely, `SNMP Receive-GetRequest`, is used to construct the ENF by leveraging the following two request methods:

$$GetRequest(ipRouteDest, tcpNoPorts)$$

$$GetRequest(ipRouteDest, udpNoPorts)$$

On the other hand, the task of maintaining the ENF could be divided into two subtasks. The first is when we need to update the list of active IPs/hosts and the second is when we need to modify the list of listening TCP and UDP ports for a specific host. To accomplish this, another SNMP procedure is presented, namely `SNMP Receive-SetRequest` [94] .

The procedure `SNMP Receive-SetRequest` is issued by an SNMP entity on behalf of a management station. It has the same PDU exchange pattern and the same format as the `GetRequest` PDU. However, the `SetRequest` is used to write an object value rather than reading or retrieving one.

In this work, we exploit `SNMP Receive-SetRequest` to update the ENF; based on a predefined update threshold, and whenever there is an update in the hosts (changing status from active to non-active or vice-versa) or their corresponding listening ports, we issue a `SetRequest` PDU to reflect the changes. For instance, if we notice that an active (i.e., connected) host with an IP address of 10.0.0.1 is no longer active (i.e., disconnected), the following SNMP request [94] is issued to remove that host from the ENF:

$$SetRequest(ipRouteDest.10.0.0.1 = invalid)$$

The above two procedures provide methods to construct and maintain what we have defined as the enterprise network facade. Recall that this characterizes the **training**

period of our proposed non-attribution detection technique. Since the management of the ENF is dependent solely on the enterprise network services and is totally decoupled from any external traffic, our approach is advantageous in two core areas. First, it requires almost negligible time to stabilize which renders its implementation very operationally feasible. Second, it relies on the observation and manipulation of a protocol (SNMP) found in every network, where its actual overhead on network bandwidth and hardware is minimal even in large network environments [96, 97].

Using ENF for Scan Detection

Once the training period has completed and an ENF is constructed, the anomaly detection phase commences. Scan detection is performed by monitoring external incoming TCP or UDP connection attempts. The attempts could be destined to the following targets: (1) an unallocated IP address, (2) an allocated IP address with a port combination not found in the ENF, (3) an allocated IP address with a port combination found in the ENF and (4) an allocated or an unallocated IP address outside of the monitored zone. In our approach, the detection occurs when we notice target 2 occurring, namely, an attempt to an allocated IP address with a port combination not found in the ENF. If the latter case occurs, we flag the connection attempt and log its corresponding details such as source and destination IP and port, protocol, and the timestamp. Target 1 is referred to as dark IPs [43] and their analysis is outside the scope of this work. Target 3 is as well excluded from the analysis. The exclusion of this target, at a first glance, seems to carry a limitation of our work in that scans to valid services (i.e., entries in the ENF) will not be detected. For instance, a DNS scan towards a naming (DNS) server is considered a valid activity and thus would not be considered a scan. However, this type of scan would indeed be detected using our approach as the same scan would almost certainly also occur against other hosts in the network not offering DNS. The scanning activity would not be detected if it were directed, although unlikely, solely at the naming server. However, we would consider the latter activity to be an actual attack (i.e., such as a denial of service attack) rather than a scan. Finally, target 4 depends on our monitored zone and intuitively we do not detect scans outside the monitored areas.

Discussion

In this part, we provide a discussion that is related to the technique's training and detection periods.

The training period is the period during which we first construct the ENF. Hence, as is the case with any technique that requires a training period, it is possible that malicious hosts activity may become part of the reference baseline. For example, if a trojan horse program [98] has been maliciously installed and has been running on one of the corporates's network servers, then the program would typically open up listening ports that are otherwise not supposed to be listening. To avoid this, we can match or verify the LHF with the enterprise network's security policy. Any inconsistencies are removed from the LHF to securely build the ENF.

Moreover, our technique's training period is efficient as the ENF only needs to record and maintain the state of the network services. To further improve this, we can manipulate SNMP to gather and record information only about specific hosts within the enterprise network. For example, we can build a *custom ENF* that includes only some of the network servers and to exclude other servers and workstations (we refer to those selected servers as belonging to within the boundaries of the monitored zone).

On the other hand, our proposed anomaly scanning detection approach does not rely on the identification of the scanning source. Therefore, it can detect certain classes of sophisticated scanning techniques (such as distributed and slow scanning) that make determining the root cause of the scanning activity impractical. Furthermore, the detection technique requires only a single packet to flag an attempt as a scan event and requires minimalistic system state storage especially if used with a *custom ENF*. Additionally, our approach is transport protocol-independent and hence can detect both TCP and UPD scans. Recall, that the outcome of this first technique is detected scans with minimal false positive rate.

3.1.2 Evaluation: Datasets, Methodologies and Results

For the purpose of empirically validating our approach, we utilized and examined two real network traffic datasets and implemented two experimental environments.

We used a dataset that consists of unsolicited one-way telescope/darknet traffic retrieved in real-time. The traffic originates from the Internet and is destined to numerous /24 and /16 network sensors. The data were collected during the period of November 1, 2012 and December 1, 2012. Tables 3.1 and 3.2, and Figure 3.1 show some network, transport and application level statistical information about the dataset.

TCP	UDP	ICMP	Others
86.3%	11.7%	1.8%	0.2%

Table 3.1: Protocols Distribution

	Usage (%)	
Class	Source	Destination
A	63.3	0.3
B	21.2	9.5
C	15.5	90.2

Table 3.2: IP Class Distribution

We selected part of the traffic that is destined to a /24 network collected at the sensor. We assumed that an operational/corporate network, having the same IP configuration as the incoming traffic, exists behind the sensors. Consequently, we built the network that is illustrated in Figure 3.2. The network has a Classless Inter-Domain Routing (CIDR) address of 192.168.1.0/24 and is composed of 250 active hosts divided into 245 workstations and 5 servers. We as well took advantage of the SNMP procedures of Section 3.1.1

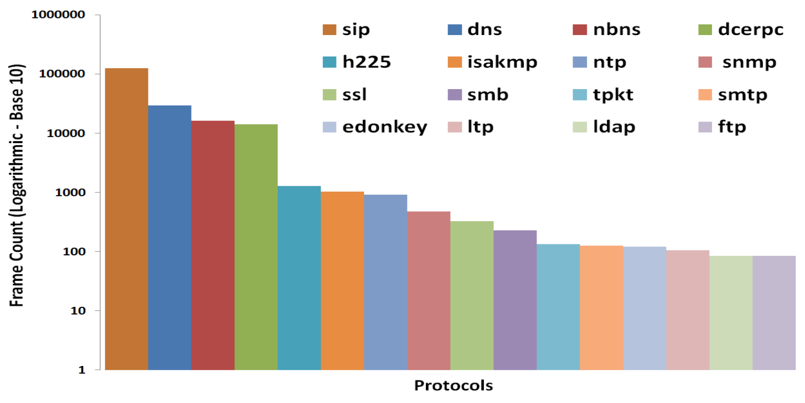


Figure 3.1: Application Layer Protocols

to develop an SNMP tool. The tool is based on the software components provided by eMarksoft SNMP [99].

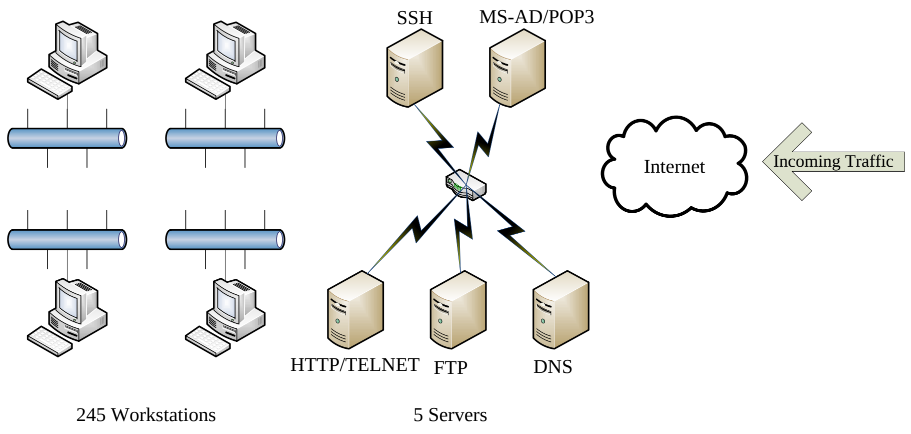


Figure 3.2: Enterprise Network

We first used the developed tool to execute the training period of our proposed approach. The ENF was populated with 5 LHFs (the other workstations are not offering any services) as illustrated in Table 3.3. The task was completed in 0.32 seconds and required 80 bytes of state memory.

To validate the detection capabilities of our approach, we experimented with a one day sample traffic captured from our dataset. We also compared our approach with Snort's

Host	TCP Ports	Description
Server 1	80, 23	HTTP/TELNET
Server 2	21	FTP
Server 3	53	DNS
Server 4	23	SSH
Server 5	445, 110	MS-Active Directory/POP3

Table 3.3: ENF Details

sfPortscan preprocessor using the same day sample. sfPortscan [100], a preprocessor plugin for the open source network intrusion and detection system Snort [101], provides the capability to detect TCP, UDP, and ICMP scanning. The sfPortscan preprocessor detects scans by counting RST packets from each perceived target during a predetermined timeout interval. Before declaring a scan, 5 events (i.e., RST packets) are required from a given target within a window. The sliding timeout window varies from 60 to 600 seconds by sensitivity level; at the highest level, an alert will be generated if the 5 events are observed within 600 seconds. We have chosen to compare our approach with Snort's sfPortscan preprocessor since Snort is one of the most broadly deployed intrusion detection/prevention technology worldwide and has become a de-facto standard.

According to the results, using our approach with this specific data sample, we were able to detect 4215 unique scans (unique IP/port pairs). Moreover, Figure 3.3 illustrates the top 6 scanned TCP ports. Scans towards those services could indicate that they are vulnerable to exploits.

To elaborate on the results, we subsequently present an analytical discussion on our technique's false negatives and false positives.

False Negatives: We fed the same dataset as an input to Snort's sfPortscan. We

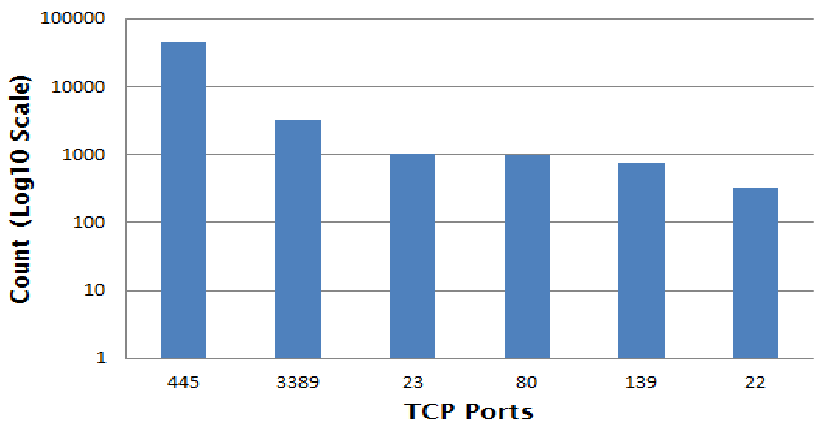


Figure 3.3: Top 6 Scanned TCP Ports - One Day Sample

relied on the output as a baseline for our comparison. Snort's sfPortscan technique detected 3690 unique scans. After a semi-automated analysis and comparison that was based on the logged scanning traffic flows (i.e., source and destination IP and port, protocol, and timestamp), we identified that all the 4215 scans that our approach detected include sfPortscan's 3690 scans. Therefore, relative to this technique and experimenting with this specific data set, we confirm that our approach yielded no false negative.

False Positives: Our approach flags an attempt as a scan whenever a connection is made to a host or service not offered by the network. The following can exist as sources of false positive: (1) User error and network misconfiguration; the intent was not to perform a scan but rather to access a legitimate service that have failed. Since there exists no scientific way to judge the connection intention, we have to classify those attempts as scans. (2) Backscattered traffic [102] destined to the corporate network; such traffic commonly refers to unsolicited traffic that is the result of responses to denial of service attacks with spoofed source IP addresses. To avoid this false positive, we can investigate such traffic using the proposed method in [86], which uses flags in packet headers, such as TCP SYN+ACK, RST, RST+ACK, and ACK, to accomplish the filtering. (3) Attempts to newly available services that were not part of the training period; to reduce the occurrences of this, we can optimize the update threshold of an ENF to include the new services.

Moreover, our approach can detect certain types of scans that were not included at the time of the experiment, and by default, in Snort's sfPortscan definitions. These include scans from a single host to a single port on a single host, slow scans and a specific host scanning multiple ports on multiple hosts. In general, we claim that a certain limited, acceptable and a manageable number of false positives will occur. Although future manual packet inspection needs to be performed to get the exact number of false positives, we need as well to consider Snort's sfPortscan false negatives and the different types of scans that our approach was able to detect.

To further evaluate the proposed non-attribution anomaly detection approach that was presented in Section 3.1.1, we consider another network scenario. We selected part of the traffic that is destined to a /24 network collected at the sensor. We assumed that a De-Militarized Zone (DMZ) perimeter network, having the same IP configuration as the incoming traffic, exists behind the sensors. A DMZ network typically refers to a logical sub-network that contains and exposes an organization's external-facing (i.e., public) services to a larger untrusted network, such as the Internet. Consequently, we built the network that is illustrated in Figure 3.4.

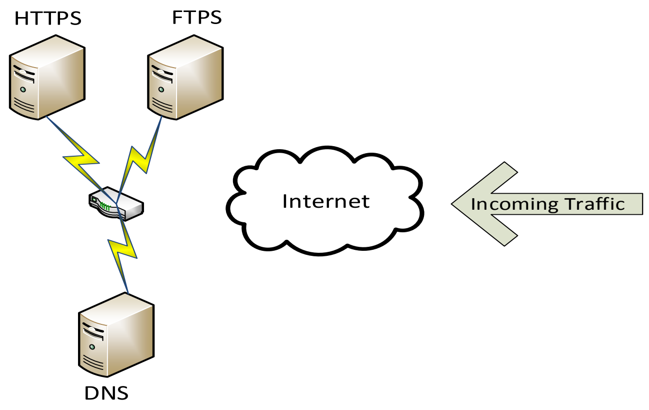


Figure 3.4: The DMZ Network

The DMZ network consists of three public servers. The corresponding ENF is summarized in Table 3.4. The procedure to build the ENF executed in 0.23 seconds and required 54 bytes of state memory.

Host	TCP Ports	Description
Server 1	443	HTTPS
Server 2	989, 990	FTPS
Server 3	53	DNS

Table 3.4: ENF Details

To validate the detection capability of our approach, we experimented with a one day sample traffic captured from our dataset. We also compared our approach with Snort's sfPortscan preprocessor using the same data sample. Table 3.5 summarizes the findings. After a semi-automated analysis and comparison that was based on the logged scanning traffic flows (i.e., source and destination IP and port, protocol, and timestamp), we identified that all the 3421 scans that our approach detected include sfPortscan's 3112 scans. Therefore, relative to this technique and experimenting with this specific data set, we confirm, once again, that our approach yielded no false negative.

Detected Scans-Proposed Approach	Detected Scans-Snort's sfPortscan
3421	3112

Table 3.5: Summary-Detection Capability

3.2 On Fingerprinting Internet-scale Probing Activities

Motivated by recent cyber attacks that were facilitated through probing, limited cyber security intelligence and the lack of accuracy that is provided by scanning detection systems, this section presents a new approach to fingerprint Internet-scale probing activities. It investigates whether the perceived traffic refers to probing activities and which exact scanning technique is being employed to perform the probing. Further, this work strives to examine probing traffic dimensions to infer the 'machinery' of the scan; whether the probing is random or follows a certain predefined pattern; which probing strategy is being employed; and whether the probing activity is generated from a software tool or from a worm/bot. The approach leverages a number of statistical techniques, probabilistic distribution methods and observations in an attempt to understand and analyze probing activities. To prevent evasion, the approach formulates this matter as a change point detection problem that yielded motivating results. Evaluations performed using 55 GB of real darknet traffic shows that the extracted inferences exhibit promising accuracy and can generate significant insights that could be used for mitigation purposes.

3.2.1 Proposed Approach

Internet security operators on the global scale are interested in generating insights and inferences concerning any probing activities that they might perceive. It is significant for them to have the capability to fingerprint probing events. Thus, they would benefit from knowing if the perceived traffic is related to scanning or not and if it is, exactly which scanning technique has been employed. This section aims at tackling these two issues.

The rationale of the proposed method states that regardless of the source, strategy and aim of the probing, the reconnaissance activity should have been generated using a certain literature-known scanning technique (i.e., TCP SYN, UDP, ACK, etc. [45]). We observe that a number of those probing techniques demonstrate a similar temporal correlation and similarity when generating their corresponding probing traffic. In other words, the observation states that we can cluster the scanning techniques based on their traffic correlation statuses. Subsequently, we can differentiate between probing and other malicious traffic (i.e., Denial of Service (DoS)) based on the possessed traffic correlation status. We can as well attribute the probing traffic to a certain cluster of scanning techniques (i.e., the probing activity, after confirmed as probing, can be identified as being generated by a certain cluster of techniques that possess similar traffic correlation status). To identify exactly which scanning technique has been employed in the probing, we statistically estimate the relative closeness of the probing traffic in comparison with the techniques found in that cluster.

To enable the capturing of traffic signals correlation statuses, the proposed method employs the Detrended Fluctuation Analysis (DFA) technique. DFA was first proposed in [103] and has since been used in many research areas to study signals correlation. Very limited work in the areas of cyber security and malicious traffic detection has utilized DFA [104, 105], and to the best of our knowledge, no work has leveraged the DFA technique to tackle the problem of fingerprinting probing traffic.

The DFA method of characterizing a non-stationary time series is based on the root mean square analysis of a random walk. DFA is advantageous in comparison with other methods such as spectral analysis [106] and Hurst analysis [107] since it permits the detection of long range correlations embedded in a seemingly non-stationary time series. It avoids as well the spurious detection of apparent long-range correlations that are an artifact of non-stationarity. Another advantage of DFA is that it produces results that are independent of the effect of the trend [108].

Given a traffic time series, the following steps need to be applied to implement DFA:

- Integrate the time series; The time series of length N is integrated by applying

$$y(k) = \sum_{i=1}^{k}[B(i) - B_{ave}] \tag{3.1}$$

  where $B(i)$ is the i^{th} interval and B_{ave} is the average interval.

- Divide the time series into "boxes" (i.e., bin size) of length n.

- In each box, perform a least-squares polynomial fit of order p. The y coordinate of the straight line segments is denoted by $y_n(k)$.

- In each box, detrend the integrated time series, $y(k)$, by subtracting the local trend, $y_n(k)$. The root-mean-square fluctuation of this integrated and detrended time series is calculated by

$$F(n) = \sqrt{\frac{1}{N}\sum_{k=1}^{N}[y(k) - y_n(k)]^2} \tag{3.2}$$

- Repeat this procedure for different box sizes (i.e., time scales) n

The output of the DFA procedure is a relationship $F(n)$, the average fluctuation as a function of box size, and the box size n. Typically, $F(n)$ will increase with box size n. A linear relationship on a log-log graph indicates the presence of scaling; statistical self-affinity expressed as $F(n) \sim n^{\alpha}$. Under such conditions, the fluctuations can be characterized by a scaling exponent α, which is the slope of the line relating $logF(n)$ to $log(n)$. The scaling exponent α can take the following values, disclosing the *correlation status* of the traffic time series.

- $\alpha < 0.5$: anti-correlated.
- $\alpha \approx 0.5$: uncorrelated or white noise.
- $\alpha > 0.5$: correlated.
- $\alpha \approx 1$: $1/f$-noise or pink noise.
- $\alpha > 1$: non-stationary, random walk like, unbounded
- $\alpha \approx 1.5$: Brownian noise.

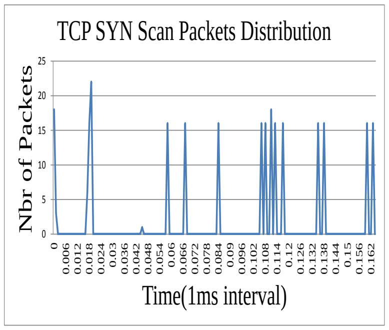
We proceed by fingerprinting the scanning techniques. For the scope of the current work, we have selected 10 cyber scanning techniques. To accomplish the task, we created an experimental environment that includes two virtual machines, a scanning and a target machine. Note that the virtual environment was hosted by VMware software while the machines were running Ubuntu Linux 10 with 2GB of RAM and 2.1GHz dual core CPUs. The machines are isolated from any external networks to prevent any noise in the generated signal. The target machine does not operate any special service. We have also setup

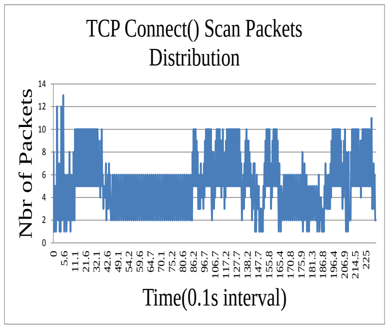
Cyber Scanning Technique	Nmap Command Flags
TCP SYN Scan	-sS
TCP connect() Scan	-sT
FIN Scan	-sF
Xmas Scan	-sX
Null Scan	-sN
UDP Scan	-sU
IP Protocol Scan	-sO
ACK Scan	-sA
Window Scan	-sW
RPC Scan	-sR

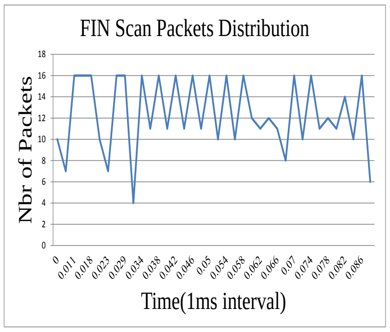
Table 3.6: Selected Cyber Scanning Techniques and Nmap Command Flags

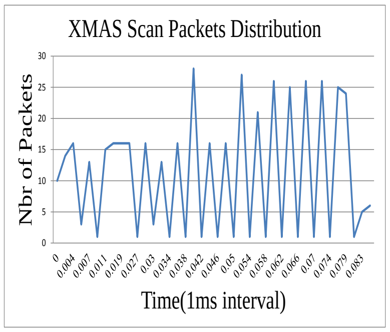
a TCPDUMP [109] sink on the target to collect the network traffic data originating from the scanning machine. It is worthy to note that we do not record the traffic response from the target as our intention is to capture the scanning techniques' traffic regardless of the offered services by the probed machine. To execute the scanning activity, we have utilized Nmap [110], an open source utility for network scanning and discovery. We ran Nmap 10 times, once for each scanning technique, and collected the generated traffic in 10 packet capture (pcap) [109] traffic files. The selected scanning techniques and the corresponding required Nmap command flags to execute each of the techniques are summarized in Table 3.6. For detailed information about the concerned scanning techniques, we refer the reader to [45, 110]. The generated packets' distribution of the 10 scanning techniques is illustrated in Figure 3.5. Subsequently, we applied the DFA technique on each of the scanning techniques' traffic signals. To achieve that, we have utilized the DFA MATLAB code found in [111] and used **1ms** as the bin size **for all** the 10 scanning techniques. The outcome of applying the DFA on the previous scanning traffic time series distributions is shown in Figure 3.6 and the output of the scaling exponents α is summarized in Table 3.7.

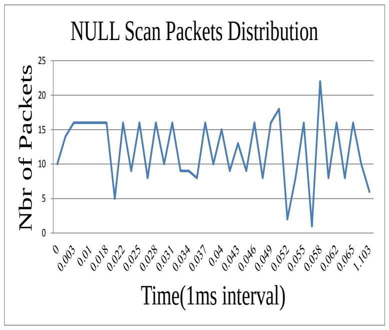
From Table 3.7 and the information relating the scaling exponent α to the correlation

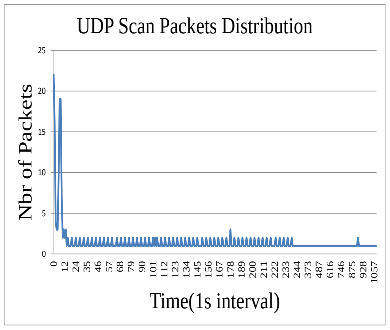

(a) TCP SYN Scan

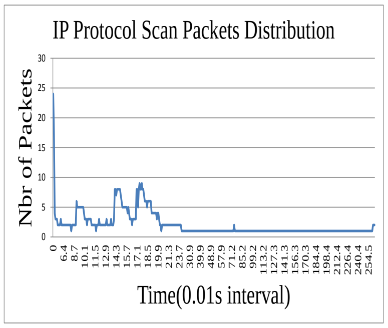

(b) TCP connect() Scan

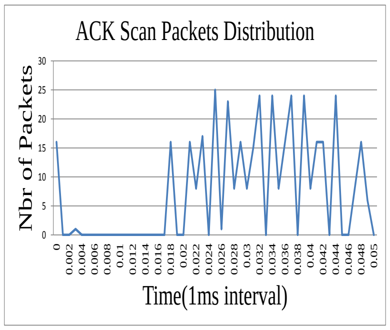

(c) FIN Scan

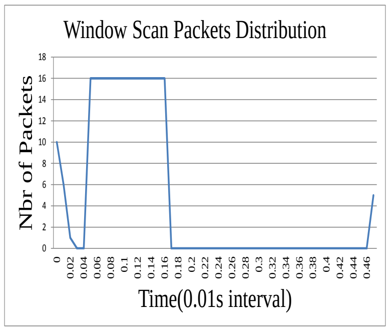

(d) Xmas Scan


(e) Null Scan


(f) UDP Scan


(g) IP Protocol Scan


(h) ACK Scan


(i) Window Scan

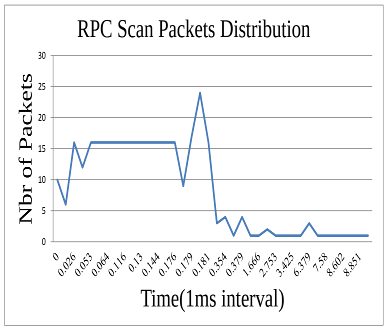
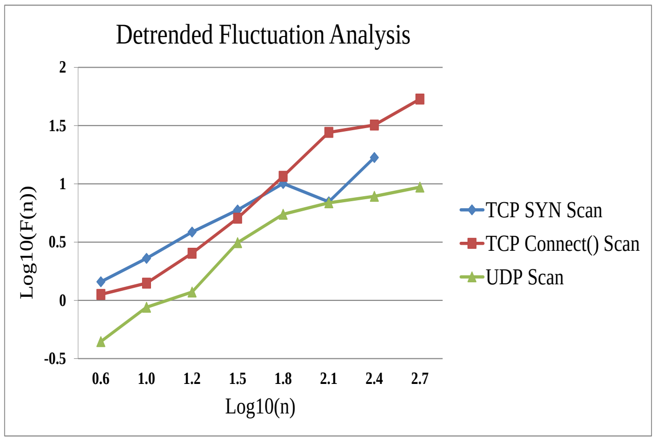
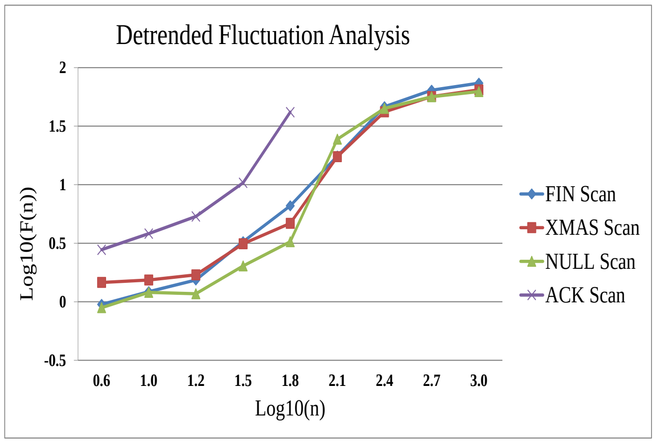

(j) RPC Scan

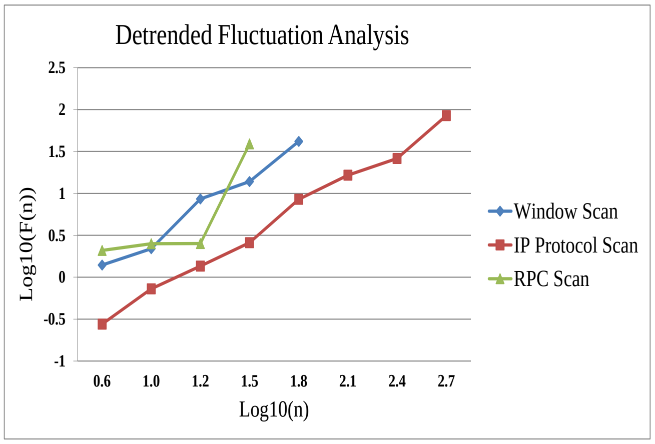
Figure 3.5: Packets' Distribution generated by the Scanning Techniques



(a)



(b)



(c)

Figure 3.6: Applying DFA on the Scanning Techniques Traffic Signals

Cyber Scanning Technique	Scaling Exponent
TCP SYN Scan	0.57
TCP connect() Scan	0.87
FIN Scan	0.31
Xmas Scan	0.30
Null Scan	0.37
UDP Scan	0.66
IP Protocol Scan	1.13
ACK Scan	0.44
Window Scan	1.24
RPC Scan	1.31

Table 3.7: Summary of the DFA Scaling Exponent α

status, we can produce Table 3.8 that discloses that a number of scanning techniques in fact demonstrated a similar temporal correlation and similarity when generating their corresponding probing traffic.

It is significant to pinpoint that such results are independent from the used scanning tool. In this work, we have used Nmap since it is the most widely adopted and well established scanning tool. Moreover, it provided a simple mechanism to generate the scanning traffic. We argue that same scanning techniques will generate a somehow similar traffic distribution regardless of the tool used and hence will output similar DFA results. To support this statement, we executed an experiment using Fing [112], another network scanning and discovery tool. We also selected the TCP SYN Scan since it is the most popular scanning technique [72]. We repeated the same experimentation as we did with the other scanning techniques. The output of the DFA scaling exponent α was $= 0.55$. Therefore, the correlation status was shown to be 'correlated' which is coherent with the DFA results that we have previously obtained with the TCP SYN Scan when we used Nmap. We can generalize such result to other techniques as well; since DFA operates on

Correlation Status	Cyber Scanning Techniques
Anti-Correlated	FIN Scan Xmas Scan Null Scan ACK Scan
Correlated	TCP SYN Scan TCP connect() Scan UDP Scan
Non-Stationary	IP Protocol Scan Window Scan RPC Scan

Table 3.8: Cyber Scanning Techniques and Corresponding Correlation Statuses

packets distribution in a time series, where similar scanning techniques, when following the protocol and technique standards, will possess similar distributions when probing their target, then we can expect similar DFA results regardless of the tool used.

It is also noteworthy to mention that attackers/scanners will not be able to avoid detection by modifying the correlation status of a probing signal because of two reasons. First, since they are going to employ one of the techniques of Table 3.7, which presents a comprehensive list of scanning techniques, regardless of whether they use a tool or anything else, they will indeed generate a probing signal that will be fingerprinted by the proposed approach. Second, the fact the we aim to leverage a darknet space to perform the inference in which the scanners have no knowledge about its existence, will cause the scanners to be detected. We argue that scanners will not go into the trouble of developing entirely new probing techniques that do not rely on the techniques of Table 3.6 and at the same time possess the capability to detect darknets, where both tasks are known to be hard [113], if not impossible, and impractical from a scanner perspective.

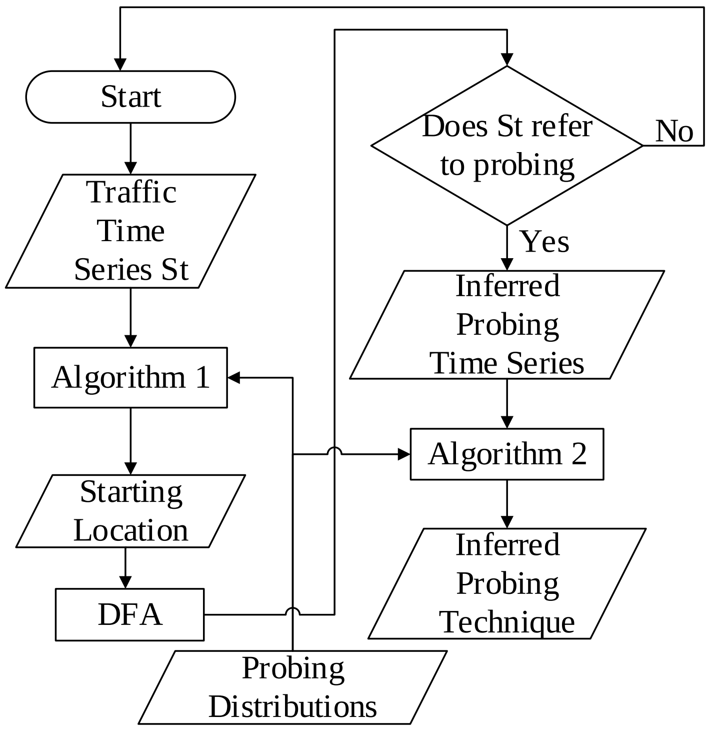


Figure 3.7: Employed System Process

We proceed by presenting Figure 3.7, which depicts the system process that is adopted by the proposed approach to permit the inference of the probing activities as well as the employed probing techniques. One of the issues that arises is where to apply DFA given a traffic time series that needs to be tested; this problem could be re-stated as follows: given a traffic time series S_t, find a starting position X and an ending position $X + \sigma$ in S_t where we can apply DFA on. Assuming a 'random' or a 'predefined' window location in the time series S_t to apply DFA on will be erroneous as this will result in wrong inferences. For example, if the traffic time series that needs to be tested is of length 5 minutes, applying DFA on the entire distribution could indicate a result (suppose it was inferred that it is not scanning) while the actual scanning starts on the 3^{rd} minute; the entire distribution's correlation status appears to be close to noise (i.e., α value ≈ 1 and hence not scanning) while from the 3^{rd} minute up to the 5^{th} the signal is correlated (i.e., scanning). To tackle this, we present Algorithm 1 which discloses an approach to approximate when to apply DFA, to correctly infer whether or not the traffic refers to probing activities.

Algorithm 1: Approximating the starting location on where to apply DFA in S_t

```
input  : A time series $S_t$ of the distribution under testing; the set of time
         series $Sc_p$ of the distributions of the scanning techniques.
output: $X$, reflecting the starting location on where to apply DFA in $S_t$

1  $m$=length($S_t$);
2  for every $Sc_p$ do
3  $n$=length($Sc_p$);
4  for $i$=1 $\rightarrow (m-n)$ do
5      s[i]=compare[$S_t(1+i, \cdots, n+i)$, $Sc_p(1, \cdots, n)$];
6  S[p]=min(s[]);
7  end
8  $X$=min(S[]);
9  return ($X$);
10
11 compare(A, B)
12 for $i$=1 $\rightarrow n$ do
13     $K[i]$= $\|\mathbf{E}\| = \mathrm{d}(A(i), B(i))$;
14     $sum$+=$K[i]$;
15     return ($sum$);
```

As shown in Figure 3.7, Algorithm 1 takes as input the time series S_t of the distribution under testing and all the other distributions of the scanning techniques Sc_p of Figure 3.5. For every distribution related to the scanning techniques, it calculates the euclidean distance **E** between the points in Sc_p and S_t. Subsequently, the scanning technique distribution is moved one point in a sliding window fashion against S_t. For each sliding window, it records the distance between Sc_p and S_t. After finishing all the sliding window procedures, the algorithm stores the minimum distance between both sliding windows in all the iterations. The algorithm finally selects X as the minimum of all the distances in all sliding windows after all the techniques Sc_p have passed on S_t. This will approximate the starting position on where to apply DFA in S_t. Note that, σ in the ending position $X + \sigma$, that was previously mentioned, is the length of the scanning technique Sc_p where X was derived from. It is significant to note that we use the scanning techniques Sc_p of Figure 3.5 as a way to infer, in an apriori fashion, where to start applying DFA and hence where we estimate that the scanning is initiated; we are not completely matching the techniques with the input time series S_t. In fact this is not the intention of Algorithm 1 and hence we do not expect the techniques to completely overlap the input time series S_t. Thus, any variation of scanning techniques' distributions is tolerated and manageable, as long their correlation status, as we expect and observe, are kept stable.

After applying DFA, given the output information of Algorithm 1, we end up with a certain correlation status. We expect that the correlation status indicates a probing activity (recall Table 3.8). However, we may encounter the case where the correlation status does not indicate probing (i.e., uncorrelated, $1/f$-noise or Brownian noise). If the activity refers to probing, then the output correlation status will lead us to a certain cluster of scanning techniques (of Table 3.8) that this probing activity has been generated from. To exactly identify which scanning technique has been used to probe, we present Algorithm 2, which discloses an approach to achieve that.

As depicted in Figure 3.7, Algorithm 2 takes as input a time series S_b of the probing distribution that DFA was previously applied on; S_b refers to the time series extracted from X to $X + \sigma$. For each of the scanning techniques Sc_{bi} in the cluster Sc_b that is related to the previous output correlation status, we statistically measure the relative closeness of S_b to each scanning technique in that cluster using Bhattacharyya distance [114], **Bha**. The latter statistic test is an established and an effective metric to determine the overlap of two sample distributions [114]. Intuitively, the algorithm selects the technique's distribution that is closer to S_b, Sc_{bi}. Sc_{bi} will be identified as the scanning technique that S_b was employing.

Algorithm 2: Identifying the scanning technique

```
input  : A time series $S_b$ of the probing distribution that DFA was
         previously applied on; a cluster of time series $Sc_b$ of the
         distributions of the scanning techniques related to the correlation
         status.
output: $Sc_{bi}$, reflecting one scanning technique that is estimated to be
         generating the probing activity found in $S_b$.
1 for every $Sc_{bi}$ do
2 $Bha_{bi} = \|\mathbf{Bha}\| = \mathrm{d}(\mathbf{Sc_{bi}}, \mathbf{S_b})$;
3 end
4 $d_i = Min(Bha_{bi})$;
5 return $(Sc_{bi}|i \text{ of } d_i)$;
```

3.2.2 Observation Validation

From what has been presented in Table 3.8, we deduced that the techniques could be clustered into 3 groups based on their probing traffic correlation statuses. Thus, the null hypothesis states that the scanning techniques' traffic originates from 3 independent clusters. However, by default, the DFA approach, especially in the context of probing activity, is not an established method to identify clusters. We now aim to validate the proposed method by comparing it with the well established machine learning clustering approaches, namely, the K-means and the Expectation Maximization (EM) techniques. These techniques and the validation results are consequently discussed.

The EM algorithm is an effective and popular technique for estimating the mixture model parameters [115]. Note that, the K-means algorithm operates by minimizing the sum of squared Euclidean distances between data records in a cluster and the clusters mean vector. This assignment criterion implicitly assumes that the clusters are represented by Gaussian distributions located at the k cluster means. Moreover, since the K-means algorithm utilizes the Euclidean metric, it does not generalize to the problem of clustering discrete or categorical data. Furthermore, the K-means algorithm employs a membership function which assigns each data record to exactly one cluster. This harsh criteria does not allow for uncertainty in the membership of a data record within a cluster. On the other hand, the mixture model framework adopted by the EM relaxes these assumptions. Due to the probabilistic nature of the mixture model, clusters can be effectively represented by a suitable component density functions (i.e., Poission, non-spherical Gaussians, etc.). Additionally, categorical or discrete data is similarly handled by associating discrete data distribution over these attributes (i.e., Multinational, Binomial, etc.). In this work, we adopt both the K-means and the EM algorithm to validate the DFA approach that was previously proposed. The EM algorithm is thoroughly discussed next.

The mixture model approximates the data distribution by fitting k component density functions f_h, $h = 1, \ldots, k$, to a database D having m records and d attributes. In our work, the attributes represent packet features extracted from the scanning machines, the records represent the data instances related to those features and the database D is the complete set of data instances. Let x be a record from D, then the mixture model probability density function evaluated at x is given by:

$$p(x) = \sum_{h=1}^{k} w_h \cdot f_h(x|\Phi_h) \tag{3.3}$$

The weights w_h represent the fraction of database records belonging to cluster h and sum to one;

$$\sum_{h=1}^{k} w_h = 1, w_h \geq 0$$

The functions $f_h(x|\Phi_h)$ $h = 1, \ldots, k$ are the clusters or component density functions modeling the records of the h^{th} cluster, and Φ_h represents the specific parameters used to compute the value of f_h (i.e., for a Gaussian component density function, Φ_h is the mean and the covariance matrix).

The mixture model also allows for overlapping clusters; data records may belong to all k clusters with different probabilities of membership. The probability of membership (i.e., weight) of data record x in cluster h is:

$$w_h(x) = \frac{w_h \cdot f_h(x|\Phi_h)}{\sum_i w_i \cdot f_i(x|\Phi_i)} \tag{3.4}$$

By making the assumption that the attributes of the database are independent over records within a given cluster, the component density functions can be decomposed as a product of density functions over each attribute $j = 1, \ldots, d$:

$$f_h(x|\Phi_h) = \prod_{j=1}^{d} f_{h,j}(x_j|\Phi_h) \tag{3.5}$$

Although the framework we present in this work is general enough to address mixture model estimation having both continuous and discrete attributes, we focus on its application to continuous-valued data, modeled by multivariate Gaussians. In our current work, this choice of Gaussians for continuous-valued data is motivated by a result from density estimation theory stating that any distribution can be effectively approximated by a mixture of Gaussians [116].

Each population (i.e., cluster) is modeled by a d-dimensional Gaussian probability distribution. The multivariate Gaussian distribution for cluster $h = 1, \ldots, k$ is parametrized by the d-dimensional mean vector μ_h and $d \times d$ covariance matrix Σ_h:

$$f_h(x|\mu_h, \Sigma_h) = \frac{1}{\sqrt{(2\Pi)^d |\Sigma_h|}} exp\left\{\wp\right\} \tag{3.6}$$

where

$$\wp = -\frac{1}{2}(x - \mu_h)^T (\Sigma_h)^{-1} (x - \mu_h),$$

x and μ_h are column vectors, the superscript ^{T} indicates the transpose to a row vector, $|\Sigma_h|$ is the determinant of Σ_h and $(\Sigma_h)^{-1}$ is its matrix inverse. The Gaussian mixture model parameters consist of the means and covariance matrices for each cluster $h = 1, \ldots, k$ along with the weights w_h associated with each cluster. Let $\Phi = \{(w_h, \mu_h, \Sigma_h), h = 1, \ldots, k\}$ be the collection of mixture model parameters. The quality of a given set of parameters Φ is a measure of how well the corresponding mixture model fits the data. This is quantified by the log-likelihood of the data given the mixture model:

$$L(\Phi) = \sum_{x \in D} log \left\{ \sum_{h=1}^{k} w_h \cdot f_h(x|\mu_h, \Sigma_h) \right\} \tag{3.7}$$

The EM algorithm begins with an initial estimation of Φ and iteratively updates it. The sequence of Φ-values is such that $L(\Phi)$ is non-decreasing at each iteration. We next outline the operation of the EM algorithm.

Given a database D with m records with d continuous-valued attributes, a stopping tolerance $\epsilon > 0$ and a mixture model parameters Φ^j at iteration j, compute Φ^{j+1} at iteration $j+1$ as follows:

1. For each database record $x \in D$, compute the membership probability of x in each cluster $h = 1, \ldots, k$:
$$w_h^j(x) = \frac{w_h^j \cdot f_h(x|\mu_h^j, \Sigma_h^j)}{\sum\limits_i w_i^j \cdot f_i(x|\mu_i^j, \Sigma_i^j)}$$

2. Update the mixture model parameters:

$$w_h^{j+1} = \sum_{x \in D} w_h^j(x),$$

$$\mu_h^{j+1} = \frac{\sum\limits_{x \in D} w_h^j(x) \cdot x}{\sum\limits_{x \in D} w_h^j(x)} \Sigma_h^{j+1}$$
$$= \frac{\sum\limits_{x \in D} w_h^j(x)(x - \mu_h^{j+1})(x - \mu_h^{j+1})^T}{\sum\limits_{x \in D} w_h^j(x)}$$

stopping criteria: if $|L(\Phi^j) - L(\Phi^{j+1})| \leq \epsilon$, stop. Else set $j \leftarrow j+1$ and go to 3.3. $L(\Phi)$ is given in

3.7.

Note that, steps 1 and 2 are respectively referred to as the estimation and maximization steps. Moreover, a full data scan is required at each iteration of the EM algorithm to compute the membership probability for each data record in each cluster. The number of iterations required before the stopping criteria is satisfied is dependent upon the initial parameter values and the actual data distribution. In general the number of iterations is arbitrary but the procedure is guaranteed to converge. Note that, all the mathematical notations that were presented in this section coupled with their corresponding definitions are summarized in Table 3.9.

Mathematical Notation	Definition
$p(x)$	Probability density function evaluated at a data record x
w_h	The probability of a data record belonging to a cluster h
$f_h(x\|\Phi_h)$	Component density functions
Φ_h	The parameters of the component density function
$\prod_{j=1}^{d} f_{h,j}(x_j\|\Phi_h)$	The product of density functions over the attributes
μ_h	Mean vector
Σ_h	Covariance matrix
$L(\Phi)$	The mixture model

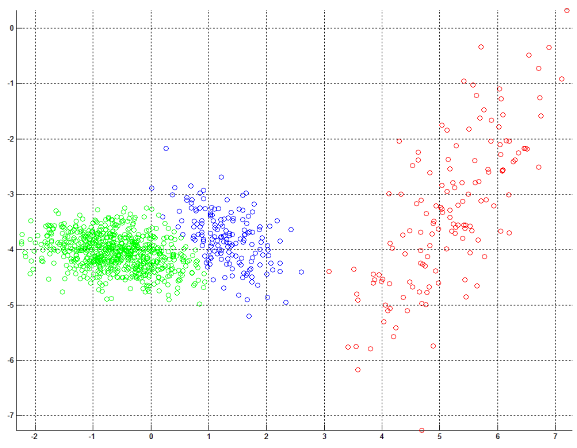
Table 3.9: Mathematical Notations and Definitions

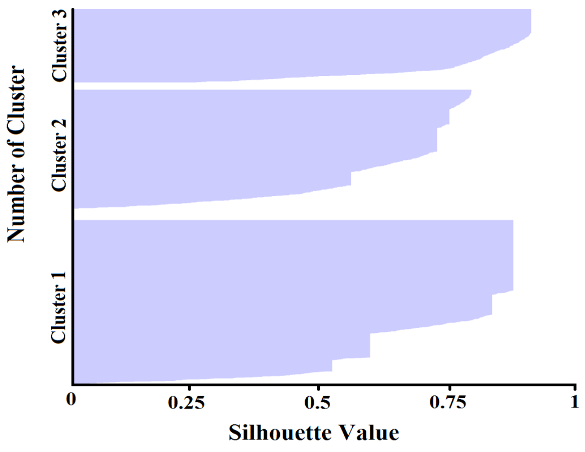
Recall that the aim is to validate the soundness of our proposed method by comparing it with the discussed machine learning clustering approaches. We proceed by going back to our scanning traffic pcap files that we have previously collected. Subsequently, we extracted from them a total of 29 data link, network and transport layer packet features as summarized in Table 3.10. This feature extraction procedure was achieved using the open source jNetPcap API [117]. We consequently compiled the extracted features into a unified data file of 7138 data instances. To apply the K-means and the EM algorithm on our data file, we have respectively used MATLAB's default clustering functionality and the WEKA data mining tool [118]. The output of those procedures is depicted in Figure 3.8.

Figure 3.8 clearly shows the formation of 3 clusters. This result provides evidence that the traffic originates from 3 different classes. To further test the validity of this result, we produced a silhouette graph of the K-means clusters as shown in Figure 5.23b.

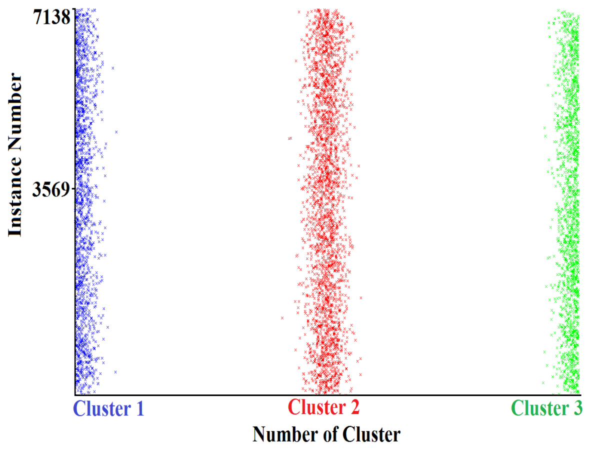
Features		
Data Link Features	1	Delta time with previous capture packet
	2	Packet Length
	3	Frame Length
	4	Capture Length
	5	The flag 'frame' is marked
Network Layer Features	6	IP Header length
	7	IP Flags.
	8	IP Flags: reversed bit
	9	IP Flags: do not fragment bit
	10	IP Flags: more fragments bit
	11	IP Fragment offset
	12	IP Time to live
	13	IP Protocol
Transport Layer Features	14	TCP Segment length
	15	TCP Sequence number
	16	TCP Next sequence number
	17	TCP Acknowledgement number
	18	TCP Header length
	19	TCP Flags
	20	TCP Flags: congestion window reduced
	21	TCP Flags: ECN-Echo
	22	TCP Flags: Urgent
	23	TCP Flags: Acknowledgement
	24	TCP Flags: Push
	25	TCP Flags: Reset
	26	TCP Flags: Syn
	27	TCP Flags: Fin
	28	TCP Window size
	29	UDP Length

Table 3.10: Features Description


(a) K-means Clusters



(b) K-means Silhouette



(c) EM Clusters

Figure 3.8: Method Validation through Unsupervised Learning

Typically, a silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. A value of 1 indicates that the points are very distant from neighboring clusters, a value of 0 informs that the points are not distant from other clusters while a negative value indicates that the points are erroneously placed in that cluster. From Figure 5.23b, it is shown that a significant amount of points in all the 3 classes have a large silhouette value, greater than 0.6 [119], indicating that the clusters are separated from neighboring clusters. This provides incentives to validate the quality of the formed K-means clusters. Further, the output of the EM clustering procedure on the same data file is shown in Figure 3.8c. Similar to the K-means results, we can notice the formation of 3 distinct classes. These results relatively validate the proposed method by revealing that the scanning traffic originates from 3 distinct classes; we accept the null hypothesis that stated that the scanning techniques' traffic originates from 3 independent clusters.

Is the probing random?

When probing activities occur, it would be interesting to possess insights related to how it is being generated; does the scanning occur in a random manner or does it follow a certain predefined pattern. Patterns existence provide cyber security intelligence about bots orchestration behavior. The latter is specifically important when trying to obtain inferences related to cyber scanning campaigns, such as the one reported in [120]. To answer this question, we proceed as follows. For each distinct pair of hosts retrieved from the probing traffic, we test for randomness in the generated traffic using the Wald-Wolfowitz (also known as the runs) statistic test [121]. If the result is positive, we record it for that specific session and apply the test on the remaining probing traffic. If the outcome is negative, we infer that the generated traffic follows a certain pattern. To capture the specific employed pattern, in this work, we model the probing traffic as a Poisson process[1] and retrieve the maximum likelihood estimate intervals (at a 95% confidence level) for the Poisson parameter λ that corresponds to that traffic. The choice to model the traffic as a

[1]The modeling approach is not very significant but rather the consistency of adopting one approach on all the probing sources.

Poisson distribution is motivated by [122], where the authors observed that probe arrivals is coherent with that distribution.

How are the targets being scanned?

As revealed in [120, 123], coordinated probing bots employ various strategies when probing their targets. These strategies could include IP-sequential [124], reverse IP-sequential [125], uniform permutation [126] or other types of permutations. In an attempt to capture the probing strategies, we execute the following. For each probing source in the probing traffic, we extract its corresponding distribution of target IPs. To differentiate between sequential and permutation probing, we apply the Mann-Kendall statistic test [127], a non-parametric hypothesis testing approach, to check for monotonicity in those distributions. The rational behind the monotonicity test is that sequential probing will indeed induce a monotonic signal in the distribution of target IPs while permutation probing will not. Further, in this work, we set the significance level to 0.5% since a higher value could introduce false positives. To differentiate between (forward) IP-sequential and reverse IP-sequential, for those distributions that tested positive for monotonicity, we also record the slope of the distribution; a positive slope defines a forward IP-sequential strategy while a negative one implies a reverse IP-sequential strategy. For those distributions that tested negative for monotonicity (i.e., not a sequential strategy), we leverage the chi-square goodness-of-fit statistic test. The latter insight will inform us whether or not the employed strategy is a uniform permutation; if the test fails, then the employed strategy will be deemed as a permutation; uniform permutation otherwise.

Who is generating the probing activity?

When an enterprise fingerprint probing (i.e., detect activity and identify the technique), it is of value as well to infer about the 'machinery' behind the source of the probing. Specifically, it would be significant to pinpoint whether the scanning is generated by a scanning tool or a worm/botnet. One use case for this, is for instance, when an Internet Service

Provider (ISP) notices that one of its customers is generating probing where that probing is generated by a worm/botnet. Keeping in mind that most probing activity is generated from non-spoofed Internet Protocol (IP) addresses (so the actual attacker/scanner can essentially get back the probing results), then the ISP can react on that and provide the suitable anti-malware solution to that specific customer.

From the two previous questions, we can infer those probing events that are random and monotonic. It is known that monotonic probing is a behavior of probing tools in which the latter sequentially scan their targets (IPs and ports) [128]. Furthermore, for random events, the monotonic trend checking can help filter out traffic caused by the non-bot scanners [129]. Thus, we deem a probing source as leveraging a probing tool if their traffic is randomly generated and if they adopt a sequential probing strategy (i.e., including reverse IP-sequential); a worm/bot otherwise. Although in the current work we do not directly differentiate between scanning generated by worms or bots, however, future work would couple such information with real malware data (that we possess as well) to provide more accurate inferences, including the type of malware or the botnet orchestration pattern.

Note that the 'machinery' of the scan aim to provide inferences and insights related to probing activities targeting a certain organization. Although the latter could be interesting to the organizations being scanned, we envision in future work that such proposed approach could be leveraged to automatically infer large-scale probing campaigns, such as the one analyzed in [120]. More specifically, we intend to employ that latter coupled with other inferences that we are currently developing to cluster the probing sources that possess similar probing behaviors.

3.2.3 Empirical Evaluation

We possess real darknet data that we are receiving on a daily basis from a trusted third party. Such traffic originates from the Internet and is destined to numerous /13 network sensors. The darknet sensors cover more than 12 countries and monitor around half a million dark IPs. The data mostly consists of unsolicited TCP, UDP and ICMP traffic. It might contain as well some DNS traffic. In a nutshell, darknet traffic is Internet traffic destined to routable but unused Internet addresses (i.e., dark sensors). Since these addresses are unallocated, any traffic targeting them is suspicious. Darknet analysis has shown to be an effective method to generate cyber threat intelligence [130, 131]. We use one week of data (55GB), extracted from multiple /24 networks, that was collected during the duration of February 24th to March 3rd 2013, to empirically evaluate our approach. Darknet traffic is typically composed of three types of traffic, namely, scanning, backscattered and misconfiguration [132]. Scanning arises from bots and worms while backscattered traffic commonly refers to unsolicited traffic that is the result of responses to denial of service attacks with spoofed source IP addresses. On the other hand, misconfiguration traffic is due to network/routing or hardware/software faults causing such traffic to be sent to the darknet sensors.

The first aim is to filter out misconfiguration data. We use a simple metric that records the average number of sources per destination darknet address. This metric should be significantly larger for misconfiguration than scanning traffic. However, although it differentiates misconfiguration from scanning traffic, it could include as well backscattered traffic as they also can possess a large average number of sources per destination (i.e, in case of a DoS). To cope with this issue, we observe, per the technique in [132], flags in packet headers, such as TCP SYN+ACK, RST, RST+ACK, ACK, etc., that resemble backscattered traffic [132]. Subsequently, we filter out flows that lack that observation, deeming them as misconfiguration.

Second, we aggregate the connections into sessions using an approach similar to the first step algorithm by Kannan et al. [133]. We consider all those connections within T_{aggreg} of each other as part of the same session for a given pair of hosts. We used the same threshold, $\mathrm{T}_{aggreg} = 100$ seconds, and found that this seems to correctly group the majority of connections between any given pair of hosts. For each day in our data set, we

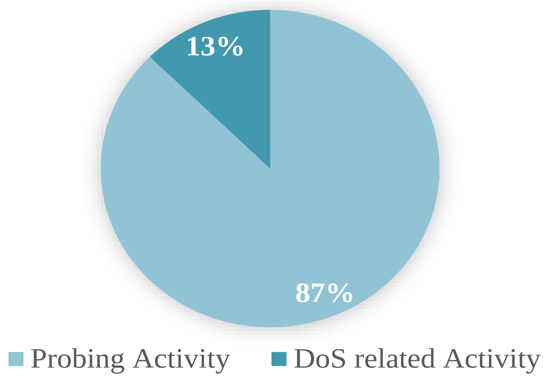


Figure 3.9: Sessions Distribution

extracted 100 sessions for a total of 700 sessions.

We setup an experimental environment using Java and implemented Algorithms 1 and 2. For all the statistical test techniques, including DFA, Bhattacharyya distance, Mann-Kendall and Wald-Wolfowitz, we employed their MATLAB implementations [111, 134, 135, 136].

We first applied the approach to attempt to differentiate between scanning and backscattered traffic (i.e., DoS related activity). Recall, that we have identified scanning as having the correlation statuses of Table 5.2. Figure 4.7 represents how the 700 sessions were distributed and fingerprinted. It is shown that probing activity corresponds to 87% (612) of all the sessions. This scanning to backscattered traffic ratio is somehow coherent with other darknet studies [132]. Note that in Figure 4.7, DoS related activity was fingerprinted as such since it was shown from its DFA results that 13% (88) of the sessions possessed correlation statuses corresponding to either uncorrelated, $1/f$-noise or Brownian noise. The fact that DoS related traffic demonstrated noise or Brownian noise is compliant with what was found in [104] when the authors performed DFA analysis on DoS traffic. To further validate such inference, we implemented the DoS detection algorithm by Moore et al. [137] and applied it on the 88 sessions. 77 sessions out of the 88 were detected as DoS related. Thus, with our approach, we have erroneously fingerprinted 11 sessions as

DoS related (assuming the mentioned DoS detection algorithm did not produce any false positive). To understand why that occurred, we inspected the 11 sessions. 7 sessions out of the 11 possessed a DFA scaling exponent α ranging from 1.51 to 1.59, and accordingly were fingerprinted as Brownian noise (i.e., DoS related). However, after inspecting their traffic packets, they were shown to be a rare type of RPC scanning traffic. This suggests that one should not consider large α values as valid results or at least keep those incidents in a 'quarantine' for further automated post-processing. The remaining 4 sessions that were also erroneously fingerprinted seem to be misconfiguration that apparently, were not previously filtered as expected.

To evaluate the scanning fingerprinting capabilities of our approach, we experimented with Snort's sfPortscan preprocessor using the same 612 sessions that were previously fingerprinted as probing. sfPortscan [100], a preprocessor plugin for the open source network intrusion and detection system Snort [101], provides the capability to detect TCP, UDP, and ICMP scanning. The sfPortscan preprocessor detects scans by counting RST packets from each perceived target during a predetermined timeout interval. Before declaring a scan, 5 events (i.e., RST packets) are required from a given target within a window. The sliding timeout window varies from 60 to 600 seconds by sensitivity level; at the highest level, an alert will be generated if the 5 events are observed within 600 seconds. We have chosen to compare our approach with Snort's sfPortscan preprocessor since Snort is one of the most broadly deployed intrusion detection/prevention technology worldwide and has become a de-facto standard.

We relied on sfPortscan's output as a baseline for our comparison. Snort's sfPortscan detected 590 scans. After a semi-automated analysis and comparison that was based on the logged scanning traffic flows (i.e., source and destination IP and port, protocol, and timestamp), we identified that all the 612 scans that our approach fingerprinted as probing activity include sfPortscan's 590 scans. Therefore, relative to this technique and experimenting with this specific data set, we confirm that our approach yielded no false negative. Moreover, according to the results, our proposed approach generated 22 sessions that are

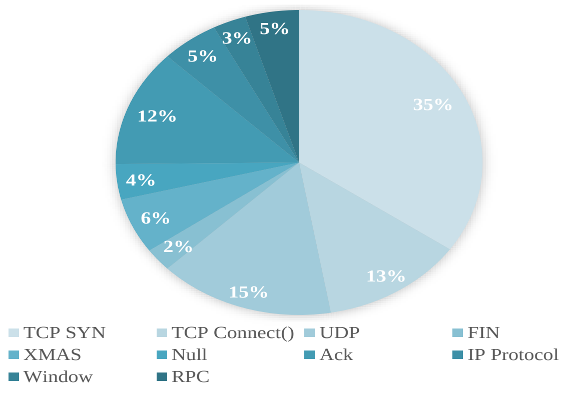


Figure 3.10: Probing Techniques Distribution

considered as false positive. It is worthy to pinpoint that our approach can detect certain types of scans that were not included at the time of the experiment, and by default, in Snort's sfPortscan definitions. These include scans from a single host to a single port on a single host, slow scans and a specific host scanning multiple ports on multiple hosts. In general, we claim that a certain limited, acceptable and a manageable number of false positives might occur (taking into consideration the system that we compare our approach with). We need as well to consider Snort's sfPortscan false negatives and the different types of probing that our approach is able to fingerprint.

We next applied the proposed approach to identify which techniques were leveraged in the previous fingerprinted probing activity. Figure 4.8 reveals that TCP SYN scanning leads with 35% (212) of all the sessions, followed by UDP, TCP connect() and ACK scanning. FIN, Xmas Tree, and Null scanning are typically considered as members of the 'stealth' scans because they send a single frame to a TCP port without any TCP handshaking or any additional packet transfers. They are relatively effective in evading firewall detection and they are often employed. The fact that the latter techniques were found to be among the least leveraged in the previous fingerprinted probing activity in our data set is quite puzzling.

We proceed by attempting to answer the questions that were raised in Sections 3.2.2,

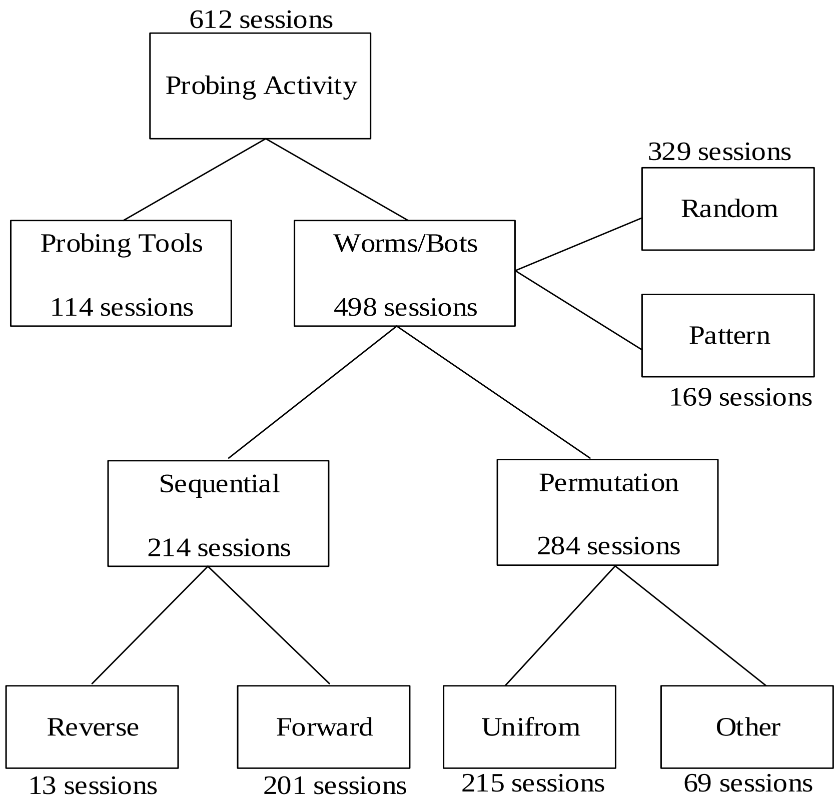


Figure 3.11: Probing Activity Dimensions Analysis

3.2.2 and 3.2.2. We applied the proposed approach which yielded the output of Figure 3.11. The results disclose that around 81% of the probing activity is being generated by worms or bots. Only 21% are being generated by probing tools. These percentages infer that leveraging real malware data, in a future study, could reveal substantial cyber security insights. The results also elaborate on the manner in which worms/bots generate their probing traffic. It is demonstrated that 66% of that probing traffic follows a random approach, while the remaining 34% follow a certain pattern when scanning their targets. Concerning the employed probing strategy, it is shown that 57% of the probing sources leveraged a permutation while the remaining adopted a sequential strategy when probing their targets. Of those that employed a permutation, 76% used a uniform permutation while 24% adopted other types of permutations. The majority ($\approx$ 95%) of those that employed a sequential strategy were found to adopt a forward IP-sequential strategy while only 5% adopted a reverse IP-sequential strategy. The latter insights allows us to 1) track the probing activity that possess similar scanning patterns and strategies and perhaps attribute it to the same campaign, 2) apply similar mitigation steps to probing activity with similar patterns and techniques and 3) provide inferences, although not decisive, that the probing is being generated by an orchestrated botnet. Note that we also generate supplementary material related to the above mentioned probing activity (i.e., worm, bots, probing patterns) including geo-location information per real source, organization, ISP, city, region and country. However, we refrain from publishing those due to sensitivity/legal issues.

3.2.4 Evasion Prevention

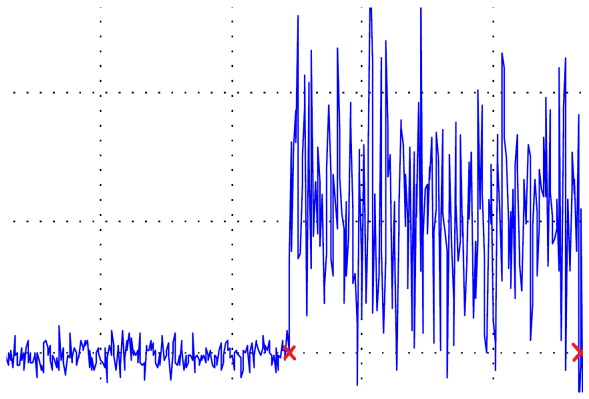
To fingerprint probing activity (i.e., detect activity and identify the technique), our approach, as stated in Section 3.2.1, leverages the DFA time series technique and operates on traffic distributions. However, it is realistic to pinpoint that the approach could be evaded in two ways. First, by a malicious attacker who deliberately injects a number of packets while performing his probing activity. Second, due to network fluctuations (i.e., delay), the distribution could be distorted. In both cases, our approach might erroneously miss the probing, attribute the probing to the wrong technique cluster or fail to identify the exact technique. We project that by formulating this problem as a time series change point detection problem, we can detect if and where the distribution has been susceptible to any sudden modifications. The time series change point detection problem has been excessively reviewed in the literature [138, 139]. For the sake of this work, as a proof of concept, we selected the work from Adams et al. [139] to experiment the effectiveness of such approach. We decided to leverage this specific work since it adopts a statistical approach which is coherent with the theme of our approach, is highly respected in its domain, and is directly applicable to our work that is related to time series distributions. Further, concerning its performance, in the worst-case, it is linear in space and time complexity according to the number of data points observed before the sudden fluctuation, which renders it practically viable. We employed the authors' MATLAB implementation, experimented with three different types of scanning traffic, namely, TCP SYN, UDP and ACK, and emulated a malicious attacker by injecting packets in the probing distributions using ‘packit’ [140], a packet analysis and injection tool.

Figure 3.12 shows how the TCP SYN, UDP and ACK scan distributions is being generated with a low frequency distribution. Suddenly, the malicious attacker injects random packets. The marked ‘X’s on the Figure demonstrate how the change point detection technique was successfully able to detect the change. In the context of our work, to prevent distribution modifications, we can simply remove the distributions between the first marked ‘X’ and the second marked ‘X’ to retrieve the original distributions, namely the probing distributions. Although we admit that further experimentation should be undertaken to thoroughly authenticate the effectiveness of such change point detection techniques in providing evasion prevention to our approach, the obtained preliminary results seem to be promising and thus motivating.

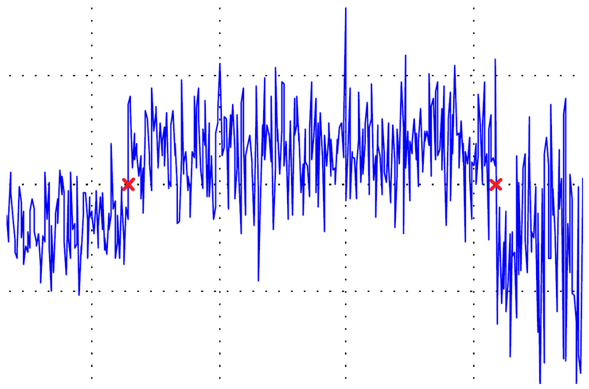
3.2.5 Approach Limitations

We acknowledge a number of limitations in our proposed approach. First, although in this work, the approach exhibited promising accuracy when evaluated using darknet (i.e, malicious) data, we have not tested the approach using normal two way traffic. Normal network traffic (i.e., benign http and ftp traffic for example) is known to be self-similar (i.e., possesses long traffic correlations). This directly affects the accuracy of our approach. We believe this point is manageable by applying pre-processing filtering mechanisms to filter out the benign traffic before applying our proposed approach. The payload analysis techniques in [141] seem viable to accomplish that. To verify this, we have executed the following experiment. We obtained a mixture of normal/malicious traffic data set from DARPA[2]. Subsequently, we executed the approach on that dataset after employing the payload filtering technique [141]. The approach’s accuracy scored around 83% in comparison with DARPA’s ground truth information. We believe that such percentage of accuracy is motivating, keeping in mind that our approach is primarily targeted towards darknet analysis. Second, in its current state, our approach does not fingerprint ICMP scanning. The latter constitutes a significant portion of today’s probing activity. We can overcome

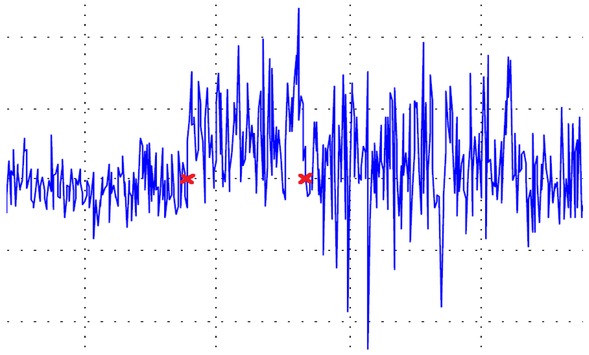
[2]http://tinyurl.com/lzbdd7h



(a) TCP SYN



(b) UDP



(c) ACK

Figure 3.12: Approach Evasion Prevention using a Change Point Detection Technique

this limitation by analyzing/fingerprinting the distributions related to such traffic and performing experimentation validation as we did to other scanning techniques. Third, the approach does not differentiate between worms and bots probing. If accomplished, it will yield significant cyber security intelligence for attribution and mitigation purposes. This task is currently work in progress. Finally, bots probing orchestration is not yet confirmed. Coupled with probing patterns that we have generated in this work, it could be used for tracking cyber scanning campaigns and for effective mitigation.

3.3 Related Work

In this section, we discuss cyber scanning related detection and clustering techniques and subsequently pinpoint the drawbacks of attribution-based approaches. Further, we pinpoint few works related to probing detection/analysis using statistical approaches.

Zhang et al. [1] proposed a scan detection method based on a distributed cooperative model. Their technique is composed of feature-based detection, scenario-based detection and statistic-based detection. Their proposed architecture is decomposed into 5 layers (sensors, event generators, event detection agents, a fusion center and a control center) that collaborate to achieve the intended task. The technique's statistic-based detection employs predefined thresholds that allows the detection of both scan and denial of service attacks. A positive aspect of this work is that the proposed technique is well suited to distributed large-scale environments. However, the presented work was based on an illustrated described scenario and the authors did not discuss its applicability on real data samples. In [80], Bhuyan et al. presented the adaptive outlier based approach for coordinated scan detection (AOCD). First, the authors used the principal component analysis feature reduction technique to identify the relevant feature set. Second, they employed a variant of the fuzzy c-means clustering algorithm to cluster information. The authors tested their algorithm using different real-life datasets and compared the results against other available literature techniques. Their approach assumes that the target of the scanning is a set of contiguous addresses, which is not always the case. In another work, Baldoni et al. [3] proposed a collaborative architecture where each target network deploys local sensors that send alarms to a collaborative layer. This, in turn, correlates this data with the aim of (1) identifying coordinated cyber scanning activity while (2) reducing false positive alarms and (3) correctly separating groups of attackers that act concurrently on overlapping targets. The soundness of the proposed approach was tested on real network traces. Their proposed system is designed to leverage information coming from various network domains to detect distributed scanning. Hence, the collaborative layer appears to be ineffective when the adversary is acting only against one network domain. In a more general work, Dainotti et al. [120] presented the measurement and analysis of a 12-day world-wide cyber scanning campaign targeting VoIP (SIP) servers. The authors used darknet/telescope data collected at the UCSD network telescope to exclusively focus on the analysis and reporting of that SIP scanning incident.

Most of the aforementioned detection and clustering techniques and other literature work [84, 78, 72] could be noted as being attribution-based; they detect and cluster distributed scanning based on the last perceived scanning source. Hence, they might encounter one of the following issues:

- Determining attribution is not always possible, which might decrease the effectiveness of such techniques.
- The scans may either be so slow or so broadly distributed that they exhaust the finite computational state of scanning detection systems or fail to exceed some predefined alert threshold.
- A significant amount of system state (i.e., memory, network topology information, storage) needs to be maintained by the monitoring system in order to perform effectively (reducing the detection time window to accommodate network traffic fluctuations might cause excessive false negatives and false positives).

Further, our work that is related to Internet-scale probing inference is different from the above related work as it does not rely on identifying the scanning source and is independent from the scanning strategy. Moreover, the proposed approach does not rely on a certain predefined alert threshold, the transport protocol used or the number of probed destinations and ports. Additionally, we attempted to go further than detection by analyzing probing traffic dimensions, namely, employed technique, monotonicity and randomness.

3.4 Summary

This chapter proposed a non-attribution anomaly detection technique. Motivated by the shortcomings of attribution-based approaches to cyber scan detection, this technique presented an alternative view of the problem/solution. The idea is to focus on what is being offered by the network and hence on what is being scanned rather than who is performing the scanning. To characterize this, we introduced and elaborated on the notion of enterprise net- work facade. To construct and maintain the ENF, we leveraged the SNMP by presenting certain management procedures. The approach's training period is decoupled from any external traffic which makes its implementation very operationally feasible, in addition to having fast stabilization time yet requiring minimalistic system state storage. The technique's detection period is attribution-independent, which allows the detection of sophisticated reconnaissance activity, requires only a single packet to detect a scan and allows the detection of both TCP and UDP scans. To evaluate our technique, we experimented using a real network traffic dataset and implemented a proof-of-concept environment. The results demonstrated that for a class C network with 250 active hosts and 5 monitored servers, the proposed technique's training period required a stabilization time of less than 1 second and a state memory of 80 bytes. Moreover, in comparison with Snort's sfPortscan technique, it was able to detect 4215 unique scans and yielded zero false negative.

This chapter also presented a new method to fingerprint Internet-scale probing activity. It aims at detecting the cyber scanning activity and identifying the exact technique that was employed in the activity. Further, it analyzes certain probing traffic dimensions such as monotonicity and randomness to generate inferences related to the ‘machinery’ of the scan (i.e, probing tools Vs worms/bots), the approach of the scanning (i.e., randomness Vs. probing patterns) and the employed probing strategy (i.e., sequential Vs. permutation). The approach leverages and employs several statistical techniques to achieve the required tasks. Empirical evaluations performed using real darknet traffic showed that the extracted inferences exhibit promising accuracy.

In the next chapter, we attempt to tackle the problem of attributing the inferred probing activities to certain malware for attribution purposes.

Chapter 4

Probing and Event Attribution

In this chapter, we describe the design and implementation of a correlation mechanism between probing and malware activities for attribution purposes. To this end, we also report on an Internet-scale malicious probing event by executing a multifaceted approach that correlates three types of real cyber security data.

4.1 Inferring Internet-scale Infections by Correlating Malware and Probing Activities

This section presents a new approach to infer worldwide malware-infected machines by solely analyzing their generated probing activities. In contrary to other adopted methods, the proposed approach does not rely on symptoms of infection to detect compromised machines. This allows the inference of malware infection at very early stages of contamination. The approach aims at detecting whether the machines are infected or not as well as pinpointing the exact malware type/family. The latter insights allow network security operators of diverse organizations, Internet service providers and backbone networks to promptly detect their clients' compromised machines in addition to effectively providing them with tailored anti-malware/patch solutions. To achieve the intended goals, the proposed approach exploits the darknet Internet space and initially filters out misconfiguration traffic targeting such space using a probabilistic model. Subsequently, the approach employs statistical methods to infer large-scale probing activities as perceived by the dark space. Consequently, such activities are correlated with malware samples by leveraging fuzzy hashing and entropy based techniques. The proposed approach is empirically evaluated using a recent 60 GB of real darknet traffic and 65 thousand real malware samples. The results concur that the rationale of exploiting probing activities for worldwide early malware infection detection is indeed very promising. Further, the results, which were validated using publically available data resources, demonstrate that the extracted inferences exhibit noteworthy accuracy and can generate significant cyber security insights that could be used for effective mitigation.

4.1.1 Motivation and Contributions

Today, the safety and security of our society is significantly dependent on having a secure infrastructure. This infrastructure is largely controlled and operated using cyberspace. Although tremendous efforts have been carried out to protect the cyberspace from diverse debilitating, intimidating and disrupting cyber threats, such space continues to host highly sophisticated malicious entities. The latter could be ominously leveraged to cause drastic Internet-wide and enterprise impacts by means of large-scale probing campaigns [120], distributed denial of service attacks [142], advanced persistent threats [143] and spamming botnets [144]. According to Panda Security, a staggering 33% of worldwide Internet machines are infected by malware [145]. Moreover, McAfee records over 100 thousand new malware samples every day; a momentous 69 threats every minute or around one new threat every second [146].

Network security operators of private and governmental organizations, Internet Service Provides (ISPs) and content delivery networks as well as backbone networks face, on a daily basis, the crucial challenge of dealing with their clients' malware-infected machines. The latter not only hinders the clients' overall experience and productivity but also jeopardizes the entire cyber security of the provider (i.e., causing vulnerabilities or opening backdoors in the internal network). Further, it significantly degrades the provided quality of service since the compromised machines will most often cause excessive increase in bandwidth that could be rendered by extreme Peer to Peer (P2P) usage, spamming, command-and-control communications and malicious Internet downloads. Additionally, if providers' networks were used to trigger, for instance, a malware-orchestrated spamming campaign, then such providers could as well encounter serious legal issues for misusing their infrastructure (i.e., for example, under the Canadian House Government Bill C-28 Act [147]). Consequently, this will immensely adversely affect the operators' business, reliability and reputation.

Thus, network security operators are interested in possessing a cyber security capability that generates inferences and insights related to their clients' malware-infected machines. It is significant for them to be able to pinpoint such machines in addition to extract intelligence related to the exact malware type/family. The latter will facilitate the distribution of suitable and tailored anti-malware solutions to those compromised clients.

Indeed, this cyber security capability should possess the following requirements. First, it should be prompt; it must possess the ability to detect the infection as early as possible in an attempt to thwart the creation of botnets and to limit the sustained possible collateral damage and any symptoms of infection. Second, it should be cost-effective; the approach should not overburden the provider with implementation scenarios and their corresponding supplementary costs. In fact, this last point is extremely imperative and decisive; ISPs are frequently accused of ignoring their clients' malware infections because the task to detect and disinfect them is tedious, prolonged and undoubtedly expensive [148, 149]. This section elaborates on such a cyber security capability that satisfies the mentioned requirements. Specifically, we frame the contributions as follows:

- Proposing a new probabilistic model to preprocess telescope/darknet data to prepare it for effective use. The aim is to fingerprint darknet misconfiguration traffic and subsequently filter it out. The model is advantageous as it does not rely on arbitrary cut-off thresholds, provide separate likelihood models to distinguish between misconfiguration and other malicious darknet traffic and is independent from the nature of the source of the traffic.

- Proposing a new approach to infer Internet-scale malware-compromised machines. The approach aims at detecting such machines as well as identifying their exact infection type. The approach achieves its aims without recording or analyzing the symptoms of infection (i.e., spamming, excessive bandwidth usage, etc.), which renders it efficient from both space and processing perspectives. Further, it exploits probing activities to attain early detection of contamination incidents in addition to requiring no implementation at the providers' premises, eliminating the cost burden.

- Leveraging the darknet Internet space, around half a million routable but unallocated IP addresses, which permits the observation and identification of worldwide probing activities and thus malware-infected machines, without requiring any providers' aid or information.

- Correlating malware and probing network activities to achieve the intended goals by employing numerous statistical, fuzzy hashing and entropy based techniques.

- Evaluating the proposed approach using a recent 60 GB of real darknet traffic and 65 thousand real malware samples.

4.1.2 Proposed Approach

In this section, we elaborate on the proposed approach as depicted in Figure 4.1. Specifically, we discuss its rationale, describe its components and present its employed mechanism, methods and techniques.

The rationale behind the proposed approach stems from the need to detect the infection at early stages of contamination. In this context, probing or scanning activities are known to be the very first symptoms of infection [35, 7]. On the other hand, the Internet dark space (i.e., dark sensors) has shown to be an effective method to generate Internet-scale cyber threat intelligence [130, 43]. In brief, darknet traffic is Internet traffic destined to routable but unused Internet addresses. Since these addresses are unallocated, any traffic targeting them is deemed as suspicious. Thus, in a nutshell, the proposed approach aims at extracting probing activities as received by a darknet and subsequently correlating

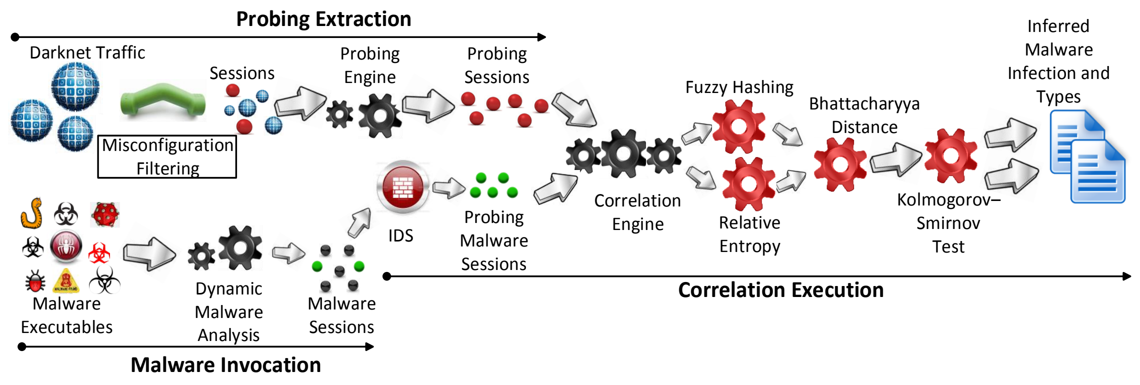


Figure 4.1: The Components of the Proposed Approach

them with malware samples. By leveraging geo-location information, the approach strives to generate insights related to worldwide compromised machines in addition to identifying their exact infected malware type/family. The approach is envisioned to be operated by a central authority, for example, an incident response center, where the latter will distribute, in real-time, the extracted inferences to concerned parties. In the sequel, we elaborate on each component of the proposed approach in accordance with Figure 4.1.

Misconfiguration Filtering

As mentioned, Internet traffic destined to routable yet unallocated IP addresses is commonly referred to as telescope or darknet data. Such malicious traffic is frequently, abundantly and effectively exploited to generate various cyber threat intelligence related but not limited to, scanning activities, distributed denial of service attacks and malware identification. However, such data typically contains a significant amount of misconfiguration traffic caused by network/routing or hardware/software faults. The latter immensely affects the purity of darknet data, which hinders the accuracy of detection algorithms that operate on such data in addition to wasting valuable storage resources. This section proposes a probabilistic model to preprocess telescope data to prepare it for effective use. The aim is to fingerprint darknet misconfiguration traffic and subsequently filter it out. The model is advantageous as it does not rely on arbitrary cut-off thresholds, provide separate likelihood models to distinguish between misconfiguration and other darknet traffic and is independent from the nature of the source of the traffic. To the best of our knowledge, the proposed model renders a first attempt ever to tackle the problem of preprocessing darknet traffic.

In a nutshell, the model aims at computing an access probability distribution for each darket IP address, derived across all remote sources that target those dark IPs. Thus, the model initially estimates the degree to which access to a given dark IP address is unusual. The model further considers the number of distinct dark IP addresses that a given remote source has accessed. Subsequently, the joint probability is formulated, computed and compared. If the probability of the source generating a misconfiguration is higher than that of the source being malicious (i.e., scanning or backscattered), then the source is deemed as misconfigured, subsequently flagged, and its corresponding generated darknet flows are filtered out.

Let $D = \{d_1, d_2, d_3, \cdots\}$ represent the set of darknet IP addresses and D_i a subset of those accessed by source s_i. First, the model captures how unusual the accessed destinations are. The idea behind this metric stems from the fact that misconfigured sources access destinations that have been accessed by few other sources. Thus, the model estimates the distribution of a darknet IP d_i being accessed by such a source as

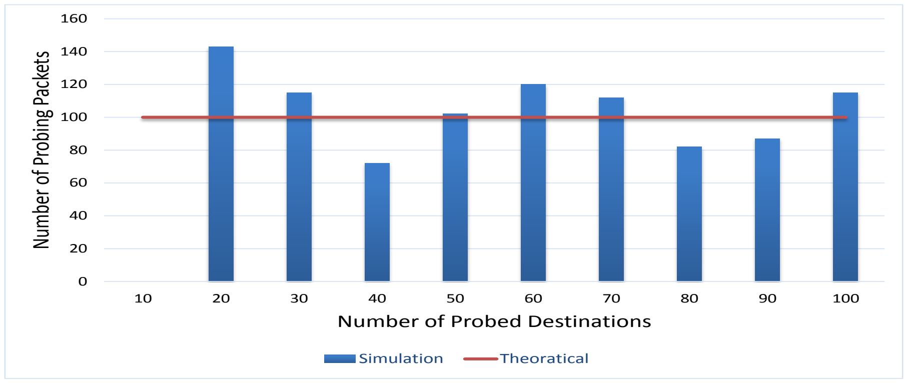
$$P_{misc}(d_i) = \frac{n_s(d_i)}{\sum\limits_{\forall d_j \in D} n_s(d_j)} \tag{4.1}$$

where $n_s(d_i)$ is the number of sources that have accessed d_i. In contrary, a malicious darknet source will access a destination at random. Typically, defining a suitable probability distribution to model the randomness of a malicious source targeting a specific darknet destination is quite tedious and unsystematic; often a simplistic assumption is applied to solve this issue. In this context, a very recent work by Durumeric et al. [87] assumed that darknet sources will probe their targets following a random uniform distribution. By adopting that assumption, one can model the probability of a darknet destination accessed by a malicious source as

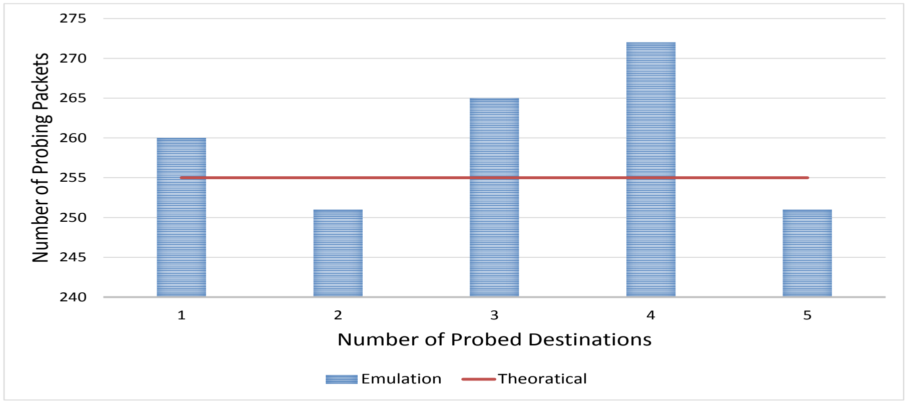
$$P_{mal}(d_i) = \frac{1}{|D|} \tag{4.2}$$

However, in this work, we thought it would be beneficial and more precise to verify the soundness of that assumption before completing the description of the proposed model. To successfully achieve the latter, we perform three experiments using simulation, emulation and real malicious darknet traffic. The first experiment is rendered by a simulation executed using Opnet Modeler[1]. The simulation is comprised of a probing source and 100 probing destinations/targets. The source and the destinations are represented using commodity machines. The probing source is instructed to generate three types of probing towards the targets, namely, TCP SYN, UDP and ACK scanning, for a duration of 15 minutes. The latter are typically common types of probing activities [35]. Figure 4.2a represents the outcome of this experiment. According to the assumption given by the uniform distribution, each target should theoretically receive around 100 packets. On average, the goodness-of-fit was around 76%, significantly below the acceptable 95% threshold [150]. To further assess the accuracy of the uniform distribution in modeling the target access distribution, we executed a second experiment. In this experiment, we employed Nmap, an open source scanning tool, to emulate the probing traffic. The emulation environment included a probing machine running the tool in addition to 5 target Virtual Machines (VMs). The probing machine repetitively executed Null, FIN and Xmas scanning towards the virtual machines for a duration of 10 minutes. The depiction of Figure 4.2b clearly demonstrates that the uniform distribution does not appear to be a good fit for such traffic. Although the latter simulation and emulation experiments demonstrated that the assumption of modeling malicious packets using the random uniform distribution is not quite accurate, we thought it would be interesting and more realistic to utilize real darknet traffic to assess that hypothesis. Subsequently, we extract one day of darknet data ($\approx$ 8GB) from April, 2014, divide it into slots of 4 hours and monitor the number of probing packets targeting 20 darknet destinations over the 6 slots. Figure 4.2c illustrates the outcome of this experiment. According to the uniform distribution, each target

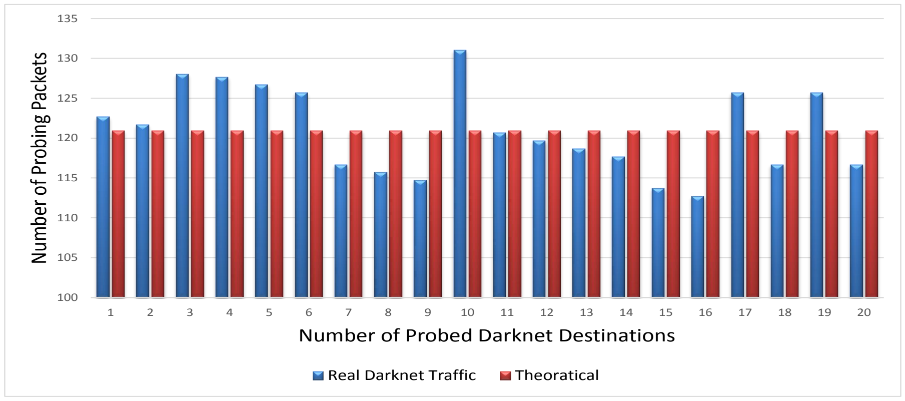
[1]http://tinyurl.com/nclm3gp



(a) Simulation Experiment



(b) Emulation Experiment



(c) Real Traffic Experiment

Figure 4.2: Uniform distribution of malicious packets in all experiments

should theoretically receive around 120 probing packets. On average, the goodness-of-fit was less than 78%. Such results from all the above experiments concur that the assumption of modeling darknet malicious packets towards darknet destinations by adopting the uniform distribution is not accurate; there indeed exists an imperative requirement to determine the most appropriate probability distribution model that best fits malicious darknet packets.

Recall, that the latter requirement needs to be fulfilled to accurately model darknet destinations targeted by a malicious source in order to continue the elaboration of the proposed model. To achieve that, we proceed by performing another set of experiments. Such experiments also rely on simulation, emulation and the utilization of real darknet traffic, and their corresponding setup environments is quite similar to those of the previous experiments. One difference is that we execute the experiments for 10 times and average the outcome. Another difference is related to the number of targeted destinations, in which we increase the latter number, for accuracy purposes, from 100 to 500 targeted destinations, 5 to 12 VMs and 20 to 110 monitored darknet destination, in the simulation, emulation and real traffic experiments, respectively. To determine the best fit model, we utilized a generic Matlab parametric probability distributions' fitting function and calculated the Bayesian Information Criterion (BIC) [151] for each of the models in relation to our data. The latter metric is an established criterion for model selection among a finite set of models. Typically, the lower the BIC is, the better is the fit. Figures 4.3a to 4.3c demonstrate the outcome probability density estimations in the three experiments. In the simulation experiment, it is evident that the Gaussian or the Normal distribution provides the best fit for modeling malicious darknet packets. In the second experiment, namely, the emulation experiment, it is revealed that the Pareto distribution provides the best fit; this is quite interesting as such distribution is often employed in modeling normal (i.e., benign) network traffic. Figure 4.3b further demonstrates that the inverse Gaussian distribution, which is analogous to the Normal distribution, also shows a positive BIC, resembling a good fit. Finally, the third experiment that employs real darknet traffic undoubtedly validates that the Gaussian distribution provides the best fit to model malicious packets targeting darknet destinations. It is worthy to note that neither of the three experiments portray the uniform distribution between the top 6 models providing a best fit. Such results concur that it is relatively accurate and practical to adopt the Gaussian distribution instead of the uniform distribution to model targeted darknet destinations.

Thus, at this point, equation (4.2) could be adjusted to match the probability density function of a Normal distribution. Consequently, we can now model the probability of a darknet destination accessed by a malicious source as

$$P_{mal}(d_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{4.3}$$

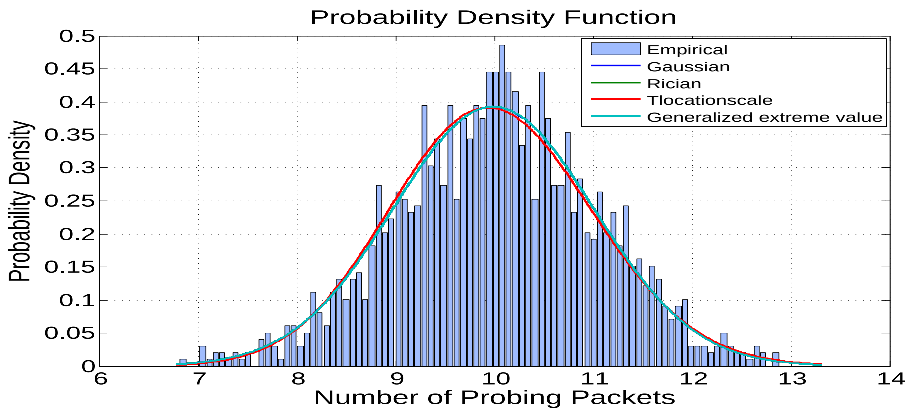
where σ is the standard deviation, μ is the mean, σ^2 is the variance and x is the location of the darknet destination following the distribution. Recall that equations (4.1) and (4.3) allows the model to initially capture how unusual the accessed destinations are. However further, the model considers how many darknet destinations have been accessed by a given source. The latter will be subsequently described.

Given a set of D_i, darknet destinations accessed by a specific source s_i, the model eventually aims at measuring two probability distributions, namely, $P_{misc}(D_i)$ and $P_{mal}(D_i)$. The former being the probability that D_i has been generated by a misconfigured source while the latter is the probability that D_i has been generated by a malicious darknet source.

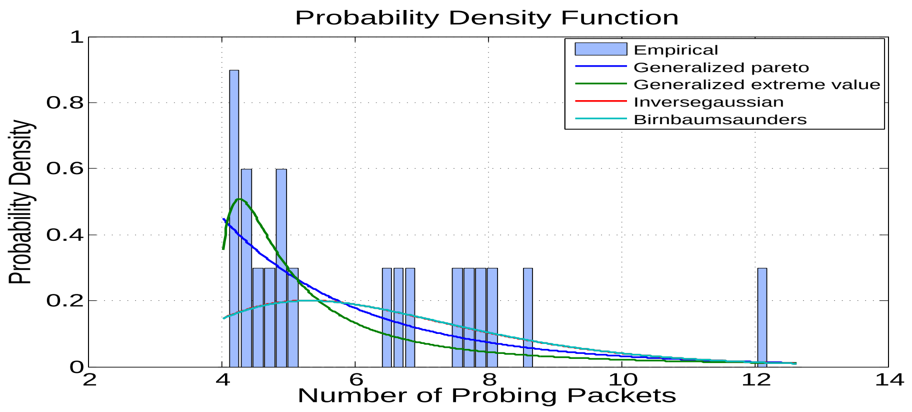
Let $D_1 = \{d_{i1}, d_{i2}, d_{i3}\}$ be those darknet addresses accessed by s_1. The model captures the probability $P(D_1)$ of the source generating $\{d_{i1}, d_{i2}, d_{i3}\}$ as the probability of s_1 accessing this specific combination of destinations knowing that it targeted three destinations multiplied by the probability of s_1 accessing any three destinations. The latter could be formalized as

$$P(D_i) = P(D_i = \{d_{i1}, d_{i2}, \cdots, d_{in}\} \mid |D_i| = n) \times P(|D_i| = n) \tag{4.4}$$

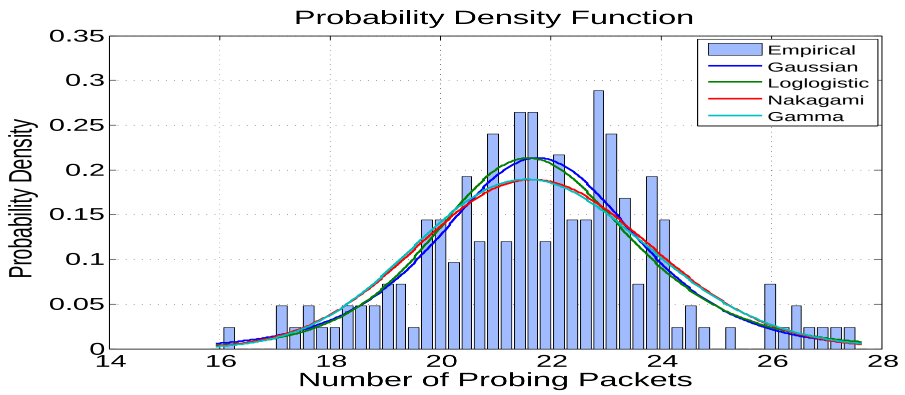
For both, a misconfigured and a malicious source, the first term of equation (4.4)



(a) Simulation Experiment



(b) Emulation Experiment



(c) Real Traffic Experiment

Figure 4.3: Model fitting sorted by BIC in all experiments

could be modeled as

$$P_{misc}(D_i = \{d_{i1}, d_{i2}, \cdots\} \;\mid\; |D_i|) = \frac{1}{K} \prod_{\forall d_j \in D_i} P_{misc}(d_i) \tag{4.5}$$

$$P_{mal}(D_i = \{d_{i1}, d_{i2}, \cdots\} \;\mid\; |D_i|) = \frac{1}{K} \prod_{\forall d_j \in D_i} P_{mal}(d_i) \tag{4.6}$$

where K, a normalization constant which is solely employed to allow the probabilities to sum to 1, could be defined as

$$K = \frac{|D|!}{n!\,(|D|-n)!} \times \frac{1}{|D|^n} \tag{4.7}$$

The likelihood that a source will target a certain number of darknet destinations (i.e., the second term of equation (4.4)) depends on whether the source is malicious or misconfigured. Characteristically, misconfigured sources access one or few destinations while malicious sources access a larger pool of destinations. We have modeled such distributions as

$$P_{misc}(|D_i|) = \frac{1}{(e-1)|D_i|!} \tag{4.8}$$

$$P_{mal}(|D_i|) = \frac{1}{|D|} \tag{4.9}$$

where the term $(e-1)$ in equation (4.8) guarantees that the distribution will sum to 1. It is noteworthy to mention that equation (4.8) ensures that the probability will significantly decrease as the number of targeted destinations increases. In contrast, equation (4.9) captures a malicious darknet source accessing a random number of darknet addresses.

By combining the above equations, we can model the probability of a source being a misconfigured or malicious, given a set of darknet destination addresses,

$$P_{misc}(D_i) = \frac{1}{K(e-1)|D_i|!} \prod_{\forall d_j \in D_i} P_{misc}(d_i) \tag{4.10}$$

$$P_{mal}(D_i) = \frac{1}{K|D|} \prod_{\forall d_j \in D_i} P_{mal}(d_i) \tag{4.11}$$

Algorithm 3: Inferring misconfiguration flows using the probabilistic model

```
Data: Darknet Flows, $DarkFlows$
Result: Flag, $MiscFlag$, indicating that the $DarkFlow$ is originating from a
        misconfigured source
1  for $DarkFlows$ do
2      $MiscFlag$=0
3      i=$DarkFlows$.getUniqueSources()
4      Amalgamate $DarkFlows_i$ originating from a specific source $s_i$
5      Update $s_i(D_i)$
6      Compute $P_{misc}(D_i)$, $P_{mal}(D_i)$
7      if $P_{misc}(D_i)>P_{mal}(D_i)$ then
8          $MiscFlag$=1
9      end
10 end
```

Indeed, Algorithm 3 provides a simplistic mechanism to infer misconfigured sources by employing the proposed darknet preprocessing model. It is worthy to note that step 6 of the algorithm (i.e., the computation of $P_{misc}(D_i)$ and $P_{mal}(D_i)$) is easily accomplished in practice by computing the negative log-likelihoods,

$$\begin{aligned} L_{misc}(D_i) &= -lnP_{misc}(D_i) \\ L_{mal}(D_i) &= -lnP_{mal}(D_i) \end{aligned} \tag{4.12}$$

Thus, Algorithm 3 deems a source and its corresponding flows as misconfiguration if $L_{mal}(D_i) - L_{misc}(D_i) > 0$.

Probing Extraction

In accordance with Figure 4.1, another component of the proposed approach is probing extraction. In Section 3.2 of Chapter 3, we have proposed a new approach to fingerprint Internet-scale probing activities. The approach aimed at detecting the probing activity and identifying the exact technique that was employed in the activity. Recall that the approach is advantageous in comparison with other methods [78, 72, 1, 3, 84] as it does not rely on identifying the scanning source and is independent from the scanning strategy (remote to local, local to remote, local to local), the scanning aim (wide range or target specific) and the scanning method (single source or distributed). Further, the proposed method does not depend (for detection) on a certain predefined alert threshold, the transport protocol used (TCP or UDP) or the number of probed destinations and ports. When empirically evaluated using a significant amount of real darknet data, the approach yielded 0 false negative and 2% false positive [7] in comparison with the leading Network Intrusion Detection System (NIDS), Snort[2]. In this work, and to successfully extract probing activities from darknet traffic, we adopt and leverage an enhanced version of that previously proposed approach. In what follows, we 1) perform another empirical evaluational of that approach in comparison with the Bro NIDS[3] for validation purposes and 2) describe the enhancements of the previously proposed approach (of Section 3.2) that are employed in this work.

1) Although the previously proposed approach was empirically evaluated and validated against Snort NIDS [7], we thought it would be also interesting in this section to further evaluate it against another NIDS, namely, Bro [152]. This procedure aims at providing more confidence in the approach in addition to significantly motivating its use as one of the initial components in Figure 4.1.

To achieve this procedure, we experimented with Bro NIDS. Paxson's Bro [152] identifies sources as scanners if they execute N failed connection attempts to a configurable

[2] `http://www.snort.org/`
[3] `http://www.bro.org/`

list of services. A connection is considered to have failed if it is unanswered or if it generates a TCP reset in response. For other services, Bro records information for all connection attempts. If the number of connection attempts for these services exceed N, this source is also identified as a scanner. The list of services and N are configured in various policy scripts implemented by Bro. For the sake of this work, we deployed Bro 2.3 with default configuration and employed the provided scanning script[4]. We use one week of darknet data (52 GB) that was collected during the duration of March 1st to March 7th 2014, to empirically evaluate the approach in [7] against Bro. Figure 4.4 demonstrates the outcome of this experiment.

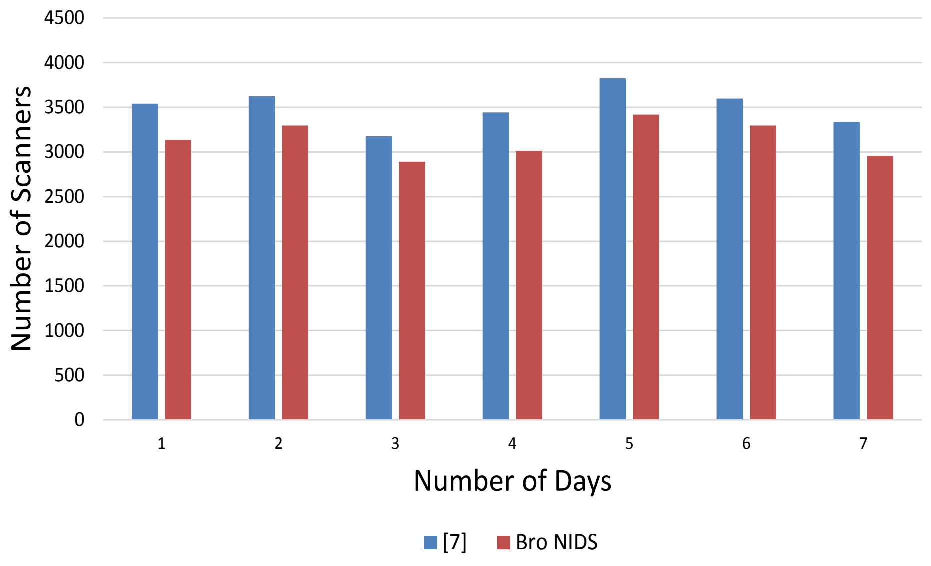


Figure 4.4: Empirical Evaluation of [7] against Bro NIDS

By investigating the sources of the scanners, it was disclosed that the approach in [7], similar to the outcome when previously compared with Snort NIDS, yielded 0 false negative; all the scanners detected by Bro were also detected by the fingerprinting approach. Further, it could be noted that the approach generated an average of 8% false positive. It is worthy to pinpoint that the approach, contrary to Bro NIDS, can detect various types of scans, which could include scans from a single host to a single port on a single host, slow scans and a specific host scanning multiple ports on multiple hosts. Nevertheless, in

[4]https://github.com/bro/bro-scripts/blob/master/scan.bro

an attempt to reduce the number of false positives, we enhance the approach as explained in the following.

2) We perform a crucial enhancement to the previously proposed approach of Section 3.2 of Chapter 3 that we employ in this work. The modification is rendered by UDP investigation. One of the issues related to the previously proposed approach is that it does not differentiate between UDP probing and UDP packets arising from distributed reflective denial of service attacks [153]; this can explain the relatively high percentage of false positives. Indeed, such attacks are an emerging form of distributed denial of service attacks that rely on the use of publicly accessible UDP servers as well as bandwidth amplification factors to overwhelm a victim with UDP traffic [41]. The idea is to send simple queries to such resolvers in which the replies, that aim at flooding the victim, are orders of magnitude larger. Such approach is behind the notorious 300 and 400 Gbps attacks that hit the Internet in the last few years [92]. Figure 4.5 depicts a specific scenario where the resolvers are open Domain Name System (DNS) servers.

The compromised machines are directed to execute a reflective dos attack against Org1. To achieve that, they initially spoof their identities by using those of Org1 and subsequently send simple `ANY` queries to the DNS open resolvers. The latter DNS query intends to pull all the available information from the DNS resolvers related to a requested domain. The domain in the request trace is often a noteworthy one that possess a significant amount of information. Commonly, the compromised machines will spay such queries on the Internet space in a hope to reach as many open resolvers as possible in order to increase the overall amplification factor. Intuitively, some of those requests will hit the network telescope/dark space and hence will be captured. Requests that actually reach open resolvers will be amplified and directed towards Org1.

Typically, the UDP services that could be leveraged for UDP amplification attacks are summarized in Table 4.1. To deal with this issue, we extend the previously proposed approach of Section 3.2 of Chapter 3 by employing the post-processing Algorithm 4.

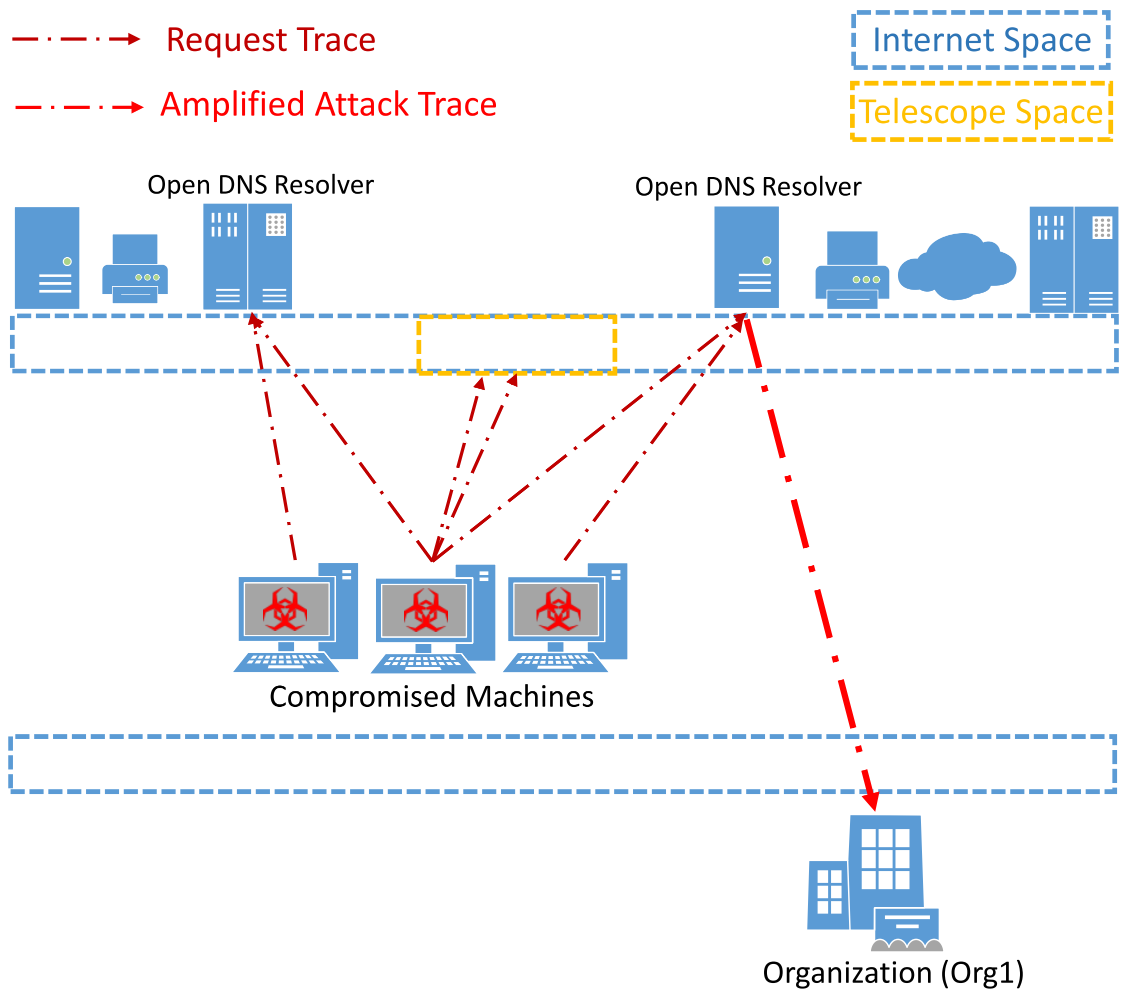


Figure 4.5: A Network Telescope pinpointing Sources of Reflective DoS Attacks

The algorithm aims at validating whether the inferred UDP probing flow or session from the outcome of [7] is indeed a probing activity or it is actually a false positive related to UDP packets originating from distributed reflective denial of service attacks. The algorithm achieves this by initially assuming that the flow is indeed a probing flow. Subsequently, it attempts to negate the latter assumption by relying on two observations. First, it monitors if the UDP flow in question is targeting one of those services of Table 4.1; a positive result, at this stage, suggests that the UDP flow is either probing that service or it is attempting to amplify a denial of service attack. If a positive result is recorded from

UDP Service	Port Number
DNS	53
NTP	123
SNMPV2	161
NetBIOS	137
SSDP	1900
CharGEN	19
QOTD	17
Quake Network Protocol	26000

Table 4.1: UDP vulnerable services and corresponding ports

the latter, the algorithm then verifies whether the source is spoofed or not. The rational behind that verification is based on the fact that probing packets are not spoofed (so the scanner/attacker can actually retrieve back the result of the scan) while those of a denial of service attack are indeed spoofed for identity anonymization purposes. The algorithm accomplishes this by leveraging the work of Templeton et al. [154] that maintains a track of the Time-To-Live (TTL) value of the packets generated by the same source. The core idea behind that approach is rendered by the fact that packets' TTL values fluctuate with source IP addresses; that is, given a non-spoofed IP address, its packets' TTL should remain constant (or within 10% change) throughout its communication period. It is worthy to mention that in terms of complexity, the algorithm possesses O(PF) space complexity and O(N) time complexity where PF is the number of probing flows and N is the number of UDP flows. By implementing the proposed Algorithm 4, we recorded a decrease of 3.5% in false positives when we re-executed the empirical evaluation (against Bro NIDS). Such result demonstrates that 1) the algorithm is effective in distinguishing between UDP probing and UDP packets originating from amplified denial of service attacks and 2) that the majority of the false positives were indeed caused by this issue.

Please recall that the outcome of this procedure (i.e., probing extraction in Figure

Algorithm 4: Verifying if the UDP flow is a probing activity or related to UDP distributed reflective denial of service attack

```
Data: Probing Flows, $ProbingFlows$;
List of Vulnerable UDP Services, $UDPVulList$
Result: Flag, $ProbFlag$, indicating whether the UDP flow, $UDPFlow$, is a
        true probing or not
for $UDPFlow$ in $ProbingFlows$ do
    $ProbFlag$=0;
    if $UDPFlow.getTarget().getPort()$ in $UDPVulList$ then
        $TTL$=$UDPFlow$.getTTL();
        for $UDPFlow$ in $ProbingFlows$ do
            if $UDPFlow.getTTL()!=TTL$ then
                $ProbFlag$=1;
            end
        end
    end
end
```

4.1) are accurate and validated probing sessions in packet capture format (i.e., pcaps).

Malware Invocation

In accordance with Figure 4.1, another component of the proposed approach is malware invocation. We operate a dynamic malware analysis module that is based on ThreatTrack Security sandbox environment[5] (i.e., controlled environment). After receiving daily malware samples from ThreatTrack feeds, they are interactively sent to the sandbox, where they are executed by client machines. The clients could be virtual or real and possess the capability to run Windows or Unix, depending on the malware type under execution. The behavior of each malware is monitored and all its corresponding activities (i.e., created files, processes, network traffic, etc.) are recorded. For the sake of this work, we extract

[5]`http://www.threattracksecurity.com/`

the network traffic generated by approximately 65 thousand unique and recent malware samples as pcaps. The pcaps contain communication traffic generated from the malware to other internal or external hosts. Those malware samples belong to diverse malware types including, Trojan, Virus, Worm, Backdoor, and AdWare coupled with their corresponding families and variants. We rely on Kaspersky for a uniform malware naming convention[6].

Correlation Execution

Consistent with Figure 4.1, the correlation engine formulates the problem of correlating probing and malware sessions as follows. Given a probing session that is extracted from darknet traffic, 1) investigate whether or not the session originates from a malware and 2) identify the exact or probable malware type/family that is generating such probing, if it was shown that the session is malware-related. In an attempt to address this problem, we perform the following. We leverage Snort's probing engine, the sfPortscan pre-processor, to detect which malware pcaps possess any signs of probing activity. We omit those malware pcaps that demonstrate a negative output. To attribute a specific malware to a probing session, we adopt a two-step procedure. First, we apply the notion of fuzzy hashing [155] between the probing session and the remaining malware pcaps. Fuzzy hashing is advantageous in comparison with typical hashing as it can provide a percentage of similarity between two samples rather than producing a null value if the samples are different. This popular technique is derived from the digital forensics research field and is typically applied on files or images [155, 156]; to the best of our knowledge, our approach is among the first to explore the capabilities of this technique on cyber security data. Readers that are more interested in the advatanges of fuzzy hashing and its applicability to malware research are kindly referred to [157, 158]. We further apply an information theoretical metric, relative entropy, as proposed by Lee and Xiang [159], between the given probing session and the malware pcaps. Relative entropy, which is defined by

$$d = \sum_k p_k \log \frac{p_k}{q_k}$$

[6]`http://securelist.com/en/threats/detect?chapter=136`

is a measure of the distance of the regularities between two datasets, p_k and q_k. If the relative entropy is $= 0$, this indicates that the two datasets have the same regularity. At this point, we 1) omit the probing sessions that demonstrate less than 5% similarity using both tests[7] and 2) select the top 10% malware pcaps that were found to minimize the entropy and maximize the fuzzy hashing percentage. The rational behind the latter approach stems from the need to filter out the malware pcaps that do not possess probing signs similar to the probing session. Second, using the remaining 10% malware pcaps, we extract their probing sessions as pinpointed by sfPortscan. For each of the malware probing sessions, we apply the Bhattacharyya distance [160] between those and the given probing session. The latter statistic test, which is defined by

$$d(p, p') = \sqrt{1 - L(p, p')}$$

where

$$L(p, p') = \sum_{i=1}^{N} \sqrt{p(i)p'(i)},$$

is an established and an effective metric to determine the overlap of two sample distributions, p and p'. The Bhattacharyya distance is often employed to measure the disjunction of classes in a typical classification problem and it is considered to be more reliable than other metrics, including for instance, the Mahalanobis distance [161]; when the two classes have similar means but different standard deviations, the Mahalanobis distance would tend to zero, while the Bhattacharyya distance would grow and yield better results depending on the difference between the standard deviations. By selecting 1% of malware pcaps that were shown to reduce the Bhattacharyya distance, we further significantly reduce the possible malware pcaps that the given probing session could be similar to. Finally, to exactly attribute the given probing session to a specific malware, we employ the two sample Kolmogorov-Smirnov statistic test [162] between the remaining malware probing sessions and the given probing session. The test will output 0 if a positive match occur; 1 otherwise. If a positive match occurs, this indicates that the probing session has

[7]These sessions indicate that they do not possess any malware-related behavior.

been generated from the inferred exact malware. Otherwise, we refer back to the output of the Bhattacharyya distance and select a set of probable malware pcaps that were shown to be relatively close to the given probing session.

It is worthy to mention that from a processing overhead perspective, the proposed approach is able to process and correlate one day of darknet data ($\approx$ 8 GB) with 65 thousands malware samples in less than 20 minutes. The latter information strongly advocate that the approach is practically viable in a real-world environment and that it can be easily rendered as an operational cyber security capability providing prompt near real-time inferences related to worldwide-compromised machines.

4.1.3 Empirical Evaluation

As previously mentioned in this thesis, we possess real darknet data that we are receiving on a daily basis from a trusted third party, namely, Farsight Security. Such traffic originates from the Internet and is destined to numerous /13 network sensors. The darknet sensors cover more than 12 countries and monitor around half a million dark IPs. Recall that darknet traffic is Internet traffic destined to routable but unused Internet addresses. Since these addresses are unallocated, any traffic targeting them is deemed as suspicious. The data mostly consists of unsolicited TCP, UDP and ICMP traffic. It might contain as well some DNS traffic. We use one week of data (60 GB) that was collected during the duration of April 1st to April 7th 2014, to empirically evaluate our approach.

Darknet traffic is typically composed of three types of traffic, namely, scanning, backscattered and misconfiguration [132]. Scanning arises from bots and worms while backscattered traffic commonly refers to unsolicited traffic that is the result of responses to denial of service attacks with spoofed source IP addresses. On the other hand, misconfiguration traffic, as tackled in Section 4.1.2, is due to network/routing or hardware/software faults causing such traffic to be sent to the darknet sensors.

We first aggregate the darknet traffic connections into sessions using an approach similar to the first step algorithm by Kannan et al. [133]. We consider all those connections within T_{aggreg} of each other as part of the same session for a given pair of hosts. We used the same proposed threshold as in [133], $T_{aggreg} = 100$ seconds, and found that this seems to correctly group the majority of connections between any given pair of hosts. To filter out misconfiguration data, we employ the proposed probabilistic model of Section 4.1.2 coupled with Algorithm 3. Figure 4.6 depicts the distribution of darknet sessions between misconfiguration and malicious sessions while Table 4.2 presents a sample of 10 misconfigured sources as inferred by the algorithm.

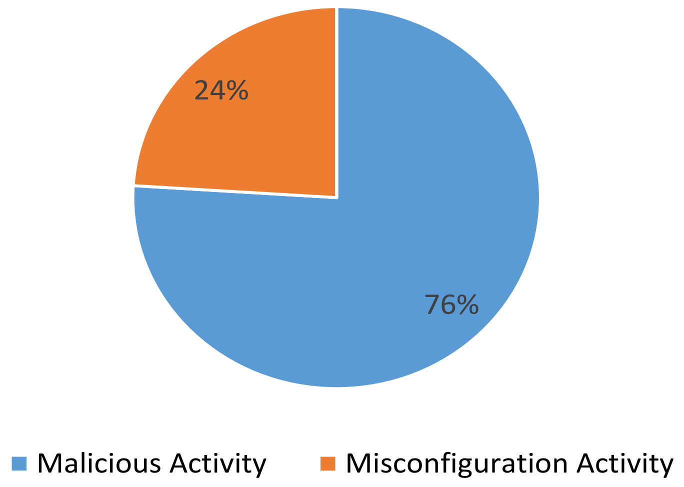


Figure 4.6: The distribution of darknet sessions

It is revealed that close to 25% of the sessions are indeed related to misconfiguration. This is relatively consistent with a previous study that we have performed in 2012 [41], which was solely based on manual verification. By further manually investigating the inferred misconfiguration sessions, it was shown that all the sessions are single flows that targeted the dark space only once in which 87% of them are malformed packets.

We further execute the enhanced probing fingerprinting approach that was highlighted in Section 10 to extract 400 probing sessions. It is noteworthy to mention, that each of those probing sessions is being generated by a unique source. Keeping in mind that most probing activity is generated from non-spoofed IP addresses (so the actual scanner

Source	$L_{mal} - L_{misc}$
1	99
2	132
3	113
4	14
5	146
6	106
7	39
8	97
9	2
10	133

Table 4.2: A sample of 10 misconfigured sources

can essentially receive back the probing results), the extracted probing sessions indeed resemble real machines.

We proceed by investigating any signs of probing activities within the 65 thousand malware pcaps. The output disclosed that 13,105 malware pcaps possessed such activities. The latter corroborates that a significant amount of malware samples in fact generate probing activities; leveraging such activities for early infection detection might be a viable and a promising approach. The distribution of the types of those probing activities within the identified malware pcaps is depicted in Figure 4.7. It is disclosed that UDP probing is the most employed probing technique; such result is consistent with other malware studies [163], where the authors revealed that UDP is the most used transport-layer protocol for malicious command-and-control communications. Further, Figure 4.8 illustrates the top 10 malware types that were found to trigger such probing activities. One interesting observation that could be extracted from such result is related to 'Virus.Win32.Sality.bh'; Dainotti et al. [120] have recently documented a large-scale probing campaign that was able to probe VoIP (SIP) servers of the entire IPv4 address space in 12 days. The authors

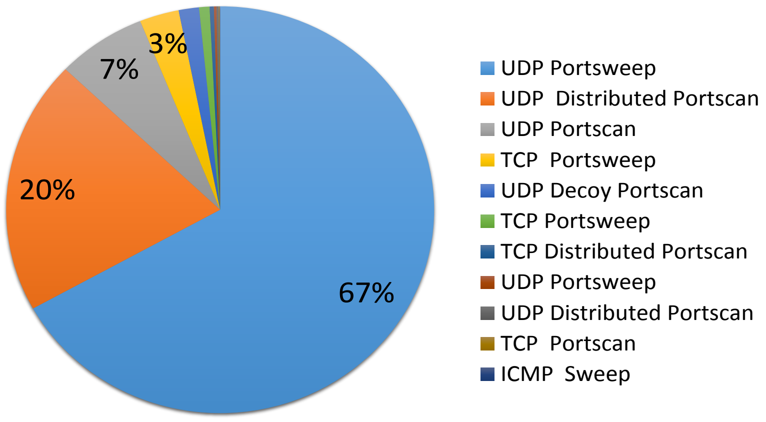


Figure 4.7: Types of Probing Activities generated by Malware

pinpointed that the malware responsible for such campaign was in fact the Sality malware; the same malware that we found, using our data set, to be generating the majority of the probing activities.

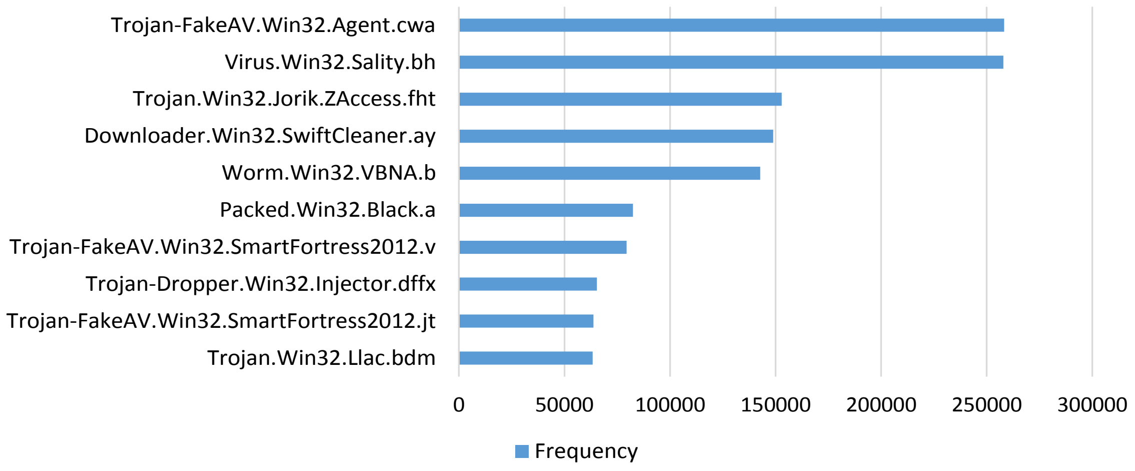


Figure 4.8: Distribution of Probing Malware Types/Families

We proceed in an attempt to reduce the number of malware pcaps that could be eventually attributed to the darknet probing sessions. In accordance with Section 11, we executed the fuzzy hashing and relative entropy approach. To accomplish the former, we leveraged deeptoad[8], a fuzzy hashing implementation, while we employed matlab[9] to accomplish the latter. Subsequently, we select the top 10% (1,310) malware pcaps that were found to minimize the entropy and maximize the fuzzy hashing percentage, in comparison with the darknet probing sessions. At this point, 212 probing sessions were filtered out, indicating that they do not possess any malware-related behavior.

For the purpose of attributing the remaining probing sessions to a manageable and a probable set of malware pcaps, in coherence with the proposed approach of Section 11, we proceed by executing the Bhattacharyya distance and selecting the 1% of malware pcaps (13 pcaps) that are shown to be statistically close to each of the probing sessions. Table 4.3 provides a specimen that couples 5 probing sources with few of their corresponding possible set of malware infections. Intuitively, we record the complete malware outcome for all the probing sources.

Note that 'Trojan-Downloader.Win32.KiayksayRen.b', that frequently appears in Table 4.3, has been confirmed by numerous anti-malware engines and services to be a significant sign of machine exploitation[10], particularly those running the Windows operating system. Thus, the proposed approach seems accurate and practical in pinpointing compromised machines in addition to disclosing the probable malware types that caused their contamination.

To exactly identify which malware sample is responsible for the darknet extracted probings sessions, we proceed by employing the Kolmogorov-Smirnov statistic test, as instructed in Section 11. Table 4.4 shows the extracted insights for a sample of 10 probing sessions while Figure 4.9 visualizes the worldwide map of the fingerprinted infections.

[8]`https://code.google.com/p/deeptoad/`

[9]`http://www.mathworks.com/matlabcentral/fileexchange/35625-information-theory-toolbox/content/relativeEntropy.m`

[10]`http://tinyurl.com/ktlqp4r`

Probing Source 1	Trojan-Downloader.Win32.KiayksayRen.b Trojan-FakeAV.Win32.SmartFortress2012.il Trojan.Win32.VBKrypt.hadj Trojan-Dropper.Win32.Agent.dtki
Probing Source 2	Trojan-Downloader.Win32.KiayksayRen.b Trojan-Dropper.Win32.Dapato.aflm Trojan-Dropper.Win32.Injector.dknf Trojan.Win32.Jorik.Shakblades.foc
Probing Source 3	Trojan-Downloader.Win32.KiayksayRen.b Worm.Win32.AutoIt.xl Trojan.Win32.Scar.furz Packed.Win32.PolyCrypt.d
Probing Source 4	DTrojan-Downloader.Win32.KiayksayRen.b Trojan-Downloader.Win32.Dapato.gje Trojan.Win32.Agent.btmu Virus.Win32.Sality.bh Trojan-FakeAV.Win32.SmartFortress2012.v
Probing Source 5	Trojan-Downloader.Win32.KiayksayRen.b Trojan-Spy.Win32.SpyEyes.acxb Trojan.Win32.FakeAV.lete Packed.Win32.PolyCrypt.d

Table 4.3: Probing Sources coupled with their probable malware samples

Note that we also generate supplementary material related to the infections including geo-location information per real source (i.e., hostname), organization, ISP, city, region and country. Although, we refrain from publishing those due to sensitivity/legal issues, we can note that the infections originate from 67 diverse operational providers, 67 distinct ISPs and 38 different countries. Such results, which we postulate to be communicated to concerned providers, concur that the proposed approach possesses the capability to infer compromised machines in addition to pinpointing the exact malware type/family that was responsible for their contamination.

Although we are unable to validate the existence of every single obtained inference

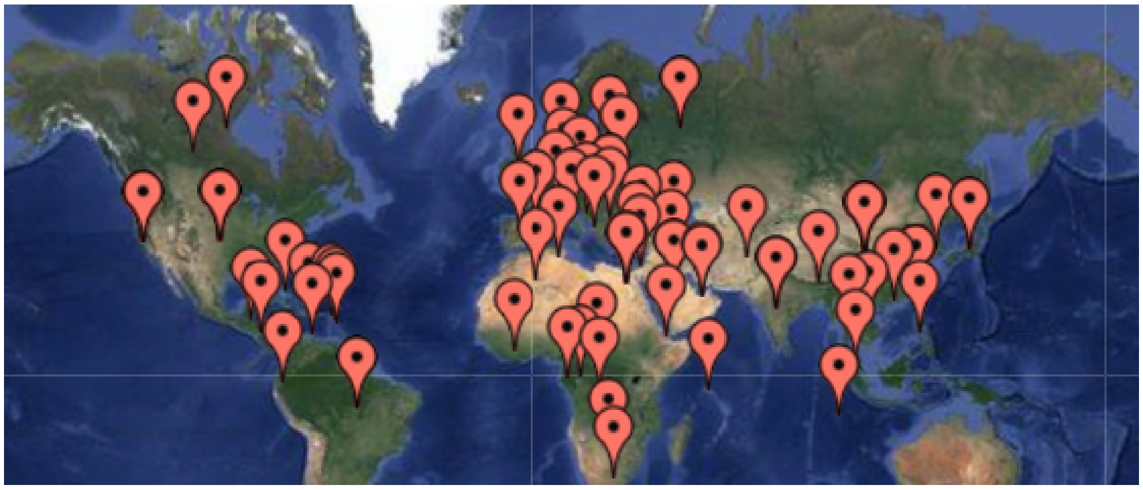


Figure 4.9: Inferred Worldwide Infections by Correlating Malware and Probing Activities

Probing Source a	Trojan-Spy.Win32.VB.gt
Probing Source b	Virus.Win32.Sality.s
Probing Source c	Trojan-FakeAV.Win32.SmartFortress2012.jv
Probing Source d	Trojan-Dropper.Win32.Injector.dpdj
Probing Source e	Backdoor.Win32.Bifrose.fur
Probing Source f	Virus.Win32.Cabanas.MsgBox
Probing Source g	Trojan.Win32.Jorik.Downloader.akr
Probing Source h	Trojan-FakeAV.Win32.Agent.cwa
Probing Source i	Trojan.Win32.Swisyn.bahq
Probing Source j	Worm.Win32.Juched.djh

Table 4.4: A Sample of Inferred Infections

related to the extracted worldwide infections due to legal and logistic constraints, we perform two activities that advocate the accuracy and completeness of the proposed approach.

First, we have observed, from the obtained results, a number of events that support the proposed approach. (1) We inferred that the majority of the infections that are related to the previously mentioned 'Virus-Win32.Sality.bh' are originating from Thailand; in [120], the authors disclosed that the bots that contributed to the large-scale VoIP probing campaign that were found to be infected by the same Sality malware, were in fact attributed to Thailand. (2) We noticed that Chinese ISPs lead in the number of generated infections. According to our results, one of the top extracted malware infections that is generated from those ISPs is the 'Trojan-Banker.Win32.Banker.adx'. This malware is a data stealing program that captures banking credentials such as account numbers and passwords from infected users. The latter insights were confirmed by McAfee in which they further concurred that China is in fact responsible from more than 45% of such contamination[11]. (3) From our results, we deduced that the malware 'Backdoor:Win32/Bifrose' was originating from a specific middle-Eastern country. The latter malware allows an external attacker to access the compromised machine to perform various malicious actions. McAfee also confirmed our findings by revealing that the same country is indeed the most contributor to such an infection[12].

Second, we relied on third party publically available data sources provided by the online services, ThreatStop[13], MxLookup[14], brightcloud[15] and ReputationAuthority[16] to validate the obtained contamination incidents. The latter cyber security data repositories provide information on Internet-scale contamination incidents per IP address. We compare the extracted malware IP addresses that were inferred by the proposed approach against those repositories. The outcome discloses that around 91% of the extracted malware IP addresses from the proposed approach were indeed found as malware-related from those repositories. The outcome also pinpointed that 57% of those overlapping confirmed incidents are still active on operational machines. Such results demonstrate that the extracted inferences from the proposed approach exhibit noteworthy accuracy and can generate significant cyber security insights that could be used for prompt mitigation.

[11]http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=853515

[12]http://www.mcafee.com/threat-intelligence/malware/default.aspx?id=1594628

[13]http://www.threatstop.com/

[14]http://mxtoolbox.com/

[15]http://www.brightcloud.com/

[16]http://www.reputationauthority.org/

4.1.4 Approach Limitations

It is realistic to acknowledge a number of limitations in the proposed approach. First, the approach leverages the dark space to infer Internet-scale probing activities. Although the monitored space is relatively large (i.e., /13), we are unable to monitor events that do not target such space. Subsequently, the approach will be unable to correlate those "unseen" activities with malware samples, and thus will fail to detect and identify their corresponding malware infections. Second, the approach relies on malware samples that actually execute probing activities. Although, from our experiments, the number of those malware seems to be significant, the approach will not be able to detect malware that do not probe. In this case, our correlation engine could be used in conjunction with already deployed approaches, similar to those that rely on honeypots to accomplish the detection. Third, in relation to the misconfiguration darknet preprocessing model, there is a need to design and implement confidence levels to assess whether the difference of the two probability estimates is large enough to safely choose one model over the other. This point is left for future work.

4.2 Multidimensional Investigation of Source Port 0 Probing

During November 2013, the operational cyber/network security community reported an unprecedented increase of traffic originating from source port 0. This event was deemed as malicious although its core aim and mechanism were obscured. This section investigates that event using a multifaceted approach that leverages three real network security feeds that we receive on a daily basis, namely, darknet, passive DNS and malware data. The goal is to analyze such event from the perspectives of those feeds in order to generate significant insights and inferences that could contribute to disclosing the inner details of that incident. The approach extracts and subsequently fingerprints such malicious traffic from the received darknet data. By executing unsupervised machine learning techniques on the extracted traffic, we disclose clusters of activities that share similar machinery. Further, by employing a set of statistical-based behavioral analytics, we capture the mechanisms of those clusters, including their strategies, techniques and nature. We consequently correlate the sources with passive DNS in order to investigate their maliciousness. Moreover, to examine if the sources are malware contaminated, we execute a correlation mechanism between the darknet data and the malware feeds. The outcome reveals that such traffic indeed is reconnaissance/probing activities originating from three different horizontal scans utilizing packets with a TCP header length of 0 or packets with odd flag combinations. The results as well demonstrate that 28% of the scanning sources host malicious/blacklisted domains as they are often used for spamming, phishing and other fraud activities. Additionally, the outcome portrays that the bot probing sources are infected by 'Virus.Win32.Sality'. By correlating various evidence, we confirm that such malware specimen is in fact responsible for part of the source port 0 probing event. We concur that this work is a first attempt ever to comprehend the machinery of such unique event and we hope that the community could consider it as a building block for auxiliary analysis and investigation.

4.2.1 Background

On November 2nd, 2013, security researchers at Cisco Systems reported that their world-wide deployed sensors have detected a massive increase in TCP source port 0 traffic[17]. They further elaborated that the magnitude observed by the sensors was elevated by 20 times than typical traffic originating from the same port and transport protocol on other days. According to the researchers, such event renders the largest spike in network traffic originating from TCP source port 0 in the last decade. In a follow-up discussion[18], the researchers noted that such port, according to its RFC, is engineered to be reserved, and that such traffic could be used to fingerprint various operating systems. Additionally, the security researchers speculated about the aim, mechanism and source of that traffic by stating that such rare event could be some sort of a research project, a malware infected probing botnet, a targeted reconnaissance event aiming to launch an immediate or a prolonged malicious task, or even a broken embedded device or a new piece of malware with

[17]http://tinyurl.com/pds443n

[18]http://tinyurl.com/n8j58hs

a bug in its scanning code.

The event was interestingly also observed by DShield/Internet Storm Center (ISC)[19]. ISC data comprises of millions of intrusion detection log entries gathered daily from sensors covering more than 500 thousand Internet Protocol (IP) addresses in over 50 countries. As shown in Fig. 4.10, the event was apparent on four days, namely, November 2nd[20], November 21st, November 24th and November 25th, 2013. The latter fact is particularly demonstrated by the peaks of the TCP ratio of port 0 on those specific days.

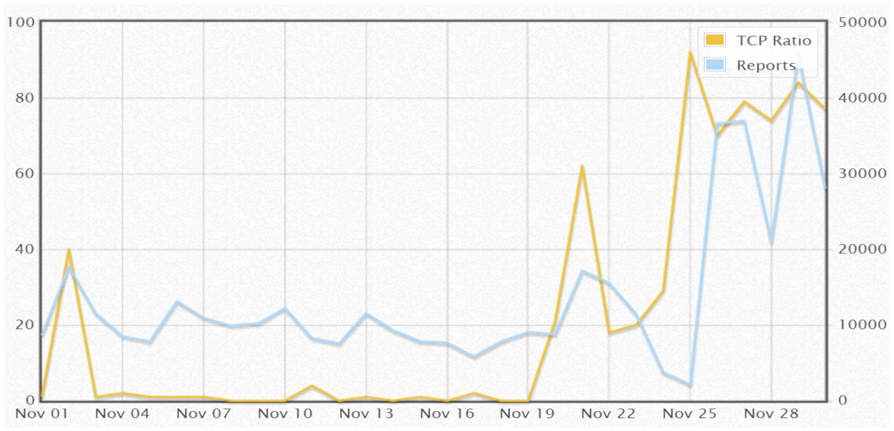


Figure 4.10: The Source Port 0 event as observed by DShield/Internet Storm Center

4.2.2 Contributions

Motivated by the requirement to shed the light on that incident in order to generate inferences and insights that could contribute in disclosing the inner details of such an unprecedented event, this section contributes by:

- Proposing a multifaceted approach that leverages three real network security feeds. The approach exploits darknet data (i.e., Internet traffic destined to half a million routable yet unallocated IP addresses) to extract, analyze and uncover the machinery of such traffic. Further, the approach employs correlation between the

[19]http://www.dshield.org/port.html

[20]coinciding with Cisco reports although not quite as significant

latter and passive DNS (i.e., Internet-wide authoritative DNS responses) to study the maliciousness of the such traffic. Moreover, the proposed approach correlates darknet-extracted traffic with malware feeds to answer questions related to contamination and attribution. To the best of our knowledge, 1) the proposed approach that correlates those three feeds in an effort to understand a cyber event has never been attempted before and 2) the yielded outcome from adopting such an approach related to this specific event is unique in the literature.

- Employing 1) machine learning data clustering techniques to partition the port 0 traffic according to similar machinery and 2) a set of novel behavioral analytics that scrutinize such traffic to capture the behavior of the sources.
- Evaluating the proposed approach using 30 GB of real darknet traffic, 1.4 billion DNS records and 30 million malware analysis reports.

4.2.3 Proposed Approach

In this section, we present the proposed approach that is composed of three mechanisms, namely, darknet analysis, passive DNS correlation and malware correlation.

Darknet Analysis

As previously mentioned, we possess real darknet data that we are receiving on a daily basis since three years from a trusted third party. Although the monitored darknet space is relatively large (i.e., /13), we were unable to notice the existence of the source port 0 event on November 2nd or November 21st. However, we were able to retrieve around 30 GB of darknet data that encompasses the event from November 24th and 25th. We base our darknet analysis approach on such data.

Traffic Extraction

In order to retrieve the packets of the source port 0 event, we created a simplistic TCPdump filter[21] that captures any darknet traffic that is utilizing TCP as the transport protocol and 0 as the source port. We applied the filter upon the 30 GB darknet data. We further refined the output by filtering out any darknet misconfiguration that could exist. To accomplish this, we adopt a metric that records the average number of sources per destination darknet address. This metric should be significantly larger for misconfiguration than probing traffic. However, although it differentiates misconfiguration from scanning, it could include as well backscattered traffic as they also can possess a large average number of sources per destination (i.e, in case of a DoS). To cope with this issue, we observe, per the technique in [132], flags in packet headers, such as TCP SYN+ACK, RST, RST+ACK, ACK, etc., that resemble backscattered traffic [132]. Subsequently, we filter out flows that lack that observation, deeming them as misconfiguration. The remaining output is rendered as the generated traffic from the source port 0 event, which is saved in a packet capture (i.e., pcap) format for further analysis.

Traffic Fingerprinting

To identify the nature of the source port 0 event, we implemented the technique from Bou-Harb et al. [33]. The latter approach specifically operates on darknet data and possesses the capability to distinguish between probing and DoS sessions. To accomplish this, the technique leverages the detrended fluctuation analysis statistical method and assigns a certain unique correlation value depending on the nature of each session. Readers who are interested in the inner details of such technique are kindly referred to [33]. By subjecting the source port 0 event traffic to the technique, the outcome revealed that 97% of the sessions are related to probing activities. We manually inspected the remaining 3%, which demonstrated that they are misconfiguration traffic that apparently, were not previously filtered as expected. We also confirmed such probing results by exposing the source port

[21]http://www.danielmiessler.com/study/tcpdump/

0 event traffic to Snort's[22] probing engine, the sfPortscan pre-processor[23], which yielded a similar result.

Traffic Clustering

For the purpose of disclosing the inner mechanisms of the source port 0 event, we refer to machine learning techniques. Such techniques allows us to efficiently, effectively and automatically uncover clusters of activities sharing similar machinery within the global event, without relying on strenuous manual analysis.
When the data observations are not pre-labeled into defined numerical or categorical classes, as in our case, two standard widely deployed algorithms for data clustering using unsupervised learning could be employed. These are the k-means [164] and the EM [165] algorithms. We proceed by going back to the source port 0 event traffic pcap file that we have previously isolated. Subsequently, we extracted from it a total of 29 data link, network and transport layer packet features as summarized in Table 3.10. The latter features have been shown to produce distinguishing characteristics when applied on network data [166]. This feature extraction procedure was achieved using the open source jNetPcap API[24]. We consequently compiled the extracted features into a unified data file and applied the k-means and the EM algorithms, leveraging MATLAB's default clustering functionality and the WEKA data mining tool[25], respectively.

Behavioral Analytics

In an attempt to capture the machinery of the probing sources/clusters, we present the following set of novel behavioral analytics. Such proposed approach takes as input the previously extracted probing sessions (recall Section 4.2.3) and outputs a series of behavioral characteristics related to the probing sources. In what follows, we pinpoint the concerned questions and subsequently present the undertaken approach in an attempt to answer those.

[22]http://www.snort.org/
[23]http://manual.snort.org/node78.html
[24]http://jnetpcap.com/
[25]http://www.cs.waikato.ac.nz/ml/weka/

Is the probing traffic random or does it follow a certain pattern? When sources generate their probing traffic, it is significant to capture the fashion in which they accomplish that. To achieve this task, we proceed as follows. For each distinct pair of hosts retrieved from the probing sessions (probing source to target), we test for randomness in the generated traffic using the non-parametric Wald-Wolfowitz statistic test. If the result is positive, we record it for that specific probing source and apply the test for the remaining probing sessions. If the outcome is negative, we infer that the generated traffic follows a certain pattern. To capture the specific employed pattern, we model the probing traffic as a Poisson process and retrieve the maximum likelihood estimate intervals (at a 95% confidence level) for the Poisson parameter λ that corresponds to that traffic. The choice to model the traffic as a Poisson distribution is motivated by [129], where the authors observed that probe arrivals is coherent with that distribution. After the test has executed for all the probing sources, we apply the CLUstEring based on local Shrinking (CLUES) algorithm on the generated patterns. CLUES allows non-parametric clustering without having to select an initial number of clusters. The outcome of that operation is a set of specific λ intervals. The aim of this is to map each probing source that was shown to employ a pattern to a certain λ interval by removing overlapping values that could have existed within the initially generated λ intervals.

How are the targets being probed? As revealed in [120], coordinated probing sources employ various strategies when probing their targets. These strategies could include IP-sequential, reverse IP-sequential, uniform permutation or other types of permutations. In an attempt to capture the probing strategies, we execute the following. For each probing source, we extract its corresponding distribution of target IPs. To differentiate between sequential and permutation probing, we apply the Mann-Kendall statistic test, a non-parametric hypothesis testing approach, to check for monotonicity in those distributions. The rational behind the monotonicity test is that sequential probing will indeed induce a monotonic signal in the distribution of target IPs while permutation probing will not. Further, in this work, we set the significance level to 0.5% since a higher value could introduce false positives. To differentiate between (forward) IP-sequential and reverse IP-sequential, for those distributions that tested positive for monotonicity, we also record the slope of the distribution; a positive slope defines a forward IP-sequential strategy while a negative one renders a reverse IP-sequential strategy. For those distributions that tested negative for monotonicity (i.e., not a sequential strategy), we leverage the chi-square goodness-of-fit statistic test. The latter insight will inform us whether or not the employed strategy is a uniform permutation; if the test fails, then the employed strategy will be deemed as a permutation; uniform permutation otherwise.

What is the nature of the probing source? It is significant as well to infer the nature of the probing source; is it a probing tool or a probing bot. From the two previous questions, we can infer those probing events that are random and monotonic. It is known that monotonic probing is a behavior of probing tools in which the latter sequentially scan their targets (IPs and ports). Furthermore, for random events, the monotonic trend checking can help filter out traffic caused by the non-bot scanners [129]. Thus, we deem a probing source as leveraging a probing tool if their traffic is randomly generated and if they adopt a sequential probing strategy (i.e., including reverse IP-sequential); a bot otherwise.

Is the probing targeted or dispersed? When sources probe their targets, it would be interesting to infer whether their probing traffic is targeted towards a small set of IPs or dispersed. To answer this, for each probing source b, we denote $\text{GF}(b)$ as the collection of flows generated by that specific source that target the dark space. The destination target IPs used by the flows in $\text{GF}(b)$ induce an empirical distribution. Subsequently, we borrow the concept of relative uncertainty, an information theoretical metric and apply it on those distributions. The latter index is a decisive metric of variety, randomness or uniformity in a distribution, regardless of the sample size. An outcome that is close to 0 defines that the probing source is using a targeted approach while an outcome value close to 1 means that its corresponding probing traffic is dispersed.

It is evident that the latter set of behavioral analytics significantly depend on numerous statistical tests and methods to capture the behavior of the probing sources. We assert that such approach is arguably more sound than heuristics or randomly set thresholds. It is also worthy to mention that all the employed statistical tests assume that the data is drawn from the same distribution. Since the approach operates on one type of data, namely, darknet data, we can safely presume that the values follow and are in fact drawn from the same distribution.

Passive DNS Correlation

We are also receiving on a daily basis around 1.3 million Domain Name System (DNS) messages. Such data is collected by observing DNS traffic between recursive DNS resolvers on the Internet. The latter is often dubbed as passive DNS data, which constitutes the successful translations and associations between domains and IP addresses. Passive DNS analysis has shown to be an effective approach to generate cyber threat intelligence [167]. We amalgamate a database that contains the last three-year period of such traffic ($\approx$ 1.4 billion records) in order to investigate the source port 0 probing event.

The rationale of employing passive DNS correlation is rendered by the requirement to contribute in investigating and perhaps attributing the probing sources of such an unprecedented event to certain malicious entities (i.e., malicious domains, for instance). Particularly, we aim to extract the following information about the suspicious IP addresses (previously retrieved from darknet analysis) that have participated in the probing event.

- Hosting capability: Malicious entities typically host a significant number of services and malicious domains on a limited number of IP addresses for cost-effectiveness reasons. Thus, by inferring domains that are resolved to a specific IP address, we can pinpoint and analyze these hosted domains to reveal the level of maliciousness of that IP address.
- Intensity: By computing the number of DNS messages that utilize a certain IP address, we can deduce the levels of accessibility and involvement of that IP address

in malicious activities.

- Aliveness: By recording the first and last timestamp that a specific IP address has been observed in DNS traffic (i.e., resolved to certain domain(s)), we can infer the participation period and aim of that IP address. Further, we can refine the analyzed passive DNS interval for the purpose of investigating and attributing that IP address with a certain cyber event.

Malware Correlation

We possess as well dynamic malware analysis reports (i.e., XML reports) of malware binaries for the last four years. We are receiving such feed on a daily basis with an average of 30 thousand XML malware reports from ThreatTrack Security[26]. Up to the event date in November 2013, we have accumulated more than 30 million malware analysis reports since January 2010. The XML reports are produced by analyzing the malware binaries in a controlled environment. Each XML report corresponds to only one malware sample. It is worthy to mention that these reports contain the executed activities by the malware samples at the network and system levels. On one hand, the network level activities refer to the connections and the exchanged packets, including IP addresses, port numbers, urls, visited domains and the actual payload data that has been sent. On the other hand, the system level activities constitute the list of Dynamic-link Library (DLL) files that are utilized by the malware, the key registry changes, and the memory usage.
We leverage such XML reports to investigate the source port 0 probing event. Specifically, we attempt to answer the following two questions by presenting their corresponding approaches:

Which malware has infected the probing machines before or during the occurrence of the event? In order to infer which malware has infected the probing machines, we present Algorithm 5. Simplistically, the Algorithm parses the XML reports mining for those malware that utilize the probing machine IP addresses (previously retrieved from darknet analysis) as per the destination IP address criterion. The Algorithm

[26]http://www.threattracksecurity.com/

Algorithm 5: Extracting malware samples that have infected the probing machines

```
Input: Dynamic malware analysis reports: $XMLs$;
List of the scanning source IPs: $ProbingIPs$;
Output: List of malware samples that infected the probing machines: $MalwareList$
for xml in XMLs do
    if xml.getMalware().getdestIP() in ProbingIPs then
        if xml.getMalware().getTime()==Nov, 2013 then
            MalwareList.add(xml.getMalware().getName());
        end
    end
end
```

further refines the output by only considering those malware that connect to the probing IPs in November, 2013.

Which malware generated the probing traffic? In an attempt to attribute the probing machines to a certain malware, we perform the following. We filter the entire set of malware XML reports by focusing on those samples that execute traffic from TCP source port 0. We subsequently match the outcome from the latter procedure with the list of malware that infected the probing machines that was derived from Algorithm 5. The rational behind this approach states that if a certain machine has been infected by a specific malware sample, in which it was derived that such machine is generating TCP source port 0 traffic, then it is highly probable that this specific malware is causing such traffic.

4.2.4 Empirical Evaluation

This section abides by the proposed approach that was previously discussed to disclose various inferences from the perspective of the three data feeds.

Darknet Inferences

The source port 0 event traffic was rendered by more than 1 million probing packets originating from TCP source port 0 destined to the monitored darknet space. It is significant to note that we typically observe, on other days, less than 1000 packets per darknet day originating from TCP source port 0. The traffic originates from 27 unique hosts, 17 distinct countries, 24 diverse Internet Service Providers (ISPs) and from within 25 operational organizations. We refrain from publishing statistics and rigorous information related to the latter due to sensitivity and legal issues. We next investigated some characteristics related to those packets. We noticed that the Time to Live (TTL) values of the packets change with the source IP addresses. This advocates that IP spoofing is less likely or non-existent [154]. The fact that the source IP addresses are not spoofed corroborates that such packets are indeed related to scanning/probing activities (so the actual scanner can essentially receive back the probing results), as it was inferred in Section 4.2.3. We also noticed that the packets arrival rate is slow, averaging around 3 packets per second. This is compliant with slow scanning activities, which are known to be difficult to be detected [168]. Upon a closer investigation of the packet headers, we observed that the majority of the packets either include a TCP header length of 0 or are malformed. Further, almost all the packets contain odd flag combinations (i.e., FIN, SYN, RST, PSH, ACK, FIN, URG, PSH, Reserved). Typically, probing by employing the latter flags is considered as performing 'stealthy' scanning activities as they are engineered to evade firewall detection by only sending a single frame to a TCP port without any TCP handshaking or any additional packet transfers [35].

We proceed by executing the clustering approach in coherence with Section 4.2.3. Recall, that the aim is to disclose traffic clusters that share similar machinery. The output of the EM algorithm is depicted in Fig. 4.11; we omit the output of the k-means since it revealed a similar result. Such outcome provides evidence that the traffic originates from 4 different classes. To further test the validity of this result, we produced a silhouette graph of the EM clusters as shown in Fig. 4.12. Commonly, a silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. A value of 1

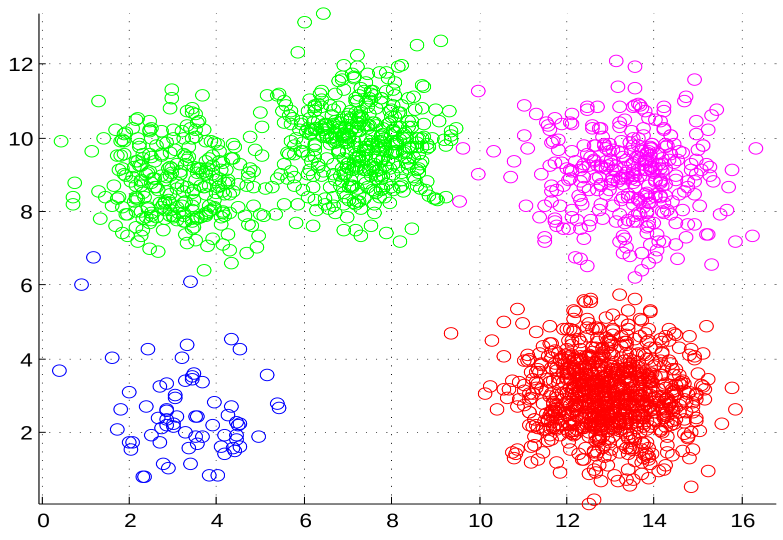


Figure 4.11: Port 0 Event Traffic Clusters

indicates that the points are very distant from neighboring clusters, a value of 0 informs that the points are not distant from other clusters while a negative value indicates that the points are erroneously placed in that cluster. From Fig. 4.12, it is shown that a significant amount of points in all the 4 classes have a large silhouette value, greater than 0.6, indicating that the clusters are separated from neighboring clusters. This provides incentives to validate the quality of the formed EM clusters. By closely investigating each cluster, we determined that the first cluster represents a horizontal scan focused on destination port 0 from a single IP address located in Germany targeting around 800 thousand unique destination addresses. The use of destination port 0 is a frequently employed technique by scanners to fingerprint the operating systems of the targeted destinations in order to tailor future attacks based on that retrieved information. Further, the second cluster discloses a probing campaign targeting more than 60 thousand destination ports and originating from a single Dutch IP address. The latter insight was also observed and confirmed by Cisco[27]. The third cluster renders another horizontal scan that specifically targeted three destination ports, namely, TCP ports 445, 22 and 3389, which respectively represent the Microsoft directory, the secure shell (i.e., ssh) and the remote desktop protocol services.

[27]http://tinyurl.com/kndnj82

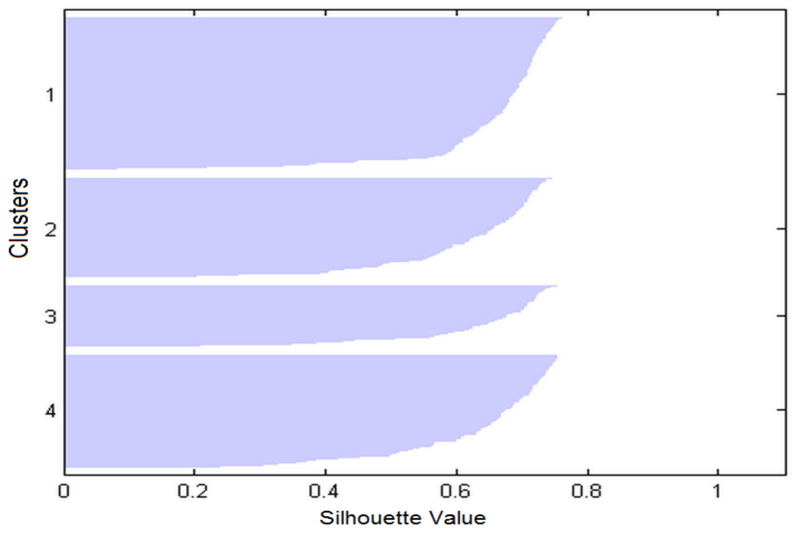


Figure 4.12: A Silhouette Plot of the EM Clusters

The latter are known to suffer from various vulnerabilities and are often exploited[28]. It is worthy to mention that this horizontal scan targeted 9 thousand destinations on port 445, 7 thousand destinations on port 22 and around 5.5 thousand destinations on port 3389. Recall, that all the probing activities in the three previous clusters originate from TCP source port 0. The last minor cluster captured darknet misconfiguration traffic advocating the obtained result of Section 4.2.3.

To further investigate the mechanisms of the probing sources, we invoked the behavioral analytics that were presented in Section 4.2.3. It was revealed that 62% of the probing sources used certain patterns when generating their probing traffic. Concerning the employed probing strategy, it is shown that 57% of the probing sources leveraged a permutation while the remaining adopted a sequential strategy when probing their targets. Of those that employed a permutation, 76% used a uniform permutation while 24% adopted other types of permutations. The majority ($\approx$ 98%) of those that employed a sequential strategy were found to adopt a forward IP-sequential strategy while only 2% adopted a reverse IP-sequential strategy. It is noteworthy to mention that in [169], the researchers dismissed the possible use of this strategy since, as they noted, the strategy

[28]http://tinyurl.com/kkfs6pq,http://tinyurl.com/m5684jo

is difficult to be used to extrapolate certain metrics from especially when dealing with partial probes. Further, the analytics disclosed that $\approx 55\%$ of the sources were probes from bots while the remaining were generated from probing tools. Moreover, it was inferred that all the probing sources were generating probing that is dispersed as opposed to targeting a small set of IPs. To the best of our knowledge, the previously generated inferences represent the first comprehensive empirical results of probing behaviors. In the context of the probing clusters that were disclosed in the previous section, it was shown that clusters one and two were generating random probing traffic as opposed to using a certain pattern, employed a sequential strategy when probing their targets and were found to be leveraging a probing tool. The latter insight proposes that such probing activities have been executed by the same 'initiator/author' since they are adopting the same mechanism and probing characteristics, which could reveal that they both intended to achieve the same desired goal. Conversely, in the third cluster, the majority ($\approx 92\%$) were found to adopt certain patterns when generating their probing traffic, employed a permutation when probing their targets and were inferred to be bots. This suggests that such probing activities could be malware-orchestrated [120] and thus there is a momentous need to explore and investigate their malevolence.

Passive DNS Inferences

We employ the approach of Section 4.2.3 to investigate the maliciousness of all the probing sources. Fig. 4.13 reveals the top 10 probing IP addresses that were shown to host (i.e., resolve to) the most domains. It could be inferred that the most significant number of resolved domains ranges from around 13 thousand domains to peaking at around 110 thousand domains per the probing IPs. We also investigated a subset of those domains that are related to malicious activities. To achieve this task, we correlated the extracted domains with publically available datasets and resources, namely, the Malware Domain List[29], Zeus Tracker[30] and McAfee's siteAdvisor[31]. The outcome is also depicted in Fig. 4.13, which reveals the number of blacklisted domains. The fact that 28% of the probing

[29]http://www.malwaredomainlist.com/

[30]https://zeustracker.abuse.ch/

[31]http://www.siteadvisor.ca/

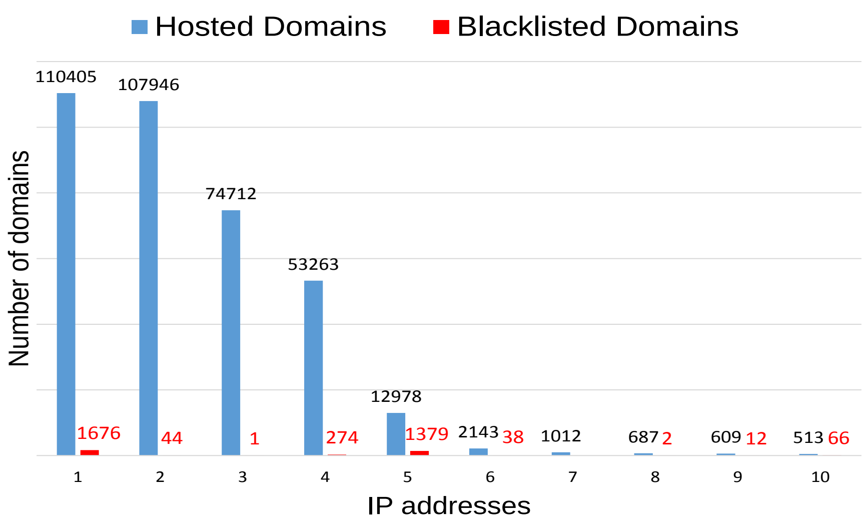


Figure 4.13: Hosted and blacklisted domains of the probing sources

IP addresses were shown to host numerous blacklisted/malicious domains provides an alarming signal that the source port 0 probing traffic could be originating from a malicious entity with malevolent goals. To identify the nature of those blacklisted domains, we leveraged the Web of Trust reputation system[32]. The outcome from such a procedure is depicted in Fig. 4.14. The results demonstrate that more than half of the malicious

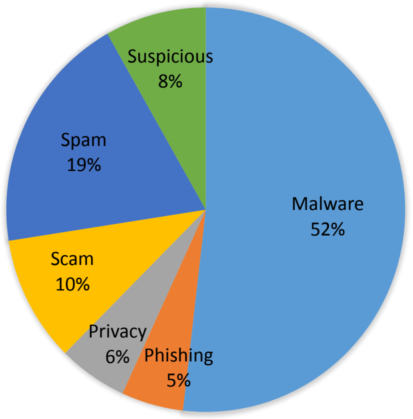


Figure 4.14: The nature of the blacklisted domains

domains could be attributed to malware; this advocates our decision to leverage malware

[32]https://www.mywot.com/

data to investigate the event in question. It was also shown that those malicious domains are blacklisted as they are often used for spamming, phishing and other fraud activities. We further investigated those malicious domains by analyzing their aliveness and intensity as discussed in Section 4.2.3 and exposed in Fig. 4.15. One can notice, on one hand, that some IPs with their corresponding blacklisted resolved domains retain less active days but possess high accessibility, which render them extremely effective in their maliciousness. On the other hand, some domains have a prolonged online presence yet possess low access counts. We envision that such malicious domains are intentionally not intended to be accessed but are rather playing a hosting or a supporting (i.e., back-end) role for other malicious tasks and services.

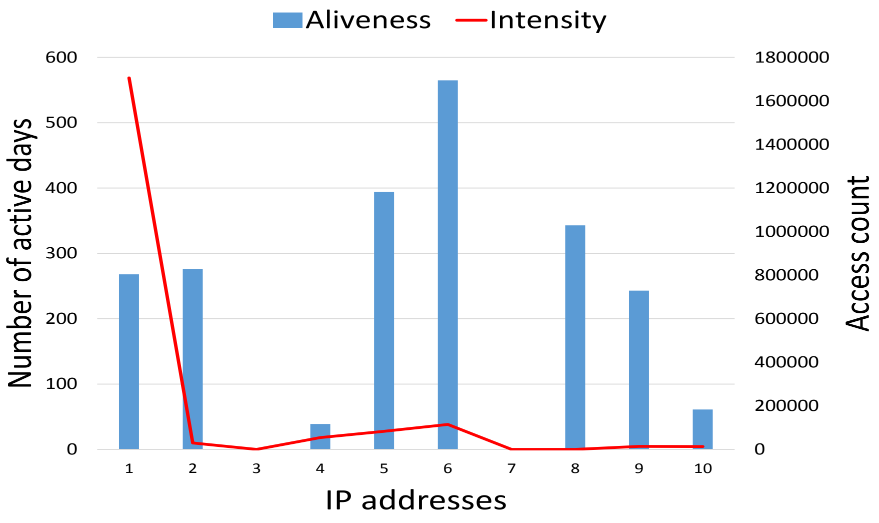


Figure 4.15: Investigating the aliveness and access count of the malicious domains

Malware Inferences

Motivated by the fact that the sources of the 3rd probing cluster, as revealed in Section 4.2.4, were inferred to be bots coupled with the conclusion that more than half of the probing sources resolve to malware-infected domains as demonstrated in Fig. 4.14, in this section, we investigate the source port 0 probing event from the malware feed perspective, in accordance with the proposed approach of Section 4.2.3.

Fig. 4.16 depicts the malware samples and their corresponding number of connections to the probing machines. It could be noted that the malware specimens, namely, Sality and Ngrbot, indeed refer to bot families[33]. Further, Table 4.5 reveals the malware samples that generated TCP source port 0 traffic.

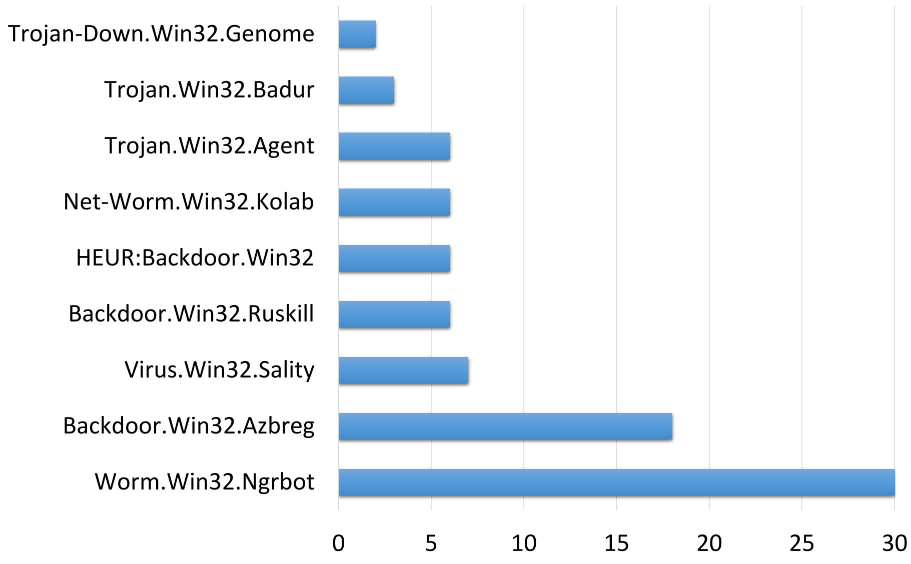


Figure 4.16: Malware and their corresponding number of connections

By correlating Fig. 4.16 and Table 4.5, we can notice that ‘Virus.Win32.Sality’ is the common factor. In other words, such malware sample has infected some of the probing machines and is simultaneously generating TCP source port 0 traffic. It is noteworthy to mention that such sample has been previously attributed to malicious activities; Dainotti et al. [120] had documented a large-scale probing campaign that was able to probe the entire IPv4 address space in 12 days. Interestingly, the authors pinpointed that the malware responsible for such campaign was in fact the Sality malware. Thus, from all the extracted insights and by leveraging the three data feeds, we strongly postulate that ‘Virus.Win32.Sality’ is responsible for part of the TCP source port 0 event.

[33]http://www.symantec.com/connect/blogs/all-one-malware-overview-sality

`Email-Worm.Win32.Mydoom`
`Worm.Win32.AutoRun`
`Virus.Win32.Sality`
`Virus.Win32.Expiro`
`Backdoor.Win32.Xtoober`
`Trojan-Downloader.Win32.Agent`
`Trojan-Dropper.Win32.Small`
`Trojan.Win32.Pincav`
`Trojan.Win32.Jorik`
`Trojan-Downloader.Win32.Delf`
`Trojan-Downloader.Win32.Genome`
`Backdoor.Win32.Gbot`
`Backdoor.Win32.Popwin`
`Email-Worm.Win32.Rays`
`Email-Worm.Win32.Runouce`
`Packed.JS.Agent`
`Trojan-Banker.Win32.Banker`
`Trojan-Downloader.Win32.FlyStudio`
`Backdoor.Win32.Banito`
`Backdoor.Win32.VB`
`HackTool.Win32.Injecter`

Table 4.5: Malware samples generating TCP source port 0 traffic

4.3 Related Work

In this section, we review some literature work related to malware and probing correlation analysis. Further, we briefly highlight two approaches that are adopted in the industry for the purpose of detecting malware-infected machines. Additionally, we pinpoint several proposed methods for inferring worm infections.

Nakao et al. [170] were among the first to exploit the idea of correlating malware and probing activities to detect zero-day attacks. The authors leveraged the nicter framework [171] to study the inter-relations between those two activities. They developed scan profiles by observing the dark space and correlated them with malware profiles that had been generated in a controlled environment. Their work seems limited in a number of points. First, the authors did not validate the accuracy of the extracted probing activities from the dark space. Second, the extracted profiles were based on few textual network and transport-layer features, where the actual correlation engine's mechanism was obscured. Third, their experiments were based on only one malware sample. In contrary, in this work, we first apply a validated statistical approach to accurately extract probing activities from darknet traffic. Second, in a first attempt ever, we correlate probing and malware activities by applying fuzzy hashing and information theoretical based techniques on the entire network traffic that was generated by those activities. Third, our experiments involve around 65 thousand malware samples. Fourth, the aim of this work differs as it is rendered by the capability to provide network security operators, worldwide, with the ability to rapidly and cost-effectively detect their clients' infections, without requiring the providers to maintain an implementation nor provide any aid or disclose any sensitive network related information. In an another closely related work, Song et al. [172] carried out correlation analysis between 10 spamming botnets and malware-infected hosts as observed by honeypots. They disclosed that the majority of the spamming botnets have been infected by at least four different malware. The authors as well developed methods to identify which exact malware type/family has been the cause of contamination. Our work differs from this work as we are correlating probing activities rather than spamming for early infection detection. Further, we are leveraging the darknet space instead of honeypots to extract Internet-scale cyber security intelligence. In a slightly different work, Eto et el. [173] proposed a malware distinction method based on scan patterns by employing spectrum analysis. The authors stated that by observing certain probing patterns, one can recognize the similarities and dissimilarities between different types of malware. The authors noted that the latter could be used as a fingerprint to effectively infer infection. The authors however, did not perform any correlation but rather limited their work to observation and analysis.

The industry has also developed approaches to identify malware compromised machines. For instance, True Internet, one of Thailand's largest ISPs, had adopted a behavioral approach to infer its clients infected machines. Their approach monitors the symptoms of infection, including but not limited to, spamming, excessive P2P usage and Denial of Service (DoS) attempts, and subsequently triggers an alert towards a controller, which then automatically quarantine the client. Although such approaches might be effective, they are typically late in detection, which might cause serious vulnerabilities within the provider. Moreover, they are usually not cost-effective as the provider ought to purchase and maintain other detection systems. Another example would be NetCologne, an ISP and cable provider in Germany, that took a different approach to automate how it deals with subscribers that are infected with malware. NetCologne setup and maintained a honeypot; infected machines often attempt to attack other computers on the same network and hence the honeypot is an accessible target that allows the identification of compromised machines. While this approach seems practical in detecting infected machines, it is neither cost-effective since the provider needs to implement and maintain the honeypot nor it is able to identify the exact malware type/family that had contaminated those machines. Moreover, honeypot evasion approaches are known to be effective [88] and are often adopted by sophisticated malware.

In the context of inferring worm infections, Gu et al. [174] presented the design and implementation of BotHunter, a perimeter monitoring system for real-time detection of Internet malware infections. The core of BotHunter is rendered by a correlation engine that performs alert consolidation and evidence trail gathering for investigating numerous infections. In a slightly different work, Whyte et al. [175] correlated Domain Name System (DNS) queries with outgoing connections from an enterprise network to detect worms propagation attempts. Through empirical evaluations, their proposed approach yielded efficient and accurate results and enabled automatic containment of worm propagation at the network egress points. In an alternative work, schechter et al. [176] presented a hybrid approach to detect scanning worms. Their approach, which is based on sequential hypothesis testing and rate limiting algorithms, demonstrated low false positive rates when evaluated in a real operational network environment. A possible drawback of the previous approaches is that they require to be deployed and maintained at the perimeter of the enterprise network to be effective.

4.4 Summary

On one hand, this chapter presented a new approach to infer malware-infected machines. The approach aims at providing network operators with a cyber security capability to detect their clients' compromised machines in addition to pinpointing the exact malware type that caused their contamination. The approach is efficient as it does not record or analyze the symptoms of infection. Further, it is prompt as it exploits probing activities, which are the very first indications of contamination. Moreover, the proposed approach is cost-effective as it does not require any implementation or maintenance costs at the providers' sides. To accomplish its goals, the proposed approach exploits the dark space to infer and validate Internet-scale probing activities after filtering out misconfiguration traffic using a newly proposed probabilistic model. Consequently, it correlates probing activities with malware samples by uniquely employing various statistical, fuzzy hashing and information theoretical metrics. The approach was empirically evaluated using a significant amount of real darknet and malware samples. The extracted inferences and insights revealed promising accuracy in addition to concurring that the rationale of exploiting probing activities for worldwide early malware infection detection is indeed practically viable.

On the other hand, this chapter also investigated a rare cyber event that was rendered by excessive traffic originating from source port 0. The goal was to shed the light on the inner details of that event to uncover its mechanism and nature of its sources. To achieve those goals, we proposed a multifaceted approach that exploited and correlated a significant amount of real network security data, including, darknet, passive DNS and malware information. The outcome revealed three probing clusters, in which one of them was shown to be originating from infected bots. By analyzing the maliciousness of the probing sources, the approach uncovered that 28% of those are related to malicious domains. Further, by correlating darknet and malware data, the approach was capable of attributing part of the event to the Sality malware. We envision that such approach could be applicable to analyze other cyber events with similar nature.

In the next chapter, we attempt to tackle the problem of inferring and attributing large-scale probing campaigns by solely observing network telescopes.

Chapter 5

Inferring and Attributing Probing Campaigns

In this chapter, we focus on the problem of identifying and attributing large-scale probing campaigns, which render a new era of probing events.

5.1 Background

Lately, there has been a noteworthy shift towards a new phenomenon of probing events which, throughout this section, is referred to as Cyber Scanning Campaign(s) (CSC(s)). These are distinguished from previous probing incidents as (1) the population of the participating bots is several orders of magnitude larger, (2) the target scope is generally the entire Internet Protocol (IP) address space, and (3) the bots adopt well-orchestrated, often botmaster-coordinated, stealth scan strategies that maximize targets' coverage while minimizing redundancy and overlap. Very recently, Dainotti et al. [177] from the Cooperative Association for Internet Data Analysis (CAIDA) presented a pioneering measurement and analysis study of a 12-day Internet-wide CSC targeting VoIP (SIP) servers. In another work [178], the same authors admitted that they have detected the reported SIP CSC including the malware responsible for its actions (i.e., Sality malware) "serendipitously" (i.e., luckily and accidentally) while analyzing a totally unrelated phenomenon. They also stated that since *currently there exist no cyber security capability to discover* *such large-scale probing campaigns*, other similar events targeting diverse Internet and organizational infrastructure are going undetected. In another inquisitive, well executed work, an “anonymous” presented and published online [123] what they dubbed as the “Carna Botnet”. The author exploited poorly protected Internet devices, developed and distributed a custom binary, to generate one of the largest and most comprehensive IPv4 census ever.

The aforementioned two CSC studies differ on various key observations. The work by Dainotti et al. disclosed that the bots were recruited into the probing botnet by means of a new-generation malware while the Carna Botnet was augmented using a custom code binary. Moreover, Dainotti et al. discovered that the bots were coordinated by a botmaster in a Command-and-Control (C&C) infrastructure where the bots used a reverse IP-sequential strategy to perform their probing, while the Carna Botnet was C&C-less and its bots used an interleaving permutation method to scan its targets. Further, the work by Dainotti et al. documented a horizontal scan that targeted world-wide SIP servers, while the Carna Botnet did not focus on one specific service but rather attempted to retrieve any available information that was associated with any host and/or service. Readers that are interested in more details related to the discussed CSCs are kindly referred to [177, 123]. We project that current undetected and unreported CSCs as well as future occurrences could be ominously leveraged to cause drastic Internet-wide and enterprise impacts as precursors of various amplified, debilitating and disrupting cyber attacks including, but not limited to, distributed and reflective denial of service attacks, advanced persistent threats and spamming campaigns.

5.2 CSC-Detector: A System to Infer Large-Scale Probing Campaigns

As previously mentioned in this thesis, we have been receiving, for the past three years, data that amount for a daily total of 12 GB of malicious real darknet traffic from more than 12 countries. This section leverages a portion of such large cyber security data to propose a novel system, CSC-Detector, that aims at identifying Cyber Scanning Campaigns. Recall that the latter define a new phenomenon of probing events that are distinguished by their orchestration (i.e., coordination) patterns. To achieve its aim, CSC-Detector adopts three engines. Its fingerprinting engine exploits a unique observation to extract probing activities from darknet traffic. The system's inference engine employs a set of behavioral analytics to generate numerous significant insights related to the machinery of the probing sources while its analysis engine fine-grains the previously obtained inferences to automatically infer the campaigns. CSC-Detector is empirically evaluated and validated using 240 GB of real darknet data. The outcome discloses 3 recent, previously obscured and unreported large-scale probing campaigns targeting diverse Internet services. Further, one of those inferred campaigns revealed that the sipscan campaign that was initially analyzed by CAIDA is arguably still active, yet operating in a stealthy, very low rate mode. We envision that the proposed system that is tailored towards darknet data, which is frequently, abundantly and effectively used to generate cyber threat intelligence, could be used by network security analysts, emergency response teams and/or observers of cyber events to infer large-scale orchestrated probing campaigns. This would be utilized for early cyber attack warning and notification as well as for simplified analysis and tracking of such events.

5.2.1 Proposed System

This section introduces CSC-Detector as significantly simplified in Figure 5.1. In a nutshell, the system takes as input darknet traffic and outputs detected well-defined CSCs. We refer to the inferred CSCs as being "well-defined" since the probing sources of those CSCs will be identified and correlated through patterns that are automatically generated by the analysis engine in which each of those patterns is defined by specific, clearly identifiable, similar probing behavioral characteristics. This section elaborates on the goals, approaches, and methods that are adopted by each of CSC-Detector's three engines that permits the system to achieve its aim.

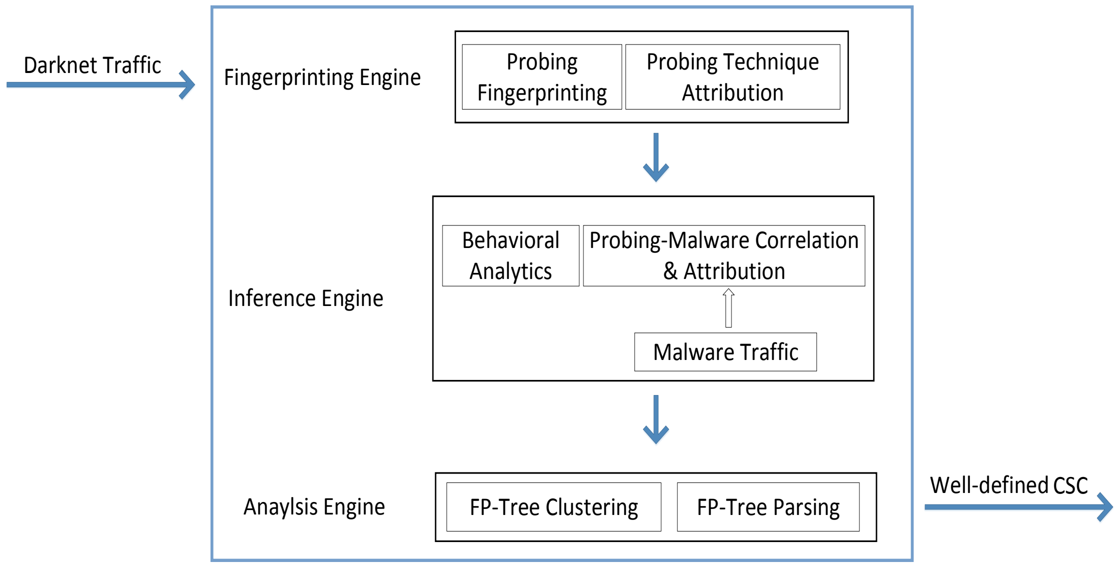


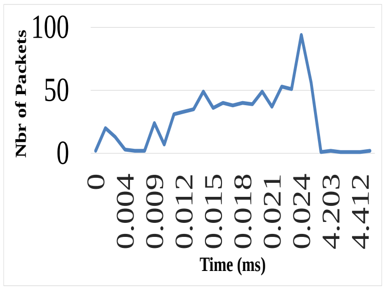
Figure 5.1: CSC-Detector: System Architecture

The Fingerprinting Engine

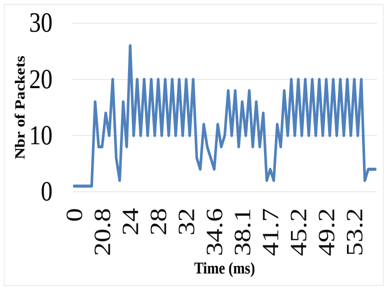
In Section 3.2 of Chapter 3, we have proposed a new approach to fingerprint Internet-scale probing activities. CSC-Detector leverages this approach to extract probing activities from darknet data. Hence, the outcome of the fingerprinting engine are probing activities (i.e., probing sessions) coupled with their inferred probing techniques.

In addition to the validation of the observation of that approach that was performed in Section 3.2 of Chapter 3, we further validate the observation of the approach as follows.

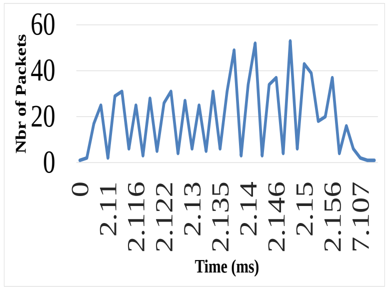
We executed 5 experiments using real and synthetic probing traffic. The Real probing traffic Data set (RD) was retrieved from the probes of the Carna Botnet [123] while the remaining four Synthetic Data sets (SD) were respectively generated using four probing tools, namely, Nmap, Unicornscan, AATools Port Scanner and Superscan, in a lab environment. For RD, the various probing technique sessions were automatically extracted by leveraging Snort's probing engine [100] and confirmed by manual inspection. For SD, the tools were invoked 100 times to generate their supported probing techniques in which a tcpdump sink was used to collect the traffic. Figure 5.2 illustrates the generated distribution of some of the probing techniques using RD. Note that, the signals refer to the



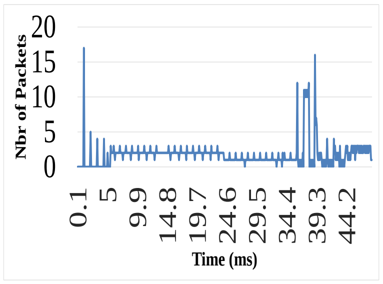
(a) TCP SYN Scan



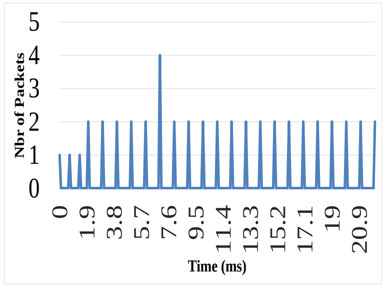
(b) TCP connect() Scan



(c) Xmas Scan



(d) UDP Scan

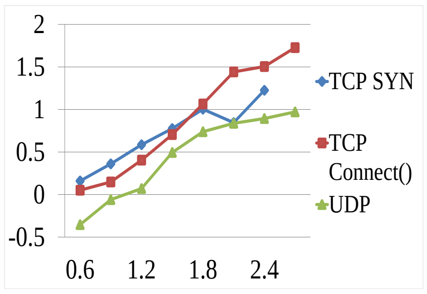


(e) ICMP Scan

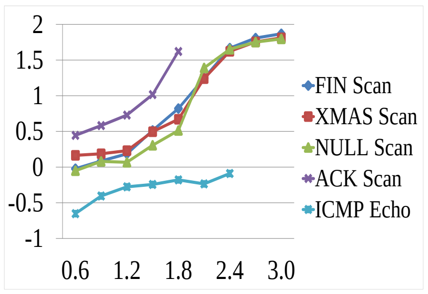
Figure 5.2: Packets' distribution generated by some of the probing techniques using RD

time series distributions (i.e., time in ms Vs number of packets) for each of the probing techniques. Subsequently, DFA was applied on each of the distributions by leveraging the technique in [111] and using 1ms as the bin size. The outcome for RD is shown in Figure 5.3 while the complete set of experiments are summarized in Table 5.1, where the results of SD represent the average of all the executions. It is worthy to mention that such list of probing techniques is comprehensive [35]; any probing traffic in any environment should have been generated by one of the techniques on the list.

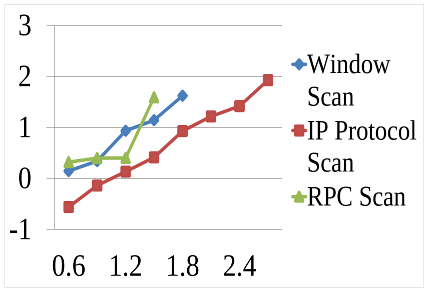
From Table 5.1 and the information relating the scaling exponent α to the correlation status, we can produce Table 5.2 that validates the observation that is exploited by CSC-Detector's fingerprinting engine; a number of probing techniques indeed demonstrate a similar and a distinct temporal correlation and similarity when generating their



(a)



(b)



(c)

Figure 5.3: DFA application on the probing techniques using RD

corresponding probing traffic. Using RD, for the TCP Connect() probing technique, the observation was confirmed with 92% confidence while for the RPC probing technique, in the same experiment, it was established for 95%. On average, for all the probing techniques, the observation was respectively validated by 97% and 93% for the RD and SD experiments.

The Inference Engine

CSC-Detector's inference engine leverages (1) the behavioral analytics that were previously proposed in Section 4.2.3 of Chapter 4 to capture the machinery of the probing sources and (2) the malware correlation approach of Section 4.1 for attribution purposes.

The Analysis Engine

Previous works [179] suggested that coordinated bots within a botnet campaign probe their targets in a similar fashion. CSC-Detector's analysis engine exploits this idea by

Probing Technique	RD: α	SD: α
TCP SYN	0.57	0.74
TCP Connect()	0.87	0.69
FIN	0.31	0.24
Xmas	0.30	0.27
Null	0.37	0.41
UDP	0.66	0.58
IP Protocol	1.13	1.22
ACK	0.44	0.29
ICMP	0.25	0.29
Window	1.24	1.18
RPC	1.31	1.29

Table 5.1: Probing Techniques & DFA output

automatically building patterns that consist of similar probing behavioral characteristics. The latter aims at identifying and correlating the probing sources into well-defined campaigns. The engine considers the criteria (i.e., features) that are summarized in Table 5.3. Inferred from the fingerprinting engine, the employed probing technique is a significant behavior; [177] demonstrated that all the CSC bots used UDP scanning. Further, the analysis engine consumes all the previously inferred probing machinery that is derived from the inference engine. It considers the fashion of the generated probing traffic; whether random or not, and which specific pattern has been adopted if not random; which probing strategy has been employed; whether the probing source is a probing tool or a bot; whether the probing is targeted or dispersed; whether or not the probing sources demonstrate any signs of malware infection and which specific malware type/family/variant if they do. Additionally, bots/sources in a CSC are postulated to possess similar probing rates and ratios of destination overlaps; we consider a 90% confidence interval as being similar. Finally, the analysis engine considers the target port as a significant criterion; [177] disclosed that all the CSC bots used port 5060.

Correlation Status	Probing Techniques
Anti-Correlated	FIN Probing Xmas Probing Null Probing ACK Probing ICMP Echo Probing
Correlated	TCP SYN Probing TCP Connect() Probing UDP Probing
Non-Stationary	IP Protocol Probing Window Probing RPC Probing

Table 5.2: Techniques & Correlation Statuses

To automatically infer orchestrated probing campaigns, the analysis engine leverages all the previously extracted inferences and insights related to the probing sessions/sources to build and parse a Frequent Pattern (FP) tree [180]. In such a tree, each node after the root represents a feature extracted from the probing sessions, which is shared by the sub-trees beneath. Each path in the tree represents sets of features that co-occur in the sessions, in non-increasing order of frequency of occurrences. Thus, two sessions that have several frequent features in common and are different just on infrequent features will share a common path on the tree. The analysis engine also employs the FP tree based mining method, FP-growth [180], for mining the complete set of generated frequent patterns. As an outcome, the generated patterns represent frequent and similar probing behavioral characteristics that correlate the probing sources into well-defined campaigns.

We should emphasize that such an approach that is employed by CSC-Detector's analysis engine possess the following advantages. First, the generated patterns are *not defined apriori;* they are naturally and automatically detected. This permits CSC-Detector

Employed probing technique
Probing traffic (Random Vs Patterns)
Employed pattern
Adopted probing strategy
Nature of probing source
Type of probing (Targeted Vs Dispersed)
Signs of malware infection
Exact malware type/variant
Probing rate
Ratio of destination overlaps
Target port

Table 5.3: Criteria adopted by the Analysis Engine

to infer novel probing campaigns, without requiring previous knowledge about their specifications. The latter is a crucial feature, especially with the continuous evolution of probing campaigns and their employed strategies. Second, the FP-Tree not only provides the capability for the system to correlate the probing sources into campaigns, but also semantically describes how the probing sessions have been constructed. Third, by engineering parsing algorithms that traverse the FP-Tree in various ways, the system can infer horizontal probing campaigns, similar to the CSC in [177], as well as campaigns that do not focus on one port but rather probe multiple targets on various ports, similar to the CSC in [123]. Fourth, from a performance perceptive, the employed approach is scalable since probing sessions are not compared pairwise, which could lead to a quadratic complexity [180]. In fact, the cost of the algorithm is the cost of inserting probing session features in the FP-Tree, which is linear.

5.2.2 System Evaluation

5.2.3 Implementation

We implemented the proposed CSC-Detector as a prototype in Java. We utilized the jNet-Pcap SDK for packets/sessions processing. For all the statistical distances and tests as well as entropy and Poisson fitting, we employed their MATLAB implementations and included them in Java using the MATLAB Builder JA. We also used ssdeep, a fuzzy hashing implementation written in C, and wrapped it unto java using SWIG. We as well leveraged a python implementation of the FP-growth algorithm. We adopted a MongoDB database to save the instances of the probing sessions coupled with their generated inferences.

Performance Evaluation

We setup CSC-Detector on a lab machine equipped with an Intel Core i7 CPU with eight cores and 12GB of RAM. We profiled CSC-Detector's performance in terms of execution/processing time, CPU utilization and memory consumption using 7 days of logged darknet data. Using this hardware setup, on average, CSC-Detector was able to process one day of darknet data ($\approx$ 12 GB) in about 3.5 hours. The latter task is from the time the system first receives the darknet pcap file to the time all the probing sessions and their inferences are stored in the database. During these 3.5 hours, throughout the 7 days, CSC-Detector's CPU utilization recorded an average of 70.4%, a minimum of 45% and a maximum of 96%. Further, its memory consumption measured an average of 975 MB, a minimum of 700 MB and a maximum of 1.15 GB. The fluctuations in both metrics can be attributed to the varying system's idle state and the variable size of the processed logged darknet pcap file. Since the system allows an arbitrary number of worker CPUs, we noticed that the primary bottleneck was reading and writing to disc. Further, the fingerprinting and inference engines highly depend on numerous statistical tests and techniques that are currently implemented in MATLAB; a native fast programming language implementation of the latter can be expected to boost the processing performance. Concerning the analysis engine, it required around 12 seconds to automatically build the FP-tree using 240 thousand probing instances. The latter information strongly advocate that CSC-Detector

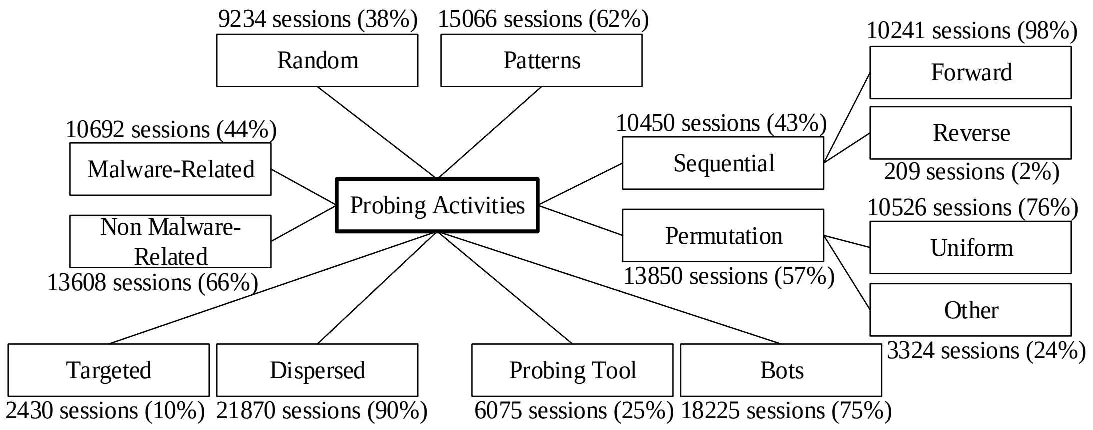


Figure 5.4: The outcome of the probing behavioral analytics

is practically viable in a real-world environment.

Empirical Evaluation

We evaluate CSC-Detector using one month of darknet data (240 GB) that was collected during the recent duration of April 1$^{\text{st}}$ to April 30$^{\text{th}}$, 2014. For each day in the dataset, we extracted 1,000 sessions for a total of 30,000 sessions to experiment with. Note that, the 30,000 sessions are selected to be generated from unique sources.

Evaluating the inference engine

We proceed by invoking the behavioral analytics that are employed by CSC-Detector's inference engine on the 24,300 probing sessions[1]. The outcome is summarized in Figure 5.4. It is revealed that 62% of the probing sources used certain patterns when generating their probing traffic. Applying the CLUES clustering algorithm on the generated patterns resulted in 12 specific non-overlapping Poisson λ intervals. We associate the probing sources to those intervals. Concerning the employed probing strategy, it is shown that 57% of the probing sources leveraged a permutation while the remaining adopted a sequential strategy when probing their targets. Of those that employed a permutation, 76% used a uniform permutation while 24% adopted other types of permutations. The majority ($\approx$ 98%) of those that employed a sequential strategy were found to adopt a

[1]As extracted from the fingerprinting engine

forward IP-sequential strategy while only 2% adopted a reverse IP-sequential strategy. It is noteworthy to mention that Dainotti et al. [177] reported that the SIP CSC indeed used a reverse IP-sequential strategy; the authors deemed that as rare and novel. Other studies such as [169] dismissed the possible use of this strategy since, as they noted, the strategy is difficult to be used to extrapolate certain metrics from especially when dealing with partial probes. Further, the inference engine disclosed that $\approx 75\%$ of the sources were probes from bots while only 15% were generated from probing tools. Moreover, it was inferred that 90% of the sources were generating probing that is dispersed while only 10% of the sources were generating probing targeted towards a small set of IPs; the average relative uncertainty of 21,870 probing sources was shown to be > 0.72. Last but not least, the inference engine employed the approach of Section 11 to investigate any signs of malware infection in the probing sessions. The output demonstrated that 44% of the probing sessions exhibited such signs. The top 5 malware contamination that caused the probing were attributed to the following: `Trojan-FakeAV.Win32.Agent.cwa`, `Virus-Win32.Sality.s`, `Trojan-Win32.Jorik.Shakblades.foc`, `Trojan-Downloader.Win32.KiayksayRen.b` and `Worm-Win32.VBNA.b`. It is noteworthy to pinpoint that the latter inference that aims at coupling probing and malware by only analyzing their generated traffic has never been attempted before. The obtained results also expressed that the average rate of all the probing sources was around 80 packets/sec, the Domain Name System (DNS) on port 53 and the Session Initiation Protocol (SIP) on port 5060 were the most probed UDP services, while TCP ports 80 and 1433 that respectively represent the HTTP and Microsoft (MS)-SQL services were the most targeted TCP services. To the best of our knowledge, the above generated inferences represent the first comprehensive empirical results of probing behaviors and characteristics. Although we are unable to validate every single inference due to the lack of its ground truth, the 3 CSCs that will be uncovered by the analysis engine (that consumes these generated inferences) will be validated using a publically available data source, namely, DShield data, advocating the relative accuracy of such insights.

	CSC1	**CSC2**	**CSC3**
Employed probing technique:	UDP	TCP SYN	ACK
Probing traffic:	Random	Random	Pattern
Employed pattern:	Null	Null	[19.17-21.23]
Adopted probing strategy:	Reverse IP-sequential	Uniform Permutation	Forward IP-Sequential
Nature of probing source:	Bot	Bot	Tool
Type of probing:	Dispersed	Dispersed	Targeted
Signs of malware infection:	Yes	Yes	No
Exact malware type/variant:	Virus.Win32.Sality.bh	Trojan.Win32.Jorik	Null
Probing rate (in pps):	12	118	77
Target port:	5060	80	1433
Number of Probing Bots/Sources:	846	817	438

Table 5.4: The automatically inferred patterns capturing three large-scale orchestrated probing campaigns

Evaluating the analysis engine

We proceed by invoking the analysis engine. Recall that this engine consumes the generated insights from the fingerprinting and inference engines to automatically build patterns that consist of similar probing sources behavioral characteristics. For the sake of this work, we have devised a simplistic parsing algorithm which automatically builds patterns from the FP-tree that aims at capturing orchestrated probing events that probe horizontally; probe all IPs by focusing on specific ports.

Inferred Campaigns: CSC-Detector automatically inferred the patterns that are summarized in Table 5.4, which respectively captured three different CSCs. The first pattern permitted the detection, identification and correlation of 846 unique probing bots into a well-defined orchestrated probing event that targeted the VoIP (SIP) service. It is shown that this event, that was initiated on the 17th of April, 2014, adopted UDP scanning, probed around 65% of the monitored dark space (i.e., 300,000 dark IPs) where all its bots did not follow a certain pattern when generating their probing traffic. Further, the results demonstrate that the bots employed a reverse IP-sequential probing strategy when probing their targets. Moreover, the malware responsible for this event was shown to be attributed to the Sality malware. The second pattern successfully captured and correlated 817 unique probing bots to form a campaign that targeted the HTTP (Web) service. This campaign, that first occurred on April 3$^{\text{rd}}$, leveraged the TCP SYN scanning technique and did not adopt a pattern when generating its probing traffic. Moreover, all its bots were found to uniformly permute the dark IP space. Additionally, the malware responsible for this event was shown to be attributed to the Jorik trojan. The third campaign that was identified by the third pattern on April 16$^{\text{th}}$ was unique by targeting the MS-SQL service, leveraging ACK scanning, employing a clearly identifiable pattern, focused on a small set of targeted probing IPs, where its 438 probing sources leveraged a scanning tool and a forward IP-sequential strategy. Note that we also generate supplementary material related to the above inferred CSCs including geo-location information per real sources, organizations, Internet Service Providers (ISPs), cities, regions and countries. However, we refrain from publishing those due to sensitivity/legal issues.

Validation: Since currently there exist no cyber security capability to discover such large-scale orchestrated probing campaigns [178], we are unable to directly compare the inferred CSCs with other systems or approaches. However, in an attempt to validate our results, we resort to publicly accessible data that is provided by DShield/Internet Storm Center (ISC). ISC data comprises of millions of intrusion detection log entries gathered daily from sensors covering more than 500 thousand IP addresses in over 50 countries.

From such data, we extract Figures 5.5, 5.6 and 5.7 which respectively depict probe counts generating probing activities towards UDP port 5060 and TCP ports 80 and 1433 during the month of April, 2014. Figure 5.5 indeed reveals a significant peak on the 17$^{\text{th}}$ of April consisting of an increasing number of probe counts targeting the SIP service; this is the same day where the first orchestrated probing campaign, that was previously inferred by CSC-Detector, was detected to be targeting the SIP service. Further, Figure 5.6, according to DShield's data, shows a significant number of probes targeting the Web service on April 3$^{\text{rd}}$; this is as well coherent with the second CSC that was inferred by CSC-Detector, that targeted the same service on the same day. Similarly, Figure 5.7

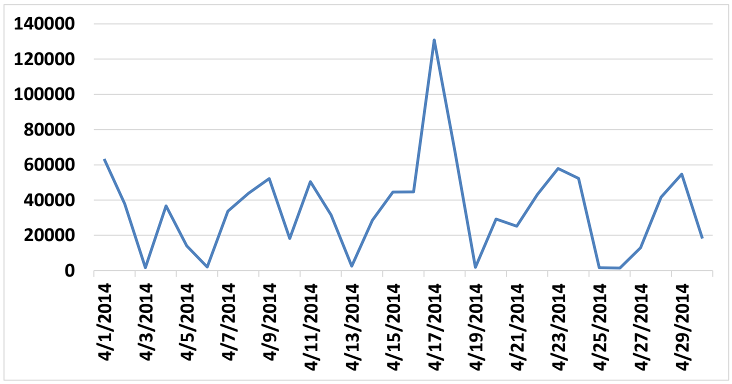


Figure 5.5: Probe counts extracted from DShield/ISC data (April 2014)

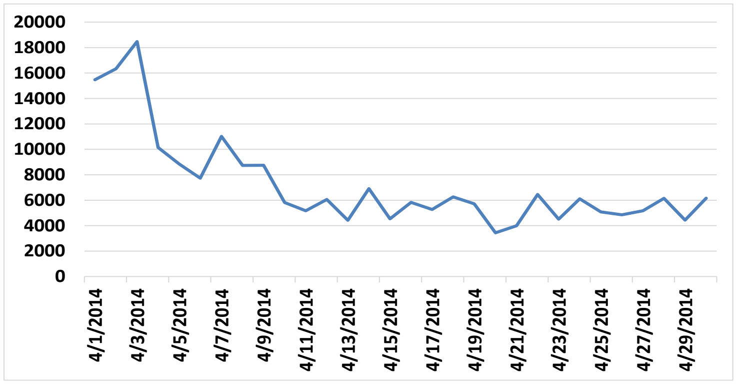


Figure 5.6: Probe counts extracted from DShield/ISC data (April 2014)

demonstrates a peak of probing packets targeting the MS-SQL service on the same day (i.e., April 16th) when the third CSC was detected to be launching its probing activities towards the MS-SQL service. Additionally, the target scope of those CSCs, also retrieved from DShield's data on those particular days, recorded an unprecedented numbers reaching millions of targets (compared with hundreds on other days). The latter information strongly advocate that the proposed system was indeed accurately successful in inferring those unusual large-scale events. Note that, those inferred probing campaigns went undetected and unreported in the cyber security community until now.

Moreover, in an attempt to further validate our results, we have performed two additional experiments.

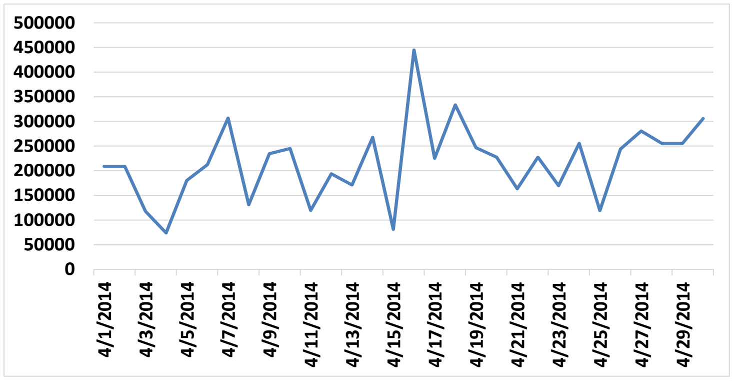


Figure 5.7: Probe counts extracted from DShield/ISC data (April 2014)

First experiment: We extracted all the 2101 source IP addresses that were pinpointed to be part of the three inferred large-scale probing campaigns. We fed those IP addresses to third party publically available data sources provided by the online services, ThreatStop[2], MxLookup[3], brightcloud[4] and ReputationAuthority[5]. The latter cyber security data repositories provide information on Internet-scale incidents (i.e., scanning, malware, spamming, etc.) per IP address. In particular, ThreatStop indexes full, present and historical dshield data. Since the three probing campaigns were inferred in April 2014, we verified the existence of the inferred source IP addresses that belong to the campaigns against those reported in the online services in that specific time frame. The outcome of such experiment, from all the four sources, demonstrated that (1) 98% of the IP addresses of the campaigns were indeed found and flagged as malicious in those repositories and (2) around 91% of those IP addresses were specifically flagged as being involved in scanning activities.

Second experiment: In an attempt to further validate the phenomenon by correlating activities of source IPs, we extracted the packet features[6] generated by the IP

[2]http://www.threatstop.com/
[3]http://mxtoolbox.com/
[4]http://www.brightcloud.com/
[5]http://www.reputationauthority.org/
[6]Adopted from [166], where they have been shown to produce distinguishing characteristics when applied

addresses for each of those three campaigns. Subsequently, the k-means data clustering technique was applied on such features. The outcome is illustrated in Figure 5.8. The figure advocates the creation of three unique clusters, representing the three inferred campaigns.

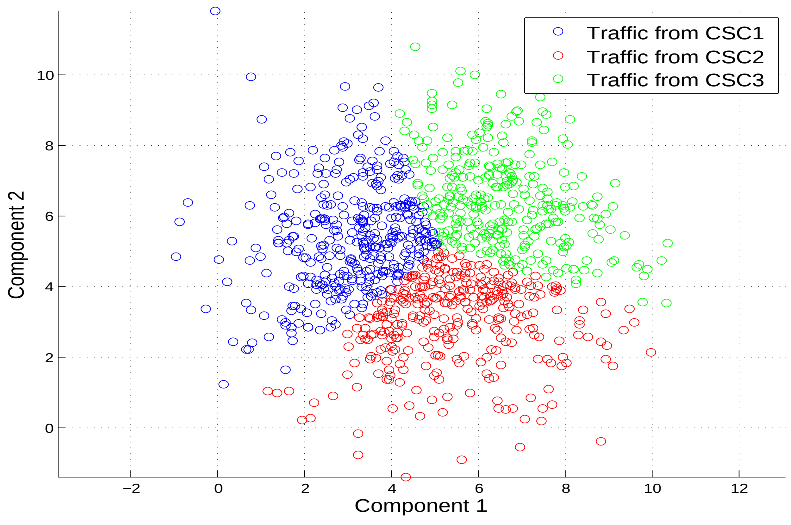


Figure 5.8: K-means output

Thus, on one hand, the first experiment validates that the source IP addresses of the inferred campaigns are indeed malicious where the majority of them were found to be involved in probing activities in April 2014. On the other hand, the second experiment demonstrates that the inferred campaigns are indeed well constructed where their IP addresses coupled with their generated probing traffic are strongly correlated as revealed by the formed independent clusters.

Discussion: The aforementioned inferred CSC1 is indeed interesting; as elaborated at the beginning of this section, in [177], CAIDA presented an analysis study of an orchestrated probing campaign that targeted VoIP (SIP) servers. According to CAIDA, the event occurred from January 31st, 2011 till February 12th, 2012. CSC1 that the proposed system was able to *automatically* infer possesses the exact characteristics of that CAIDA

on network data

event. They both were attributed to the Sality malware, generated UDP scanning, targeted SIP servers on port 5060, and used a reverse IP-sequential probing strategy. The latter behavior is predominantly stimulating because, as previously mentioned, that strategy is known to be extremely under-employed [169]. However, one distinguishing feature between those two events is that the probing bots of the campaign that the proposed system was able to infer has considerably lower probing rate than those of CAIDA's event; on average, the bots of the inferred campaign recorded 12 packets per second (pps) while those of CAIDA measured around 60 pps. Such information (1) arguably proves that CAIDA's event is indeed still active, yet operating in a stealthy, very low rate mode in an attempt to achieve its reconnaissance task without being detected or (2) a new instance of the same orchestrated event commanded by the same C&C took place in April, 2014 without any cyber security body reporting it. In either cases, we find the latter information motivating and to a certain degree bewildering. Thus, we aim in the near future work to track that event to verify and elaborate on its inner details. It is worthy to mention that in the month of April 2014, DShield, the media or other sources did not announce a campaign that CSC-Detector was not able to infer. Moreover, although we did not have the opportunity to operate and experiment with CSC-Detector for a long period of time, however, we were able to extract some points that we recently have had the chance to observe:

- We are often faced with the case in which the probing bot sends two probing packets to the same destination but with two different destination port numbers. For example, if the intention of the scanner/campaign is to probe for the SIP service, the scanner might send two probing packets towards ports 80 and 5060. This was also observed and concurred in [177]. CSC-Detector is frequently identifying those bots as belonging to two different CSCs. One solution to compensate this issue would be to permit multiple port number assignments in the target IP criteria through the analysis engine.

- Most of the probing bots that the system is identifying as belonging to the same CSC are often very geographically dispersed. This suggests that it is very hard if not impossible to mitigate their probes or their possible subsequent cyber attacks by blocking them based on their location, ISP or Autonomous System Number (ASN).

- Typically, we would expect that probing bots within the same correlated CSC to possess similar probing rates, as they are coordinated by the same botmaster in a C&C infrastructure that characteristically adopt a unified probing strategy. For the three CSCs that the system was able to infer, the latter was found to be **false** (using a 90% confidence interval); perhaps this implies that the same botmaster is requiring different bot groups within the same campaign to adopt different probing strategies in an attempt to avoid being attributed to the same campaign. In general, we deem this behavior as stimulating, uncommon and somehow confusing to understand and interpret.

5.3 Time Series Interpolation and Prediction For Inferring Orchestrated Probing Campaigns

This section also aims at inferring probing campaigns by investigating traffic destined towards network telescopes. The objective is to provide a more systematic methodology to infer, in a prompt manner, whether or not the perceived probing packets belong to an orchestrated campaign. Additionally, the methodology could be easily leveraged to generate network traffic signatures to facilitate capturing incoming packets as belonging to the same inferred campaign. Indeed, this would be utilized for early cyber attack warning and notification as well as for simplified analysis and tracking of such events. To realize such goals, the proposed approach models such challenging task as a problem of interpolating and predicting time series with missing values. By initially employing trigonometric interpolation and subsequently executing state space modeling in conjunction with a time-varying window algorithm, the proposed approach is able to pinpoint orchestrated probing campaigns by only monitoring few orchestrated flows. After verifying the accuracy of the individually-employed theoretical based methods using simulated data, we consequently empirically evaluate the effectiveness of the complete proposed model using 330 GB of real telescope data. By comparing the outcome with a previously validated work, the results indeed demonstrate the promptness and accuracy of the proposed model.

Thus, motivated by the imperative requirement to infer such probing campaigns, we frame this section's contribution as follows:

- Proposing an approach that models the complex problem of inferring probing campaigns as the task of interpolating and predicting time series in the presence of missing values. The model allows the pinpointing of orchestrated campaigns by observing just few probing packets, rendering it very prompt. Further, by only keeping a record of the probing time series, the model is efficient and lightweight in terms of memory and processing requirements, which makes it applicable to be implemented in real-time on operational data streams. It is also noteworthy to mention that the model could be easily invoked to capture newly incoming packets that belong to the same inferred campaign by generating simple yet effective network traffic signatures. To the best of our knowledge, this is the first systematic model that is specifically tailored towards the goal of inferring probing campaigns by observing network telescopes.

- Employing time series interpolation and prediction approaches based on trigonometric and state space modeling techniques, namely, the discrete Fourier transform and kalman filter, and evaluating their accuracy against other approaches using simulated data. The latter aims at authenticating the choice of those techniques as core components of the proposed model.

- Evaluating the promptness and accuracy of the proposed model using 330 GB of real telescope data in addition to validating the models' outcome and advantages by comparing it with a previous work.

5.3.1 Proposed Model

In this section, we formulate the problem, present the proposed approach and evaluate the employed theoretical-based techniques.

Problem Formulation

By observing a set of network telescopes $\{nt_1, nt_2, \cdots, nt_n\}$, one can infer unidirectional probing flows $\{pf_1, pf_2, \cdots, pf_n\}$ as previously illustrated in Figure 2.28. A probing flow refers to a series of scanning packets, generated by employing a specific probing technique [35], $r = \{s, d, \Delta t\}$, that is originating from a single Internet source s and targeting one particular darknet IP address d within a duration Δt. From a telescope perspective, such flows seem to appear autonomous and totally independent from other probing flows. However, as stated in Section 5.1, there exists evidence of the emergence of probing campaigns that aim at executing coordinated scanning activities.

Thus, the problem in-hand could be clarified using the following questions:

1. By solely observing unidirectional probes pf_n arriving at the network telescope nt_n, how can we infer orchestrated probing flows $\varphi(pf_n)$?

2. How can we distinguish orchestrated probing flows $\varphi(pf_n)$ from other flows that are also targeting the telescope? More specifically, given a possible set of $\varphi(pf_n)$, how can we achieve $\varphi(pf_n) - \epsilon$, where ϵ could refer to noise (i.e., independent flows pf_n), to produce confirmed $\varphi(pf_n)$.

3. How can we design an approach that achieves the above but at the same time is efficient (i.e., demands little processing and memory requirements) and prompt. The latter feature is crucial as we would like to possess the capability to flag an orchestrated campaign by observing as few coordinated probing flows as possible. Indeed, this would be utilized for early cyber attack warning, notification and thus mitigation as well as for simplified analysis and tracking of such events, by monitoring and capturing signatures of those campaigns in newly incoming flows.

In what follows, we elaborate on the proposed approach, as holistically illustrated in Figure 5.9, that aims at answering the above questions.

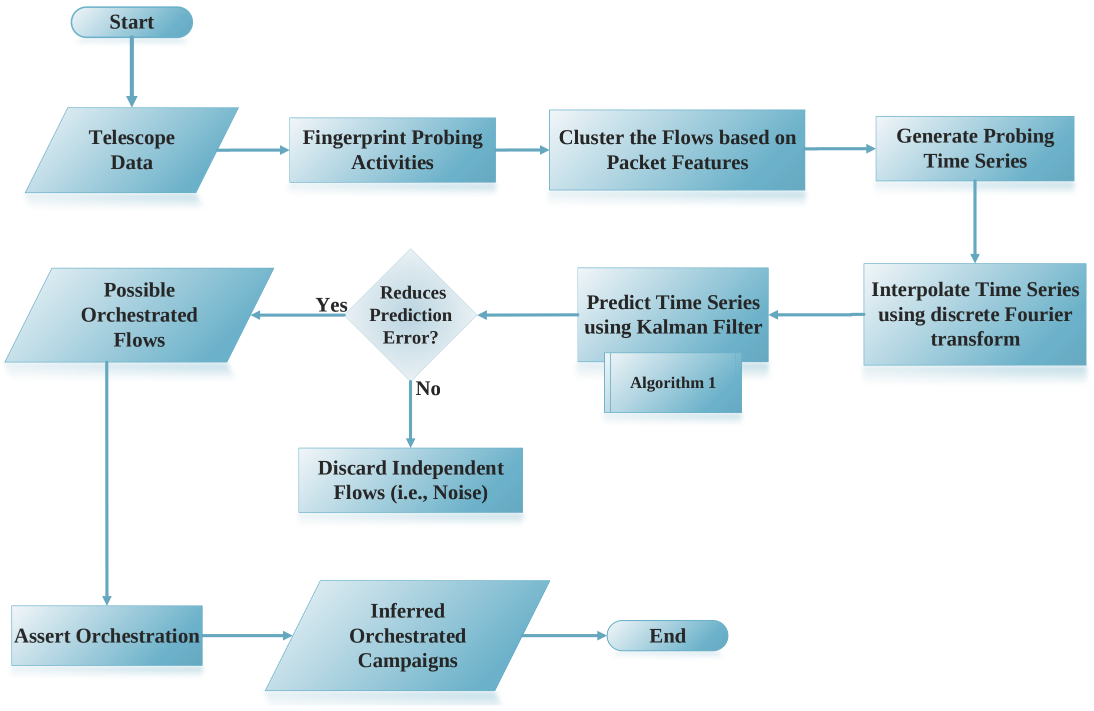

Start
Telescope Data
Fingerprint Probing Activities
Cluster the Flows based on Packet Features
Generate Probing Time Series
Interpolate Time Series using discrete Fourier transform
Predict Time Series using Kalman Filter
Algorithm 1
Reduces Prediction Error?
Yes
No
Possible Orchestrated Flows
Discard Independent Flows (i.e., Noise)
Assert Orchestration
Inferred Orchestrated Campaigns
End

Figure 5.9: A holistic view of the proposed approach

Proposed Approach

In this work, we uniquely model the task of inferring orchestrated probing campaigns as the problem of interpolating and predicting time series in the presence of missing values. The core rationale behind that approach stems from the idea that if the probing flow time series demonstrates positive predictability features, thus identifying certain probing patterns, then it might be part of an orchestrated event. To realize such rationale, the proposed approach (1) fingerprints independent probing flows as perceived by the telescope, (2) clusters and builds the probing flow time series, (3) interpolates the time series for data completion purposes, (4) predicts the time series in order to infer orchestrated flows and eliminate independent (i.e., non-orchestrated) ones and (5) confirms orchestrated flows by capturing and verifying the logic of the usage of the probing sources. In the sequel, we elaborate on the latter five steps.

Fingerprinting Independent Flows

In Section 3.2 of Chapter 3, we have proposed a new approach to fingerprint Internet-scale probing activities. In this work, and to successfully extract independent probing flows as perceived by the network telescope, we adopt and leverage the previously proposed approach.

Flow Clustering & Time Series Generation

Probing flow packets targeting the telescope have the following 8-tuple form

$$\langle t, src_{ip}, dst_{ip}, src_p, dst_p, prot, ttl, flags\rangle$$

representing the timestamp, the source and destination IP addresses and ports, the transport protocol used, the time-to-live (ttl) value and the packet flags. From recent probing campaign incidents [177, 123], it was observed that orchestrated probes possess similar values for dst_p, $prot$ and $flags$. Thus, we first amalgamate all the probing flows into different clusters sharing similar values for those packet features. Additionally, based on those reported events, it was demonstrated that a particular campaign has been generated by the same type of operating system; in [177], it was inferred that 97% of the orchestrated probing flows were originating from windows machines while in [123], it was revealed that all the flows were generated from Unix environments. Thus, it is desirable that we further cluster the previous groups by the same originating operating system. To achieve this, we investigate the *ttl* value of the probing packets as perceived by the network telescope. According to [181], most modern operating systems use only a few selected initial *ttl* values, particularly, 30, 64, 128, and 255. It is evident that most of these initial *ttl* values are far apart. Moreover, since Internet traces have shown that few Internet hosts are apart by more than 30 hops [182, 183], which is also confirmed by our own recent observations, one can determine the initial *ttl* value of a probing packet that is received at the telescope by selecting the smallest initial value in the set that is larger than its final *ttl*. For example, if the observed final *ttl* value is 112, the initial *ttl* value would be 128. Thus, based on this heuristical approach, we further refine the clusters by subdividing the probing flows according to similar *ttl* values. Recall that the latter aims at further clustering the groups based on similar originating operating system. It is important to note that probing sources/attackers can use, in theory, certain tools that can explicitly modify the *ttl* value in an attempt to evade being correctly clustered. For example, traceroute[7] is a common application which can set the IP *ttl* to any random number, independent of the operating systems' kernel's default. However, a probing source who overrides IP *ttl* to avoid detection should enforce a stable mapping of IP source to IP *ttl*, in order to prevent us from seeing a noisy IP TTL originating from its IP address, and thus subsequently flagging his IP source as *ttl*-spoofing. In practice, in our experiments, we did not notice any *ttl*-spoofing, where all the inferred *ttl* values were from those 4 default categories. Please recall, in a nutshell, that the output of the above are independent probing flows clustered based on similar dst_p, $prot$, $flags$ and ttl values.

For each of the above independent probing flows within those clusters, we generate their corresponding time series of the form

$$\langle t, dst_{ip} \rangle$$

Note that due to the simplicity of such time series, the approach is able to generate around 200 thousand of those in under 5 seconds. Further, since the time series is composed of just two packet features, namely, the timestamp and the destination telescope IP address, the storage requirements are minimal; less than 300 MB to save those 200 thousand time series. It is also noteworthy to mention that such storage requirement will not augment by time, since the approach discards the previously extracted time series after processing.

Time Series Interpolation

The above time series could be referred to as a time series with missing values in which some of its rows are not captured. This is because (1) the network telescope only covers a portion of the Internet space and thus is not able to perceive all the probes at all times

[7]http://linux.die.net/man/8/traceroute

and (2) some probing packets arriving at the telescope could be distorted and thus are filtered out before the generation of the time series. Indeed, without a complete view of the probing time series, it is very difficult, if not impossible, to devise approaches to infer orchestration. To deal with this issue, we resort to time series interpolation. This refers to numerical analysis techniques that aim at constructing new data points from and within a range of a discrete set of known data points. Although there exists several interpolation techniques in the literature [184], in this work, we employ and tailor, for accuracy[8] and efficiency reasons, trigonometric interpolation, namely, the discrete Fourier transform (dFt) [185, 186]. In the sequel, we detail the latter and demonstrate how it is used for effective interpolation.

Fourier analysis is the theory about representing general functions by means of trigonometric functions [185]. This decomposition procedure is called Fourier transform [186].

Defined on a discrete function of infinite length x_n, the discrete-time Fourier transform (DTFT) reads

$$X(\widetilde{w}) = \sum_{n=-\infty}^{\infty} x_n \cdot e^{-i\widetilde{w}n} \tag{5.1}$$

with $\widetilde{w} \in [-\pi, \pi)$ since DTFT is 2π-periodic [187]. Operating on x_n defined over all integers $n \in \mathbb{Z}$, the frequency $\widetilde{w}$ of the DTFT is always continuous. In other words, DTFT maps an infinite series of complex numbers into a finite interval. From an engineering perspective, the function values x_n can be called samples as well. We also notice the inversion

$$x_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\widetilde{w}) \cdot e^{i\widetilde{w}n} \, \mathrm{d}\,\widetilde{w}\,. \tag{5.2}$$

For a sequence $x_n, n = 0 \cdots N-1$, that is of a finite length, the discrete Fourier

[8]See Section 5.3.2

transform (DFT) is defined as

$$X(k) = \sum_{n=0}^{N-1} x_n \cdot e^{\frac{-2\pi ikn}{N}}. \tag{5.3}$$

The inverse Fourier transform (IFT) is given by

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(w) \cdot e^{iwt} \ \ \mathrm{d}w \tag{5.4}$$

and the inverse discrete Fourier transform (IDFT) is

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{\frac{2\pi ikn}{N}}. \tag{5.5}$$

By (5.2) and (5.4), we can see that DTFT provides an approximation for the FT defined in continuous time.

Furthermore, by Eulers notation [188]

$$e^{\pi i} = cos\pi + isin\pi = -1$$

and further defining

$$w_N = e^{\frac{\pi i}{N}}, \tag{5.6}$$

we may rewrite DFT and IDFT as

$$X(k) = \sum_{n=0}^{N-1} x_n \cdot w_N^{-2kn} \tag{5.7}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot w_N^{2kn} \tag{5.8}$$

Going back to our interpolation problem, given a set of $(n + 1)$ data points in the probing time series, dFt provides an interpolating function that is composed of finite sum of cosines and sine that connects all $(n + 1)$ data points, including any missing values in between. However, for the previous function to be directly applicable to the extracted probing time series, we modify the dst_{ip} packet feature in the time series from the typical IPv4 CIDR notation [189] (i.e., w.x.y.z/13) to a discrete numeric by using a simple developed mechanism.

Time Series Prediction

After interpolating the probing time series within the clusters, the next step would be to verify if such series indicate any signs of predictability. If confirmed, this would provide evidence that the probing flows indeed possess some orchestration logic. To achieve that task, we tailor and employ a recursive optimal stochastic estimator, namely, the kalman filter [190]. Although we have selected to leverage this specific theoretical-based technique because of various reasons, including, (1) its superior results in practice due to optimality and structure [191, 192, 193], (2) its convenient form that permits online real-time processing [194, 195], (3) its ease to formulate and implement and (4) its efficiency when calculating the measurement equations and the error covariances [196], the main reason, however, would be due to its significant applicability to the problem in-hand; the kalman filter infers parameters of interest from indirect, inaccurate and uncertain observations. This is coherent with our problem since the time series under predictability verification has been interpolated and thus might contain some uncertain records. The inner workings of the kalman filter are highlighted next.

The kalman filter could be simplified by

$$X_k = K_k \cdot Z_k + (1 - K_k) \cdot X_{k-1} \tag{5.9}$$

where X_k is the current state, K_k is the kalman gain, Z_k is the current measured value and X_{k-1} is the previous estimation. Note that Z_k refers to an observed or interpolated record in the probing time series. Informally, we can say that the kalman filter identifies the most optimum averaging factor for each consequent state. Additionally, it remembers some information related to the past states.

The first step of the kalman filter is rendered by evaluating the probing time series signal by using two linear stochastic equations, namely,

$$x_k = x_{k-1} + w_k \tag{5.10}$$

and

$$z_k = x_k + v_k \tag{5.11}$$

where w_k and v_k are measurement noises.[9]

The second step of the kalman filter is composed of establishing and subsequently iterating through the prediction (i.e., time update) and correction (i.e., measurement update) equations. On one side, the time update formulation is given by

$$x_k^- = x_{k-1} \tag{5.12}$$

and

$$P_k^- = P_{k-1} \tag{5.13}$$

where (5.12) projects the state ahead while (5.13) projects the error covariance ahead. On the other hand, the measurement update formulation is given by

$$K_k = \frac{P_k^-}{P_k^- + R}, \tag{5.14}$$

$$x_k = x_k^- + K_k(z_k - x_k^-) \tag{5.15}$$

and

$$P_k = (1 - K_k) \cdot P_k^- \tag{5.16}$$

[9]For simplicity, these values are extracted from a Gaussian distribution fitted into the received darknet data. Note that this assumption is not problematic since the kalman filtering algorithm will converge into correct estimations, even if the Gaussian noise parameters are roughly estimated [192, 197].

where (5.14) computes the kalman gain, (5.15) updates the estimate via z_k and (5.16) updates the error covariance.

Going back to our probing time series, recall that the aim is to infer whether or not the probes demonstrate any signs of predictability and thus orchestration. We exploit the kalman filtering error covariance, P_k, as previously computed, in conjunction with a time-varying window algorithm to achieve the latter. The pseudocode of the algorithm is presented next.

Algorithm 6 operates on the basis of two time windows. The first is used to load the probing time series within each cluster into volatile memory for processing while the second is used to compare the probing time series. The comparison is based on kalman's error covariance; the algorithm flags those flows that increase the error as demonstrating non-predictability while inferring orchestrated ones by monitoring a decrease in the error metric. From a complexity perceptive, the algorithm requires $\mathrm{O}(m)$ space complexity where m is the size of the probing time series within each cluster and $\mathrm{O}(m + n + p)$ time complexity where n is the size of the probing time series related to the comparison window and p is the required time for the kalman filter to process the time series within q fixed iterations. It is noteworthy to mention that in this work, we have chosen a q value of 10 iterations for the kalman filter in order to measure and compare the error covariance. Note that the choice of the value per say is not an issue; rather, its systematic application for all the probing time series for effective and accurate comparison. Further, as mentioned in Section 5.3.1, recall that since the probing time series is very simple, there is no compelling reason (i.e., from a memory or processing requirements) to optimize the values of the window sizes. In practice, the algorithm can process and decide upon a time series with 1 thousand records in less than 10 seconds and require, on average, around 10 MB in volatile storage.

Algorithm 6: A time-varying window algorithm to distinguish between orchestrated and independent probing flows by leveraging kalman's error covariance

```
Data: Probing Clusters, $PC$,
Probing Time Series, $PTS$
Result: List of Orchestrated Flows, $OF$
for $PTS$ in $PC$ do
    $Processing_{Window}, PW$=5;
    $Comparison_{Window}, CW$=1;
    while $PTS$ in $PW$ do
        while $PTS$ in $CW$ do
            $v1$=kalman.execute($PTS$);
            $CW$++;
            $v2$=kalman.execute($PTS$);
            if $v1 > v2$ then
                flag(($CW$- -).FirstElement();
                $OF$.add($CW$- -).OtherElements();
            end
            if $v1 < v2$ then
                flag($CW$).LastElement());
                $OF$.add($CW$).OtherElements();
            end
            $CW$++;
        end
    end
    $PW$+=5;
end
```

Orchestration Confirmation

The previous sections elaborated an approach that aims at inferring orchestrated probing flows as perceived by the network telescope. Specifically, the outcome of the previous section could be simplified by Figure 5.10. In this Figure, assume that the probing sources

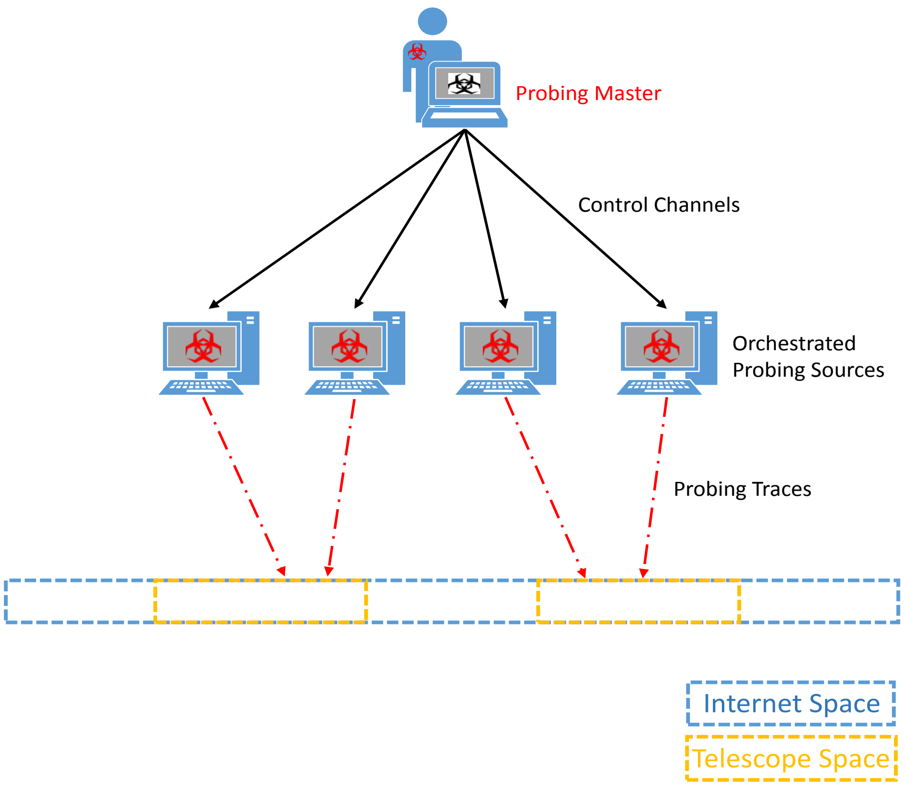


Figure 5.10: A simplified illustration of orchestrated probing flows

were inferred to be orchestrated by employing the proposed approach. Recall, that the approach only analyzed the probing traces which targeted the telescope to conclude that inference. However, to further assert such orchestration, we extend the approach by exploiting the logic of distribution or usage of the probing sources as instructed by the probing master. The latter aims at providing fortified evidence of the coordination of the probing sources. Note that, we use the generic name 'probing master' to either refer to a probing botmaster where the probing sources are deemed to be infected by a malware and thus are referred to as bots similar to the analyzed probing event in [177] or to a more relaxed naming convention which is rendered by any 'malicious manager' that is coordinating the probing activities in some matter as was illustrated in [123]. Typically, to infer the logic of the usage of the probing sources, one need access to the control channels (see Figure 5.10). However, recall that we only have access to the network telescope information. Although we could leverage the observation that coordinated bots in a unified botnet will relatively generate the same amount of traffic as was noted in [198], we argue that this approach is not quite accurate and does not apply in our case. First, since our work exploits network telescopes to infer probes orchestration, it is highly probable that we might not observe all the probes from the probing sources. Thus, even though the orchestrated probing sources might send the same amount of probes, the network telescope will not perceive such equal amount. Second, the probing master could attempt to hide the orchestration behavior by purposely instructing the probing sources to probe with very different number of probes. In either cases, that observation that relies on the generated load by the coordinated sources will fail to assert orchestration. Therefore, in this work, we rely on another behavior that is easily and more accurately captured by a network telescope to achieve that goal. From [177, 123], it was demonstrated that the probing master instructed its sources to initiate the orchestrated probing activities in a constant manner. In other words, given a queue of targets, the coordinated probing sources would, in a constant repetitive manner, select a subset of the destinations and pledge their probes towards them during a dispersed period of time. We leverage such information and devise a simple mechanism to capture the activity closeness of the orchestrated sources. This step could be considered as a post-processing step that is performed on the output of the previous section. Within a given $Processing_{Window}, PW$ (recall Algorithm 6), we record all the orchestrated probing sources as inferred by the proposed approach. For each of such probing sources, we build a simplistic data structure that holds the indices in which that specific probing source was active within PW. Figure 5.11 illustrates such structure where $K \in$ {0,1}.

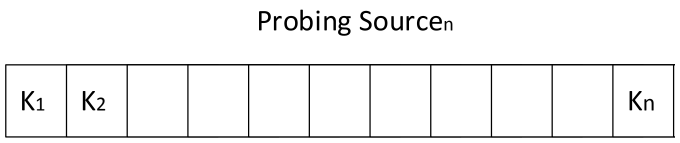


Figure 5.11: A structure that captures when the orchestrated probing sources are active

Such binary structure indeed captures when the orchestrated probing sources were active during PW, where the indices reflect a specific time metric (i.e., ms or sec). Thus, the problem in-hand is now reformulated as a pattern matching task executed on binary structures. Indeed, the aim is to utilize the assembled binary structures to infer closeness of activity between the orchestrated sources. To achieve the latter, we implement a highly efficient pattern matching algorithm that is specifically tailored to operate on binary strings. The code is based on that of [199]. The latter algorithm was found to be extremely efficient, when compared with other literature techniques, even when the binary structures are large [199] as it might occur in our case. To determine a suitable percentage of activity closeness that we can use to assert the orchestration of the probing sources, we perform the following empirical experiment. We extract 200 unique and orchestrated probing sources coupled with their real traffic flows from the Carna Botnet [123]. For all the probing sources, we generate their corresponding structure as previously described in this section and illustrated in Figure 5.11. For any two of those orchestrated sources, we execute the binary matching algorithm and record the percentage of similarity. The outcome is illustrated in Figure 5.12.

The result indicates that between any two orchestrated flows, the average of their activity closeness, using the proposed approach in this section, is around 57.4%. In this work, we employ the minimum value derived from this experiment, which is equal to 39%, as a relaxed lower bound to assert the orchestration of the probing sources. Moreover, *only for labeling purposes*, we assert that a cluster of inferred and confirmed orchestrated sources will indeed define a campaign when the number of such sources reaches a minimum of 100 unique machines; on one hand, the probing campaigns in [177, 123] demonstrated a large number of involved orchestrated sources reaching thousands and millions, while, on the other hand, other probing studies [200] revealed a more focused coordinated probing events involving just few hundreds of orchestrated sources. Thus, we deem the 100 machines threshold as a suitable parameter to flag a group of orchestrated sources as a campaign. Note that, this relatively low threshold will permit the model, at very early stages, to pinpoint orchestrated campaigns by observing just 100 coordinated and confirmed probing flows. Indeed, this would be utilized for early cyber attack warning and notification as

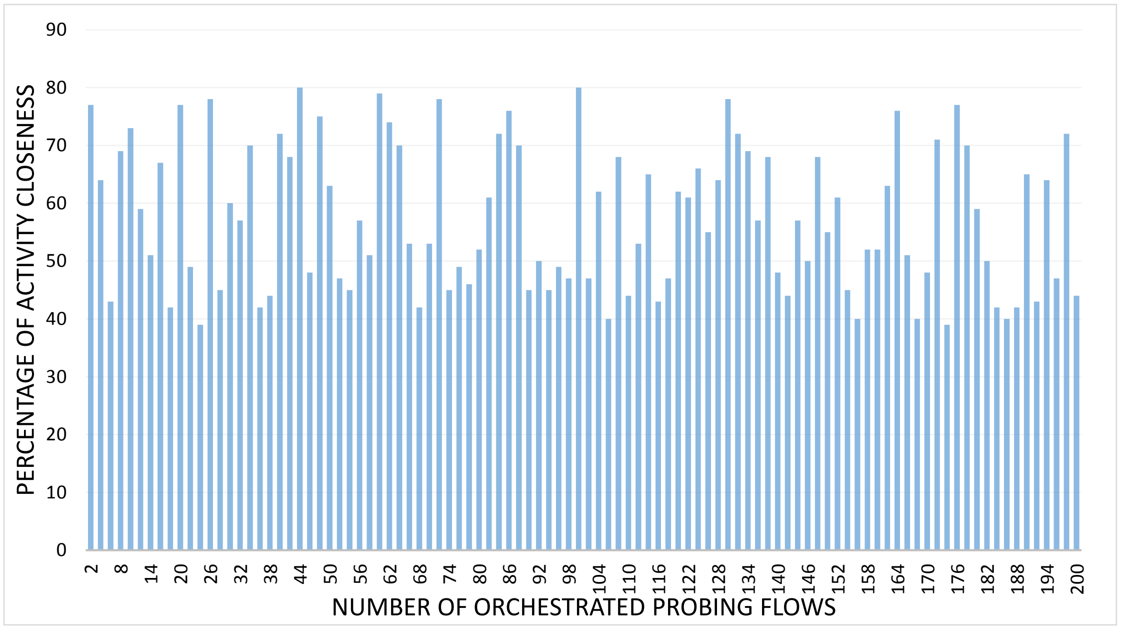


Figure 5.12: An empirical experiment to determine a suitable lower bound for the activity closeness of the orchestrated probing sources

well as for simplified and prompt analysis and tracking of such events.

5.3.2 Evaluation of Techniques

To interpolate and predict the probing time series, we have initially employed and tailored trigonometric interpolation based on discrete Fourier transform and subsequently executed state space modeling related to kalman filtering. This section compares the accuracy of those two approaches with other common approaches. This aims at verifying the choice of those two theoretical-based techniques as core components of the proposed model in solving the specific problem in-hand. To archive that, we resort to a simulation experiment.

The experiment is rendered by a simulation executed using Opnet Modeler[10]. The simulation is comprised of a probing source and 100 probing destinations/targets. The source and the destinations are represented using commodity machines. The probing source is instructed to generate three types of probing towards the targets, namely, TCP

[10]http://tinyurl.com/nclm3gp

SYN, UDP and ACK scanning, for a duration of 10 minutes. The latter ones are typically common types of probing activities [35].

We generate the probing time series targeting 20 destinations in coherence with Section 5.3.1. In case of interpolation, the aim is to verify the accuracy of the discrete Fourier transform (dFt) in two cases, namely, when the series is missing only one value and in the case where it is missing multiple values. In both cases, we compare the dFT in relation to the Least Squares Approximation (LSA) method and the Cubic Spline (CS) method. The latter are classical and thus significantly used approaches in the field of time series interpolation [201, 202, 203]. On one hand, the LSA approach attempts to minimize the sum of the squares of the errors for a polynomial of a given degree by applying the least-square principle. Such approach is coherent with the maximum-likelihood principle of statistics [204]; if the measurement errors are normally distributed and if the standard deviation is constant for all the data, the line determined by minimizing the sum of squares can be shown to have values of slope and intercept which have maximum likelihood of occurrence. On the other hand, the CS method fits a smooth curve to the known data, using cross-validation [205] between each pair of adjacent points, to initially set the degree of smoothing and subsequently estimate the missing observation by the value of the spline. Note that to implement the LSA and CS techniques, we employed two open source Java libraries[11] while we experimented with a slightly modified version of the FFTW[12] library to implement the dFT approach.

Table 5.5 and Figure 5.13 summarize the interpolated time series in the case of one missing value where t is in ms and x(t) is the targeted destination. In all the three missing value cases, we can note that the dFt approach provided the closest estimates to the actual value while the polynomial LSA approach was the least accurate. In the second and third missing value cases, the CS approach performed almost as accurate as the dfT method with the LSA approach showing much improvement. From this experiment, we

[11]http://tinyurl.com/mptwqg4, http://tinyurl.com/k59sbrm

[12]http://www.fftw.org/

can conclude that the dFT, overall, outperformed the other methods in terms of accuracy, especially when the number of values preceding the missing interpolated value is limited; this is crucial feature that is much needed when working with darknet probing flows.

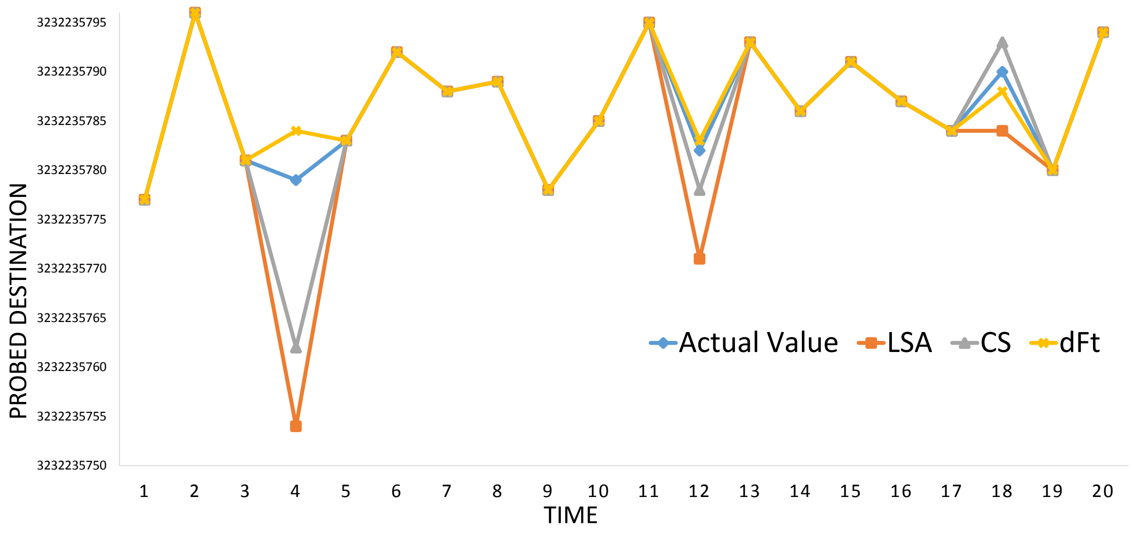


Figure 5.13: Summary of the estimates of the probing time series interpolation in the presence of one missing value

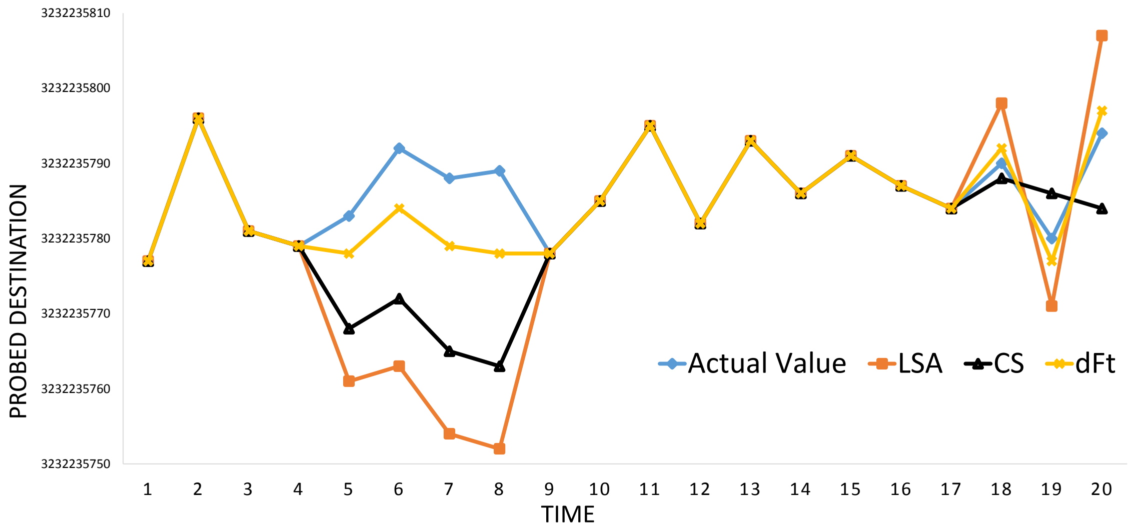


Figure 5.14: Summary of the estimates of the probing time series interpolation in the presence of multiple missing values

Time t	x(t) Actual Value	x(t)	x(t) using LSA	x(t) using CS	x(t) using dFT
1	3232235777	3232235777	3232235777	3232235777	3232235777
2	3232235796	3232235796	3232235796	3232235796	3232235796
3	3232235781	3232235781	3232235781	3232235781	3232235781
4	3232235779	**MISSING**	**3232235754**	**3232235762**	**3232235784**
5	3232235783	3232235783	3232235783	3232235783	3232235783
6	3232235792	3232235792	3232235792	3232235792	3232235792
7	3232235788	3232235788	3232235788	3232235788	3232235788
8	3232235789	3232235789	3232235789	3232235789	3232235789
9	3232235778	3232235778	3232235778	3232235778	3232235778
10	3232235785	3232235785	3232235785	3232235785	3232235785
11	3232235795	3232235795	3232235795	3232235795	3232235795
12	3232235782	**MISSING**	**3232235771**	**3232235778**	**3232235783**
13	3232235793	3232235793	3232235793	3232235793	3232235793
14	3232235786	3232235786	3232235786	3232235786	3232235786
15	3232235791	3232235791	3232235791	3232235791	3232235791
16	3232235787	3232235787	3232235787	3232235787	3232235787
17	3232235784	3232235784	3232235784	3232235784	3232235784
18	3232235790	**MISSING**	**3232235784**	**3232235793**	**3232235788**
19	3232235780	3232235780	3232235780	3232235780	3232235780
20	3232235794	3232235794	3232235794	3232235794	3232235794

Table 5.5: Verifying the accuracy of the discrete Fourier transform as applied to probing time series interpolation in the presence of one missing value

Time t	x(t) Actual Value	x(t)	x(t) using LSA	x(t) using CS	x(t) using dFT
1	3232235777	3232235777	3232235777	3232235777	3232235777
2	3232235796	3232235796	3232235796	3232235796	3232235796
3	3232235781	3232235781	3232235781	3232235781	3232235781
4	3232235779	3232235779	3232235779	3232235779	3232235779
5	3232235783	**MISSING**	**3232235761**	**3232235768**	**3232235778**
6	3232235792	**MISSING**	**3232235763**	**3232235772**	**3232235784**
7	3232235788	**MISSING**	**3232235754**	**3232235765**	**3232235779**
8	3232235789	**MISSING**	**3232235752**	**3232235763**	**3232235778**
9	3232235778	3232235778	3232235778	3232235778	3232235778
10	3232235785	3232235785	3232235785	3232235785	3232235785
11	3232235795	3232235795	3232235795	3232235795	3232235795
12	3232235782	3232235782	3232235782	3232235782	3232235782
13	3232235793	3232235793	3232235793	3232235793	3232235793
14	3232235786	3232235786	3232235786	3232235786	3232235786
15	3232235791	3232235791	3232235791	3232235791	3232235791
16	3232235787	3232235787	3232235787	3232235787	3232235787
17	3232235784	3232235784	3232235784	3232235784	3232235784
18	3232235790	**MISSING**	**3232235798**	**3232235788**	**3232235792**
19	3232235780	**MISSING**	**3232235771**	**3232235786**	**3232235777**
20	3232235794	**MISSING**	**3232235807**	**3232235784**	**3232235797**

Table 5.6: Verifying the accuracy of the discrete Fourier transform as applied to probing time series interpolation in the presence of multiple missing values

To further assess how the employed dFt approach will perform, we used the same simulation setup to execute another experiment which is rendered by the presence of multiple missing values. Table 5.6 and Figure 5.14 summarize the output of such experiment. The results pinpoint another imperative advantage of the dFt technique; while the accuracy of the other approaches very rapidly degrade when faced with multiple subsequent missing values (i.e., especially at the last position in the missing value series), the dFt approach still maintains an acceptable accuracy. It can also be noted that, similar to the previous experiment, the dFt approach outperformed the other techniques in all the individual missing value cases.

We proceed by investigating the accuracy of the kalman filter as modeled and applied to the previously interpolated probing time series. Recall, as per Section 5.3.1, we have leveraged this iterative optimal stochastic estimator coupled with a time-varying window algorithm to infer and distinguish between orchestrated and independent probing flows as perceived by the network telescope. To assess its accuracy, we compared it with typical time series prediction techniques, including the Exponential Smoothing (ES) and the Moving Average (MA) [206]. We further computed the mean squared error [207], a metric which is commonly exploited to measure and compare the forecast error when dealing with time series prediction. The latter is defined by

$$\frac{1}{n}\sum_{i=1}^{n}(\bar{Y_i} - Y_i)^2 \tag{5.17}$$

where $\bar{Y_i}$ is a vector of n predictions and Y_i is a vector of n true values.

Figure 5.15 illustrates the output of the experiment. By computing the mean squared error for the three approaches, it was revealed that, on average, the ES technique suffered from a 6.8% error rate, the MA technique recorded a 4% error rate while the kalman filter agonized only 1.52% error rate. The latter indeed demonstrate the accuracy of the kalman filter. Note that such a low error rate could be further enhanced by permitting the filter to further iterate to allow it to generate close to fully optimized estimates.

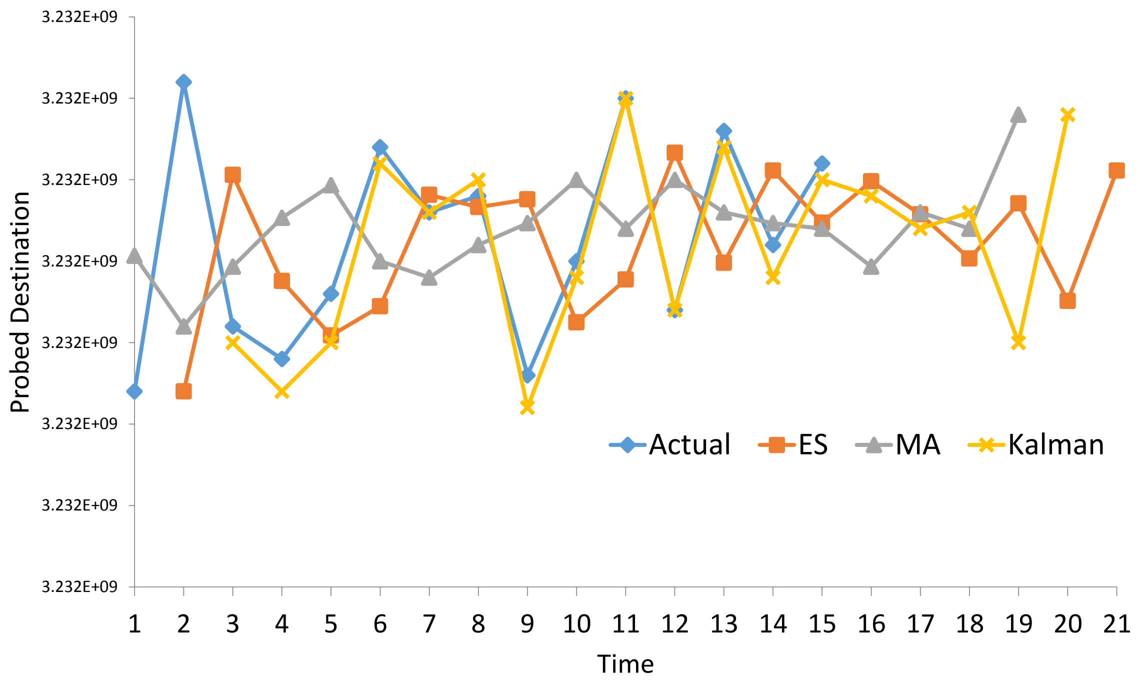


Figure 5.15: Summary of the prediction estimates of the interpolated probing time series

In a nutshell, the above three experiments clearly reveal that the dFt approach is more accurate than the LSA and CS techniques when interpolating time series with missing values. Further, the experiments strongly demonstrate that the kalman filter can generate high accurate prediction estimates in few iterations in comparison with other common forecasting techniques. This relatively authenticates the choice of both of these theoretical-based modeling techniques as core components of the proposed model.

5.3.3 Empirical Evaluation

In this section, we leverage 330 GB of darknet data extracted from the month of April 2014 to validate the promptness and accuracy of the proposed model.

Consistent with Section 5.3.1, we infer 30 thousand unique independent probing flows. Further, coherent with Section 5.3.1, around 35% of the latter were successfully clustered into 5 groups. The details of those clusters are summarized in Table 5.7.

It can be noted that the majority of the probing flows have been generated by Windows

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Probed Destination Port	5060	23	80	1433	443
Transport Protocol	UDP	TCP	TCP	TCP	TCP
Probing Flags	UDP	TCP SYN	TCP SYN	TCP ACK	TCP FIN
TTL	128	128	64	64	128
Number of Probing Flows	2698	3067	913	2983	939

Table 5.7: Summary of the clustered probing flows

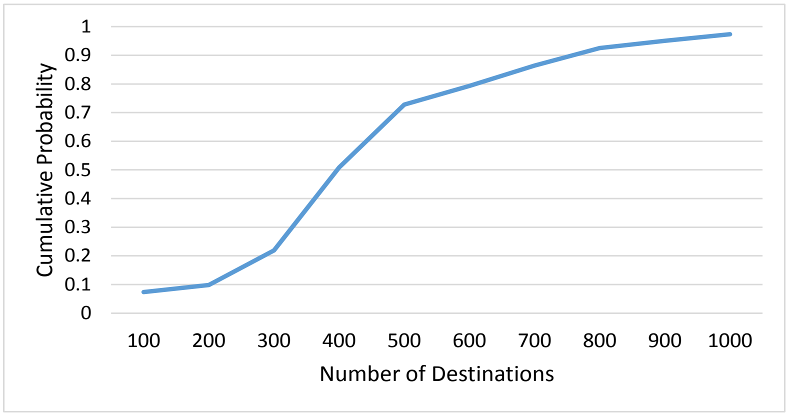


Figure 5.16: CDF of Destinations in Cluster 1

machines (i.e., TTL = 128[13]). For each probing flow within each of the 5 clusters, we generate their probing time series in accordance with Section 5.3.1. Figures 5.16 to 5.20 show the CDF of the probed destinations within each of those clusters.

From the Figures, we can infer that cluster 2 is the most dispersed; 50% of the sources probed more than 1300 destinations. Further, we can extract that clusters 3 and 5 are more focused in which most of the their sources probed a small number of destinations.

We proceed by executing the discrete-time Fourier transform interpolation approach on the time series within each of the 5 clusters as indicated in Section 5.3.1. Figure 5.21

[13]http://tinyurl.com/9u9rugw

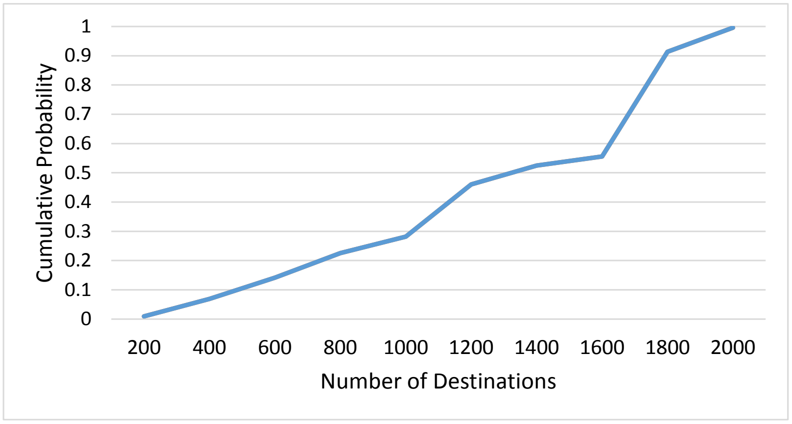


Figure 5.17: CDF of Destinations in Cluster 2

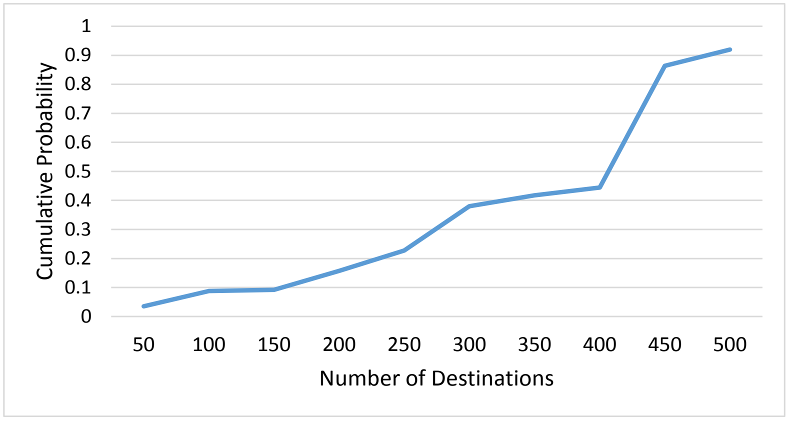


Figure 5.18: CDF of Destinations in Cluster 3

corroborates the fact that the probing flows of cluster 2 are indeed dispersed; not only they target a significant amount of destinations as was inferred from Figure 5.17, but also most of them were not captured by our leveraged /13 darkspace, which is indicated by the large percentage of interpolated multiple missing value time series.

Subsequently, consistent with Section 5.3.1, we invoke the kalman filter coupled with Algorithm 6 to distinguish between orchestrated and non-orchestrated flows. Table 5.8 summarizes one execution of the filter while Figure 5.22 represents the convergence of the kalman filter algorithm in that execution. One can pinpoint that with just 10 iterations, the filter is successfully able to converge; this advocates our chosen value q of 10 iterations

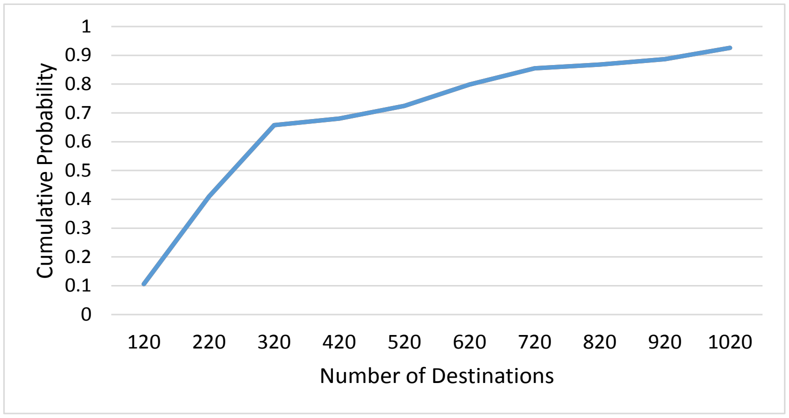


Figure 5.19: CDF of Destinations in Cluster 4

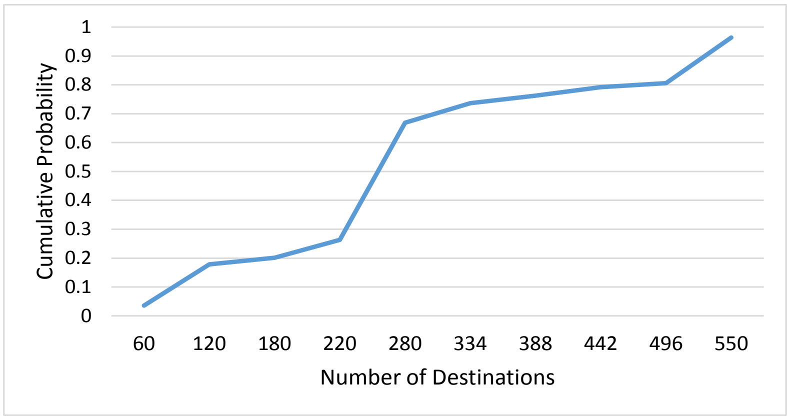


Figure 5.20: CDF of Destinations in Cluster 5

for the kalman filter related to Algorithm 6 in order to measure and compare the error covariance for the purpose of distinguishing between orchestrated and non-orchestrated probing flows (recall Section 5.3.1).

By initially executing Algorithm 6 and subsequently enforcing the orchestration confirmation technique from Section 21, the first four clusters of Table 5.7 demonstrated positive orchestration behavior and passed the 100 coordinated machines threshold. Cluster 5 was eliminated as it revealed negative coordinated behavior as enforced by the orchestration confirmation technique and employed thresholds (recall Section 21). By employing

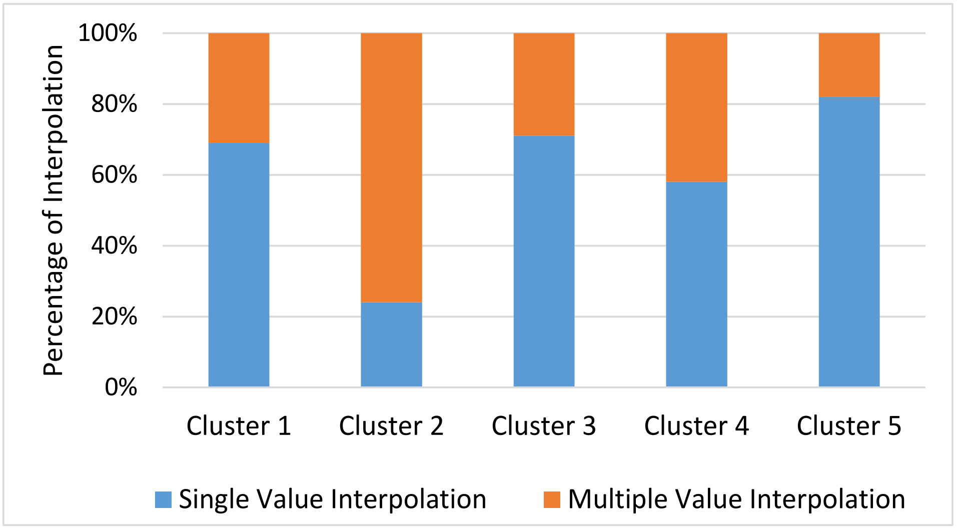


Figure 5.21: Distribution of the Types of the Interpolated Values

k	1	2	3	4	5	6	7	8	9	10
z_k	0.390	0.500	0.480	0.290	0.250	0.320	0.340	0.480	0.410	0.450
x_{k-1}	0	0.355	0.424	0.442	0.405	0.375	0.365	0.362	0.377	0.380
P^-_k	1	0.091	0.048	0.032	0.024	0.020	0.016	0.014	0.012	0.011
x_k	0.355	0.424	0.442	0.405	0.375	0.365	0.362	0.377	0.380	0.387
P_k	0.091	0.048	0.032	0.024	0.020	0.016	0.014	0.012	0.011	0.010

Table 5.8: Summary of one execution of the kalman filter

simple tcpdump[14] signatures on the data derived from the features (i.e., destination port, protocol, flags, ttl) of the first four clusters, Table 5.7 could be re-represented as orchestrated flow clusters as summarized in Table 5.9.

By comparing Tables 5.7 and 5.9, one can note the drop between clustered independent flows (Table 5.7) and orchestrated flows (Table 5.9) as achieved by Algorithm 6.

[14]http://tinyurl.com/pd2exca

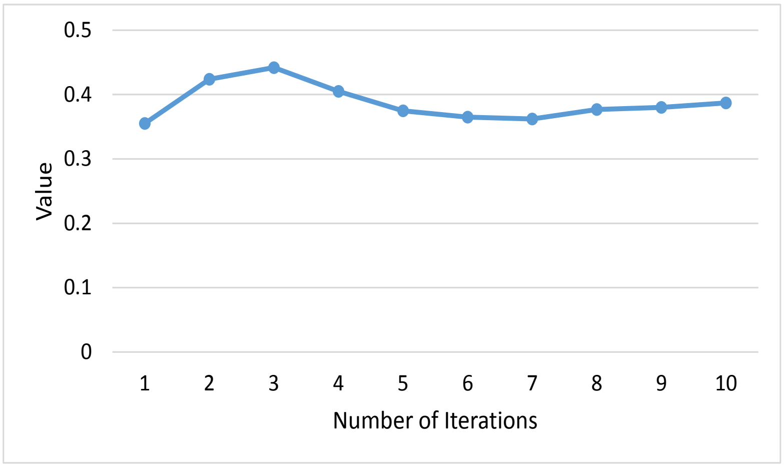


Figure 5.22: The Kalman Filter algorithm converging over few iterations

	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Probed Destination Port	5060	23	80	1433
Transport Protocol	UDP	TCP	TCP	TCP
Probing Flags	UDP	TCP SYN	TCP SYN	TCP ACK
TTL	128	128	64	64
Number of Orchestrated Probing Flows	1567	1863	858	587

Table 5.9: Summary of the inferred orchestrated probing campaigns

5.3.4 Comparison with Previous Work

Our first attempt to tackle the problem of inferring large-scale probing campaigns by observing network telescopes was rendered in [38]. In that work, as was demonstrated in Section 5.2, we proposed a set of data behavioral analytics to scrutinize probes as received by the darkspace. Please recall that the analytics were based on statistical, heuristical and fuzzy hashing approaches that aim at generating feature vectors for each of the perceived probing flows.

To compare and contrast the advantages of the presently proposed work, in this section, we compare its outcome with our previous work. To achieve that, we execute the

	C1	**C2**	**C3**
Employed probing technique:	UDP	TCP SYN	ACK
Probing traffic:	Random	Random	Pattern
Employed pattern:	Null	Null	[19.17-21.23]
Adopted probing strategy:	Reverse IP-sequential	Uniform Permutation	Forward IP-Sequential
Nature of probing source:	Bot	Bot	Tool
Type of probing:	Dispersed	Dispersed	Targeted
Signs of malware infection:	Yes	Yes	No
Exact malware type/variant:	Virus.Win32.Sality.bh	Trojan.Win32.Jorik	Null
Probing rate (in pps):	12	118	77
Target port:	5060	80	1433
Number of Probing Bots/Sources:	846	817	438

Table 5.10: The automatically inferred patterns capturing three large-scale orchestrated probing campaigns by employing the approach from Section 5.2

previously proposed approach from Section 5.2 on the same dataset of this work. The outcome is illustrated in Table 5.10. By comparing Tables 5.9 and 5.10, we can pinpoint that both approaches were able to infer orchestrated clusters 1, 3 and 4 of Table 5.9. One difference that is related to those campaigns is concerning the number of identified orchestrated flows; the number of inferred orchestrated probing sources using the proposed approach exceeded those inferred using the approach from Section 5.2. We manually investigated the additional flows as identified using the proposed approach, and we confirmed that they are indeed part of the campaign. Hence, we assert that the previously proposed approach missed some flows as belonging to the same campaign. Further, the previously proposed approach completely missed a fourth campaign, which was pinpointed by the proposed approach as cluster 2 of Table 5.9. Additionally, the proposed approach is not only more accurate but is also more prompt than the previous approach; the proposed approach can flag a campaign by observing only few orchestrated flows while the previous approach requires the creation of the complete feature pattern (i.e., processing of all the perceived data) to cluster the sources into well-defined campaigns. Moreover, the proposed approach is more efficient as it only records and process lightweight time series (recall Section 5.3.1) while the previous approach needs to maintain, in memory, the entire darknet data flows.

Last but not least, the presently proposed approach is more systematic and formal compared with the previously proposed approach as it deals with time series, trigonometric interpolation and state space modeling as apposed to relying on the output of statistical tests and heuristics. Since the previously proposed approach attributes the campaign to a certain malware family using correlation techniques between probing and malware samples (recall Section 5.2), we may port this capability to the presently proposed approach, which currently lacks that feature.

5.3.5 Model Limitations

It is realistic to pinpoint two limitations of the proposed approach. First, as previously described in this section, the model flags a campaign when the number of orchestrated flows within the campaign reach a hard threshold of 100 coordinated sources. Although the latter is not problematic, we would like to avoid hard thresholds, possibly by assigning a probability function to the campaign. In this matter, the model would be able to pinpoint a campaign when the probability reaches a certain confidence level. This interesting aim is left for future work. Second, it is evident that the model leverages the telescope space to infer orchestrated probing campaigns. Although the monitored space is relatively large (i.e., /13), the model is unable to monitor events that do not target such space. Subsequently, the approach will be unable to correlate those "unseen" activities and thus will fail to detect and identify all coordinating probing flows. However, we deem the latter as a generic limitation with any work the leverages the dark space and not a model limitation per say.

5.4 Behavioral Service Graphs: Inferring the niche of a Probing Campaign

In this section, we present an approach, by leveraging graph theoretic notions, that aim at inferring the niche of the detected probing campaigns. It is worthy to note that the niche of the campaign are rendered by the infected bots (i.e., machines) that are causing the probing botnet to expand by infecting other nodes. Such capability is highly imperative, as it will allows the prompt containment of such nodes to subsequently force the campaign to diminish.

5.4.1 Proposed Approach

We model the probing machines that show signs of infection coupled with their feature vectors using what we refer to as Behavioral Service Graphs. Please note that the probing machines are inferred as such by leveraging the approach from Section 3.2 of Chapter 3. Further, the feature vectors of each of such machines are generated by using the behavioral analytics from Section 4.2.3 of Chapter 4. Behavioral Service graphs are of the form $G = (N, E)$ where N represents the set of infected probing sources/machines (i.e., nodes) and E characterizes the edges between such nodes. It is worthy to mention that G is an undirected complete graph [208], with weights on the edges representing the probability of behavioral similarity (P_{bs}) computed by piecewise comparisons between the previously inferred feature vectors of each of the nodes. Examples of G corresponding to 8 and 20 nodes is shown in Figure 5.23. Another feature of such graphs is that they are designed to

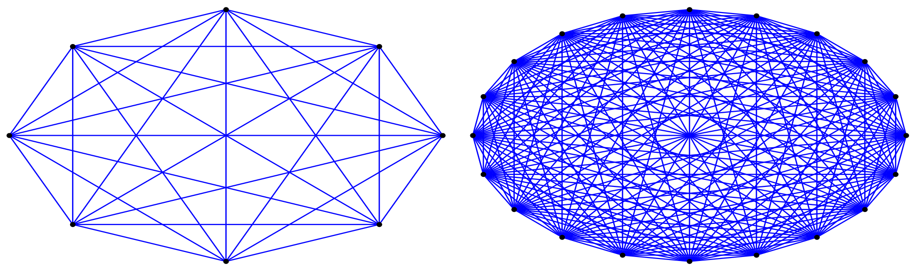

(a) K_8 Complete Graph (b) K_{20} Complete Graph

Figure 5.23: Illustration of Undirected Complete Behavioral Service Graphs

provide additional forensic evidence related to what service is being probed. The service is rendered by the destination port number inferred from the detected probing packets. Hence, the word ‘Service’ in Behavioral Service Graphs. Therefore, in essence, each constructed graph is actually modeling infected machines, their behavioral similarity and what specific network service is being probed.

The proposed approach further executes two steps to retrieve the niche number of infected machines for a given probing campaign.

First, given a complete Behavioral Service Graph $G = (N, E)$, the approach extracts a subgraph $G' = (N', E')$ where $N' = N$ and $E' \subseteq E$. This aims at reducing the number of edges while maximizing the behavior probability between the infected machines (i.e., nodes). To achieve this task, we employ the graph theoretical concept of a maximum spanning tree [209] by implementing a slightly modified version of Kruskal's algorithm [210]. Although there exists a plethora of approaches for the creation of maximum spanning trees, the latter algorithm was the basis of many and is abundantly available in numerous tool sets. For illustration purposes, if we apply the algorithm on the K_8 and K_{20} complete graphs of Figure 5.23, the outcome could be represented as in Figure 5.24. Second, to understand the structure of the subgraph formed by members of a campaign

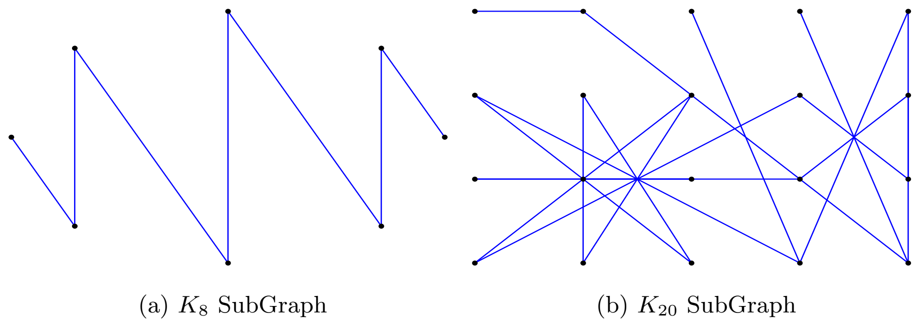

(a) K_8 SubGraph (b) K_{20} SubGraph

Figure 5.24: The application of Maximum Spanning Trees on Complete Behavioral Service Graphs

on a Behavioral Service Graph, suppose that there are m bots (i.e., infected machines) in the network, and therefore there are m corresponding nodes on the graph. Let the set $X = \{X_1, X_2, \cdots, X_m\}$ denote these nodes and P_e denote the probability of having an edge between any given X_i and X_j, for $i \neq j$ where $1 \leq i \leq m$ and $1 \leq j \leq m$. Since P_e would exist with an equal and a random probability given any pair of X_i and X_j, the subgraph formed by the nodes X_1, X_2, $\cdots$, X_m on a Behavioral Service Graph is indeed an Erdős-Rényi random graph [211, 212], where each possible edge in the graph possesses an equal probability of being created.

One interesting property shown by Erdős and Rényi is that, Erdős-Rényi graphs have a sharp threshold of edge probability for graph connectivity [212]. Simplified, if the edge-probability is greater than such threshold, then all of the nodes produced by such a model will be strongly connected. Erdős and Rényi have shown that the sharp connectivity threshold is $th_s = \frac{ln\theta}{\theta}$, where θ is the number of nodes in the graph. The proposed approach exploits this neat graph theoretic property; given the previously extracted maximum spanning tree subgraph, the approach eliminates all nodes/edges whose bot-edge probability (i.e., behavioral similarity P_{bs}) is less than th_s, deeming the rest of the nodes, given such formal forensic evidence, as the minimum number of infected machines (i.e., the niche) forming a campaign.

In conclusion, according to the random peer selection model, the niche members of the same infected campaign are expected to be closely connected to each other on a subgraph extracted from Behavioral Service Graphs.

5.4.2 Empirical Evaluation

We evaluate the proposed approach in two different deployment scenarios using two real datasets. This aims at validating the accuracy, effectiveness and simplicity of the generated network-based evidence as well as demonstrating the portability of the proposed approach.

Scenario 1: Enterprise Capability

In this first scenario, Behavioral Service Graphs are employed to infer infected machines within an enterprise network. Although the notion of an enterprise network could extend to an Internet service provider or even a backbone network, in this scenario, for simplicity purposes, we depict a small department within an organization having a deployment setting similar to what is illustrated in Figure 5.25. Such department includes 26 machines that are connected to the Internet via an enterprise commodity edge server. The proposed approach is deployed on that server.

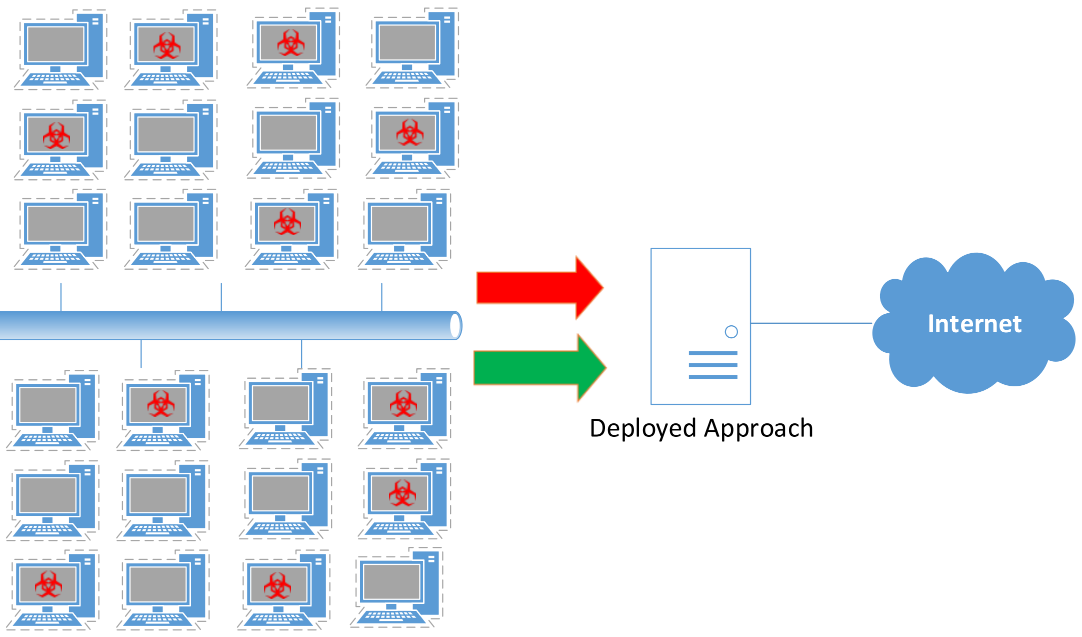


Figure 5.25: The proposed approach deployed as an enterprise edge engine

Building the ground truth

In order to systematically assess the accuracy of the proposed scheme, one needs to know the IP addresses/hosts of the members of the malicious campaign in a given network. Otherwise, nothing can be said about the true positive or false alarm rate.

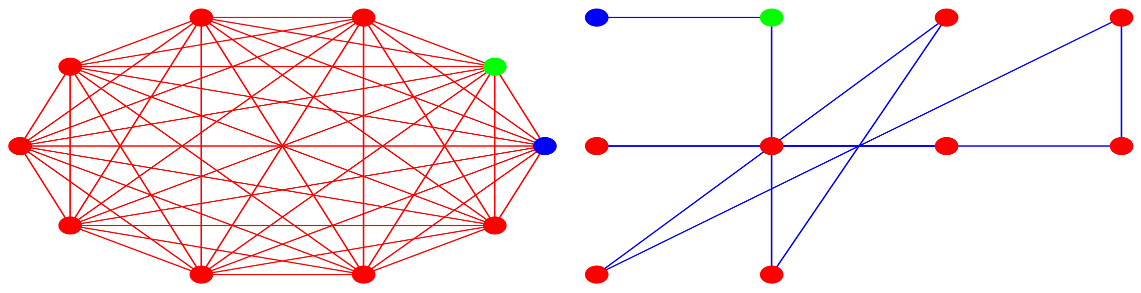
In order to establish the ground truth for our experiment, we obtained 10 GB of real probing traffic dataset retrieved from the Carna botnet[15]. The latter orchestrated campaign is rendered as one of the largest and most comprehensive IPv4 probing census ever. Subsequently, we presumed, as shown in Figure 5.25, that 10 out of the 24 machines are infected and thus are generating their malicious probing traffic towards the Web service using TCP as the transport protocol and 80 as the destination port number. We successfully achieved this by substituting the IP addresses of the assumed infected departmental machines with 10 IP addresses belonging to 10 unique sources of the Carna botnet that are probing that service. To provide a realistic evaluation scenario, we now assume that

[15]http://internetcensus2012.bitbucket.org/download.html

we have no knowledge about the infected departmental machines. Subsequently, we generated a legitimate background traffic dataset by leveraging the Security Experimentation EnviRonment (SEER) tool set[16] and randomly merged it[17] with the malicious probing traffic dataset. The newly created merged dataset could be thought of as the network data generated by the departmental machines and received by the edge server, where the proposed approach has been deployed for inference and analysis.

Evaluation

By invoking the proposed approach on the merged dataset, the Behavioral Service Graph and its corresponding maximum spanning tree were inferred as depicted in Figure 5.26.



(a) Enterprise Complete Graph (b) Enterprise SubGraph

Figure 5.26: The creation of the Enterprise Complete and Sub Graphs

A number of observations could be extracted from the complete graph. First, the number of assembled Behavioral Service Graphs is accurate; the approach generated one complete graph which is correct as the infected machines in the illustrated scenario are probing only one service, namely, the Web service. Second, the number of nodes in this Behavioral Service Graph is also precise; the approach inferred and correlated 10 infected machines which is consistent with the number of infected departmental machines. Third, after a semi-automated analysis and comparison that was based on the logged probing IP traffic flows, we identified that all the 10 machines that the proposed approach has identified are indeed the same IP addresses of the infected departmental machines (i.e.,

[16]http://seer.deterlab.net/trac

[17]using tcpslice available at http://sourceforge.net/projects/tcpslice/

the IP addresses of the Carna botnet). Therefore, based on the latter three observations, we can claim that the proposed approach yielded no false negative or false positive.

However, to further fortify the latter claim in an attempt to generalize it as it applies to various network scenarios, we performed several other experiments. Specifically, we were interested in evaluating the accuracy of the proposed approach as (1) the number of probed services increase in diversity and (2) as the number of infected machines scale up in the given network. Thus, we first augmented the number of probed services, one at a time, up to 10 various probed TCP and UDP services. The results disclosed that the number of generated Behavioral Service Graphs remained accurately reflecting the number of probed services. Moreover, to verify the scalability of the proposed approach, we increased the number of ground truth infected machines, by slots of 100, up to 1000 machines. The outcome disclosed around 82% accuracy in terms of constructed number of nodes and 100% accuracy in terms of the positive infection state of such nodes. Such experiments validate the accuracy of the proposed scheme, yet revealing that such approach might not be very scalable.

We were further concerned about the quality of the formed cluster provided by the complete Behavioral Service Graph. Since such graph is supposed to correlate the nodes based on their infection state as well as their behaviors, we thought it would be significantly beneficial to assert such grouping of nodes by employing another approach. To achieve this, we relied on an unsupervised, machine learning data clustering technique, namely, the k-means algorithm [213]. Typically, the standard k-means algorithm requires, as apriori knowledge, the number of clusters k. However, since our aim is to provide a robust evaluation methodology, we relied on an approach to automatically determine the optimal number of clusters. In particular, we leveraged the Calinski-Harabasz criterion [214] that operates by systematically verifying various number of clusters and subsequently recording the variances between and within the formed clusters. To determine the optimal number of clusters, the metric should be maximized with respect to k; the optimal number of clusters is the solution with the highest Calinski-Harabasz index value. To apply the k-means on the infected 10 nodes as previously inferred by the complete Behavioral Service Graph, we (1) retrieved their probing traffic from the merged dataset using a simplistic tcpdump[18] filter, (2) extracted their packet features [215] using the open source jNetPcap API[19] and (3) compiled the extracted features into a unified data file and then applied the k-means algorithm in conjunction with the Calinsk-Harabasz metric on such file. The outcome of the k-means execution is illustrated in Figure 5.27.

Motivated by [216] that asserted 1) that the relaxed solution of the k-means clustering, specified by the cluster indicators, could be given by the principal components from the Principal Component Analysis (PCA) technique [217] and 2) that the PCA subspace spanned by the principal directions is identical to the cluster centroid subspace, Figure 5.27 reveals the formed cluster on the first two principal axes of the PCA. One can notice the formation of one, and only one, relatively strongly correlated cluster. This result strongly advocates that the nodes possess strong similarity characteristics. In summary, such outcome that was generated by approaching the clustering problem from another perceptive, indeed validates the grouping capability of the infected machines (i.e., nodes) that is provided by the constructed complete Behavioral Service Graph.

Thus, up to this stage, an enterprise forensic investigator can leverage such precise and actionable evidence for prompt detection, containment and mitigation of such infected machines from the concerned network. Intuitively, an investigator can as well isolate such machines to collect additional evidence by monitoring their network and system activities for auxiliary data analysis or extraction of digital artifacts. The latter evidence could be exploited to generate signatures of attacks or for thoroughly analyzing a specific security phenomenon of interest.

Extracted from the Behavioral Service Graph, the enterprise subgraph that is depicted in Figure 5.26b also provides noteworthy inferences. First, it was able to generate

[18]http://www.tcpdump.org/

[19]http://jnetpcap.com/

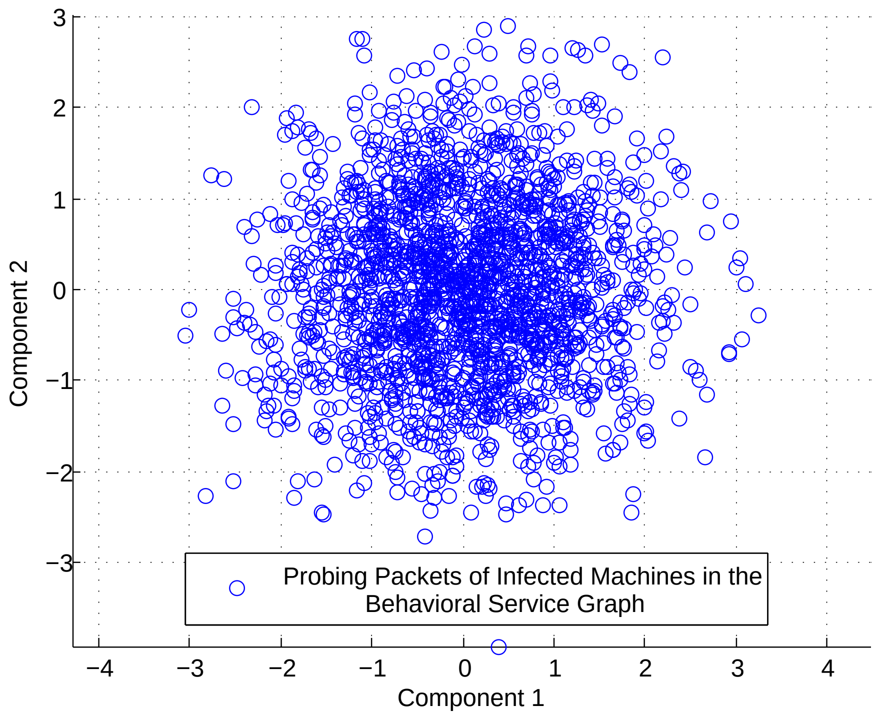


Figure 5.27: Validating the clustering capability of the complete Behavioral Service Graph

formal forensic evidence that clustered the machines into a well-defined orchestrated infected campaign. Recall, that the formality arises from the fusion of the bots behavioral similarly as previously extracted by the behavioral analytics of Section 4.2.3 coupled with the graph theoretical notion of a maximum spanning tree. Second, and more importantly in our opinion, the approach was able, by leveraging Erdős-Rényi random graphs, to infer the niche members of that infected campaign. From Figure 5.26, one can notice that 2 out of the 10 nodes are colored differently than red (i.e., in blue and green). In fact, the proposed approach revealed that these two nodes render the creation of the campaign. Thus, in theory, these nodes should have caused the creation of the campaign in the first place. To validate this, we manually investigated those IP addresses by going back to the Carna botnet dataset. Our investigation showed that these two IP addresses are used as root nodes in the botnet to infect other machines for propagation purposes. The latter fact was further validated as these two nodes were among the top 3 nodes to generate most of the probing traffic in the dataset. The latter formal forensic evidence could be promptly exploited by investigators to prioritize the eradication of those two nodes in order to seize the expansion of the campaign on their networks. This would indeed significantly limit any present or future possible sustained collateral damage and any symptoms of infection that could be caused by the infected bots.

Scenario 2: Global Capability

In the previous scenario, we have demonstrated how the proposed scheme can be exploited to operate within the context of an enterprise network. In this section, we port the approach to a global scale and elaborate on how it can be employed to monitor, infer and distribute Internet-scale forensic intelligence. Thus, in this scenario, the approach is envisioned to operate in a model similar to what is dubbed as a global Security Operation Center (SOC). Typically, such operational centers have access to significant various real-time and raw data streams from around the globe. They often exploit such data for analysis, correlation and generation of intelligence that would be distributed to concerned parties for alert and mitigation purposes. Such centers were initially formed as global independent entities to combat an increasing trend of external (in contrary to internal) threats and attacks.

Thus, in this second scenario, Behavioral Service Graphs are postulated to be deployed as an additional forensic capability in one of those SOC centers. In this context, we operate the scheme by investigating darknet data.

Evaluation

From our darknet data repository, we extract one week of data (≈ 80 GB) retaining to the period of October 4^{th} to October 11^{th}, 2012. The aim is to employ the proposed approach on such data to evaluate the scheme's capability and effectiveness in disclosing insights related to that reported campaign.

By executing the proposed approach on the extracted probing traffic from our darknet dataset, the outcome demonstrated that one of the Behavioral Service Graphs was

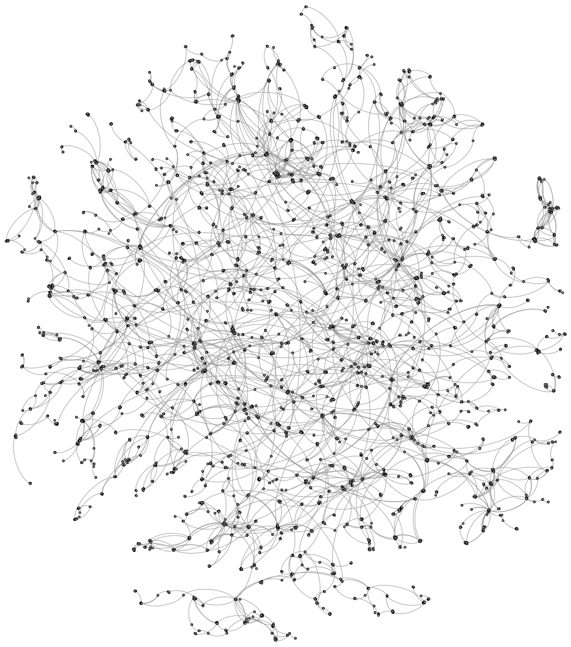


Figure 5.28: The proposed approach revealing the bots of the SQL probing campaign

indeed able to infer and correlate around 800 unique sources[20] targeting the SQL service. Further, the behavioral analytics (1) showed strong behavioral similarity between those sources and (2) inferred that those sources were indeed bots, thus providing strong evidence that such campaign was triggered from Internet-wide infected machines. The latter inferred bots could be visualized as in Figure 5.28. Additionally, it might be interesting to mention that the proposed approach deemed 84 bots as the niche of the campaign.

Thus, provided with such forensic evidence, SOC analysts can demand an immediate take-down of those 84 bots to limit the expansion of such campaign on the global Internet. Supplementary, they can promptly notify concerned parties to employ mitigation approaches against the abuse of SQL servers.

[20]Since the monitored darknet space is a /13, we are only able to see a portion of the campaign using that dataset

5.5 Related Work

In this section, we review the related work on various concerned topics.

Extracting Probing Events: Li et al. [129] considered large spikes of unique source counts as probing events. The authors extracted those events from darknet traffic using time series analysis; they first automatically identified and extracted the rough boundaries of events and then manually refined the event starting and ending times. At this point, they used manual analysis and visualization techniques to extract the event. In an alternate work, Jin et al. [218] considered any incoming flow that touches any temporary dark (grey) IP address as potentially suspicious. The authors narrowed down the flows with sustained suspicious activities and investigated whether certain source or destination ports are repeatedly used in those activities. Using these ports, the authors separated the probing activities of an external host from other traffic that is generated from the same host. In contrast, in this work, we propose a method that exploits a unique observation related to the signal correlation status of probing events. By leveraging this, CSC-Detector's fingerprinting engine is able to differentiate between probing and other events and subsequently extract the former from incoming darknet traffic.

Analyzing Probing Events: The authors of [218, 219] studied probing activities towards a large campus network using netflow data. Their goal was to infer the probing strategies of scanners and thereby assess the harmfulness of their actions. They introduced the notion of gray IP space, developed techniques to identify potential scanners, and subsequently studied their scanning behaviors. In another work, the authors of [129, 220, 221] presented an analysis that drew upon extensive honeynet data to explore the prevalence of different types of scanning. Additionally, they designed mathematical and observational schemes to extrapolate the global properties of scanning events including total population and target scope. In contrary, we aim at inferring large-scale probing campaigns rather than focusing on analyzing probing events. To achieve that, CSC-Detector's inference and analysis engines collaborate to scrutinize the behavior of probing events as perceived by a darknet and subsequently automatically build patterns that aim at correlating the probing sources into well-defined campaigns.

Probing Measurement Studies: In addition to [177, 123], Benoit et al. [222] presented the world's first Web census while Heidemann et al. [223] were among the first to survey edge hosts in the visible Internet. Further, Pryadkin et al. [224] offered an empirical evaluation of IP address space occupancy whereas Cui and Stolfo [225] presented a quantitative analysis of the insecurity of embedded network devices obtained from a wide-area scan. In a slightly different work, Leonary and Loguinov [169] demonstrated IRLscanner, a tool which aimed at maximizing politeness yet provided scanning rates that achieved coverage of the Internet in minutes. In this work, as previously mentioned, we strive to infer large-scale probing campaigns rather than solely providing measurements of particular probing events.

Botnet Detection Frameworks: A number of botnet detection systems have been proposed in the literature [226, 227, 228, 229]. Some investigates specific channels, others might require deep packet inspection or training periods, while the majority depends on malware infections and/or attack life-cycles. To the best of our knowledge, none of the proposals is dedicated to tackle the problem of inferring large-scale probing campaigns. Further, in this work, we aim to achieve that task by analyzing the dark IP space and by focusing on the machinery of the probing sources, without requiring content analysis or training periods.

Indeed, as stated by CAIDA in [178], the capability to infer large-scale orchestrated probing campaigns does not exist, rendering the proposed approaches in this section as novel contributions.

5.6 Summary

First, this chapter presented CSC-Detector, a systematic effort towards the challenging goal of detecting and identifying large-scale orchestrated probing campaigns specifically tailored towards the dark IP space. The system was primarily motivated by the prompt need for such a cyber security capability as stated by CAIDA [177, 178]. The system's fingerprinting engine exploits a unique observation to extract probing activities from darknet traffic. Consequently, CSC-Detector's inference engine employs a set of behavioral analytics to generate insights that capture numerous characteristics of the probing sources. Last but not least, the system's analysis engine leverages several criteria, methods and approaches to automatically identify and correlate the probing sources into well-defined campaigns. CSC-Detector was empirically evaluated using significant amount of real darknet data. The recently inferred CSCs, which were validated using DShield data, indeed demonstrates that the system exhibits promising accuracy and can generate substantial evidence to infer diverse large-scale probing campaigns.

Second, this chapter also approached the problem of inferring orchestrated probing campaigns but from a time series perspective. The motivation behind that was two-folds. First, from an efficiency perspective, we thought it would be desirable to work with the artifact of the data rather than the data itself. Second, scientifically, it is typically more sound to generate inferences and insights using formal methods and approaches rather than depending on statistical inferences or heuristics. Thus, also in this section, we uniquely modeled the challenging task of inferring probing campaigns as a problem of interpolating and predicting time series in the presence of missing values. Initially, the model extracts independent probing flows as perceived by the telescope. Subsequently, the model builds numerous time series clusters sharing similar traffic features. For data completion purposes, the model employs trigonometric interpolation on such probing time series. To infer possible orchestration behavior, the model executes state space modeling in conjunction with a time-varying window algorithm. Finally, to assert orchestration, the model exploits the usage of the coordinated sources by leveraging a similarity of closeness technique. Simulation analysis have authenticated the choice of the employed theoretical-based techniques as core components of the proposed model, while empirical evaluations indeed demonstrated the promptness and accuracy of the proposed model in comparison with a previous work.

Third, this chapter had devised Behavioral Service Graphs, an approach that is able to effectively process, analyze and correlate large volumes of network traffic to generate, in a very prompt manner, formal, highly-accurate and actionable network forensic evidence that could be leveraged by investigators to infer the niche of the the probing botnets. Rigorous empirical evaluations with real data under two different deployment scenarios indeed verified the accuracy and effectiveness of the approach.

Chapter 6

Conclusion

The ever increasing population and adoption of cyberspace has been a great asset both socially and economically. The complete embracing of cyberspace technologies allowed the creation and implementation of new ideas that tremendously facilitate everyday tasks. Critical infrastructure heavily depend on information and communication technologies to operate successfully. However, recent events demonstrated that cyberspace could be subjected to amplified, debilitating and disrupting attacks that might lead to severe security issues with drastic consequences.

This thesis was dedicated to tackle the ever increasing cyber security concern rendered by probing activities, which are a core facilitator of numerous cyber security incidents. We successfully achieved the latter by employing passive monitoring of the Internet IP space, also known as network telescopes.

In particular, in this thesis, we primarily reviewed the literature in terms of probing approaches, probing techniques as well as distributed probing detection methods. Consequently, we concentrated our research work towards the problem of inferring enterprise and Internet-scale probing activities. After inferring such malicious activities, we attempted to correlate the latter with malware samples for attribution and containment purposes. The last problem that we tackled in this thesis was rendered by inferring and attributing large-scale probing campaigns, which define a new era of of such malevolent events.

From the conducted research, we can extract the following points/research gaps:

- Inferring and attributing botnets or malicious campaigns by solely monitoring the dark IP space is very challenging due to the passive nature of such IP space. Therefore, other interactive techniques such as honeypots could be used in conjunction with darknet analysis to enhance botnet investigation.
- Differentiating between scan-based amplification attacks and probing activities is still partially in the gray area. More robust and practical approaches are required to investigate the latter.
- Packet analysis is the only technique employed on darknet data to investigate spoofing activities. This method is rendered by inspecting ICMP packets and TTL values. Minimal research has been executed to study spoofing events through darknet analysis. Therefore, spoofing is still a noteworthy malicious activity that needs more attention from the security research community.
- Filtering darknet misconfiguration is still not thoroughly investigated in the literature and hence requires more attention from the research community.
- Mobile darknet is a new research trend that has a promising future in the area of passive monitoring. Future deployment could include mobile-based VoIP darknets.
- Despite the existence of few collaborative darknet projects, more darknet resources and information sharing efforts should emerge to infer and attribute large-scale cyber activities. Indeed, establishing a worldwide darknet information exchange is a capability that requires collaboration and trust; however, this collaboration necessitates the implementation of numerous global policies and undoubtedly would raise serious privacy concerns.

As for future work, we aim at tackling the following research and development topics:

- The design and implementation of cyber threat intelligence approaches for critical infrastructure protection by leveraging big data analytics.

- The analysis of the implications of IPv6 assignment on Internet passive monitoring.
- The design and implementation of robust non-heuristical correlation mechanisms between various cyber security data feeds.

Bibliography

[1] W. Zhang, S. Teng, and X. Fu. Scan attack detection based on distributed cooperative model. In *Computer Supported Cooperative Work in Design, 2008. CSCWD 2008. 12th International Conference on*, pages 743–748. IEEE, 2008.

[2] H.U. Baig and F. Kamran. Detection of port and network scan using time independent feature set. In *Intelligence and Security Informatics, 2007 IEEE*, pages 180 –184, may 2007.

[3] R. Baldoni, G. Di Luna, and L. Querzoni. Collaborative Detection of Coordinated Port Scans. Technical report, 2012. `http://www.dis.uniroma1.it/~midlab`.

[4] D. Whyte, P.C. Oorschot, and E. Kranakis. Tracking darkports for network defense. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 161–171. IEEE, 2007.

[5] S. Robertson, E.V. Siegel, M. Miller, and S.J. Stolfo. Surveillance detection in high bandwidth environments. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 130–138. IEEE, 2003.

[6] H. Choi, H. Lee, and H. Kim. Fast detection and visualization of network attacks on parallel coordinates. *computers & security*, 28(5):276–288, 2009.

[7] E. Bou-Harb, M. Debbabi, and C. Assi. A statistical approach for fingerprinting probing activities. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 21–30, Sept 2013.

[8] Government of Canada. Canada's Cyber Security Strategy Report, 2010. `http://www.capb.ca/uploads/files/documents/Cyber_Security_Strategy.pdf`.

[9] The Whitehouse. CyberSpace Policy Review, 2013. `http://www.whitehouse.gov/assets/documents/Cyberspace_Policy_Review_final.pdf`.

[10] Government of Canada. Service Canada, 2012. `http://www.servicecanada.gc.ca/eng/home.shtml`.

[11] Stephen Hinde. The law, cybercrime, risk assessment and cyber protection. *Computers & Security*, pages 90–95, 2003.

[12] Yoo Chung. Distributed denial of service is a scalability problem. *SIGCOMM Comput. Commun. Rev.*, 42(1):69–71, January 2012.

[13] M.K. Daly. Advanced persistent threat. *Usenix, Nov*, 4, 2009.

[14] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, CCS '12, pages 833–844, New York, NY, USA, 2012. ACM.

[15] Chenfeng Vincent Zhou, Christopher Leckie, and Shanika Karunasekera. A survey of coordinated attacks and collaborative intrusion detection. *Computers & Security*, 29(1):124 – 140, 2010.

[16] Symantec. W32.Stuxnet Dossier, 2012. `http://tinyurl.com/36y7jzb`.

[17] DefenseTech. Cyber War 2.0, Russia v. Georgia, 2012. `http://tinyurl.com/8l7cvm8`.

[18] GovCon Technology. Cyber Attacks Fifth Dimension of Warfare, Says NATO Official, 2012. `http://tinyurl.com/yad4z49`.

[19] Softpedia. Google Bolivia Domains Defaced by Turkish Hacker. `http://news.softpedia.com/news/Google-Bolivia-Domains-Defaced-by-Turkish-Hacker-396405.shtml`.

[20] Hackmageddon.com. Brandon University Cyber Attack. `http://hackmageddon.com/tag/brandon-university/`.

[21] Hackmageddon.com. October 2013 Cyber Attacks Timeline. `http://hackmageddon.com`.

[22] HackRead. African Petroleum Producers Association Website Hacked by Fallaga Team Tunisia. `http://hackread.com/fallaga-team-tunisia-hacks-african-petroleum-site/`.

[23] Reuters. U.S. looking into cybersecurity incidents targeting Obamacare website. `http://www.reuters.com/article/2013/11/13/us-usa-healthcare-security-idUSBRE9AC16M20131113`.

[24] Chicago Tribune. Hackers plan attack on Russian government sites. `http://articles.chicagotribune.com/2012-05-04/business/sns-rt-russia-hackersanonymousl5e8g45ui-20120504_1_cyber-attack-government-websites-pro-kremlin`.

[25] Yerkir Media. Azerbaijan hired hackers to attack Armenian websites and had certain success. `http://www.yerkirmedia.am/?act=news&lan=en&id=9438`.

[26] Action Plan 2010-2015 for Canada's Cyber Security Strategy. `http://tinyurl.com/plxmua8`.

[27] New York Times Internal Network Hacked. `http://tinyurl.com/cvnrsac`.

[28] WordPress sites targeted by mass brute-force attack. `http://tinyurl.com/cxmgjax`.

[29] PlayStation Network Outage Caused By 'External Intrusion'. `http://tinyurl.com/6cbcldv`.

[30] Iran hacks energy firms. `http://tinyurl.com/opjw79c`.

[31] S. Panjwani, S. Tan, K.M. Jarrin, and Michel Cukier. An experimental evaluation to determine if port scans are precursors to an attack. In *Proceedings of the International Conference on Dependable Systems and Networks. DSN 2005*, pages 602–611, 2005.

[32] Elias Bou-Harb, Nour-Eddine Lakhdari, Hamad Binsalleeh, and Mourad Debbabi. Multidimensional investigation of source port 0 probing. *Digital Investigation*, 11, Supplement 2(0):S114 – S123, 2014. Fourteenth Annual {DFRWS} Conference.

[33] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. On fingerprinting probing activities. *Computers & Security*, 43(0):35 – 48, 2014.

[34] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. A systematic approach for detecting and clustering distributed cyber scanning. *Computer Networks*, 57(18):3826 – 3839, 2013.

[35] Elias Bou-Harb, Mourad Debbabi, and Chadi Assi. Cyber scanning: A comprehensive survey. *Communications Surveys Tutorials, IEEE*, 16(3):1496–1519, Third 2014.

[36] Elias Bou-Harb, Claude Fachkha, Mourad Debbabi, and Chadi Assi. Inferring internet-scale infections by correlating malware and probing activities. In *Communications (ICC), 2014 IEEE International Conference on*, pages 640–646, June 2014.

[37] E. Bou-Harb, M. Debbabi, and C. Assi. On detecting and clustering distributed cyber scanning. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pages 926–933, July 2013.

[38] E. Bou-Harb, M. Debbabi, and C. Assi. Behavioral analytics for inferring large-scale orchestrated probing events. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 506–511, April 2014.

[39] E. Bou-Harb, C. Fachkha, M. Pourzandi, M. Debbabi, and C. Assi. Communication security for smart grid distribution networks. *Communications Magazine, IEEE*, 51(1):42–49, January 2013.

[40] Claude Fachkha, Elias Bou-Harb, and Mourad Debbabi. On the inference and prediction of ddos campaigns. *Wireless Communications and Mobile Computing*, 2014.

[41] C. Fachkha, E. Bou-Harb, and M. Debbabi. Fingerprinting internet dns amplification ddos activities. In *New Technologies, Mobility and Security (NTMS), 2014 6th International Conference on*, pages 1–5, March 2014.

[42] C. Fachkha, E. Bou-Harb, and M. Debbabi. Towards a forecasting model for distributed denial of service activities. In *Network Computing and Applications (NCA), 2013 12th IEEE International Symposium on*, pages 110–117, Aug 2013.

[43] C. Fachkha, E. Bou-Harb, A Boukhtouta, S. Dinh, F. Iqbal, and M. Debbabi. Investigating the dark cyberspace: Profiling, threat-based analysis and correlation. In *Risk and Security of Internet and Systems (CRiSIS), 2012 7th International Conference on*, pages 1–8, Oct 2012.

[44] Richard J Barnett and Barry Irwin. Towards a taxonomy of network scanning techniques. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08, pages 1–7, New York, NY, USA, 2008. ACM.

[45] M.H. Bhuyan, DK Bhattacharyya, and JK Kalita. Surveying port scans and their detection methodologies. *The Computer Journal*, 54(10):1565–1581, 2010.

[46] W.R. Stevens and G.R. Wright. *TCP/IP Illustrated: the protocols*, volume 1. Addison-Wesley Professional, 1994.

[47] R. Thurlow. Rpc: Remote procedure call protocol specification version 2. 2009.

[48] J. Medeiros, A. Brito, and P. Pires. A data mining based analysis of nmap operating system fingerprint database. *Computational Intelligence in Security for Information Systems*, pages 1–8, 2009.

[49] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. *Computer Networks*, 53(1):81–97, 2009.

[50] D. Stuttard and M. Pinto. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws.* Wiley, 2011.

[51] Xianghua Xu, Jian Wan, Wei Zhang, Chao Tong, and Changhua Wu. Pmsw: a passive monitoring system in wireless sensor networks. *International Journal of Network Management*, 21(4):300–325, 2011.

[52] G. Combs. Wireshark network analyzer-user's guide. 2008.

[53] M. Shelton. Passive Asset Detection System, 2008. `http://passive.sourceforge.net/about.php`.

[54] T. Socolofsky and C. Kale. Tcp/ip tutorial. Technical report, RFC 1180, Spider Systems Ltd, 1991.

[55] H. Al-Bahadili and A.H. Hadi. Network security using hybrid port knocking. *IJCSNS*, 10(8):8, 2010.

[56] Elias Bou-Harb, Makan Pourzandi, Mourad Debbabi, and Chadi Assi. A secure, efficient, and cost-effective distributed architecture for spam mitigation on lte 4g mobile networks. *Security and Communication Networks*, 2012.

[57] P. Li, M. Salour, and X. Su. A survey of internet worm detection and containment. *Communications Surveys & Tutorials, IEEE*, 10(1):20–35, 2008.

[58] A. Boulanger. Unauthorized intrusions and denial of service. *Cybercrimes: A Multidisciplinary Analysis*, pages 27–44, 2010.

[59] N. Hachem, Y. Ben Mustapha, G.G. Granadillo, and H. Debar. Botnets: lifecycle and taxonomy. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–8. IEEE, 2011.

[60] Zhichun Li, Anup Goyal, Yan Chen, and Vern Paxson. Automating analysis of large-scale botnet probing events. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 11–22, New York, NY, USA, 2009. ACM.

[61] K. Ko, H. Jang, B. Park, and Y. Eom. Analysis of the propagation pattern of a worm with random scanning strategy based on usage rate of network bandwidth. *Information, Security and Cryptology–ICISC 2009*, pages 374–385, 2010.

[62] C. Gates. Coordinated scan detection. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS 09)*, 2009.

[63] L. Aniello, G. Di Luna, G. Lodi, and R. Baldoni. A collaborative event processing system for protection of critical infrastructures from cyber attacks. *Computer Safety, Reliability, and Security*, pages 310–323, 2011.

[64] S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan. Tcp fast open. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 21. ACM, 2011.

[65] W. John, S. Tafvelin, and T. Olovsson. Trends and differences in connection-behavior within classes of internet backbone traffic. *Passive and Active Network Measurement*, pages 192–201, 2008.

[66] J. Gadge and A.A. Patil. Port scan detection. In *Networks, 2008. ICON 2008. 16th IEEE International Conference on*, pages 1–6. IEEE, 2008.

[67] I. Van Beijnum. An ftp application layer gateway (alg) for ipv6-to-ipv4 translation. 2011.

[68] G. Gont, C. Pignataro, and F. Gont. Recommendations for filtering icmp messages. 2012.

[69] K. Egevang and P. Francis. The ip network address translator (nat). Technical report, RFC 1631, may, 1994.

[70] R. Droms. Automated configuration of tcp/ip with dhcp. *Internet Computing, IEEE*, 3(4):45–53, 1999.

[71] MIT. 1999 DARPA Intrusion Detection Evaluation Data Set. `http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.html`.

[72] S. Staniford, J.A. Hoagland, and J.M. McAlerney. Practical automated detection of stealthy portscans. *Journal of Computer Security*, 10(1/2):105–136, 2002.

[73] C. Gates. Coordinated scan detection. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS 09)*, 2009.

[74] D. Whyte, P.C. Oorschot, and E. Kranakis. Tracking darkports for network defense. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 161–171. IEEE, 2007.

[75] V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: Global characteristics and prevalence. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 138–147. ACM, 2003.

[76] J. Treurniet. Detecting low-profile scans in tcp anomaly event data. In *Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services*, page 17. ACM, 2006.

[77] D. Hanselman and B.C. Littlefield. *Mastering MATLAB 5: A comprehensive tutorial and reference*. Prentice Hall PTR, 1997.

[78] J. Treurniet. A network activity classification schema and its application to scan detection. *Networking, IEEE/ACM Transactions on*, 19(5):1396–1404, 2011.

[79] A. Dainotti, R. Amman, E. Aben, and K.C. Claffy. Extracting benefit from harm: using malware pollution to analyze the impact of political and geophysical events

on the internet. *ACM SIGCOMM Computer Communication Review*, 42(1):31–39, 2012.

[80] M.H. Bhuyan, D.K. Bhattacharyya, and J.K. Kalita. Aocd: An adaptive outlier based coordinated scan detection approach. *International Journal of Network Security*, 14(6):339–351, 2012.

[81] J.C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2):191–203, 1984.

[82] K. Stockinger, E. Bethel, S. Campbell, E. Dart, and K. Wu. Detecting distributed scans using high-performance query-driven visualization. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 82. ACM, 2006.

[83] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[84] G. Conti and K. Abdullah. Passive visual fingerprinting of network attack tools. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 45–54. ACM, 2004.

[85] David Moore, Colleen Shannon, Geoffrey M Voelker, and Stefan Savage. *Network telescopes: Technical report.* Department of Computer Science and Engineering, University of California, San Diego, 2004.

[86] Eric Wustrow, Manish Karir, Michael Bailey, Farnam Jahanian, and Geoff Huston. Internet background radiation revisited. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 62–74. ACM, 2010.

[87] Zakir Durumeric, Michael Bailey, and J Alex Halderman. An internet-wide view of internet-wide scanning. In *USENIX Security Symposium*, 2014.

[88] Evan Cooke, Michael Bailey, Farnam Jahanian, and Richard Mortier. The dark oracle: Perspective-aware unused and unreachable address discovery. In *NSDI*, volume 6, 2006.

[89] Michael Bailey, Evan Cooke, Farnam Jahanian, Andrew Myrick, and Sushant Sinha. Practical darknet measurement. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 1496–1501. IEEE, 2006.

[90] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Exit from hell? reducing the impact of amplification ddos attacks. In *USENIX Security Symposium*, 2014.

[91] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Hell of a handshake: abusing tcp for reflective amplification ddos attacks. In *USENIX Workshop on Offensive Technologies (WOOT)*, 2014.

[92] Largest ever ddos attack peaks at 400 gbps. *Info Securityl*. `http://tinyurl.com/nljf3ct`.

[93] Security Information Exchange (SIE), Farsight Security Inc. `https://www.farsightsecurity.com`.

[94] Internet Engineering Task Force (IETF). A Simple Network Management Protocol (SNMP), 1990. `http://www.ietf.org/rfc/rfc1157.txt`.

[95] William Stallings. *SNMP,SNMPV2,Snmpv3,and RMON 1 and 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1998.

[96] Internet Engineering Task Force (IETF). SNMP Overhead and Performance Impact, 2003. `http://tools.ietf.org/html/draft-breit-snmp-overhead-00`.

[97] L. Andrey, O. Festor, A. Lahmadi, A. Pras, and J. Schönwälder. Survey of snmp performance analysis studies. *International Journal of Network Management*, 19(6):527–548, 2009.

[98] Symantec. Trojan Horse. `http://www.symantec.com/security_response/writeup.jsp?docid=2004-021914-2822-99`.

[99] eMarkSof Inc. eMarksoft SNMP Component, 2002-2012. `http://www.emarksoft.com/mib-snmp-component.htm`.

[100] Daniel Roelker, Marc Norton and Jeremy Hewlett. sfportscan, 2004. `http://projects.cs.luc.edu/comp412/dredd/docs/software/readmes/sfportscan`.

[101] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[102] D. Moore, G.M. Voelker, and S. Savage. Inferring internet denial-of-service activity. Technical report, DTIC Document, 2001.

[103] C.-K. Peng, S. V. Buldyrev, S. Havlin, M. Simons, H. E. Stanley, and A. L. Goldberger. Mosaic organization of dna nucleotides. *Phys. Rev. E*, 49:1685–1689, Feb 1994.

[104] U. Harder, M.W. Johnson, J.T. Bradley, and W.J. Knottenbelt. Observing internet worm and virus attacks with a small network telescope. *Electronic Notes in Theoretical Computer Science*, 151(3):47–59, 2006.

[105] K. Fukuda, T. Hirotsu, O. Akashi, and T. Sugawara. Correlation among piecewise unwanted traffic time series. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008.*, pages 1–5, 2008.

[106] M.B. Priestley. Spectral analysis and time series. 1981.

[107] J.A.O. Matos, S. Gama, H.J. Ruskin, A.A. Sharkasi, and M. Crane. Time and scale hurst exponent analysis for financial markets. *Physica A: Statistical Mechanics and its Applications*, 387(15):3910–3915, 2008.

[108] K. Hu, P.C. Ivanov, Z. Chen, P. Carpena, and H.E. Stanley. Effect of trends on detrended fluctuation analysis. *Physical Review E*, 64(1):011114, 2001.

[109] V. Jacobson, C. Leres, and S. McCanne. The tcpdump manual page. *Lawrence Berkeley Laboratory, Berkeley, CA*, 1989.

[110] G.F. Lyon. Nmap network scanning: The official nmap project guide to network discovery and security scanning author: Gordon fyodor l. 2009.

[111] M. Little, P. McSharry, I. Moroz, and S. Roberts. Nonlinear, biophysically-informed speech pathology detection. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings.*, volume 2, page II.

[112] Fing, the ultimate network toolkit. `http://www.overlooksoft.com/fing`.

[113] Simon Woodhead. Monitoring bad traffic with darknets. *Network Security*, 2012(1):10 – 14, 2012.

[114] Thomas Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology,*, 15(1):52–60, 1967.

[115] R.M. Neal and G.E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES*, 89:355–370, 1998.

[116] D.W. Scott. Multivariate density estimation. *Multivariate Density Estimation, Wiley, New York, 1992*, 1, 1992.

[117] jNetPcap. Sli Technologies. `http://jnetpcap.com/userguide`.

[118] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[119] k-Means Clustering. `http://www.mathworks.com/help/stats/k-means-clustering.html`.

[120] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescap. Analysis of a "/0" Stealth Scan from a Botnet. In *Internet Measurement Conference (IMC)*, Nov 2012.

[121] Jerome H Friedman and Lawrence C Rafsky. Multivariate generalizations of the wald-wolfowitz and smirnov two-sample tests. *The Annals of Statistics*, pages 697–717, 1979.

[122] Zhichun Li, Anup Goyal, and Yan Chen. Honeynet-based botnet scan traffic analysis. In *Botnet Detection*, pages 25–44. Springer, 2008.

[123] Internet Census 2012-Port scanning /0 using insecure embedded devices. `http://tinyurl.com/c8af8lt`.

[124] Genevieve Bartlett, John Heidemann, and Christos Papadopoulos. Understanding passive and active service discovery. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 57–70. ACM, 2007.

[125] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. Census and survey of the visible internet. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 169–182. ACM, 2008.

[126] Stuart Staniford, Vern Paxson, Nicholas Weaver, et al. How to own the internet in your spare time. In *Proceedings of the 11th USENIX security symposium*, volume 8, pages 149–167, 2002.

[127] Maurice George Kendall. Rank correlation methods. 1948.

[128] Sushil Jajodia. *Cyber Situational Awareness: Issues and Research.* 2012.

[129] Zhichun Li, Anup Goyal, Yan Chen, and Vern Paxson. Towards situational awareness of large-scale botnet probing events. *IEEE Transactions on Information Forensics and Security*, 6(1):175–188, 2011.

[130] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, D. Watson, et al. The internet motion sensor: A distributed blackhole monitoring system. In *Proceedings of the 12th ISOC Symposium on Network and Distributed Systems Security (SNDSS)*, pages 167–179, 2005.

[131] Yegneswaran, Vinod et al. On the design and use of internet sinks for network abuse monitoring. In *Recent Advances in Intrusion Detection.* 2004.

[132] Eric Wustrow, Manish Karir, Michael Bailey, Farnam Jahanian, and Geoff Huston. Internet background radiation revisited. In *Proceedings of the 10th annual conference on Internet measurement*, pages 62–74. ACM, 2010.

[133] Jayanthkumar Kannan, Jaeyeon Jung, Vern Paxson, and Can Emre Koksal. Semi-automated discovery of application session structure. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 119–132. ACM, 2006.

[134] Yi Cao. Bhattacharyya Distance Measure for Pattern Recognition. `http://tinyurl.com/bveualz`.

[135] Simone Fatichi. Mann-Kendall Test. `http://tinyurl.com/cstvpwa`.

[136] MathWorks. Run test for randomness. `http://tinyurl.com/d6gtykz`.

[137] David Moore, Colleen Shannon, Douglas J Brown, Geoffrey M Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems (TOCS)*, 24(2):115–139, 2006.

[138] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 33–42. ACM, 1999.

[139] Ryan Prescott Adams and David J.C. MacKay. Bayesian online changepoint detection. Cambridge, UK, 2007.

[140] Darren Bounds. Packit - Packet analysis and injection tool. `http://linux.die.net/man/8/packit`.

[141] Vinod Yegneswaran, Paul Barford, and Vern Paxson. Using honeynets for internet situational awareness. In *Proceedings of the Fourth Workshop on Hot Topics in Networks (HotNets IV)*, pages 17–22, 2005.

[142] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[143] M Daly. Advanced persistent threat. *Usenix, Nov*, 4, 2009.

[144] Yinglian Xie, Fang Yu, Kannan Achan, Rina Panigrahy, Geoff Hulten, and Ivan Osipkov. Spamming botnets: signatures and characteristics. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 171–182. ACM, 2008.

[145] Panda Security. Worldwide infected machines. `http://tinyurl.com/o24ky8t`.

[146] ScMagazine-McAfee. The state of malware in 2013. `http://tinyurl.com/ohjprsc`.

[147] Parliament of Canada. BILL C-28. `http://tinyurl.com/avh9vzv`.

[148] Shuaibu Hassan Usman. A review of responsibilities of internet service providers toward their customers' network security. *Journal of Theoretical and Applied Information Technology*, 49(1), 2013.

[149] ZDNet. ISPs accused of ignoring botnet invasion. `http://tinyurl.com/lt48jzl`.

[150] Kevin Dowd, Andrew JG Cairns, David Blake, Guy D Coughlan, David Epstein, and Marwa Khalaf-Allah. Evaluating the goodness of fit of stochastic mortality models. *Insurance: Mathematics and Economics*, 47(3):255–265, 2010.

[151] Yosiyuki Sakamoto, Makio Ishiguro, and Genshiro Kitagawa. Akaike information criterion statistics. *Dordrecht, The Netherlands: D. Reidel*, 1986.

[152] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[153] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of the 2014 Network and Distributed System Security (NDSS) Symposium*, February 2014.

[154] Steven J Templeton and Karl E Levitt. Detecting spoofed packets. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 164–175. IEEE, 2003.

[155] Jesse Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, 3, Supplement(0):91 – 97, 2006. The Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06).

[156] Yo-Ping Huang, Tsun-Wei Chang, and F.-E. Sandnes. An efficient fuzzy hashing model for image retrieval. In *Fuzzy Information Processing Society, 2006. NAFIPS 2006. Annual meeting of the North American*, pages 223–228, 2006.

[157] Jesse Kornblum. Fuzzy Hashing. `http://jessekornblum.com/presentations/cdfsl07.pdf`.

[158] Jesse Kornblum.]A Fuzzy Future in Malware Research. `http://tinyurl.com/obdkyg5`.

[159] Wenke Lee and Dong Xiang. Information-theoretic measures for anomaly detection. In *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, pages 130–143, 2001.

[160] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology,*, 15(1):52–60, 1967.

[161] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. The mahalanobis distance. *Chemometrics and intelligent laboratory systems*, 50(1):1–18, 2000.

[162] Hubert W Lilliefors. On the kolmogorov-smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62(318):399–402, 1967.

[163] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of p2p traffic. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04, pages 121–134, New York, NY, USA, 2004. ACM.

[164] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[165] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[166] Riyad Alshammari and A. Nur Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer Networks*, 2011.

[167] Leyla Bilge et al. Exposure: Finding malicious domains using passive dns analysis. In *NDSS 2011*.

[168] Jung Jaeyeon et al. Fast portscan detection using sequential hypothesis testing. In *2004 IEEE S&P*.

[169] Derek Leonard and Dmitri Loguinov. Demystifying service discovery: implementing an internet-wide scanner. In *The 10th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2010. ACM.

[170] Koji NAKAO, Daisuke INOUE, Masashi ETO, and Katsunari YOSHIOKA. Practical correlation analysis between scan and malware profiles against zero-day attacks based on darknet monitoring. *IEICE Transactions on Information and Systems*, 92(5):787–798, may 2009.

[171] D. Inoue, M. Eto, K. Yoshioka, S. Baba, K. Suzuki, J. Nakazato, K. Ohtaka, and K. Nakao. nicter: An incident analysis system toward binding network monitoring with malware analysis. In *Information Security Threats Data Collection and Sharing, 2008. WISTDCS '08.*, pages 58–66, 2008.

[172] Jungsuk Song, Jumpei Shimamura, Masashi Eto, Daisuke Inoue, and Koji Nakao. Correlation analysis between spamming botnets and malware infected hosts. *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*, 0:372–375, 2011.

[173] Masashi Eto, Kotaro Sonoda, Daisuke Inoue, Katsunari Yoshioka, and Koji Nakao. A proposal of malware distinction method based on scan patterns using spectrum

analysis. In ChiSing Leung, Minho Lee, and JonathanH. Chan, editors, *Neural Information Processing*, volume 5864 of *Lecture Notes in Computer Science*, pages 565–572. Springer Berlin Heidelberg, 2009.

[174] Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security*, volume 7, pages 1–16, 2007.

[175] David Whyte, Evangelos Kranakis, and Paul C van Oorschot. Dns-based detection of scanning worms in an enterprise network. In *NDSS*, 2005.

[176] Stuart E Schechter, Jaeyeon Jung, and Arthur W Berger. Fast detection of scanning worm infections. In *Recent Advances in Intrusion Detection*, pages 59–81. Springer, 2004.

[177] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescap. Analysis of a "/0" Stealth Scan from a Botnet. *IEEE/ACM Transactions on Networking*, 2014.

[178] Alberto Dainotti, Alistair King, and Kimberly Claffy. Analysis of internet-wide probing using darknets. In *Proceedings of the 2012 ACM Workshop on Building analysis datasets and gathering experience returns for security*, BADGERS '12, pages 13–14, New York, NY, USA, 2012. ACM.

[179] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, IMC '06, pages 41–52, New York, NY, USA, 2006. ACM.

[180] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 1–12, New York, NY, USA, 2000. ACM.

[181] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. Bgp routing stability of popular destinations. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 197–202. ACM, 2002.

[182] Bill Cheswick, Hal Burch, and Steve Branigan. Mapping and visualizing the internet. In *USENIX Annual Technical Conference, General Track*, pages 1–12. Citeseer, 2000.

[183] K Claffy, Tracie E Monk, and Daniel McRobb. Internet tomography. *Nature*, 7(11), 1999.

[184] Thomas Martin Lehmann, Claudia Gonner, and Klaus Spitzer. Survey: Interpolation methods in medical image processing. *Medical Imaging, IEEE Transactions on*, 18(11):1049–1075, 1999.

[185] Antoni Zygmund. *Trigonometric series*, volume 1. Cambridge university press, 2002.

[186] Ronald Newbold Bracewell and RN Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[187] Apurba Das. Discrete time transformations: Dtfs and dtft. In *Signal Conditioning*, pages 147–158. Springer, 2012.

[188] C Edward Sandifer. The early mathematics of leonhard euler. *Washington, DC*, 2007.

[189] Vince Fuller and Tony Li. Classless inter-domain routing (cidr): The internet address assignment and aggregation plan. 2006.

[190] Greg Welch and Gary Bishop. An introduction to the kalman filter, 1995.

[191] SY Chen. Kalman filter for robot vision: a survey. *Industrial Electronics, IEEE Transactions on*, 59(11):4409–4420, 2012.

[192] Dan Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, 2010.

[193] G Wayne Morrison and David H Pike. Kalman filtering applied to statistical forecasting. *Management Science*, 23(7):768–774, 1977.

[194] Kristof Van Beeck, Toon Goedemé, and Tinne Tuytelaars. Towards an automatic blind spot camera: robust real-time pedestrian tracking from a moving camera. In

Proceedings of the twelfth IAPR Conference on Machine Vision Applications, pages 528–531, 2011.

[195] Chris A Lightcap and Scott A Banks. An extended kalman filter for real-time estimation and control of a rigid-link flexible-joint manipulator. *Control Systems Technology, IEEE Transactions on*, 18(1):91–103, 2010.

[196] David N DeJong, Roman Liesenfeld, Guilherme V Moura, Jean-François Richard, and Hariharan Dharmarajan. Efficient likelihood evaluation of state-space representations. *The Review of Economic Studies*, 80(2):538–567, 2013.

[197] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: theory and practice using MATLAB*. John Wiley & Sons, 2011.

[198] Zhichun Li, Anup Goyal, and Yan Chen. Honeynet-based botnet scan traffic analysis. In *Botnet Detection*, pages 25–44. Springer, 2008.

[199] Simone Faro and Thierry Lecroq. An efficient matching algorithm for encoded dna sequences and binary strings. In *Combinatorial Pattern Matching*, pages 106–115. Springer, 2009.

[200] Zhichun Li, Anup Goyal, Yan Chen, and Vern Paxson. Towards situational awareness of large-scale botnet probing events. *Information Forensics and Security, IEEE Transactions on*, 6(1):175–188, 2011.

[201] Gregory C Chow and An-loh Lin. Best linear unbiased interpolation, distribution, and extrapolation of time series by related series. *The review of Economics and Statistics*, pages 372–375, 1971.

[202] Roque B Fernandez. A methodological note on the estimation of time series. *The Review of Economics and Statistics*, pages 471–476, 1981.

[203] Milton Friedman. The interpolation of time series by related series. *Journal of the American Statistical Association*, 57(300):729–757, 1962.

[204] Catherine Forbes, Merran Evans, Nicholas Hastings, and Brian Peacock. *Statistical distributions*. John Wiley & Sons, 2011.

[205] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[206] George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[207] Zhou Wang and Alan C Bovik. Mean squared error: love it or leave it? a new look at signal fidelity measures. *Signal Processing Magazine, IEEE*, 26(1):98–117, 2009.

[208] Josep Díaz, Jordi Petit, and Maria Serna. A survey of graph layout problems. *ACM Computing Surveys (CSUR)*, 34(3):313–356, 2002.

[209] Kenta Ozeki and Tomoki Yamashita. Spanning trees: A survey. *Graphs and Combinatorics*, 27(1):1–26, 2011.

[210] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[211] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[212] Paul Erd6s and A Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

[213] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[214] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.

[215] Riyad Alshammari and A Nur Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer networks*, 55(6):1326–1350, 2011.

[216] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning*, page 29. ACM, 2004.

[217] Jonathon Shlens. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*, 2014.

[218] Yu Jin, György Simon, Kuai Xu, Zhi-Li Zhang, and Vipin Kumar. Grays anatomy: Dissecting scanning activities using ip gray space analysis. *Usenix SysML07*, 2007.

[219] Yu Jin, Zhi-Li Zhang, Kuai Xu, Feng Cao, and Sambit Sahu. Identifying and tracking suspicious activities through ip gray space analysis. In *Proceedings of the 3rd annual ACM workshop on Mining network data*, MineNet '07, pages 7–12, New York, NY, USA, 2007. ACM.

[220] Zhichun Li, Anup Goyal, Yan Chen, and Vern Paxson. Automating analysis of large-scale botnet probing events. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ASIACCS '09, pages 11–22, New York, NY, USA, 2009. ACM.

[221] Vinod Yegneswaran, Paul Barford, and Vern Paxson. Using honeynets for internet situational awareness. *In Proc. of ACM Hotnets IV*, 2005.

[222] Darcy Benoit and André Trudel. World's first web census. *International Journal of Web Information Systems*, 3(4):378, 2007.

[223] John Heidemann, Yuri Pradkin, Ramesh Govindan, Christos Papadopoulos, Genevieve Bartlett, and Joseph Bannister. Census and survey of the visible internet. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, IMC '08, pages 169–182, New York, NY, USA, 2008. ACM.

[224] Y Pryadkin, R Lindell, J Bannister, and R Govindan. An empirical evaluation of ip address space occupancy. *USC/ISI Technical Report ISI-TR*, 598, 2004.

[225] Ang Cui and Salvatore J. Stolfo. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In *Proceedings of the 26th Annual*

Computer Security Applications Conference, ACSAC '10, pages 97–106, New York, NY, USA, 2010. ACM.

[226] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: detecting malware infection through ids-driven dialog correlation. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, pages 12:1–12:16, Berkeley, CA, USA, 2007. USENIX Association.

[227] Jan Goebel and Thorsten Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets (USENIX HotBots)*, pages 8–8. Cambridge, MA, 2007.

[228] Peter Wurzinger, Leyla Bilge, Thorsten Holz, Jan Goebel, Christopher Kruegel, and Engin Kirda. Automatically generating models for botnet detection. In Michael Backes and Peng Ning, editors, *Computer Security ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pages 232–249. Springer Berlin Heidelberg, 2009.

[229] Florian Tegeler, Xiaoming Fu, Giovanni Vigna, and Christopher Kruegel. Botfinder: finding bots in network traffic without deep packet inspection. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, CoNEXT '12, pages 349–360, New York, NY, USA, 2012. ACM.