Securing the Internet by Analysing and Controlling DNS Traffic: Email Worm Detection and Mitigation

vorgelegt von
Diplom-Ingenieur
Nikolaos Chatzis

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Hans-Ulrich Heiß
Berichter:	Prof. Dr. Radu Popescu-Zeletin
Berichter:	Prof. Dr. Jean-Pierre Seifert
Berichter:	Prof. Dr. Dimitrios Serpanos

Tag der wissenschaftlichen Aussprache: 17. November 2010

Berlin 2010

D 83

Abstract

The Domain Name System (DNS) is a critical infrastructure of the Internet because almost all applications that run on Internet-connected machines depend on the name resolution service it provides to work. The DNS consists of three components: the domain name space, the name servers, and the clients, formally referred to as resolvers. Due to its critical nature, the domain name space and the name servers have been for many years very attractive targets for attackers seeking to inflict widespread damage. To deal with this state of affairs, substantial attention and investment have been directed at enhancing the security of and protecting the DNS to ensure its continuous, reliable and efficient operation. This, in conjunction with a notable shift in the motivation and profile of attackers have led in recent years to a considerable change in the Internet attack landscape. Attacks have gradually become more sophisticated and focused, and financial gain has evolved into the major driving force behind them. In this new era, attackers have realised that misusing the name servers or exploiting the name resolution service comes with greater damage or economic profit than directly attacking the components of the DNS or disrupting the name resolution service. As an immediate consequence, the vast majority of Internet attacks nowadays produce an observable effect on the DNS traffic that traverses the Internet, the operation of the name servers, or in some cases on both. In the present study, it is shown that this observation opens a new and very promising perspective for effectively detecting and mitigating a wide variety of Internet attacks.

To demonstrate the value of this perspective, the present study is devoted to detecting and mitigating Internet worms that along with bot software are the two major Internet threats network operators and end users face. The focus is particularly on email worms, which have been, and remain, a very popular medium for attackers to achieve their ends and, therefore, the most prevalent type of Internet worms and malicious software in general. The attackers' ends include installing bot software designed to distribute unsolicited emails or launch targeted distributed denial of service attacks, stealing private information and destroying key data. In this thesis, a method for detecting user machines that are compromised by email worms on the local name servers is introduced. The method uses clustering and similarity search over time series derived from the DNS query streams of user machines. It is demonstrated that the method overcomes the limitations of the existing methods, exhibits remarkable accuracy and negligible false alarm rate, and can be effective in the long run. In addition, a method for containing email worms is introduced. The method uses a traffic control mechanism to regulate the DNS response streams that the local name servers return to user machines and, thereby, limit the rate at which compromised user machines spread email worms further. It is shown that the method has the potential to slow down the epidemics of email worms and contribute to reducing the illegitimate email and DNS traffic compromised user machines send to the Internet with minimally, if at all, affecting their legitimate traffic.

Zusammenfassung

Das Domain Name System (DNS) ist eine für das Internet unentbehrliche Infrastruktur, weil fast alle Anwendungen, die auf mit dem Internet verbundenen Maschinen laufen, von der Namensauflösung, die es zur Verfügung stellt, abhängen. Das DNS besteht aus drei Komponenten: dem Domain-Namensraum, den Nameservern und den Klienten, formal Resolver genannt. Wegen seiner kritischen Natur waren der Domain-Namensraum und die Nameserver jahrelang sehr attraktive Ziele für Angreifer, die versuchen, weit verbreiteten Schaden zuzufügen. Um diesen Stand der Dinge zu überwinden, wurde große Aufmerksamkeit auf die Verbesserung der Sicherheit sowie den Schutz des DNS gerichtet, und bedeutende Investitionen getätigt, um seinen durchgehenden, zuverlässigen und effizienten Betrieb sicherzustellen. In Verbindung mit einer bemerkenswerten Verschiebung in der Motivation und im Profil der Angreifer hat dies in den letzten Jahren zu einer beträchtlichen Änderung in der Internet-Angriffslandschaft geführt. Die Angriffe wurden schrittweise verfeinert und fokussiert, und finanzieller Gewinn hat sich zur ihrer treibenden Hauptkraft entwickelt. In dieser neuen Ära haben die Angreifer festgestellt, dass der Missbrauch der Nameserver oder die Ausnutzung des Namensauflösungdienstes größeren wirtschaftlichen Schaden verspricht, als direkte Angriffe auf die DNS-Komponenten oder das Stören des Namensauflösungdienstes. Als direkte Konsequenz hat heutzutage die überwiegende Mehrheit von Internet-Angriffen einen sichtbaren Effekt auf den DNS-Verkehr, der das Internet durchquert, auf den Betrieb der Nameserver, oder in einigen Fällen auf beides. In dieser Studie wird demonstriert, dass diese Beobachtung eine neue und sehr viel versprechende Perspektive bietet, um viele Internet-Angriffe effektiv zu erkennen und abzuschwächen.

Um den Wert dieser Perspektive zu demonstrieren, behandelt die vorliegende Studie die Erkennung und Abschwächung von Internet-Würmern, die zusammen mit Bot-Software die zwei Hauptbedrohungen für die Netzwerk-Betreiber und Endbenutzer im Internet sind. Der Fokus liegt besonders auf Email-Würmern, die ein sehr populäres Mittel sind, damit Angreifer ihre Zwecke erreichen, und deshalb die überwiegende Ausprägung der Internet-Würmer und Schadprogramme im Allgemeinen gewesen sind und bleiben. Diese Zwecke umfassen die Installation von Bot-Software, entworfen um unaufgefordert Emails zu verteilen und gezielte verteilte Leistungsverweigerungsangriff zu starten, den Diebstahl von privaten Informationen und das Zerstören von sensitiven Daten. In dieser Arbeit wird eine Methode für die Erkennung von Benutzermaschinen, die mit Email-Würmern angesteckt sind, auf dem lokalen Nameserver vorgestellt. Die Methode benutzt Clustering und Ähnlichkeitssuche über Zeitreihen, abgeleitet aus DNS-Anfragesträmen, die Benutzermaschinen erzeugen. Es wird gezeigt, dass sie die Beschränkungen der vorhandenen Methoden überwindet, bemerkenswerte Genauigkeit und eine unwesentliche Rate von Falsch-Positiv-Meldungen erreicht und langfristig wirkungsvoll sein kann. Zusätzlich wird eine Methode für die Eingrenzung von Email-Würmern eingeführt. Die Methode benutzt einen Verkehrssteuerungsmechanismus, um die DNS-Antwortströme zu regulieren, die lokale Nameserver zu den Benutzermaschinen zurückschicken und dadurch die Rate, mit der infizierte Benutzermaschinen Email-Würmer weiter verbreiten. Es wird gezeigt, dass sie das Potenzial hat, Email-Wurmepidemien zu verlangsamen und zur Verringerung von illegitimem Email-und DNS-Verkehr, den angesteckte Benutzermaschinen ins Internet senden, zu beitragen, mit keinem bis minimalem Einfluss auf den legitimen Benutzerverkehr.

Acknowledgements

Working toward a PhD is a lonely and difficult task, yet one that cannot be successfully completed without the assistance of others. My own experience was no different, and there are some people to whom I owe a great debt of gratitude.

First, I would like to express my profound and most sincere gratitude to my adviser, Professor Radu Popescu-Zeletin, for his time, guidance and for offering me the opportunity to carry out my PhD research work at the Fraunhofer Institute FOKUS as well as the freedom to pursue a research topic of my own interest.

I am also very thankful to Professors Jean-Pierre Seifert and Dimitrios Serpanos for their critical review and valuable suggestions on earlier versions of this thesis. I owe my deepest thanks to Associate Professor Nevil Brownlee for believing in my ideas, accepting to co-author my papers and providing constructive feedback.

During my time at the Fraunhofer Institute FOKUS, I have had the pleasure and privilege to closely interact with two great students, Enric Pujol and Silke Peters. I would like to thank them for all the stimulating discussions, invaluable input and their companionship, patience and understanding in the course of this work.

My appreciation and respect go to three current and former colleagues at the Fraunhofer Institute FOKUS, Ranganai Chaparadza, Thomas Hirsch and Mikhail Smirnov. Without their invaluable advice, encouragement and support in various ways and times, it would have taken me much longer to complete this work.

Finally, this work would never have been possible without the unconditional love and support of my friends and family. I am particularly indebted to Ilias, Enric, Angelos, Evgenia and last, but not least, to Eleni for always being there for me as a constant source of inspiration, motivation and strength over the past years.

Contents

List of Figures

List of Tables

Part I

Introduction, Background and Motivation

Chapter 1

Introduction

1.1 Research Objectives

Every country's society in general, and every individual in particular, depends everyday on a number of services provided by large logical and spatial networks, which are commonly known as critical infrastructures. Although the definition of a critical infrastructure often differs from one country to another, in almost every country the networks whose operation is of fundamental importance to the national stability and socio-economic development include: the energy, water supply, transportation, banking and finance, and telecommunications networks [1]. Due to the dependency of every country's society on the services of critical infrastructures, improving the security level of critical infrastructures has gradually become, and remains, a top priority issue (see, for instance, [46, 71]), and as such it has attracted, globally, a great deal of interest and investment in recent years.

Based on the rationale that any accidental or deliberate quality degradation or interruption of the services provided by the critical infrastructures can have serious socio-economic implications, and thus it should be avoided at all costs, attention has been mainly directed at three areas. The focus of the first area has been on ensuring that the components of critical infrastructures are adequately protected and on identifying and reducing vulnerabilities and single points of failure in them. The emphasis of the second area has been on automatically detecting as early as possible security incidents that affect the operation of the components of critical infrastructures, generating informative alarms and limiting the damage that these incidents can cause. The focus of the third area has been on developing strategies for rapid recovery and restoration of the services provided by critical infrastructures or their affected components and post-incident root cause analysis.

The ever-growing in number and sophistication measures for securing critical infrastructures on the one hand, and the increasing awareness among attackers that taking advantage of critical infrastructures can produce greater damage or economic

profit than attacking them on the other, has led to a notable change in the attack landscape. Specifically, in recent years, attacks that involve exploiting the services or misusing the components of critical infrastructures to disrupt or destroy other targets have increased in frequency and diversity. As an immediate consequence, the distinction between the use of critical infrastructures as a target and as an attack medium has been constantly blurring. As a response to this change, a number of initiatives that aim at developing strategies for preventing attackers from using critical infrastructures to achieve their illegitimate ends are currently in progress (see, for instance, [44]). Central to most of these initiatives is the analysis of information collected on the components of the critical infrastructures that makes it possible to distinguish legitimate users from those with malicious intent.

In the present study, this approach is followed and explored, and as a key result, it is shown that the shift in research emphasis it implies is useful for improving Internet security. In particular, it is demonstrated that profiling user machines based on how they use the name resolution service provided by the Domain Name System (DNS) and acting accordingly, represents a new and promising perspective to deal with a wide variety of Internet attacks in an effective manner. The justification for the analogy is as follows. The DNS is organised as an Internet-wide network of name servers and is a critical infrastructure of the Internet. Its critical nature is grounded in the fact that almost all applications that run on Internet-connected machines depend on the name resolution service to work. Most of the malicious programs that are released in the wild and cause severe damage and economic loss are not an exception to this rule. Over the past years, protecting the DNS from outages and attacks has received considerable attention and investment that has resulted in a significant decrease in the frequency of attacks targeting the DNS (see, for instance, [72, 179]). By contrast, the number of malicious programs designed to compromise user machines and launch attacks against user machines or other Internet-connected machines that appear daily steadily grows.

Internet worms have consolidated their position as a major Internet threat. The term Internet worm is used in this thesis, similarly to Kienzle et al. [98], to refer to every malicious program that propagates on a computer network independent of whether human interaction is required or not. Email is an essential communication medium for today's connected society and, as such, it has evolved into the main propagation vector of Internet worms. Although much effort has been dedicated in the research and commercial communities to come up with strategies to counter them, email worms – Internet worms that spread primarily via email – remain a serious threat to network operators and end users. From a network operators' perspective, email worms are considered responsible for waste of storage, network congestion and loss of service or degradation in the performance of network resources. The reason for this is that they are directly and indirectly associated with the high amount of unsolicited email traffic traversing the Internet. From the end users' point of view,

email worms are a serious security threat because they are the dominant medium attackers use to install malicious programs on their machines.

In light of the above discussion, the study reported in this thesis had three primary research objectives. The first of them was to look at Internet threats from a DNS perspective and uncover that many of them involve exploiting the DNS, and thereby produce an observable effect on DNS traffic. The second objective was to conduct a comprehensive and critical review of the available methods for detecting and mitigating Internet worms, and in particular email worms, which are the most prevalent type of Internet worms and one of the major threats to network operators and end users. The underlying goal in doing so was to highlight that and provide insight into why these methods have met with little success against email worms. The third objective was to demonstrate that analysing and controlling the DNS traffic that traverses the Internet is for the Internet security community a new and very promising direction to explore. To justify this, two methods that operate on the local name servers and have the potential to be effective in dealing with email worms in the long run are introduced. Specifically, one of them detects user machines compromised by email worms by analysing the traffic that local name servers receive, and the other slows down the Internet-scale epidemics of email worms by controlling the traffic that local name servers send back to user machines.

1.2 Key Contributions

This work makes the following contributions to the field of Internet security:

- It reveals that a wide variety of serious Internet threats have an observable effect on the DNS, which represents a critical infrastructure of the Internet. In connection with this, it proposes that analysing and controlling the DNS traffic that traverses the Internet can play a key role for improving Internet security.

- It focuses on combating email worms and the high amount of illegitimate traffic associated with them and makes two contributions. First, it presents a method that can automatically detect user machines compromised by email worms by analysing on the local name servers the DNS traffic they produce. Second, it introduces a method that can automatically limit the propagation speed of email worms and the illegitimate traffic compromised user machines generate by controlling the DNS traffic that local name servers return to user machines.

- It shows that exact shape-based similarity search over time series, which has attracted increasing interest within the database and data mining community and is a central element of the proposed detection method, is a very useful tool for distinguishing between normal and anomalous network activity. In addition, it presents a computational model that can be useful in obtaining

a better understanding of the Internet-scale spreading of email worms and exploring the potential efficacy of novel methods for containing email worms.

- It provides a thorough review and a detailed analysis of the merits and weaknesses of the existing methods and research directions for dealing with Internet worms. Thereby, it offers insight into the full dimensions of this Internet threat, the current solutions and main recommendations for future research directions.

1.3 Thesis Organisation

The remainder of this thesis consists of six chapters and it is organised as follows:

- Chapter 2 briefly presents the function, the overall architecture and the inner mechanisms of the DNS. Then, it demonstrates that many Internet threats produce an observable effect on the DNS traffic. Finally, it explains the reasons why the present study is devoted to email worms.

- Chapter 3 provides a thorough review and a detailed analysis of the existing methods for detecting machines compromised by Internet worms. It identifies the dimensions of their design space and uncovers their limitations that render them ineffective or inefficient against email worms.

- Chapter 4 introduces a method that classifies user machines as clean or compromised by an email worm based on the DNS traffic they generate. It presents the principle, key elements and experimental results of the method, and discusses approaches worm writers might take to bypass it.

- Chapter 5 gives a comprehensive and critical review of the existing methods for mitigating Internet worms. It focuses particularly on the existing containment methods, identifies the dimensions of their design space and uncovers why they are not effective or efficient against email worms.

- Chapter 6 describes a method that delays the propagation of email worms and limits the illegitimate traffic compromised user machines produce. It presents the principle and key elements of the method and the simulation model used to evaluate it, and reports on simulation results.

- Chapter 7 restates the key contributions that the present study makes to the field of Internet security. Then, it summarises the strengths and limitations of the two proposed methods. Finally, it highlights a number of new research directions that have been opened with this work.

Chapter 2

Background and Motivation

2.1 DNS Role, Components and Operation

The DNS is today the largest distributed system in operation and a critical infrastructure of the Internet. The service it provides to Internet-connected machines is to supply them with various types of information about Internet resources, such as Internet Protocol (IP) addresses or the email servers of each network of the Internet that are responsible for accepting incoming emails. Primarily, it maps alphanumeric (human-friendly) domain names, like www.example.com, to the corresponding numerical (machine-friendly) IP addresses, let it be 208.77.188.166 for www.example.com, and vice versa. The mapping between domain names and IP addresses is formally referred to as name resolution and is an essential element for the operation of the vast majority of applications that run on Internet-connected machines. The DNS works according to the client-server model and in connection with this it consists of three components: the domain name space (data), the name servers, and the clients, known as resolvers in DNS terminology.

The domain name space is organised as a tree structure of domain names. Each node or leaf of the tree that is administered by a single organisation is called a DNS zone. Each DNS zone is associated with a text file, called a zone file, which contains a list of entries, each of them called a resource record (RR). Each RR is a quintuple that maps a resource to a domain name. The five fields of a RR are the following: name, time-to-live (TTL), type, class, and resource record data. Name is the domain name and TTL the time after which the RR should be regarded as out of date, if it is temporarily stored (cached) on a name server or a resolver. Type and class take standard mnemonic values that determine the application and the application family of the RR, respectively; in all RRs, except in very rare cases, type is set to Internet Class (IN). The resource record data is a variable-length field that corresponds to the data portion of the RR, and is to be interpreted in the context of class and type. The precise format and the various RR types were first defined in RFC 1035, and

since then, they have been revised or extended in a number of more recent RFCs (see, for instance, RFCs 1183, 1706 and 2782).

The name servers are specialised database servers organised as an Internet-wide hierarchical network. There exist various configuration options for a name server that correspond to the various functional roles it can have in a network. Primarily, a name server can be configured as master (primary), slave (secondary), caching or forwarding (proxy). A master name server stores one or more zone files and provides authoritative responses for the RR contained in them. This means that it responds to DNS queries for the RRs of the corresponding DNS zones by reading its file system. The critical nature of the DNS requires that at least two name servers support each DNS zone. Thereby, a slave name server obtains its zone information from a master, and is the backup if the master name server fails. The master and slave name servers are referred to as authoritative name servers and the zone files stored in their file systems as authoritative data. A caching name server temporarily stores the RRs that are carried in the DNS responses it receives from authoritative name servers until their TTLs expire, and responds to the relevant DNS queries it receives from resolvers with these RRs. As its name suggests, a forwarding name server forwards the DNS queries it receives from resolvers to other name servers, and usually caches the RRs carried in the responses.

As previously mentioned, most Internet applications rely on the domain name space to work. This includes the name resolution service itself since, as shown later in this section, the DNS resolves domain names in a recursive manner. Hence, a resolver is installed on almost every Internet-connected machine. Its role is to translate the needs of the applications to DNS queries and send them to name servers. There are two types of DNS queries a resolver can issue: iterative and recursive. An iterative query results in either a complete response or a reference to a name server that is authoritative for the queried RRs or a part of them. Recursive queries ask name servers to carry out all the necessary work to return complete answers. In their vast majority user machines and other Internet-connected machines, except for the name servers, run a lightweight (stub) resolver. Stub resolvers can generate recursive queries only, forward these queries to full resolvers, and resend them if they are not answered within a short period of time. Full resolvers can generate iterative queries, as well, and are mainly installed on name servers. Actually, this is the reason why in almost every network a name server is deployed topologically near the user machines (stub resolvers) and acts for them as the essential first link in the entire chain of Internet connectivity. This name server stands at the centre of the present study and is referred to as local name server hereafter.

To provide insight into their deployment point on the Internet as well as their communications with each other and the other Internet-connected machines, some common practices for the configuration of local and authoritative name servers are described in what follows. Local name servers are typically configured as caching

name servers and respond to recursive queries originating from stub resolvers in their network only. Thereby, they are usually located physically close to the machines they serve and have private IP addresses. Three reasons justify these configuration decisions. First, stub resolvers cannot generate iterative queries. Hence, the local name servers have to be capable of generating iterative queries and sending them to authoritative name servers on their behalf and returning to them complete responses. Second, deploying local name servers close to the machines that run stub resolvers and activating caching, contribute to improving the performance of the Internet applications that run on these machines. This is because doing so reduces the overall time needed to respond to the needs of the applications. Third, assigning private IP addresses to local name servers makes them better protected against Internet attacks that exploit recursion, caching, or both, such as the DNS cache poisoning attacks discussed latter in this chapter. By contrast, authoritative name servers normally use public IP addresses, since they have to be visible to the Internet, and do not support recursive queries and caching. There are two main reasons for the latter. First, for performance purposes, authoritative name servers usually do not serve stub resolvers and, second, recursion and caching expose a publicly-visible name server to many risks. To support the analysis presented above, Fig. 2.1 illustrates the necessary steps to resolve a sample domain name.

As a concluding remark, it is worth noting that beneath the seemingly simple name resolution service provided by the DNS lies a complex logical and administrative infrastructure. In connection with this, the short description of the DNS given above is necessarily incomplete and the interested reader is referred to one of the several fine books written about the DNS for further details and references (see, for instance, [5, 7]). Nevertheless, this section is generally complete with regard to its intended purpose. This is namely to introduce the reader to the function, the overall architecture and the inner mechanisms of the DNS placing particular emphasis on the aspects that are important for the present study.

2.2 Internet Threats from the DNS Viewpoint

The DNS stores and provides Internet-connected machines with information about Internet resources that is critical for the operation of most of the applications running on them. Thereby, looking at Internet threats from the DNS viewpoint, uncovers that the effects an attacker can achieve fall into three broad categories. The effects in the first category are connected with rendering a part of the domain name space or some name servers unavailable. Depending on the particular attack details, this can result in either making some resources unreachable for every resolver on the Internet or in failure of several applications, such as Web browsing and email, that run on some Internet-connected machines. The effects in the second category are coupled with gaining unauthorised access to name servers or manipulating RRs on

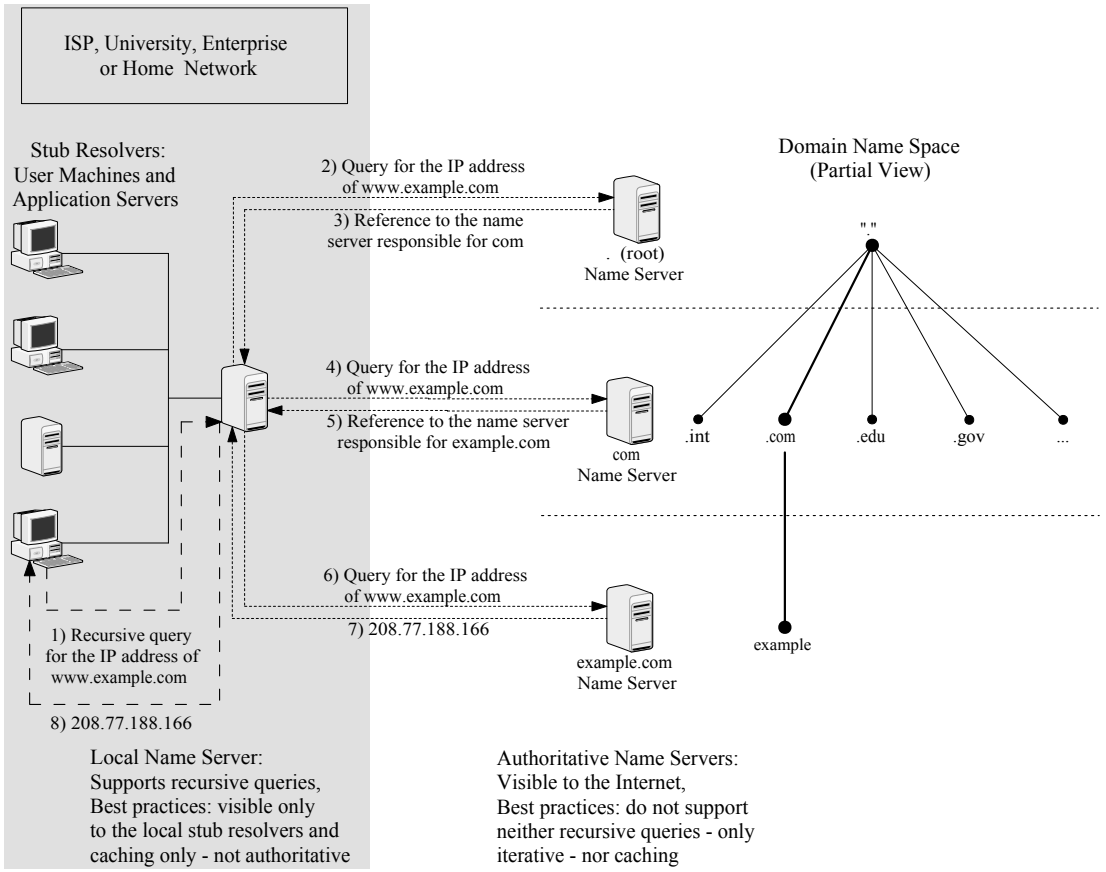

Figure 2.1: Every time an application that runs on a user machine needs the IP address of a domain name (www.example.com), the stub resolver on the user machine sends a recursive query to the local name server (Step 1). If the local name server does not have the response (or a part thereof) stored in its cache, it starts from the top of the name server hierarchy (root name servers) to sequentially query the authoritative name servers (Steps 2, 4 and 6). The authoritative name servers respond with referrals to other name servers that are lower in the hierarchy and are authoritative for a part of the queried domain name (Steps 3 and 5). Once the local name server receives the response to the query of the user machine (or a domain name not exists error) (Step 7), it forwards it to the user machine (Step 8). In the figure, some common practices for the configuration of local and authoritative name servers are listed. Unless otherwise stated, in all the figures in this chapter, the grey highlighted region marks the operational area of the local name server.

the name servers, the route between name servers and resolvers, or the resolvers. The effects in this category are typically sought by attackers willing to redirect unsuspecting end users to malicious and compromised Web sites. The effects in third category are associated with exploiting the DNS in a seemingly legitimate manner to mount attacks against various targets. This usually involves taking advantage of the

information in the domain name space in order to locate attack targets or spread malicious software, commonly referred to as malware, or using the name servers as attack weapons. As detailed later in this chapter, the attacks that relate to the effects in the third category have become increasingly popular among attackers. This is because it is hard for network operators to react to and suppress these attacks before they have already caused significant damage.

To achieve the effects in each category, attackers have the following three attack vectors at their disposal: design vulnerabilities in the DNS, vulnerabilities in DNS implementations, and exploitation of the name resolution service. For each of these attack vectors there exist several exploitation techniques. Each of these techniques is linked with one of the three components of the DNS – the domain name space, the name servers or the resolvers. A high-level overview of all the possible combinations of the three attack features mentioned so far – the vulnerable component of the DNS, the attack effect category and the attack vector used – is given in Fig. 2.2. Each of the next three subsections is devoted to presenting an attack vector and its exploitation techniques with the intent to accomplish three objectives. The first objective is to analyse the symptoms of the attack associated with each technique on the DNS. This analysis provides evidence that the majority of the attacks produce an observable effect on the DNS traffic. The second objective is to assess the severity of the attacks that are coupled with each attack vector. This involves discussing the popularity of each attack vector among attackers, the skills required for exploiting it and the existing countermeasures. This discussion is intended to uncover which attack vectors are being adequately addressed and which require more attention. The third objective is to demonstrate that the most serious and inadequately addressed Internet threats are strongly dependent on the name resolution service. In most cases, this is manifested by an observable effect on the characteristics of the DNS traffic that flows between user machines and the local name servers. This makes research on network security methods that are based on analysing and controlling DNS traffic, such as these introduced later in this thesis, of particular interest and significance. Before proceeding further, it should be noted that parts of the discussion that follows in this section have been published in [PP2].

2.2.1 Exploitation of DNS Design Vulnerabilities

During the design phase of the DNS in the early-to-mid 1980s, trustworthy communication among name servers and resolvers was not identified as a requirement (see [150] for a detailed list of the requirements). The reason for this is that at that time the Internet was a collection of military, academic and research networks serving a small community of trusted end users. Since then, however, the Internet has evolved into a massive network of networks that continues to grow rapidly and serves numerous governments, businesses and end users. As a result, today, the

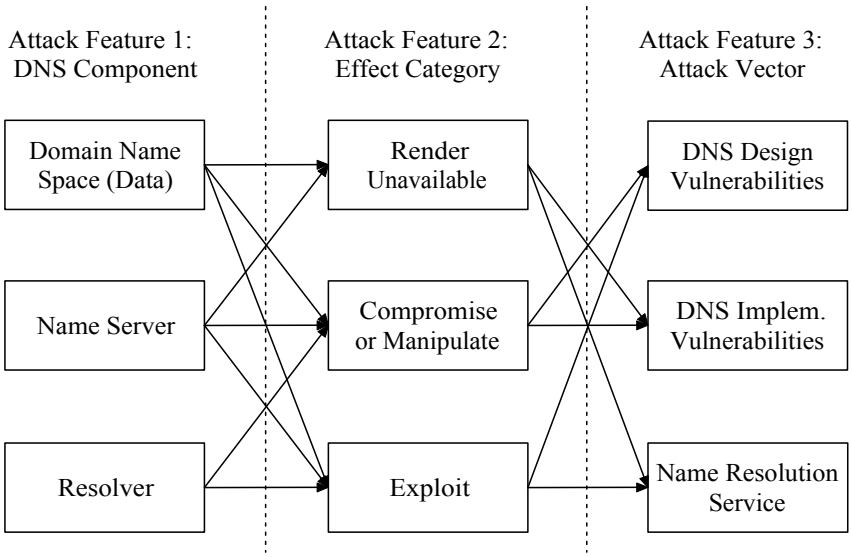

Figure 2.2: A high-level overview of all the possible combinations of the three DNS-related features – the vulnerable component of the DNS, the attack effect category and the attack vector exploited – that attackers consider for launching a large variety of Internet attacks.

implicit trust relationship among name servers and resolvers is widely regarded as a major design flaw. This is because, as will become clear in what follows, it provides attackers with an excellent avenue to launch various attacks.

Specifically, the lack of mechanisms to ensure that the RRs a resolver received came from an authoritative name server (source authentication) and that they were not modified in the network during the transit (data integrity), offers attackers a large attack surface to work with. An important side effect of these two weaknesses, which is often thought of as a serious weakness in its own right, is the absence of authenticating the non-existence of a domain name. The reason for this is that the non-existence of certain types of domain names, such as that of email servers that are responsible for accepting incoming emails (MX), results in actions from a resolver other than immediate failure. This poses a significant threat for penetrating user machines (see RFC 3833). Attackers normally take advantage of these two weaknesses to cause a denial of service (DoS) or misdirect end users to fraudulent Web sites (pharming) or proxy servers (phising). To achieve these ends, several exploitation techniques exist that can be divided into two categories. The classification criterion is if the attacker provides the manipulated RRs to the stub resolvers through a name server or directly. The techniques in the first category can be further subdivided into domain hijacking and cache poisoning (DNS spoofing).

To reserve a domain name for private or commercial use, an end user or an organisation needs to register the domain name with a registrar. A registrar is a company accredited to assign and manage domain names by the Internet Corporation

for Assigned Names and Numbers (ICANN), which is the organisation responsible for managing the DNS, or by a country code top-level domain (ccTLD) authority. The registrars maintain for each domain name registered with them all the required details, including the name of the holder and information about the authoritative name servers. Domain hijacking is the wrongful taking of control of a domain name from its rightful holder [78], commonly called registrant. In this thesis, to distinguish this illegal act from the legal re-registration of an expired domain name by another end user or organisation, domain hijacking is referred to as domain theft. Given that the process of registering a domain name is bilateral in nature, an attacker can attempt to victimise either the registrant to unwittingly transfer its domain name or the respective registrar to change the registration information. Either way, if successful, the attacker steals control of the domain name. This implies that the DNS traffic destined for the authoritative name servers that are associated with the domain name will go to, or through, illegitimate name servers.

Cache poisoning is the malicious act of inserting manipulated RRs in the cache of a name server that is configured to answer recursive queries and temporarily store the RRs that are contained in the DNS responses it receives. Thereby, in most cases, the victim servers are local name servers. Cache poisoning is not new but it is still widely used as the first step in attack sequences that result in pharming or phising. There exist the following three techniques to poison the cache of a name server: the classic, the chaining attack and the man-in-the-middle cache poisoning. The classic cache poisoning is known as Kashpureff-style after E. Kashpureff who in July 1996 diverted end users asking for InterNIC's Web site, which had the current role of ICANN until September 1998, to his AlterNIC Web site. In the Kashpureff-style cache poisoning, the attacker sends to the victim name server a DNS query for a domain name for which it controls the authoritative name servers. This forces the victim name server to query these authoritative name servers and cache the RRs it receives. The chaining attack is similar to this attack with the variation that instead of sending the initial DNS query to the victim name server directly, the attacker provokes a user machine, for example, by sending to it a crafted email, to do so. The Kashpureff-style and the chaining attack cache poisoning are illustrated in Fig. 2.3. In the man-in-the-middle cache poisoning, the attacker monitors the traffic that flows between the victim name server and authoritative name servers. Once the victim queries the authoritative name servers, the attacker attempts to send back to it a DNS response before the authoritative name servers do. The only obstacle for attackers is that they have to figure out the value of a $16-$bit ID field in the header of the DNS query that is used to match queries with responses. This process is called ID guessing. For a determined attacker checking all the 2^{16} possible combinations or a subset of them that with high probability contains the unknown ID is feasible in a short period of time. Since this attack is very much a race condition, attackers sometimes try to increase their probability to succeed by

slowing down the authoritative servers. A commonly-taken approach to achieve this is by instructing a network of compromised machines, commonly called botnet, to flood an authoritative name server with bogus queries. Both the attack scenario and its extension with the botnet are depicted in Fig. 2.4.

Instead of inserting manipulated RRs in the cache of a name server and waiting until the user machines query the name server for them, an attacker can provide the stub resolvers that run on user machines with manipulated RRs directly. In a similar way to the man-in-the-middle cache poisoning, the attacker monitors the traffic that flows between a user machine and the local name server in its network. By guessing or reading, by means of on-the wire eavesdropping, the ID in the header of a DNS query, the attacker attempts to send a crafted DNS response to the stub resolver before the local name server does. If the attacker succeeds in doing so, the legitimate response is dropped and the traffic of the victim user machine is diverted from its true destination. Another option is to corrupt the information about the local name server on the user machines. This is possible either remotely by exploiting implementation vulnerabilities in the operating system or the applications running on the user machines or by releasing a self-propagating malware specially designed to do so. In this attack scenario, the attacker masquerades as a name server or makes stub resolvers query name servers under its control. As a consequence, the local name server does not receive any DNS queries from the victim user machines. Thereby, this attack is detectable by analysing the outgoing DNS traffic from non-name servers on the gateway of a monitored network. The two attack scenarios for providing manipulated RRs to user machines are shown in Fig. 2.5.

The past several years have seen a surge of Internet security research in the field of information assurance. The main focus has been on protecting Internet information in storage, processing or transit by using authentication and encryption. In particular for the DNS, the Internet Engineering Task Force (IETF) has been working since 1996 on the DNS Security Extensions (DNSSEC) (see RFC 2535). The DNSSEC addresses cryptographically-based source authentication and integrity of DNS data, and it is aimed at securing both the DNS and end users from the threats discussed above. In more detail, the DNSSEC extends the specifications of the DNS by introducing mechanisms that provide authenticated denial of domain name existence and ensure that the RRs a resolver received are correct, consistent and came from an authoritative name server. Although some steps toward this direction have already been taken, the DNSSEC is not yet widely deployed. This is mainly due to three reasons. The first of them is that many researchers and network operators consider it being limited in scope because it does not cover confidentiality of data and authentication of the requester [40]. The second reason is that its design specifications are deemed imperfect [122], having many intrinsic weaknesses (see RFC 3833). The third reason is that, so far, no generally accepted mechanisms have been proposed for validating the public key and detecting whether the name servers

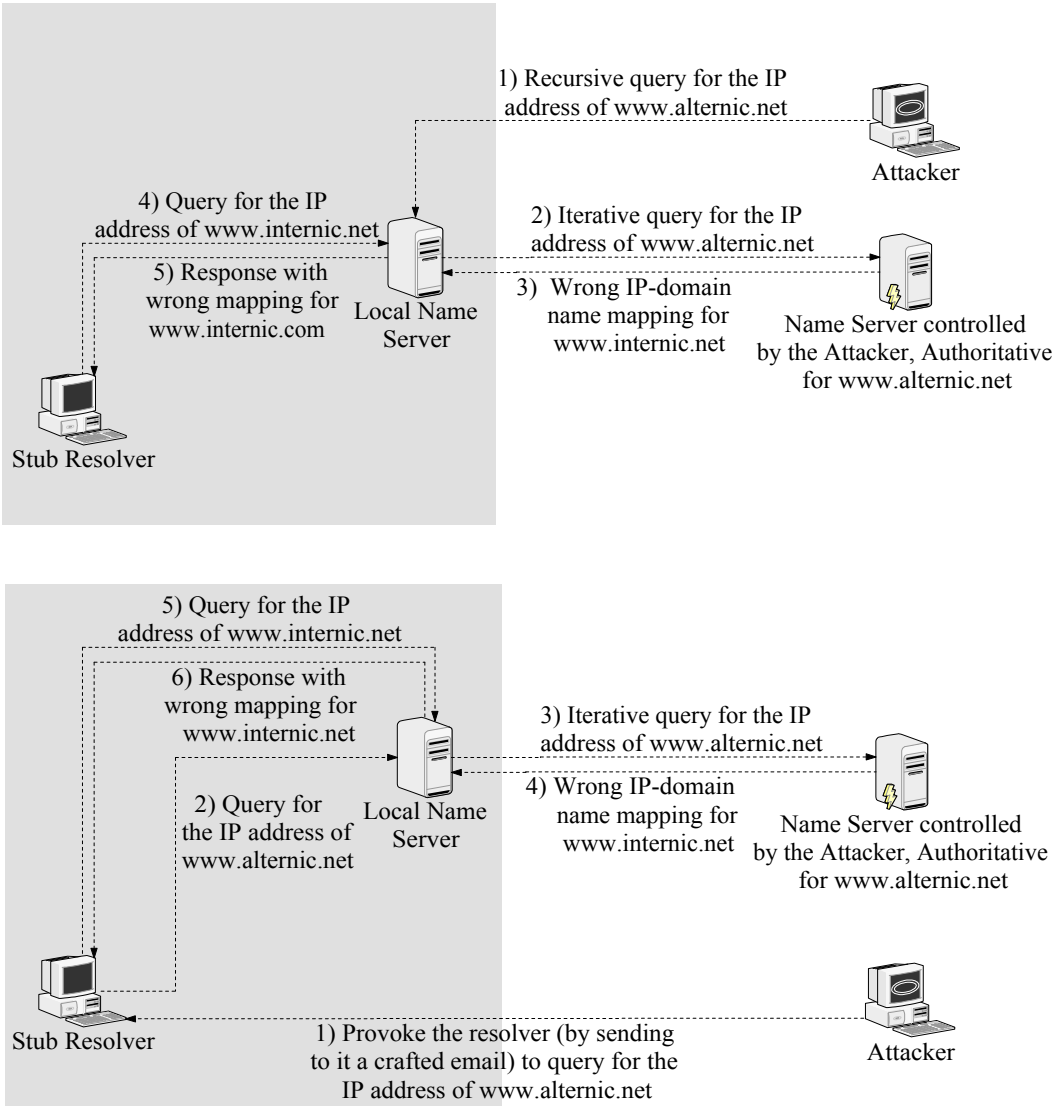

Figure 2.3: In the classic (Kashpureff-style) cache poisoning (upper plot), the attacker sends to the victim (local) name server a query for a domain name for which it controls the authoritative name servers (www.alternic.net). This makes the victim (local) name server query these authoritative name servers and cache the manipulated RRs (for www.internic.net) it receives (Steps 2 and 3). The victim (local) name provides stub resolvers with the manipulated RRs when queried for them (Steps 4 and 5). The chaining attack (lower plot) is similar to this attack with the variation that instead of sending the initial query to the victim (local) name server directly, the attacker provokes a stub resolver to do so (Step 1).

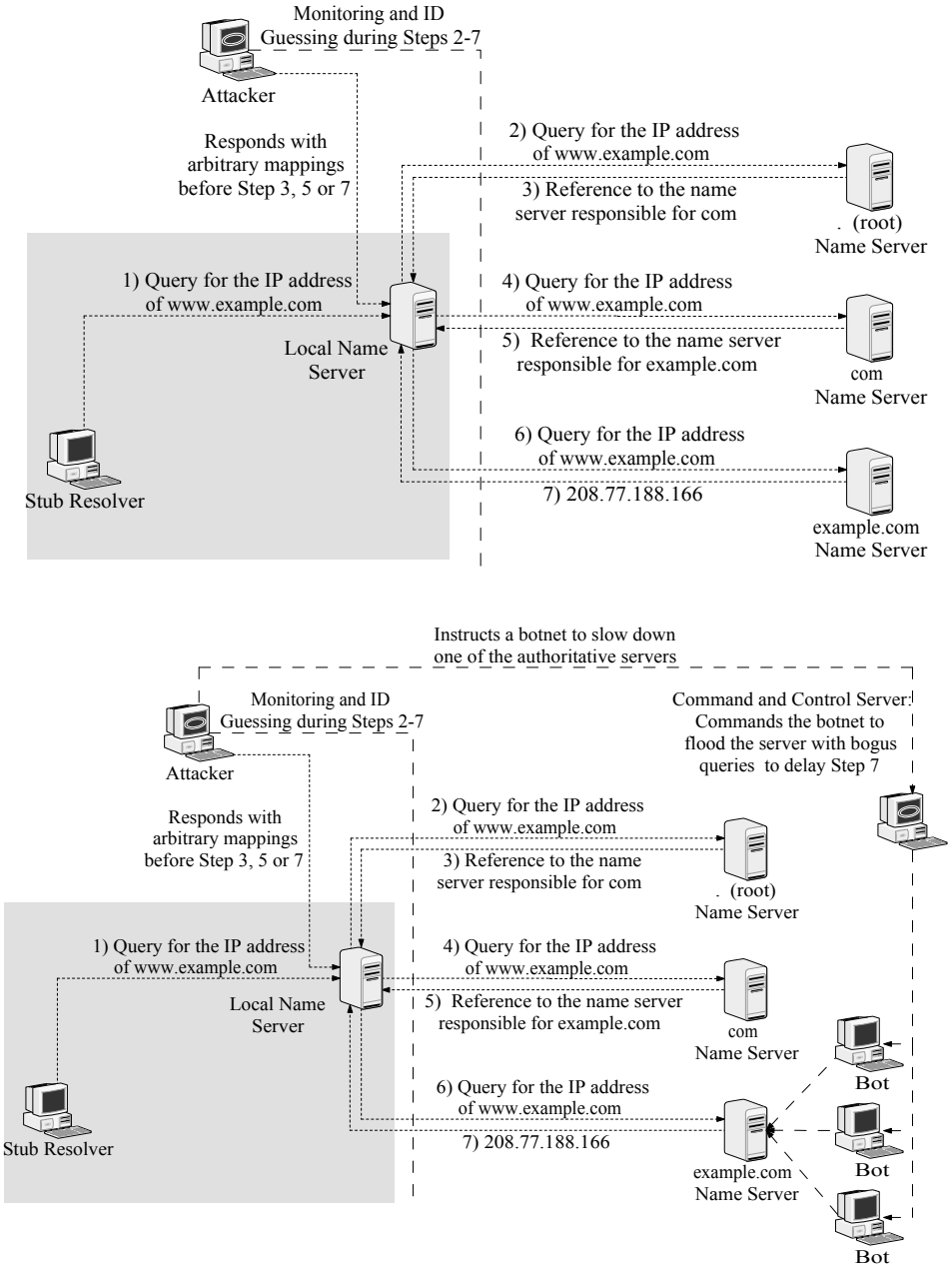

Figure 2.4: In the man-in-the-middle cache poisoning (upper plot), the attacker monitors the traffic that flows between the victim (local) name server and authoritative name servers, and attempts to guess the ID in the header of a query. If it succeeds while the resolution is still in progress (Steps 2-7), it can send manipulated RRs to the victim (local) name server. The victim (local) name server caches the manipulated RRs and drops the responses it receives from the authoritative server. To increase its probability to succeed, attackers often attempt to slow down the authoritative name server. To achieve this, they usually command a botnet to flood the authoritative name server with bogus queries (lower plot).

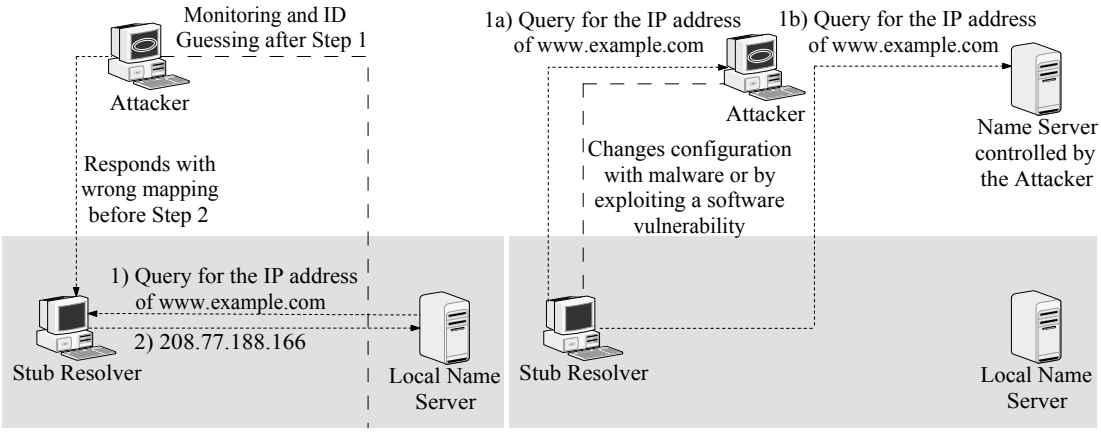

Figure 2.5: In the man-in-the-middle scenario (left plot), the attacker monitors the traffic that flows between a user machine and the local name server in its network, and attempts to guess or read the ID in the header of a query. If it succeeds while the resolution is still in progress, it can send manipulated RRs to the victim user machine. Alternatively (right plot), an attacker can release self-propagating malware specially designed to corrupt the information about the local name server on user machines to make the victim user machine send queries to the attacker directly (Step 1a) or a name server under its control (Step 1b).

or the private keys are corrupted or stolen, respectively.

Despite its limitations, the DNSSEC is being increasingly recognised as a useful tool for the Internet threats it addresses. As a result, in recent years, significant effort and investment have been committed worldwide to its promotion and deployment. In particular since 2005, when its revised RFCs (RFCs 4033, 4034 and 4035) were published, the DNSSEC is slowly but steadily gathering momentum [138]. As a matter of fact, several ccTLDs, such as Brazil (.br), Bulgaria (.bg), Czech Republic (.cz), Puerto Rico (.pr) and Sweden (.se), have already deployed the DNSSEC. This serves as a strong indicator that there is a reasonable chance that the DNSSEC will be eventually widely installed. As an immediate consequence, the risk from exploiting the design vulnerabilities in the DNS will be eliminated or at least greatly reduced. It is noteworthy that at the time of writing this thesis, taking advantage of vulnerabilities in the design of the DNS is still regarded as a serious Internet threat. However, given the current state of affairs in the global deployment of the DNSSEC, this attack vector is not considered further in this thesis.

2.2.2 Exploitation of DNS Implementation Vulnerabilities

Implementation vulnerabilities are low-level software flaws that are introduced during the coding phase of a system. The name servers (full resolvers) and stub resolvers, as any other piece of software, are vulnerable due to complexity and inevitable human

error. Exploiting these vulnerabilities, which is usually done by launching buffer overflow attacks, allows attackers to cause name servers or stub resolvers to crash or gain unauthorised access to the machines that run the vulnerable software. The latter, enables them, in turn, to execute arbitrary code on these machines. Attackers that succeed in gaining unauthorised access can, above all, reconfigure a name server or a stub resolver in any way, limited only by their skills.

The reconfiguration process can vary depending on the purpose of the reconfiguration operation that, in turn, is linked with the particular role of the vulnerable component of the DNS. The primary purpose of attempting to acquire unauthorised access to an authoritative name server is to corrupt the RRs in its zone files in order to make some Internet resources unreachable or redirect end users to malicious or compromised Web sites. The prime motivation behind manipulating the configuration files of a local name server is to alter either the way it responds to resolvers or the access rights of the name resolution service in order to subsequently exploit it to mount further Internet attacks. The next subsection describes various Internet attacks for which the latter plays a key role. Reconfiguring stub resolvers is a means to cause a DoS on the machines they run on or make them connect to compromised name servers. Despite the fact that the stub resolvers often have implementation vulnerabilities, and thus are at risk, they are only very rarely attacked. By contrast, the name servers remain particularly popular and attractive targets for attackers. This is because the damage and economic profit associated with a successful attack against a name server are much greater than those with attacking a stub resolver. Hence, in what follows, the focus is solely on name servers.

Although the number and the severity of implementation vulnerabilities differ for name servers developed by various vendors, the de facto standard name server software – the Berkeley Internet Name Domain (BIND) – is generally considered to be secure. In his "Pro DNS and BIND" [5], Ron Aitchison writes for BIND that, in contrast to the common belief, the name servers need to be protected from external sources and attacks, but in general not from their implementation. This is also supported by the fact that between its first release in 2000 and the end of 2008 only seven implementation vulnerabilities were discovered in BIND9, which is the latest version of BIND, and only three of them were classified as serious [80]. By exploiting any of these seven vulnerabilities, attackers could terminate or cause a name server to crash. All the vulnerabilities (design and implementation) of the BIND9 name server that have been publicised in the Common Vulnerabilities and Exposures (CVE) [36] and the US-CERT vulnerability notes database [174] are listed in Table 2.1. The fact that only a small number of software flaws have been discovered to date indicates that finding and exploiting vulnerabilities in BIND is a challenging and complex task, one that demands great technical skills and knowledge. Therefore, the exploitation of this attack vector is generally considered to be low risk.

Given this state of affairs, the current solution against name server vulnerabilities

Table 2.1: The design and implementation vulnerabilities that were discovered in the BIND9 name server between its first release in September 2000 and the end of 2008. Seven implementation vulnerabilities were discovered whose exploitation could result in unexpected crash or termination of the name server. The severity of only three of them was characterised as high. Except for these implementation vulnerabilities, three design vulnerabilities that allowed attackers to cache poison the name server were also discovered.

Date	CVE and CERT Identifiers	Severity	Effect
2008-07-08	CVE-2008-1447, VU#800113	High	Design Vulnerability
2008-01-18	CVE-2008-0122, VU#203611	Low	Server crash/termin.
2007-07-24	CVE-2007-2926, VU#252735	Medium	Design Vulnerability
2007-07-24	CVE-2007-2925, -	Medium	Design Vulnerability
2007-04-30	CVE-2007-2241, VU#718460	High	Server crash/termin.
2007-01-30	CVE-2007-0493, -	Low	Server crash/termin.
2007-01-30	CVE-2007-2926, -	Low	Server crash/termin.
2006-09-06	CVE-2006-4095, VU#915404	High	Server crash/termin.
2005-01-25	CVE-2005-0034, VU#938617	Low	Server crash/termin.
2002-06-04	CVE-2002-0400, VU#739123	High	Server crash/termin.

follows the install-now-patch-later model. This means that a new software update (patch) is released every time a new implementation vulnerability is discovered. Thereby, keeping a name server secure in the long run is for network operators primarily a matter of regularly updating its software and upgrading to the latest version. In addition, if a network operator follows a set of well-documented best practices for administrating and operating a name server (see, for instance, [72]), it can significantly reduce both the risk of exploitation and the potential damage to the name server and the Internet-connected machines it serves caused by successful exploitation of its implementation vulnerabilities [14].

2.2.3 Exploitation of the Name Resolution Service

In this subsection, the intent is to demonstrate that, besides its critical role for the operation of Internet applications, the name resolution service is also a key vector for a wide variety of attacks against Internet-connected machines. In contrast to the attacks previously discussed, which involve taking advantage of design or implementation vulnerabilities, these presented in what follows involve exploiting the name resolution service in a seemingly legitimate manner. According to the particular way in which the name resolution service is exploited, these attacks can be classified into three categories. In the first category, fall flooding attacks that target degrading or interrupting the name resolution service that name servers provide

to Internet-connected machines. In the attacks in the second category, the name resolution service is used as a weapon. In particular, name servers are provoked to flood application servers in order to cause DoS for their legitimate users. In the third category, fall attacks that are directed either at user machines or at application servers and involve performing preparatory, controlling or auxiliary operations that are heavily dependent on using the name resolution service.

Flooding a name server, involves forcing one or many Internet-connected machines to simultaneously send a large number of bogus queries to the victim name server until it cannot process the legitimate queries it receives. Forcing many machines instead of just one, makes it more difficult for network operators to detect all the sources of the bogus queries and mitigate the attack. Therefore, it is the dominant technique used by attackers to flood a name server, and basically any other application server. The victim can be either a local or an authoritative name server. Flooding a local name server, results in degradation or interruption of the name resolution service for the stub resolvers it serves, whereas flooding an authoritative name server in denial of the domain names it is associated with for every resolver on the Internet. Compared to flooding a local name server, flooding an authoritative one can affect more end users, is associated with economic damage for the holder of the domain name and is less challenging because the authoritative name servers have public IP addresses. Therefore, to date, the techniques described hereafter have been used primarily for attacking authoritative name servers. Nevertheless, they require minor changes to be effective against local name servers.

In more detail, to flood a name server, attackers have two techniques at their disposal: either to instruct a botnet to bombard the victim name server with bogus queries or to use a set of recursive name servers as reflectors. A reflector is any Internet-connected machine that returns a packet if a packet is sent to it. In the first scenario, the effectiveness of the attack increases significantly if the victim name server supports recursion and the botnet is instructed to generate recursive queries. The reason for this is that the name servers have a limited capacity to handle recursive queries. In the second scenario, the name servers acting as reflectors flood the victim name server with bogus responses or recursive queries [140]. To force the reflectors to flood the victim name server with bogus responses, attackers send to them spoofed queries with the victim's IP address in the source field; whereas, to make them flood the victim name server with bogus queries, they send to them recursive queries for RRs in the zone files of the victim name server. A recent survey reports that more than 75% of the name servers on the Internet are open resolvers (answer recursive queries from any Internet-connected machine) [171]. Thereby, both techniques are still very effective. Indeed, during the latest flooding attack against the 13 root name servers on February 6, 2007 six of them were affected, two of them badly [79]. The two attack scenarios are illustrated in Fig. 2.6.

Apart from flooding name servers, attackers can force one or many of them to

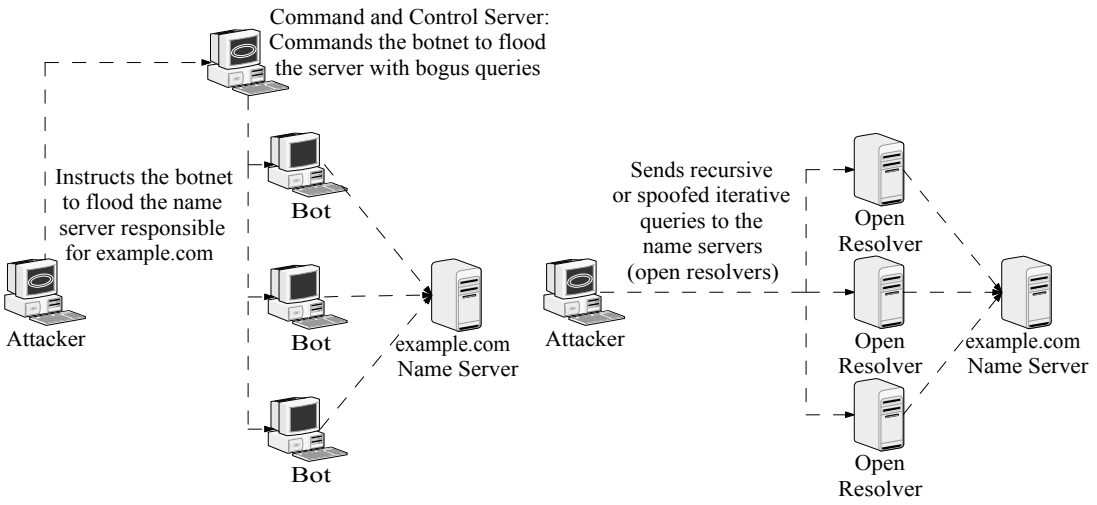

Figure 2.6: An attacker can flood a (recursive) name server by using a botnet (left plot) or a set of (recursive) name servers that respond to queries originating from any Internet-connected machine – open resolvers (right plot). In the former scenario, the attacker instructs the bots to generate many bogus (recursive) queries destined for the victim name server. In the latter scenario, the attacker sends many iterative (or recursive) queries with the IP address of the victim name server in the source field to the set of name servers. These name servers flood the victim name server with bogus responses (or recursive queries)

flood an application server. In this scenario, the name servers are not the attack target but the attack medium; nevertheless, they can be potentially seriously affected during the attack. Using this technique to cause DoS for the legitimate users of an application server has become popular among attackers mainly for two reasons. First, it produces an amplified flooding effect because small (up to 60 bytes) queries can generate relative large (512−byte or even larger, if EDNS0 is supported, see RFC 2671) responses. Second, using name servers enables attackers to launch flooding attacks in the presence of filtering mechanisms because the DNS traffic usually flows in and out of the networks of the Internet unfiltered. Based on whether the name servers support recursion or not, the attack is referred to as amplification or reflection, respectively. In the reflection attack, the attacker sends to the name servers many spoofed iterative queries with the IP address of the victim application server in the source field. As a result, the name servers flood the victim application server with DNS responses. If the name servers support recursion, the amount of flooding traffic can be significantly increased. This is because specially crafted recursive queries that contain a large buffer advertisement can make the name servers generate DNS responses of more than 4 000 bytes [177]. Usually, attackers do not query the name servers directly but use botnets to simultaneously send many queries to a large number of name servers. The reflection and amplification attacks as well as their

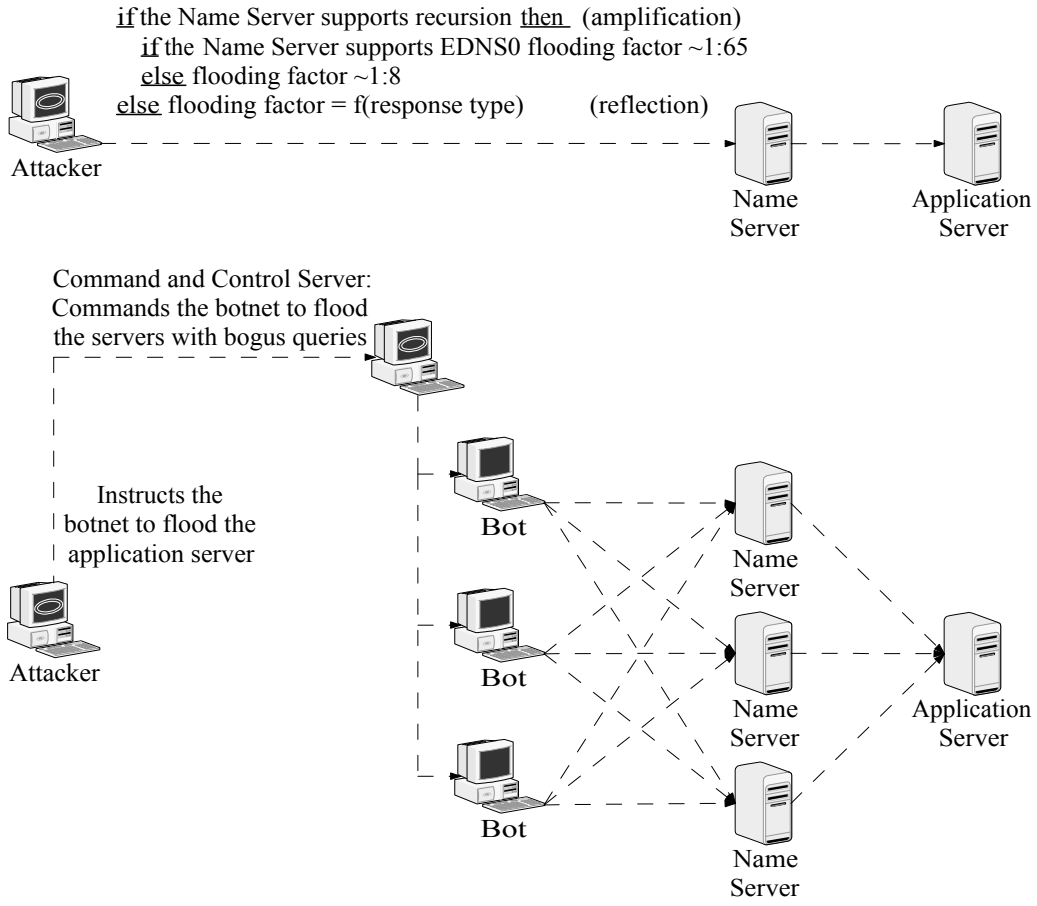

Figure 2.7: An attacker can use a name server as reflector or amplifier to flood an application server (upper plot). Small (up to 60 bytes) queries with the IP address of the victim application server in the source field can generate large (512−byte or even larger) responses destined for the victim application server, resulting in a flooding factor of up to 1:65. The amount of flooding traffic can be greatly increased if the attacker uses a botnet to query many name servers simultaneously and these name servers support recursion (lower plot).

generalisations with a botnet are depicted in Fig. 2.7.

Despite the fact that it is not as straightforward as for the attacks presented so far, botnets and modern Internet worms exploit the name resolution service in a seemingly legitimate manner, as well. Botnets nowadays are used by attackers for conducting a broad range of malicious activities. Among them, distributing spam and launching targeted distributed denial of service (DDoS) attacks are considered to be the driving forces in the economics of botnet developers [9, 216]. Although there exists no universally accepted definition for spam, this term is used throughout this thesis to describe unsolicited emails that are usually sent in bulk. Internet worms are a serious Internet threat in their own right and as the dominant medium

used by attackers to install bot software (and other malware) on Internet-connected machines [166]. This implies that the three main Internet threats at the time of writing this thesis – botnets, DDoS and Internet worms [8] – are closely interrelated. The motivation for attackers to make their malware dependent on the name resolution service is twofold. On the one hand, it enables them to hide illegitimate traffic or the true source of their activities, and it facilitates locating and accessing potential targets with less risk of being detected, on the other. In connection with this, it is largely because they use the name resolution service in a seemingly legitimate manner that the malicious activities of botnets and Internet worms are often detected after they have caused immense damage.

In more detail, using bot software that depends on the name resolution service, offers attackers, referred to as botmasters in botnet terminology, several advantages that are detailed hereafter. Once a bot is installed on a user machine, it establishes and maintains a connection to a command and control server. The command and control server is operated by a botmaster that employs it as a means to distribute commands to the bots. Botmasters want to have the flexibility to change the IP address of their command and control server in an arbitrary manner for two reasons. First, it allows them to hide the actual location of their server, and thereby increase the probability that it is not detected and shut down, and, second, it enables them to resume control of their botnets if their server is eventually detected and shut down. Therefore, bots usually have the domain names instead of the IP address of their command and control server defined in their code, and thus to connect to it they have to locate it first by querying the local name server in their network. The processes of a new bot that phones home and a botmaster that regains control of a botnet, known as dynamic DNS [173], are depicted in Fig. 2.8. In some cases, with the intent to eliminate any potentially detectable change in the DNS traffic caused by their botnets, bot writers develop bots that do not rely on querying the local name servers. Instead, they require that botmasters define the IP addresses of one or a set of name servers under their control. However, depending on the security systems deployed on the networks where the bots reside, this often leads to the opposite result. The reason is that the communication between bots and external name servers can be detected by analysing the outgoing DNS traffic from non-name servers on the gateway of a monitored network.

Once their botnets are set up, a major concern for botmasters is how to distribute commands to the bots without raising suspicion. Given that, as previously pointed out, the DNS traffic usually flows in and out of the networks of the Internet unfiltered, an effective way to do this is by means of DNS tunnelling [175]. In more detail, botmasters can hide their commands in RRs for which they control the authoritative name servers. Thereby, in response to DNS queries from the bots for RRs that are under the authority of these name servers, the local name servers unwittingly provide bots with the commands of botmasters. Other ways in which botmasters

take advantage of the name resolution service include preparing and supporting DDoS and spamming attacks. Usually, before flooding an application server, bots query the local name server in their network to resolve its IP address. This is because botmasters typically define the DDoS attack target by its domain name instead of its IP address. Doing this, increases the probability that bots will reach the victim application server, and that the victim application server will not escape the attack, even if its IP address is changed. Moreover, bots that are used for spamming, which are known as spambots, are strongly dependent on the name resolution service, as well. In particular, for harvesting email addresses from the Internet and locating the email servers that are responsible for accepting emails for these email addresses, they rely heavily on the local name servers.

Probing the IP address space for Internet-connected machines that run vulnerable software is an obsolete propagation method that only rarely recent Internet worms use. There are two main reasons that prompted worm writers to turn to alternative methods. The first reason is that IP scanners have been thoroughly studied over the past years, and as a result many effective methods for detecting and mitigating Internet worms that probe the IP address space to propagate have been proposed. The second reason is that discovering new implementation vulnerabilities to take advantage of has increasingly become a challenging and time-consuming task, one that requires advanced technical skills and knowledge. Hence, the most prevalent recent Internet worms compromise machines by victimising end users, and spread on social networks via communication channels offered by applications that contain rich social information. Thereby, in a similar manner to legitimate applications, once Internet worms compromise a machine, they exploit the name resolution service to collect the information they require to spread further. Moreover, it is common for machines compromised by Internet worms to automatically launch flooding attacks against application servers. For the same reasons as botmasters, worm writers define in the worms' code the domain name of the victim application servers. For instance, user machines compromised by any of a variety of Internet worms, like for example by email worms of the Mydoom family or peer-to-peer (P2P) worms – Internet worms that spread primarily on P2P networks – of the Antinny family, query the local name server in their network to resolve the IP addresses of several Web servers before flooding them [81]. As Internet worms constitute the core of the present study, the following section is devoted to providing the necessary background on them, discussing the current trends in their design and uncovering the dependency of their operations on querying the local name servers.

2.3 Research Topic Selection

Based on the fact that many of the operations email worms carry out affect the DNS traffic that compromised user machines generate a detection and a mitigation method

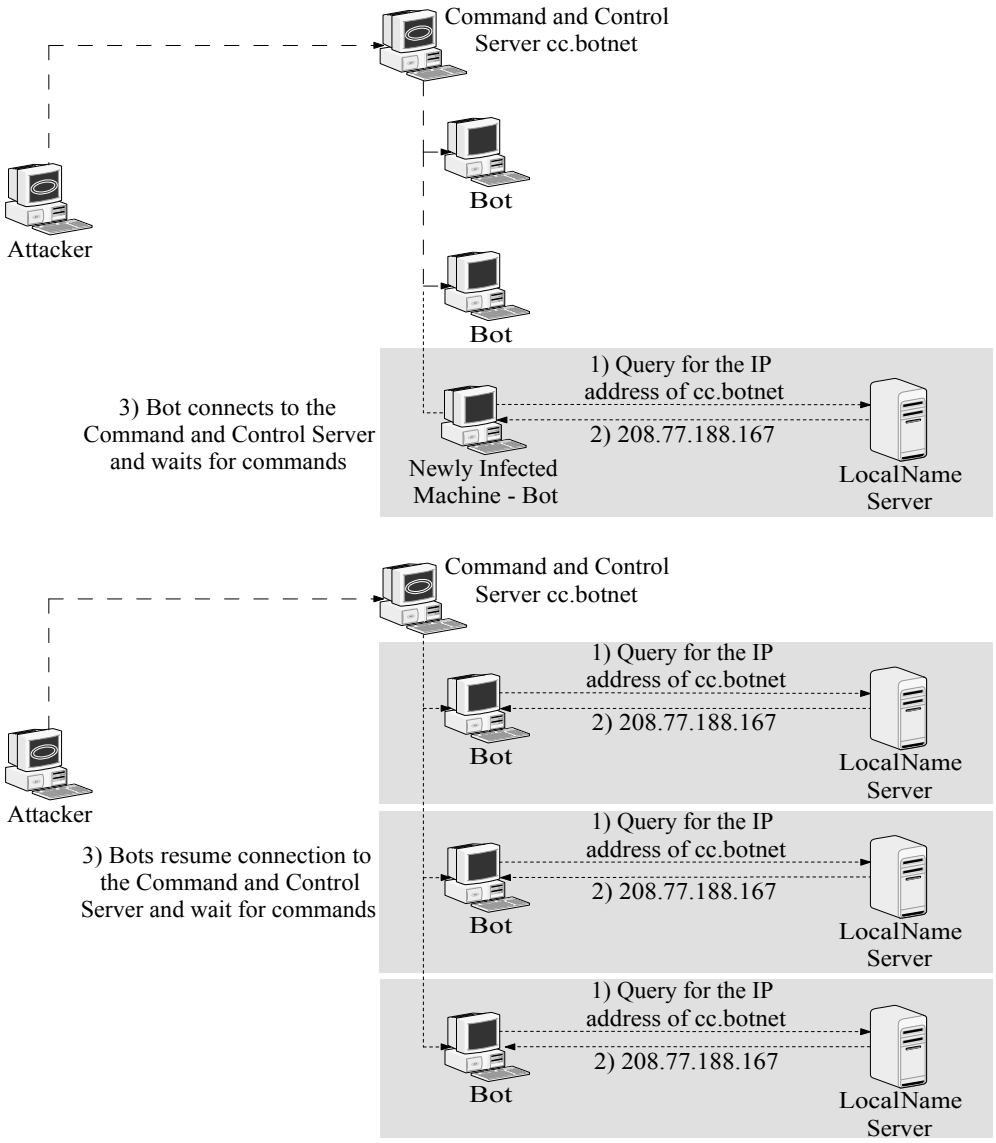

Figure 2.8: Once a bot is installed on a user machine (upper plot), it queries the local name server in its network for the IP address of the command and control server whose domain name it has defined in its code (Step 1). Once it receives the response (Step 2), it connects to the command and control server and waits for commands (Step 3). If the bots lose connection to the command and control server (lower plot), they query the local name servers in their network for its new IP address (Step 1). Once they receive the response (Step 2), they re-establish communication with the command and control server (Step 3).

that have the potential to be effective in dealing with email worms in the long run are presented later in this thesis. In this section, three key aspects are elaborated with the underlying goal to explain the motivation behind selecting this particular Internet threat as the main topic of the study reported in this thesis. First, the main trends in the design of Internet worms are discussed, and it is demonstrated that email worms are the most prevalent malware type. Second, the life cycle of email worms is analysed, and it is shown in detail which and how exactly the operations of email worms depend on the local name servers. Third, three additional reasons to the self-evident one, namely that they are one of the major Internet threats, that make email worms a useful, interesting and challenging topic to study are presented.

2.3.1 Trends in Internet Worms

Since the Morris worm was unleashed in 1988, Internet worms have consolidated their position as a major threat to network operators and end users. Often, their outbreaks make the headlines because they are associated with extensive economic damage. In recent years, Internet worms have been capable of compromising thousands of Internet-connected machines in a short time, launching spamming and DDoS attacks, stealing security information and destroying key data. Furthermore, the traffic that compromised Internet-connected machines generate causes network congestion and loss of service or degradation in the performance of network resources. For this reason it is regarded as a serious operational concern in its own right. As already mentioned in Chapter 1, similarly to Kienzle et al. [98], the term Internet worm is used throughout this thesis to cover every malware, standalone or file-infecting, that propagates on the Internet independent of whether human interaction is required or not. Based on their propagation method, Internet worms are distinguished into two classes: scanning (exploit-based) and topological [189].

Scanning worms compromise Internet-connected machines by taking advantage of implementation vulnerabilities in the operating system or the applications that run on them. In connection with this, programming Internet worms of this class is primarily a matter of finding new implementation vulnerabilities in commonly-used software. Once they compromise a user machine, scanning worms propagate by attempting to connect and send over TCP/IP a possibly evolved copy of themselves to a set of machines they select from the IP address space. If a recipient of such a copy has the vulnerable software installed, it gets compromised and starts spreading the worm further in the same way. As a consequence, the compromised machines exhibit similar network behaviours, which are typically characterised by many connection attempts in short time periods. Instead of exploiting software flaws, topological worms rely on victimising unaware end users to infect Internet-connected machines (user machines) and propagate on social networks through communication channels offered by applications that contain rich social information. Braverman [20] shows

that the three most common social networks on which topological worms spread are, in order of decreasing prevalence: the Internet-scale email network, and networks of P2P and instant messaging (IM) applications.

Until a couple of years ago, scanning worms were regarded as a significant Internet threat, and as a result they attracted considerable attention from the research and industrial communities. However, as of the mid-2004, new scanning worms appear only rarely, and a number of useful methods for detecting and mitigating them have been proposed and deployed. By contrast, topological worms have emerged as a popular medium for attackers to achieve their malicious ends, and remain an ever-evolving Internet threat. Apart from the effectiveness of the existing network security systems against scanning worms, four more reasons can be identified behind this notable change in the Internet threat landscape. First, developing topological worms is less challenging than developing scanning worms because it does not involve discovering and taking advantage of implementation vulnerabilities. Second, the propagation speed and the number of Internet-connected machines that topological worms can compromise constantly rises, as the number of end users that use social networking applications on a regular basis increases. Third, as their high incidence suggests, topological worms are nearly undetectable by the network security systems currently deployed. The reason for this is that they propagate by misusing legitimate applications; as a consequence, their traffic blends in with that of legitimate end users and the traffic of compromised machines does exhibit easily detectable anomalies. Fourth, compared to the case of scanning worms, understanding the phenomena that relate to topological worms is still in its infancy. Therefore, devising powerful methods for detecting and mitigating topological worms continues to present many challenges to researchers and industry practitioners.

Electronic mail, or email for short, is an essential communication medium for the Internet-connected society. Increasingly, individuals and organisations depend on it for a growing portion of their day-to-day communication. As an unavoidable consequence of this dependency, email gradually became, and remains, the dominant application topological worms misuse to propagate [20]. In fact, in recent years, the monthly-updated top-threat lists of all antivirus vendors have been almost exclusively populated by Internet worms that use email as their primary propagation vector (see, for instance, [91, 181]). This subclass of topological worms stands at the centre of the present study and, as previously mentioned, the members of it are referred to as email worms in this thesis. In Fig. 2.9, the most prevalent malware families for the years 2004–2008 are given and, if applicable, the propagation vectors of each family are shown. The source of the presented data is the technical journal Virus Bulletin [180], which tracks developments on the prevention, detection and removal of malware. Virus Bulletin claims to have full editorial independence and is widely recognised as a reliable information source within the antivirus industry. A close look at the figure reveals that email worms have been around for several years and still

present for attackers a popular means to achieve their ends. This clearly indicates that the currently-deployed network security systems to detect and mitigate Internet worms yield only meagre results against email worms.

As a consequence of this state of affairs, the implications of the persistent high incidence of new and rewrites of known email worms continue to pose manifold threats to network operators and end users. From the perspective of network operators, the traffic that compromised user machines generate is considered responsible for network congestion and performance deterioration in network infrastructural elements. The reason for this is that email worms are associated directly and indirectly with the high amount of illegitimate email traffic on the Internet, which according to the Messaging Anti-Abuse Working Group (MAAWG) accounts for more than 80% of all the email traffic since the first quarter of 2006 [125]. Intuitively, during the outbreaks of email worms significant, sudden increases in the email traffic that traverses the Internet are observed. Apart from this direct cause-effect relationship, there also exists a substantial indirect relationship between email worms and illegitimate email traffic. Namely, worm writers and spammers are steadily joining forces by using email worms to compromise user machines with the goal to turn them into spambots. In recent years, spambots are regarded as the principal source of spam [146]. From the end users' point of view, the problem is stabilising, as only a fraction of that 80% actually reaches their inbox [57]. However, email worms remain a serious problem for them because they are the dominant way attackers use to remotely install on their machines malware designed to steal their confidential information (spyware), launch phishing attacks and exploit their resources [164].

2.3.2 Email Worm Life Cycle

Increasingly, email worms are being written to induce financial gain through the abuse of compromised user machines rather than merely to spread. In connection with this, their operations relate either to their propagation or to their intended malicious activities. As far as their propagation is concerned, email worms bear many similarities to infectious diseases. Therefore, in consistency with the terminology that is used to describe the state of an individual with respect to an infectious disease, a compromised user machine is commonly referred to as infected. Email worms perform their malicious activities upon execution of a specific part of their code, known as the payload. From a macroscopic point of view, executing the payload results in two types of operations: destructive and supportive. The former are targeted against the infected user machines or other Internet-connected machines; whereas, the latter are aimed at hiding the traces of the malicious activities. Worm writers have a steadily growing number of pieces of malware that perform a vast array of destructive operations at their disposal. This primarily includes bot software for distributing spam or launching DDoS attacks and spyware.

Year: 2004

1. Netsky	1-2
2. Bagle	1-2
3. Sober	1
4. Mydoom	1-2
5. Zafi	1-2
6. Dumaru	1
7. Mimail	1-2
8. Klez	1-2-5-6
9. Opasoft	2-4
10. Bagz	1

Year: 2005

1. Sober	1
2. Netsky	1-2
3. Mytob	1-2-3-4
4. Bagle	1-2
5. Mydoom	1-2
6. Bagz	1
7. Zafi	1-2
8. Lovgate	1-2
9. Funlove	6
10. Tenga	6

Year: 2006

1. Detnat	2
2. Mytob	1-2-3-4
3. Netsky	1-2
4. Bagle	1-2
5. Nyxem	1-2
6. Lovgate	1-2
7. Mydoom	1-2
8. Zafi	1-2
9. Bagz	1
10. Warezov	1

Year: 2007

1. Detnat	2
2. Netsky	1-2
3. Mytob	1-2-3-4
4. Bagle	1-2
5. Nyxem	1-2
6. Virut	6
7. Zafi	1-2
8. Mydoom	1-2
9. Lovgate	1-2
10. Warezov	1

Propagation Vectors

1	Email
2	Peer-to-Peer
3	Instant messaging
4	Scan for software vuln.
5	Removable devices
6	Infect files
-	No propagation

Year: 2008

January		February		March		April		May		June	
1. Netsky	1-2	1. Netsky	1-2	1. Netsky	1-2	1. Netsky	1-2	1. Netsky	1-2	1. Agent	-
2. Mytob	1-2-3-4	2. Cutwail	-	2. Mytob	1-2-3-4	2. Cutwail	-	2. Agent	-	2. Netsky	1-2
3. Psyme	-	3. Mytob	1-2-3-4	3. Virut	6	3. Onlinegames	-	3. Rays	1	3. Cutwail	-
4. Bagle	1-2	4. Mydoom	1-2	4. Mydoom	1-2	4. Mytob	1-2-3-4	4. Mytob	1-2-3-4	4. Clagger	-
5. Agent	-	5. Bagle	1-2	5. Bagle	1-2	5. Virut	6	5. Onlinegames	-	5. Rays	1
July		**August**		**September**		**October**		**November**		**December**	
1. Netsky	1-2	1. Agent	-	1. Agent	-	1. Agent	-	1. Agent	-	1. Netsky	1-2
2. Agent	-	2. Zbot	-	2. Inject	-	2. Mytob	1-2-3-4	2. Goldun	-	2. Agent	-
3. Zbot	-	3. Delf	-	3. Autorun	-	3. Netsky	1-2	3. Netsky	1-2	3. Mytob	1-2-3-4
4. Bifrose	-	4. Netsky	1-2	4. Goldun	-	4. Goldun	-	4. Autorun	-	4. Virut	6
5. Mytob	1-2-3-4	5. Autorun	-	5. Netsky	1-2	5. Autorun	-	5. Zbot	-	5. Zbot	-

Figure 2.9: The most prevalent malware families for the years 2004-2008, as publicised by Virus Bulletin. For 2004-2007, the ten most prevalent families, in order of decreasing prevalence, are shown on a yearly basis. For 2008, the five most prevalent families are shown on a monthly basis. For each family, the basic propagation vector is underlined and the email worm families are grey highlighted. The basic propagation vector is used by all the members of a family and it is the criterion antivirus vendors employ to characterise a family. Email worms are the most prevalent Internet worm type and malware in general.

Email worms usually have limited operational capabilities that are linked to the objectives of worm writers. Although the exact combination and the implementation details of these operations can differ significantly from one email worm to another, the life cycle of every email worm consists of four distinct phases: the penetration, target acquisition, payload activation and propagation. For reasons that will become clear in what follows, as a means to increase the virulence and survival chances of their email worms, worm writers program them so that they repeat all the phases, except for the penetration phase, according to a prearranged pattern or when triggered by a system event. Such events are, for instance, the reopening of an infectious email by an email user or a system restart. In this thesis, the email worms that repeat their phases are referred to as reinfecting to distinguish them from the non-reinfecting, which are those that go through them once. Notably, almost all the email worms that have recently appeared in the wild are reinfecting [203].

Email worms circulate in either of two possible ways via specially crafted (infectious) emails, which infected user machines generate. Specifically, they spread

either as attachment files or the infectious emails contain links to compromised Web sites. Once such an email reaches the inbox of an email user, the email worm it carries enters the penetration phase. In this phase, the email worm attempts to install malware on the user machine. Doing this, typically involves tricking the end user, mainly by means of social engineering techniques, into executing the files attached to the email or clicking fraudulent links in it. The social engineering techniques provoke end users to make wrong choices, and they are based on specific attributes of human decision-making, known as cognitive biases [127]. As will become clear in what follows, the penetration phase is the only one in the life cycle of email worms that normally does not affect the network behaviour of the victim user machine. Once the malware is installed, the user machine becomes infected and the payload activation or the target acquisition phase can begin.

The payload activation phase can be decomposed into two independent subphases: the reinfection and attack. The purpose of the email worms when they are in the reinfection subphase is to render the infection persistent. To accomplish this, they perform a number of operations that can be thought of as being auxiliary to those they carry out when they are in the other phases and subphases of their life cycle. Common examples of such operations include the manipulation of configuration parameters, the creation or modification of files, the installation of additional malware for automatically maintaining and updating the malware originally installed and the weakening of the security settings on the infected user machines. In the attack subphase, the email worms execute their destructive operations, which, as previously mentioned, depend heavily on the intent of worm writers. Above all, recent email worms are designed to automatically launch flooding attacks once they compromise a user machine and install remotely controlled bot software that can be used at a later time to distribute spam or mount DDoS attacks.

Once the email worms enter the target acquisition phase, they start collecting email addresses and information about the email servers that are responsible for delivering emails to them. Two clearly distinguishable subphases can be identified that occur either simultaneously or in succession: the harvesting and name lookup. When the email worms are in the first of these subphases, they perform several operations to compile a list with as many as possible email addresses that are then targeted for infection in the propagation phase. In particular, they are usually programmed to obtain email addresses by searching in the address book and files with predefined extensions on the infected user machines, guessing, and scanning Web pages. In this context, guessing refers to the attempt of email worms to discover email addresses of email users by combining the user and domain names they harvest from the file system of the infected user machines or the scanned Web pages and commonly-used names worm writers define in their code.

In the name lookup subphase, as its name implies, the email worms query the local name server on the networks of the infected user machines to resolve the IP

addresses of the email servers that deliver emails to the email addresses collected during the harvesting subphase (potential victims). It is worth noting that although some email worms were programmed to query other recursive name servers on the Internet, the vast majority of them rely on the local name servers. Despite the fact that a few (two to four) DNS queries are needed to resolve the IP address of one email server that delivers emails to a potential victim, email worms typically produce more DNS queries than those really needed. This is due to a variety of reasons. The most common one is that email worms are usually programmed to resolve not only the IP address of the default email server, which is the one with the highest priority, but that of all the email servers associated with a potential victim. The rationale of worm writers behind programming email worms to do so is that an email server with lower priority is more likely than the default one not to be equipped with antivirus or antispam software. In connection with this, the probability that email worms will succeed in reaching the potential victims by using it is higher.

At any time after the target acquisition phase has started, the propagation phase is initiated. In this phase, the email worms sequentially send one infectious email to every email address in the list of potential victims compiled during the harvesting subphase. As far as the sending operation of email worms is concerned, worm writers have three options at their disposal. These are namely to program email worms to use either open mail relays (email servers that allow anyone on the Internet to send emails through them) or the email clients that run on the infected machines in a seemingly legitimate manner or to arm email worms with a built-in email engine, and thereby make them capable of bypassing the outgoing email servers. The last option provides the advantage that email worms can escape detection by antivirus and antispam software installed on the outgoing email servers, and thus it is the one most often used. In addition, to increase the virulence and propagation speed of email worms, worm writers program them to spread over secondary communication channels. Such channels are offered by applications that, similarly to email, contain rich social information, like for instance P2P and IM applications.

In light of the analysis presented above, it becomes clear that many operations in the life cycle of email worms are heavily dependent on the name resolution service. Specifically, in the target acquisition phase, the email worms query the local name servers for the IP addresses of Web pages (harvesting subphase) and email servers (name lookup subphase). In addition, as detailed in the previous section, the pieces of malware that launch spamming and DDoS attacks (attack phase) usually are also heavily dependent on the name resolution service. The phases and subphases of the email worm life cycle, their timing relationship and their most common operations are shown in Fig. 2.10. Moreover, the operations that affect the DNS traffic that a user machine generates are marked. The figure is necessarily incomplete, simply because new tactics may arise; however, it is generally complete with regard to the situation at the time of writing this thesis, as it includes all the basic operations

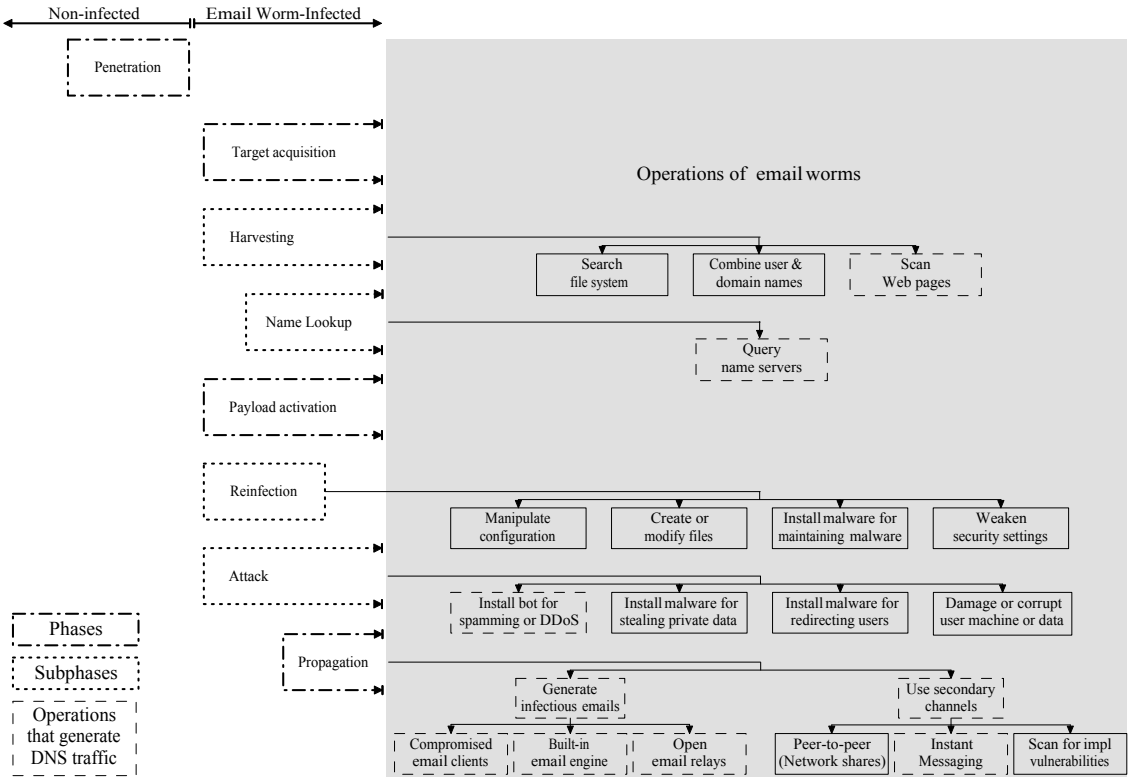

Figure 2.10: The life cycle of email worms consists of four distinct phases: the penetration, target acquisition, payload activation and propagation. The target acquisition and payload activation phases start after the penetration phase has been completed, and each of them can be divided into two subphases: the harvesting and name lookup, and the reinfection and attack subphases, respectively. The propagation phase starts after the target acquisition phase has started. The timing relationship of these phases and subphases (white region) and the most common operations email worms perform (grey highlighted region) are shown.

email worms perform. Overall, it is justifiable to conclude that once a user machine becomes infected, the characteristics of the DNS traffic it generates change. Showing that this change provides a strong basis for detecting user machines infected with email worms in the long run is one key contribution of this work.

2.3.3 Email Worms vs. Botnets

In recent years, email worms are considered to be one of the two major Internet threats. Although this is enough to justify devoting this work to finding methods to effectively detect and mitigate email worms, three additional reasons for doing so are listed below. These reasons serve as a subjective motivation to focus on studying email worms instead of botnets used to distribute spam and launch DDoS attacks.

1. The Internet-scale epidemics of email worms typically precede spamming campaigns and targeted DDoS attacks launched by botnets. This is because email worms have emerged into the dominant medium for attackers to automatically install bot software on a large number of user machines without the end users' knowledge or permission. Thereby, preventing the propagation of email worms, can eliminate one powerful option attackers have to compromise user machines.

2. Email worms and spambot software perform many common operations for finding email addresses and sending out emails. Therefore, it is reasonable to assume that the network behaviour of user machines infected with email worms bears many similarities to that of spambots. In connection with this, the methods for detecting user machines infected with email worms and mitigating the degree to which they pollute the Internet can be useful against spambots.

3. Email worms are, in contrast to bot software, particularly interesting technological constructs characterised by an intriguing mathematical structure and complexity. Understanding their operations and designing effective detection and mitigation methods, have received less attention than botnets, and the results have been rather poor so far. Hence, email worms are a fascinating topic to research, one that has many scientific and technical aspects to investigate.

2.4 Summary

The DNS consists of three components: the domain name space, the name servers, and the resolvers. It is a critical infrastructure of the Internet because the vast majority of the applications that run on Internet-connected machines rely on the name resolution service it provides to work. The data that compose the domain name space are stored on the authoritative name servers, and the contact point for an Internet-connected machine to retrieve them is the local name server in its network. Due to the critical nature of the DNS, the domain name space and the name servers have always been popular targets for attackers seeking to inflict widespread damage. However, in recent years, the motivation behind attacks and, as a result, their nature has changed considerably. Specifically, attacks have become more sophisticated and focused, and financial gain has evolved into the major driving force behind them. As far as the DNS is concerned, attackers have realised that exploiting it comes with more advantages than damaging its components or disrupting the name resolution service. As a consequence, a wide variety of Internet attacks produce an observable effect on the DNS traffic that traverses the Internet.

To achieve their malicious ends, attackers take advantage of vulnerabilities either in the design or in widely-used implementations of the DNS or exploit the name resolution service in a seemingly legitimate manner. Over the last years, the threats

associated with taking advantage of design or implementation vulnerabilities have sharply declined. The main reasons for this are that the deployment of the DNSSEC is steadily gaining ground and that the BIND, which is the de facto standard name server, is an extensively tried and tested software. By contrast, the two major threats to network operators and end users – Internet worms and botnets – make use of the name resolution service for performing a variety of destructive and supportive operations. Email worms are by far the most prevalent type of malware, since attackers employ them as their dominant medium to carry out a number of malicious activities. In this chapter, it is demonstrated that many of the operations in the life cycle of email worms involve querying the local name servers.

The traffic that email worms generate as they propagate causes network congestion and loss of service or degradation in the performance of network resources. In addition, user machines infected with email worms are commonly used to distribute spam or launch DDoS attacks. Despite the high incidence of email worms and the real threat they present to network operators and end users, the currently-deployed network security systems yield only poor results against this type of malware. In Chapter 3, with the underlying objective to give the main reasons for this state of affairs, a comprehensive, critical review of the existing methods for detecting Internet worms is presented. Based on the assumption that user machines infected with email worms exhibit different communication patterns with local name servers from non-infected, a novel method for effectively detecting user machines infected with email worms in the long run is introduced in Chapter 4.

Part II

Detection of Internet Worms

Chapter 3

Literature Review

3.1 General Detection Methods

Depending on their scope, the methods for detecting Internet worms that are proposed in the literature fall into either of two categories: general or specialised. The general methods are useful against any Internet worm, independent of the class it belongs to (scanning or topological). The specialised methods are based on the concept of behavioural signatures. A behavioural signature is a traffic pattern that characterises the activity of Internet worms belonging to a certain class. As a consequence, in contrast to the general detection methods, the specialised detection methods are capable of identifying the activity of either scanning or topological worms only. In this section, a comprehensive, critical review of the existing general detection methods is presented, and their strengths and weaknesses are discussed in detail. Following this, the next section is devoted to describing the available specialised methods and highlighting their strengths and weaknesses.

The most widely adopted and used method for detecting Internet worms is misuse detection, and in particular its most typical example, signature-based detection. The principle of this method involves searching the packets that flow on a monitored network for explicit indications of Internet worm activity [168]. For signature-based detection, the indications are known as attack signatures and are byte sequences that appear frequently in the packets of machines infected with Internet worms captured and analysed in the past. Thereby, any machine in a monitored network whose packets contain attack signatures is automatically classified as infected. With respect to this, the two main strengths of signature-based detection are its simplicity and its capability of identifying machines infected with known Internet worms in real time. On the other side, an intrinsic weakness of this method is that new attack signatures are generated by analysing Internet worm traffic, which is possible only after the respective Internet worm outbreaks have occurred. Moreover, to keep the detection accuracy high over time, the signature-based detection systems, such as

most of the commercial antivirus software, need to maintain a database with attack signatures. This database has to be constantly, manually or otherwise, updated with attack signatures to enable the systems to detect emerging Internet worms.

Despite its wide adoption and use, signature-based detection is not a complete solution for Internet worms. The reason for this is that, apart from the weakness pointed out above, it has two more significant weaknesses. The first of them is that it is ineffective against new, previously unseen worms, known as zero-day worms. This is because the attack signatures take effect after they have been generated and disseminated by antivirus vendors and verified and installed by network operators or end users. However, as mentioned above, this process starts after the Internet worms have been released, and it normally takes hours or even days. Therefore, it is highly likely that a zero-day worm succeeds in reaching its infection peak before a suitable attack signature has been widely installed. The second weakness of signature-based detection is its ineffectiveness against polymorphic worms. To evade signature-based detection systems, Internet worms of this type use several techniques, such as self-encryption, to alter their code as they propagate. Hence, one typical attack signature is not enough for identifying all the illegitimate packets sent to the Internet by machines that are infected with the same polymorphic worm. Detecting polymorphic worms, becomes even harder as generating specialised attack signatures for polymorphic worms presents still insuperable challenges [169].

From a deployment perspective, signature-based detection systems operate either on user machines or on other dedicated Internet-connected machines. Running a signature-based detection system on every, or at least most, user machines on a network is considered to be a very powerful solution for effectively protecting it from Internet worms. Doing so, however, is associated with high deployment costs and requires that end users update the attack signature databases on a regular basis. The problem with this is that it is often the case that end users do not keep an eye on the update releases or are reluctant to install updates for various reasons. One of them is, for instance, fear that installing updates will negatively impact their machines. The analysis presented in [129] supports this fact by providing evidence that even for well-publicised Internet worms the rate at which the Internet-connected machines that run the vulnerable software are updated is rather slow. In addition, irrespective of their deployment point, the signature-based detection systems usually operate on the network of the potential victims. This implies that they do not target reducing the amount of illegitimate traffic traversing the Internet, which is for network operators the most serious concern associated with Internet worms.

In recent years, a great deal of work has been carried out to develop signature-based detection systems that can detect zero-day worms. Most of the studies in this direction have been devoted to reducing the dependency of signature-based detection on human interactions. With respect to this, these studies are primarily focused on automatic signature generation [157, 158, 58, 114, 176, 6, 103, 37] and, to a lesser

extent, on automatic signature dissemination [182]. As its name suggests, automatic signature generation targets automatically generating useful attack signatures from the packets that flow on a monitored network. To accomplish this, the available methods are based on a set of predetermined rules. In particular, Earlybird [157, 158], TreeCount [58] and the methods presented in [114, 176] use as attack signatures byte strings that appear frequently in packets that belong to many traffic flows. Anagnostakis et al. [6] determines a slightly different set of rules. It uses namely as attack signatures byte strings that appear frequently in small packets sent from one source to many destinations. Instead of relying on determining rules, which requires domain knowledge, HoneyStat [37] and Honeycomb [103] generate attack signatures automatically by analysing the traffic captured by honeypots. As discussed later in this section, deploying honeypots is another general method for detecting Internet worm activity. Automatic signature dissemination aims at reducing the time that elapses between when a new attack signature is generated and when it is installed in the databases of signature-based detection systems. Instead of waiting for network operators and end users to pull them from a central repository, this method involves automatically pushing every new attack signature to the systems.

Despite the potential efficacy of automatic signature generation in dealing with zero-day worms, the available methods have several weaknesses that have to be overcome before they can be widely adopted and used. One of them is that they inherently assume that an invariant long substring appears in all the packets associated with an Internet worm. However, this assumption does not hold true for polymorphic worms. Although a few studies have investigated ways to address this issue [108, 135], there is still much to be done in this direction. Moreover, to generate reliable attack signatures, the available methods require monitoring and performing deep packet inspection on an extensive amount of traffic. In practice, these operations introduce high processing overhead, which renders these methods unsuitable for high speed, busy networks. Furthermore, the automatically generated attack signatures are normally less reliable than those generated by humans. In fact, false positives occur if byte strings that are automatically classified as suspicious appear equally frequent in legitimate traffic or even in the absence of Internet worm traffic. This limits the applicability of the available methods because they essentially require that human experts whitelist byte sequences that appear in legitimate traffic. In addition, most of the studies in the literature address scanning worms only. Hence, the methods they propose are ineffective against topological worms.

Honeypots are dedicated Internet-connected machines that appear to be part of a network but are actually isolated, unprotected, and capture the traffic that is sent to them. The ability of honeypots to detect Internet worm activity is based on the concept that no Internet-connected machine should be using or interacting with them because they have no legitimate activity. In connection with this, any transaction or interaction with them is, by definition, suspicious and potentially

malicious. In contrast to the methods discussed above, honeypots are not intended to work standalone but rather complementary to other Internet worm detection methods. Hence, they can be essentially regarded as providing an additional level of security. Based on the purpose of their deployment, honeypots fall into two categories: production and research [161]. The production honeypots target mitigating the risk of Internet worms in organisations, whereas the research honeypots are designed to gather information on Internet worms. Honeypots are further divided into low-interaction and high-interaction depending on the amount of freedom they offer to Internet worms. The high-interaction honeypots, such as the Honeynets [170], provide entire operating systems and applications for Internet worms to interact with. Thereby, they can be compromised completely, which involves allowing an attacker to remotely gain root access to the machines they run on. In contrast to the high-interaction honeypots, the low-interaction honeypots, such as the Honeyd [144], the Specter [160] and the KFSensor [97], emulate operating systems and applications and do not permit attackers to remotely gain root access.

Deploying honeypots, is simple and can significantly contribute to accurately detecting Internet worm traffic originating from infected machines inside or outside a monitored network. This includes traffic generated by Internet-connected machines that are infected with zero-day or polymorphic worms. In addition, honeypots can operate on busy, high speed networks in real time because they typically capture a small amount of traffic. However, honeypots also have a number of weaknesses inherent to their design and functioning. Their most significant weakness is that they capture traffic destined for them only. As a consequence, even though they can reliably identify the activity of scanning worms, they are almost blind to infection attempts made by topological worms. The reason for this is that topological worms do not, or only rarely, probe the IP address space to find Internet-connected machines running vulnerable software to infect. A second weakness of honeypots is that they are subject to risks. In more detail, they can be hijacked, controlled and used to attack, infiltrate or harm other Internet-connected machines. Although the risk level varies for the different types of honeypots, the risk of takeover has increased considerably, especially since commercial anti-honeypot tools, such as the Honeypot Hunter [102], have surfaced. Furthermore, for the same reasons discussed above in connection with the signature-based detection systems, honeypots go to no lengths to reduce the amount of illegitimate traffic on the Internet.

For completeness' sake, network telescopes, which are conceptually very similar to honeypots, are briefly presented in what follows. Central to network telescopes are the notions of a darknet and a greynet. A darknet is a (large) block of IP addresses in which no active services or servers reside; whereas, a greynet is a (large) block of IP addresses that is sparsely populated with darknet addresses interspersed with active IP addresses [67]. Similarly to honeypots, network telescopes can accurately identify Internet-connected machines infected with scanning worms but only rarely

capture infection attempts made by topological worms. As a matter of fact, over the past years, network telescopes, such as the UCSD Network Telescope [128] and the Internet Motion Sensor [11], have monitored the Internet-scale spread of many scanning worms. By doing so, they have provided researchers with the necessary information to model the Internet-scale propagation of Internet worms that use various scan techniques. Nevertheless, due to the limitations of the public IP address space, only big organisations and universities involved in Internet security research generally operate network telescopes on public IP addresses.

3.2 Behaviour-Based Detection Methods

The limited usefulness of the general detection methods against zero-day, polymorphic and, above all, topological worms, has motivated researchers and security professionals to look for alternative methods. Specifically, a great deal of work has been directed at behaviour-based (anomaly-based) detection, which is regarded as the counterpart of misuse detection. Instead of searching for explicit indications of Internet worm activity, behaviour-based detection involves modelling in terms of network traffic the normal behaviour of non-infected machines. Thereby, any machine whose traffic patterns deviate from the normal behaviour is automatically detected and classified as suspicious. However, these unexpected traffic patterns, referred to as anomalies, can be caused by a variety of factors. Therefore, to decide whether a misbehaving machine is infected or not, behaviour-based detection requires loosely correlating certain anomalies with Internet worm activity. This, in turn, needs some specifications of the expected behaviour of infected machines, which, as mentioned before, are known as behavioural signatures. In most cases, these specifications are determined by analysing the operations of Internet worms that relate to their propagation; thus, they are worm class-specific. Hence, behavioural signatures determined for Internet worms of a class are usually ineffective for detecting machines infected with Internet worms that belong to other classes (or subclasses).

With respect to this, in what follows, the behaviour-based detection methods proposed in the literature are discussed for each class of Internet worms separately. It is worth noting that there exists no behaviour-based detection method that has the potential to be effective in identifying activity of topological worms that belong to all, or at least more than one subclass. This is because, as will become clear later in this chapter, the operations that relate to the propagation of topological worms that spread on different social networks differ significantly. Therefore, after first presenting the available behaviour-based detection methods for scanning worms, one subsection is devoted to describing the available behaviour-based detection methods for each of the three subclasses of topological worms (IM, P2P and email worms). Since this work introduces a new behaviour-based detection method for email worms, the discussion below is intended to elaborate on two key points that motivate doing

so. The first point concerns explaining the reasons why the available behaviour-based detection methods for Internet worms of other classes, and subclasses, cannot be directly applied or adapted to detect email worm activity. The second point involves uncovering the inherent common weaknesses of the available behaviour-based detection methods for email worms that had to be taken into account during the design phase of the method introduced later in this thesis.

3.2.1 Methods for Scanning Worms

To determine behavioural signatures that are useful for detecting Internet-connected machines infected with scanning worms, the focus has been mainly on the following three directions: monitoring the TCP/IP connections, finding meaningful correlations between the incoming and outgoing traffic and analysing the traffic of critical network services associated with Internet-connected machines. The vast majority of the behavioural signatures that have been proposed are in the simple form of inferring scanning worm activity if the value of a metric of interest measured over a sliding time window exceeds its long-term average.

Based on the assumption that the infected machines probe the IP address space for potential victims at the highest possible speed, the Snort [147] keeps track of the TCP SYN packets (requests to establish a connection) that the machines in a monitored network generate within a time interval of a few seconds. Any machine whose average connection rate (connections per second) to unique destinations exceeds a predetermined (static) threshold value is classified as infected. Using the same principle, Wagner et al. [185] employs entropy as a measure of randomness of the unique destination IP addresses. Yu et al. [210] advances Wagner et al. by using the statistical entropy of the distribution of the destination IP addresses in conjunction with a set of dynamic decision rules to detect scanning worms that, like the Atak worm, are capable of varying their probing rate as they spread. Sekar et al. [152] introduces a multiresolution method that has the potential to detect scanning worms with various probing rates. This method uses a set of static threshold values for the connection rate instead of only one. The Bro [139] is based on the observation that the failed connections rate (failed connections per second) of an Internet-connected machine is a more accurate indicator of scanning worm activity than its connection rate. The rationale is that probing the IP address space for Internet-connected machines running vulnerable software unavoidably results in a high number of failed connections. The reason for this is that the IP address space is sparsely populated. Venkataraman et al. [178] uncovers that the Snort and Bro have high memory requirements since they store the complete list of the source-destination IP address pairs within the time interval. Motivated by this observation, it presents a system that relies on the same behavioural signatures as these two systems but has low memory and processing requirements.

More sophisticated connection-oriented methods are based on the theory of sequential hypothesis testing [149, 85, 165, 84]. The key idea is that using dynamically adaptable instead of static threshold values leads to higher accuracy. In particular, the relevant studies suggest that the value of a metric of interest can be modelled as a random walk on one of two stochastic processes corresponding to the successful and failed connections. Thereby, it is possible to adapt the threshold value dynamically according to the traffic that the machines in a monitored network generate. In connection with this, Jung et al. [149, 85] introduces the Threshold Random Walk (TRW), which allocates a certain number of connection credits to each machine in the monitored network. Within a time interval, an Internet-connected machine is allowed to connect to other Internet-connected machines it has not contacted before only if its credit balance is positive. The machine gains one credit for each successful connection, whereas each failed connection depletes one credit. Studer et al. [165] introduces the Adaptive Rate Scan Detection (ARSD), which is a refinement of the TRW. The ARSD looks at the credit reward and penalty rates, which are derived from the connection success and failure statistics, respectively. Jung et al. [84] advances the TRW to cover the case of Internet worms that are capable of spreading extremely fast, even by means of relatively low-rate probing.

The detection methods that inspect only the outbound connections of Internet-connected machines have three common weaknesses. The first of these weaknesses is that they are easily evaded by stealthy worms. The term stealthy worm is used in the literature and what follows to refer to a scanning worm that probes the IP address space over a long time period. The second weakness is that they do not have the potential to distinguish the activity of scanning worms from other occasional legitimate scan activities. The third weakness is that, based on their threshold values, they impose either too much constraint on legitimate traffic or allow too much illegitimate traffic to pass. For these reasons, a number of connection-oriented methods that look at more indicative symptoms of scanning worm activity, primarily correlations between the incoming and outgoing traffic of Internet-connected machines, are proposed in the literature [33, 60, 29, 31, 43].

In more detail, Cheung et al. [33] analyses connection graphs to infer causality relationships among connections in a monitored network that reveal scanning worm activity. Gu et al. [60] is based on the observation that a machine is being probed prior to becoming infected and, once infected, it probes the IP address space for machines running vulnerable software. Thereby, it classifies as infected any machine that receives a packet on a certain port and after some seconds starts sending out packets destined for the same port with an increasing rate. Chen et al. [29] introduces the WormTerminator. The WormTerminator is a cloned virtual machine of the machine it runs on that appears to be the destination of all outgoing traffic. Once the machine becomes infected, it attempts to infect the WormTerminator. Chen et al. [31] suggests matching the destination port numbers between the inbound and

outbound connections and looking for sudden increases in the number of distinct destination IP addresses of outbound connections to suspicious ports. Ellis et al. [43] identifies and uses the following three behavioural signatures of scanning worms: their packets have similar payloads, they are spread by Internet-connected machines acting as clients and exhibit tree-like propagation and reconnaissance.

Other studies focus on the traffic of critical network services and demonstrate that the information it carries provides a solid basis for identifying scanning worm activity [15, 16, 191, 53, 198, 193]. Namely, the TRAFEN [15, 16] involves capturing the Internet Control Message Protocol (ICMP) destination unreachable (ICMP-T3) packets on the gateway of a monitored network. The ICMP-T3 packets are associated with failed connection attempts. In connection with this, the premise of this method is that a high rate of failed connection attempts from an Internet-connected machine results in a high number of ICMP-T3 packets destined for this machine. Thereby, the TRAFEN classifies as infected any machine whose IP address appears frequently in the destination IP address field of the captured ICMP-T3 packets. Whyte et al. [191] examines DNS traffic captured on the local name server of enterprise networks. The method proposed in that study builds on Ganger's observation, which is introduced in [53] and states that the non-existence of DNS queries before a connection attempt is a telltale sign of scanning worm activity. Wong et al. [198] introduces a similar method based on an extension of Ganger's observation. Whyte et al. [193] presents a method that detects scanning worm activity on enterprise networks by inspecting the outgoing Address Resolution Protocol (ARP) traffic of user machines.

As previously stated, the behavioural signatures that are useful for detecting the activity of Internet worms belonging to a specific class are in most cases ineffective against Internet worms of other classes (or subclasses). This implies that the methods described above are ineffective in detecting email worm activity. However, to make this clear further elaboration on this issue is provided below. Email and scanning worms differ in four important ways. First, in contrast to scanning worms, email worms do not attempt to establish TCP/IP connections to their potential victims but propagate via email servers. Thereby, user machines infected with email worms exhibit different connection patterns from those that are infected with scanning worms. Second, to discover their potential victims, email worms exploit the social information they harvest from the machines they infect instead of probing the IP address space. Therefore, their propagation is not associated with many connection failures. Third, email worms operate on significantly longer time scales than scanning worms because their propagation depends on interactions with the end users. Fourth, email worms propagate in a seemingly legitimate manner by misusing email. As a consequence, the traffic they produce blends in with legitimate email traffic.

The first two differences mentioned above, render the methods that look for anomalies in the TCP/IP connection patterns of Internet-connected machines directly (high connection or failed connection rates) or their side effects (many ICMP-T3

packets) ineffective in identifying user machines infected with email worms. Furthermore, the third difference implies that the methods that are based on exploiting either the causality of connections or the correlations between the incoming and the outgoing traffic of Internet-connected machines are also ineffective in identifying email worm activity. This is because these methods are designed to operate within relatively short time intervals. They are namely suited for identifying the occurrence of an event of interest that depends on the occurrence of another event that happened just a few seconds before. Finally, the last difference between email and scanning worms makes the methods that build on Ganger's observation or analyse ARP traffic unsuitable for detecting user machines infected with email worms. The reason for this is that email worms propagate in a legitimate communication channel.

3.2.2 Methods for IM Worms

IM provides real time message delivery and rich presence information service over the Internet. The classic IM applications work according to either the client-server or the P2P model. In applications based on the former model, communications between the IM clients occur via IM servers. By contrast, in those based on the latter model, connection requests are sent via IM servers but messages are exchanged between the IM clients directly. The client-server model is by far the most popular one, as it is used by the MSN Messenger, Yahoo! Messenger and AOL Instant Messenger. Therefore, only methods for detecting IM worms that spread on public client-server IM networks are discussed hereafter. The IM worms of this type typically propagate by connecting via the IM server to all the user machines for which an IM address exists in the online contact list of the user machines they infect. After connecting to the target user machines, the IM worms use one or more of the following three methods to propagate: they send URL-embedded chat messages, initiate automated file transfer requests or exploit vulnerabilities in the IM clients, the operating system or the applications that run on the target user machines [118].

Despite the fact that the ever-increasing adoption of public IM applications leads to an ever-growing threat of IM worm infections, only a handful of studies propose behaviour-based detection methods for IM worms [112, 202, 204]. Specifically, Liu et al. [112] focuses on IM worms that send URL-embedded chat messages to all the IM addresses in the online contact list of the user machines they infect. The proposed method involves tracing back and detecting infected user machines by analysing the multicasted chat messages that IM servers receive. In particular, all the multicasted chat messages received by an IM server within a time interval are captured and for each of them a tree-like directed acyclic graph is constructed. The nodes of the graph represent the recipients of the chat message. If the depth and the width of a tree exceed some predetermined threshold values, the multicasted chat message under consideration is classified as being generated by an IM worm. Then, all the

infected user machines are discovered by traversing the tree. Liu et al. elaborates also on two refinements of this method that involve extracting attack signatures and measuring the prevalence of multicasted chat messages. Xie et al. [202] introduces HoneyIM that, as its name suggests, is based on the concept of honeypots. HoneyIM assumes a set of decoy IM addresses that should not receive URL-embedded chat messages or file transfer requests. Thereby, any user machine running an IM client that attempts to communicate with these decoy IM addresses is classified as infected. Yan et al. [204] concentrates on both the number of URL-embedded chat messages and the number of file transfer requests. It suggests that abrupt increases in the values of these two metrics are a sign of IM worm activity. Additionally, for catching stealthy IM worms it proposes to determine threshold values for the number of URL-embedded chat messages and the number of automated file transfer requests between IM address pairs that barely communicated in the past.

Because IM and email worms are subclasses of topological worms, they bear some similarities in that they propagate among social contacts on an overlay network made up by user machines and application servers (IM or email servers). Despite these similarities, the behaviour-based detection methods for IM worms, which are proposed in the literature, are not useful for identifying email worm activity. The reason for this lies in that all these methods are essentially based on an intrinsic property of IM networks that does not apply for the Internet-scale email network. This property is namely that, from a macroscopic point of view, the public client-server IM networks form controlled closed environments. One central IM server stands in the middle of the IM network and controls all the communications among IM clients. The existence of such a central control point is crucial for all the detection methods for IM worms described above. However, the closed environment property is not met in the Internet-scale email network. This is because, in the typical email exchange scenario, a different pair of email servers distributed on the Internet is responsible for the email communication between two email users.

3.2.3 Methods for P2P Worms

In recent years, P2P networks have evolved into a very attractive field for attackers seeking to compromise user machines by spreading Internet worms. This is because of the increasing popularity of P2P applications, which is associated with an equal growth in the number of end users that are regularly connected. Depending on how their peers are linked to each other, P2P networks are classified as unstructured if the links are established arbitrarily or as structured if the peers organise themselves in an orderly fashion. Most of the widely-used P2P networks, including Kazaa, Gnutella and BitTorrent, are unstructured networks that are used for file sharing. By contrast, the structured P2P networks, such as the Pastry [148] and the Chord [163], remain primarily an area of academic interest. As a consequence, the research on detecting

P2P worms, which has been undertaken so far, concentrates on P2P worms that spread on unstructured file-sharing networks.

Based on their propagation details, P2P worms fall into either of two categories: passive or active. The passive P2P worms compromise user machines if downloaded and opened, and then leave copies of themselves in the shared folders. To lure unsuspecting end users into downloading them, they are usually disguised as or hidden in popular files. By contrast, the active P2P worms behave much like scanning worms. In more detail, they propagate by sending copies of themselves to potential victims they find either by exploiting the P2P neighbourhood information stored on the user machines they infect or by collecting offline the IP addresses of the user machines that compose the P2P networks. In the latter case, all the infected machines share the same target list [208, 209]. Despite their similarities with scanning worms, Göldi et al. [56] shows that the detection methods for scanning worms cannot differentiate between non-infected user machines and those infected with active P2P worms. Chen et al. [26] categorises active P2P worms further into reactive and proactive. The reactive worms rely on certain end user actions and propagate hidden in legitimate traffic, whereas the proactive ones self-replicate by taking advantage of vulnerabilities. Although the passive P2P worms propagate rather slow, which often allows detecting them before they reach their full potential, the vast majority of P2P worms belong to this category [155, 87].

There are two basic observations behind the detection methods for P2P worms proposed in the literature. The first of them is that the popularity of peers and that of files in a P2P file-sharing network follow a certain distribution – the Zipf distribution [217]. The second observation is that the propagation of P2P worms distorts these distributions. Chen at al. [26] observes that the distortions are typically manifested as abrupt changes in the inbound or outbound TCP/IP connections rates of the peers. Thereby, it proposes to classify as infected any machine whose outbound connection rate exhibits sudden increases above a predetermined threshold value. Zhang et al. [212] advances Chen at al. by assuming the existence of a set of peers with specialised detection capabilities, which are referred to as guardian nodes. The guardian nodes are capable of sensing increases in the inbound and outbound connection rates of their neighbouring peers. Moreover, inspired by the work presented in Zhou et al. [214] and Malan et al. [115], the guardian nodes in Zhang et al. can recognise the infection process of P2P worms inside running applications and identify repetitive network behaviours.

Notably, the studies discussed above do not report on evaluation experiments using traffic traces of P2P worms or real network topologies. As a matter of fact, they support their claims only by means of theoretical analysis or computer simulations. As a result, their findings are necessarily idealised abstractions of how the methods they propose would operate on a real network. This constitutes a significant barrier to their wide adoption and use. If this major deficiency is put aside, the ineffectiveness

of these methods in detecting email worm activity can be easily uncovered. The reason for this is that, in a similar way as the connection-oriented behaviour-based detection methods for scanning worms, all the revised methods base detection on the TCP/IP connection rates of user machines. Nevertheless, as explained earlier in this thesis, the connection rates of a user machine provide no reliable basis for deciding whether it is infected with an email worm or not.

3.2.4 Methods for Email Worms

The main corpus of the research work on determining behavioural signatures that are useful for detecting email worm activity naturally concentrates on analysing email traffic. Driven by two different but complementary objectives, this work has proceeded in two directions. The objective of the methods in the first direction is to detect user machines that are infected with email worms as early as possible to prevent them from sending an infectious email to many potential victims (non-infected user machines). To this end, these methods inspect the emails that user machines generate either directly on the (sender) user machines or other Internet-connected machines in their network, mainly on the outgoing email servers. The objective of the methods in the second direction is to detect infectious emails before or once they reach the potential victims to prevent further infections. To accomplish this, these methods inspect the emails that are destined for user machines either directly on the (recipient) user machines or other Internet-connected machines in their network, mainly on the incoming email servers.

There is a small number of methods in the first direction that are intended to operate directly on user machines. These methods take advantage of the fact that email worms harvest the email addresses of potential victims from the file system of the user machines they infect [74, 75, 19]. In connection with this, these methods are based on hiding some bait email addresses in the file system of user machines and, then, constantly monitoring the email traffic that user machines generate. Any user machine that attempts to send an email to any of the bait email addresses is automatically classified as infected. In particular, Hu et al. [74] proposes to generate a set of bait email addresses of enough unpredictability and hide them in files with predefined extensions, such as .eml, .txt. and .htm files. Similarly, Huang et al. [75] suggests inserting in the address book of each email user an email address that is not used by any registered email user of the local domain. Boldizsár et al. [19] proposes to deploy a dedicated server that is responsible for issuing one unique bait email address for each user machine on a monitored network.

Storing bait email addresses in the file system of user machines is an effective method for detecting spam. However, it does not fit well into the case of email worms. This is because of its passive nature and the propagation characteristics of email worms. In particular, infectious emails sent to non-bait email addresses

spread an email worm further unaffected. This implies that one weakness of the methods previously described relates to the time that it takes email worms to find the bait email addresses. Nevertheless, the most serious weakness of these methods is that, as any other method designed to operate on user machines, they need an almost complete deployment throughout the networks of the Internet to significantly affect the Internet-scale epidemics of email worms. This, in turn, requires that end users voluntarily participate and take measures to protect their networks. Even though this can be very helpful, in practice, it is highly unlikely that all, or at least most, end users will participate. This is either due to their ignorance of the threats associated with Internet worms or concerns about possible negative effects on their machines. Thereby, to eliminate the need for global deployment to be effective, most of the methods in the first direction operate on other Internet-connected machines (network-based) instead of directly on user machines (host-based).

A number of these methods involve statistically analysing the email traffic that user machines generate on the outgoing email servers, and share two common elements [61, 83, 154, 119, 120, 123, 13, 164]. Namely, they are based on defining features and specifying the normal values thereof, which are the values that correspond to legitimate email activity. In this context, a feature is a statistic that describes an aspect of email activity. Thereby, any significant deviation of a feature from its normal value is considered to be an indication of email worm activity. Gupta et al. [61] defines a set of features for the volume of the email traffic and automatically specifies their normal values by means of statistical learning. Jiang et al. [83] also defines a set of features manually but relies on domain knowledge to specify their normal values. Shih et al. [154] classifies a large number of emails as legitimate or malicious by means of a commercial antivirus software. Based on the classification results, it defines as features 11 static properties of email traffic that present the highest discriminatory power for distinguishing between legitimate and email worm activity. Martin et al. [119, 120] defines numerous features with the rationale that using many features increases the discriminatory power. Masud et al. [123] applies Principal Component Analysis (PCA) to the set of features of Martin et al. to find an optimal subset. Barreno et al. [13] and Stolfo et al. [164] take a step further and define also features other from those that describe the volume and the rate of the email traffic. Barreno et al. defines features that generalise across all the user machines on a monitored network. Stolfo et al. introduces the Email Mining Toolkit (EMT). The EMT uses features that describe the social activity of email users. The principle behind the EMT is that each email user usually sends emails to a particular group of email addresses, whereas email worms to all the email addresses they harvest from the file system of the user machines they infect.

Instead of analysing only the email traffic that user machines generate, other network-based methods look also at the email traffic that is destined for user machines. These methods usually exploit correlations between the incoming and outgoing email

traffic of user machines, which are indicative of email worm activity [99, 203, 58]. Specifically, Kim et al. [99] inspects all the incoming and outgoing traffic on email servers and classifies as infectious any email that is distributed from one to many user machines shortly after it has been delivered unchanged in the same manner. Xiong [203] introduces the Attachment Chain Tracing Scheme (ACT) that involves monitoring traffic on email servers, as well, and identifies infected machines based on a two-step procedure. The ACT classifies as suspicious all the user machines on a monitored network that send out many emails and decides which of them are infected by tracing their contacts. Based on the same principle, Gopalan et al. [58] presents the SenderCount, which is based on a different two-step procedure. The SenderCount identifies prevalent signatures in the email traffic by means of a scheme that is based on the Rabin fingerprints. If both the number of user machines that send out emails containing the prevalent signatures and the sending rate thereof exceed some predetermined threshold values, the SenderCount classifies as infected any user machine that attempts to connect to many email servers.

The network-based methods discussed so far – irrespective of whether they look at statistical properties of the outgoing or correlations of the incoming and outgoing email traffic of user machines – share two common deficiencies, which limit their general utility. The first of them is that they are designed to operate on outgoing email servers. As a matter of fact, this detracts significantly from their effectiveness in identifying user machines infected with any of a large variety of recent email worms. The reason for this is that, as mentioned in Section 2.3.3, most of the email worms that appear in the wild are equipped with a built-in email engine and, thus, do not use the outgoing email servers. This would have been discovered if these methods had been evaluated against real or a large set of email worms. However, because of the scarce availability of email worm traffic traces, these methods were evaluated either by means of computer simulation or against a few email worms. Hence, their claims and assumptions regarding the false positive and negative rates are necessarily idealised performance predictions. The second deficiency of these methods is their dependency on an amount of a priori information. Namely, tuning them requires either a set of manually or otherwise classified data, or domain knowledge to empirically determine threshold values, which make it possible to detect significant deviations from the normal email activity. However, such a priori information is only rarely, if at all, readily available in practice.

One of the first methods in the second direction – consistent with the rationale of detecting infectious emails before or once they reach the potential victims to prevent further infections – is presented in Balzer [12]. Its basic idea is to enhance email clients with specialised wrappers that monitor user machines for suspicious activity every time an attachment file is received and executed. This includes tracking accesses to the file system and (Windows) system registry, connection attempts and spawned processes. Based on the same principle, the Malicious Email Tracking

(MET) [17] scans emails on the incoming email servers for known malicious programs. Sidiroglou et al. [156] proposes to execute all the suspicious attachment files inside an instrumented virtual machine. Any anomalous access to the (Windows) system registry reveals that the attachment file under consideration is malicious. Taibah et al. [167] describes a similar method. It suggests running all the attachment files with certain extensions on a virtual machine and storing their fingerprints. Every new incoming attachment file whose fingerprint has been already analysed is classified according to the result of the previous analysis. Yoo et al. [207, 206] suggests quantifying the suspiciousness of incoming emails by applying Bayesian inference to some of their attributes, such as their sender and recipient email addresses. Schultz et al. [151] introduces the Malicious Email Filter (MEF). The MEF assumes the presence of similar byte sequences in malicious executables that differentiate them from legitimate programs. Thereby, the MEF is trained offline using a set of labelled programs and, then, detects online infectious attachment files.

One common characteristic of the methods in this second direction is that they are intended to operate on the network of the user machines they protect. Therefore, they go to no lengths to reducing the amount of illegitimate email traffic traversing the Internet. However, as previously pointed out, the illegitimate email traffic is the main concern of network operators regarding email worms. Apart from this serious weakness, the methods revised above focus solely on email worms that spread in the form of (Windows) attachment files. As a consequence, they are ineffective against a large number of email worms that come as infectious emails with links to compromised Web sites. Moreover, these methods rely on a priori assumptions about the infectious attachment files or the malicious processes spawned on user machines once these attachment files are executed. Let aside that such assumptions are not justifiable because judgements on executables are usually relevant to platform implementations, in none of the presented studies details about the detectable malicious activity or experimental results against real email worms are given. The former is probably because examining the purpose of executable code is an extremely complex task. For all these reasons, the general usability and the detection efficacy of the methods proposed in these studies remain to a great extent questionable.

DNS-Based Methods for Email Worms

As previously mentioned, most of the work on determining behavioural signatures that are useful for detecting email worm activity focuses on analysing email traffic. Nevertheless, in pursuing this objective, a number of studies are concerned with providing insight into and exploiting the characteristics of the DNS traffic that infected user machines generate [197, 132, 124, 133, 192, 82]. Because research on Internet security methods that operate on the DNS traffic of user machines stands at the centre of the study presented in this thesis, the key findings and implications of these studies are discussed in more detail in what follows.

Wong et al. [197] analyses DNS traffic captured on the local name server of a university network during the outbreaks of Sobig.F and Mydoom.A. Musashi et al. [132] reports on a similar analysis that provides additional information on these two email worms. The authors of that report continue their research and extend their scope by examining the DNS traffic that user machines infected with any of Netsky.C [124], Netsky.Q and Mydoom.S [133] generate. Independent from one another, Whyte et al. [192] and Ishibashi et al. [82] inspect the DNS traffic of user machines infected with Netsky.Q. In particular, Whyte et al. analyses DNS traffic captured on the local name server of an enterprise network, whereas Ishibashi et al. DNS traffic captured on the local name servers of a large commercial Internet Service Provider (ISP). Despite their differences regarding the details of the analysed DNS traffic, all these studies essentially converge toward a common conclusion. This is namely that the application-level characteristics of the outgoing DNS traffic of a user machine change once it gets infected.

This conclusion serves in a straightforward manner as the basis for most of the work on determining behavioural signatures based on DNS traffic. As a matter of fact, the methods paired with the analyses in [132, 124, 133, 192] and the one described in Binsalleeh et al. [18] rely on the same principle. Specifically, they inspect the outgoing DNS traffic of user machines at the application level searching for specific volume-based anomalies. Their core idea is that the number of DNS queries for MX RRs and the relation between the numbers of DNS queries for pointer (PTR) RRs, MX and address (A) RRs of user machines provide a strong basis for identifying email worm activity. Although this holds true for the few email worms considered in each study, it cannot be generalised for detecting user machines that become infected with email worms in the long run. In fact, in the next chapter of this thesis, it is shown through specific examples that the number of DNS queries a user machine generates is a rather weak indicator of whether it is infected or not. Furthermore, as any other threshold-based method, these methods require domain knowledge for determining suitable threshold values that make it possible to distinguish between non-infected and infected user machines.

Instead of determining threshold values for the number of DNS queries that user machines generate, other methods that also inspect the outgoing DNS traffic of user machines at the application level use more sophisticated procedures [82, 211]. In particular, Ishibashi et al. [82] assumes a priori knowledge of a characteristic set of DNS queries that infected user machines generate. Based on this knowledge, the method proposed in that study detects user machines infected with email worms by applying Bayesian inference to the DNS queries local name servers receive. Zhang et al. [211] advances Ishibashi et al. by defining and computing a score for each user machine on a monitored network that represents the probability that it is infected. To accomplish this, the method introduced in that study builds decision trees from the DNS queries captured on local name servers within a time interval. Despite

their sound mathematical grounding, these studies have a methodological problem that brings the validity and usefulness of the methods they propose into question. This problem is that they essentially assume a priori knowledge about zero-day worms. Obviously, such knowledge is not apparent, and if it were it would allow straightforward detection.

Apart from the weaknesses already pointed out, the studies revised above share two additional common weaknesses that further restrict the usefulness of the methods they propose. The first of them is that the proposed methods perform deep packet inspection on the DNS queries that the user machines generate. This reveals that these studies overlook that the DNS queries carry sensitive end user information, which imposes many privacy requirements and, thus, is not a suitable candidate for such inspection. Besides this, this weakness renders the proposed methods unsuitable for high speed, busy networks, because of the high processing overhead associated with analysing packet payloads, and ineffective against polymorphic email worms that might appear in the future. However, the second weakness is by far the most significant. This is that the claims of these studies are supported by rather weak experimentation because of the lack of realistic benchmarks. In particular, these studies report on experiments against a small number of email worms and do not include assessment details using real network topologies. Overall, it is due to these weaknesses that, despite their positive contribution of defining a correlation between email worm activity and DNS traffic, these studies have only modestly, if at all, contributed to developing systems for detecting user machines infected with email worms in the long run.

3.3 Summary

Depending on their scope, the detection methods for Internet worms are classified as general or specialised. The general detection methods are designed to be effective against Internet worms of all classes, whereas the specialised only against those that belong to one particular class (or subclass). The specialised detection methods are based on the concept of behavioural signatures, and thereby are referred to as behaviour-based (anomaly-based). In this chapter, a thorough and critical review of the available general detection methods is presented, which reveals that all of them suffer from one or more of the following deficiencies:

- they are ineffective against zero-day worms because they rely on human interactions, which introduce significant delays,
- they can be readily evaded by polymorphic worms since they search for network footprints, which can be easily modified,

- they exhibit lack of or limited efficacy in detecting topological worms because they are primarily designed for scanning worms,
- they are not suitable for busy, high speed networks due to the high processing overhead they incur by analysing packet contents,
- they do not target limiting the worm traffic sent to the Internet since they are deployed on the networks of the potential victims,
- they are subject to high risk of being compromised, and being subsequently used to attack other Internet-connected machines.

The limited effectiveness of the general detection methods in identifying zero-day, polymorphic and, mainly, topological worm activity, shifted the focus to behaviour-based detection methods. These methods involve finding traffic patterns that indicate infection. These patterns are primarily determined by analysing the operations of Internet worms that relate to their propagation and, as a result, are useful for detecting Internet worms that belong to one class (or subclass) only. In this chapter, a comprehensive and critical review of the available behaviour-based detection methods is provided, which sheds light on the reasons that render these methods incapable of dealing with email worms in an effective manner. In particular, these methods are limited in one or more of the following ways:

- they are designed for scanning, IM or P2P worms and, as a consequence, are inherently ineffective against email worms because the traffic patterns they detect do not appear in the traffic of email worms,
- they rely on knowledge, including technical details about the operations of email worms, for setting threshold values and classifying data to train systems, which is only rarely, if at all, available in practice,
- they operate on or near the potential victims and, thereby, they cannot reduce the illegitimate email traffic that traverses the Internet, which is for network operators the main problem regarding email worms,
- they violate the privacy of end users, they cannot work with encrypted packets, and they are unsuitable for busy, high speed networks since they perform extensive traffic monitoring and deep packet inspection,
- they have limited utility because, as a result of being poorly evaluated, their scope is narrow, as their observation basis is not general enough for detecting user machines infected with email worms in the long run,

- they operate on user machines and, thus, they cannot be deployed incrementally but instead need a complete deployment throughout the networks of the Internet to affect the Internet-scale epidemics of email worms.

In light of the analysis presented in this chapter, it becomes clear that more effort is required to come up with effective methods for detecting email worm activity. In this direction, a new behaviour-based detection method for email worms that overcomes the limitations of the methods discussed in this chapter is introduced in Chapter 4. The method identifies email worm activity by analysing at the flow level the DNS traffic that user machines send to the local name servers.

Chapter 4

Proposed Detection Method

4.1 Overview

In this chapter, a new behaviour-based method for detecting user machines infected with email worms on the local name server of a monitored network is introduced. The method bases detection on the flow-level characteristics of the DNS traffic that user machines generate. Its observation basis is that most humans, regardless of their individual habits, exhibit similar Internet usage behaviours; thus, it is reasonable to expect that the outgoing DNS traffic of non-infected user machines share many of the same patterns. Likewise, email worms are programs, and as such they have only a certain number of modules that they use to harvest email addresses, propagate and attack Internet-connected machines. Even when various modules are in hand, they are limited in number. Therefore, it is also justified to assume that there have to be similarities in the outgoing DNS traffic of user machines being infected with instances of a single, or even of different email worms. In this work, it is demonstrated that depending on the flow-level characteristics of the DNS query streams they produce, user machines do indeed fall into either of two canonical profiles: legitimate user or infected with email worm. Using clustering and exact shape-based similarity search over time series derived from their DNS query streams, the method automatically classifies the user machines that query a local name server into one of these profiles. The experimental results in Section 4.4 and the discussion in Section 4.5 suggest that the method can be effective in accurately detecting user machines that become infected with email worms, even with zero-day, in the long run.

The proposed method overcomes the limitations of the available behaviour-based detection methods discussed in Chapter 3. Specifically, it has five advantages that emerge from processing the traffic of user machines on the local name server, which is for these machines the first link in their entire chain of Internet connectivity. First, it needs a single-point deployment that does not depend on end user participation. Second, it is very effective, among other reasons, because the efficacy of a behaviour-

based detection method increases as the topological proximity, measured in terms of hop-count, between its deployment point and the Internet-connected machines it protects decreases. This is supported by the fact that higher proximity implies finer-grained visibility into the network behaviour of these machines. Third, it is easy to implement and has low overhead, as the traffic that local name servers receive from user machines accounts for a small amount of data. Fourth, as will be explained in Section 4.5, it cannot be evaded by simple changes in the code of email worms. Fifth, in conjunction with a containment method, it can contribute to limiting the illegitimate traffic infected user machines send to the Internet. This is because email worms cannot hide from the local name servers the IP address of the infected user machine when they query them for the RRs they need to prepare infection attempts, spamming and DDoS attacks. The method has also three advantages that stem from analysing traffic at the flow level. First, it does not violate the privacy of end users. Second, it is less computationally demanding than methods that perform deep packet inspection. Third, it cannot be defeated by encryption; hence, it can be effective against polymorphic email worms that might appear in the future. One last advantage is that the method is highly automated since it does not require training before applying it or maintenance during its operation.

As pointed out in Section 3.2, the effectiveness of the available behaviour-based detection methods, regardless of the Internet worm (sub)class they are designed for, depends to a great extent on domain knowledge about the expected behaviour of the infected machines (behavioural signatures). This knowledge is usually required to determine proper threshold and parameter values or construct datasets to train the methods. Typically, this knowledge is obtained by analysing traffic traces captured during the outbreaks of known Internet worms that belong to the targeted Internet worm (sub)class. Let alone that inferring meaningful knowledge from traffic traces, is a very difficult and sometimes even impossible task, such traces are only rarely, if at all, made publicly available. Even when they are made available, the knowledge one can extract from traffic traces of non-infected and machines infected with known Internet worms captured on a monitored network, is not, necessarily, useful for tuning a method to detect machines infected with new Internet worms on other networks. To minimise the dependency of behaviour-based detection on domain knowledge, a new approach that uses clustering is proposed and taken in this work. Clustering organises a given set of data objects into groups so that similar objects are grouped together. By means of clustering, the proposed method assigns a set of user machines into distinct profiles based on the similarities in the flow-level characteristics of their DNS traffic. In connection with this, it essentially needs only a decision regarding which flow-level characteristic has to be considered to distinguish between non-infected and infected machines.

Based on the analysis of the email worm life cycle in Section 2.3.2, it becomes clear that once a user machine gets infected the number of DNS queries it produces

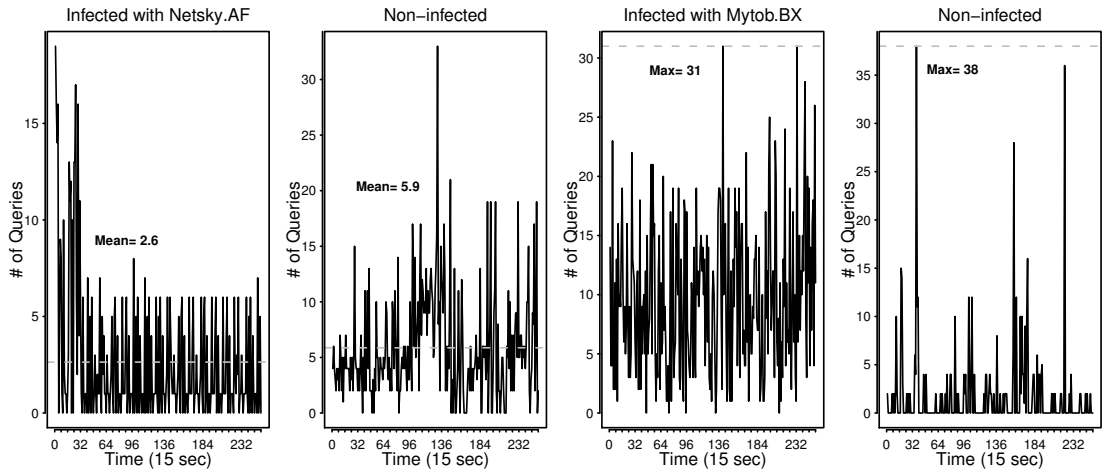

Figure 4.1: Non-infected user machines can have a greater mean number of DNS queries than those being infected with email worms (left plots) or produce DNS query spikes similar to those that infected with email worms user machines produce (right plots). Thereby, the mean and maximum numbers of DNS queries that the user machines in a monitored network generate within a time interval are not reliable indicators of email worm activity.

over time is affected. Hence, it is reasonable to assume that infected user machines exhibit a higher DNS query rate than non-infected or generate unexpectedly large DNS query spikes. Under this assumption, it is justified to expect that determining threshold values for the mean or maximum number of DNS queries that a local name server receives from a user machine within a certain time interval, suffices to classify each requesting user machine as non-infected or infected. However, the two counter-examples in Fig. 4.1 and the experimental results in Section 4.4.3 show that this assumption does not hold true in practice. Fig. 4.1 indicates that non-infected user machines can have a greater mean number of DNS queries than those being infected or produce DNS query spikes similar to those that infected user machines produce; spikes correspond to extremely large values of momentary throughput. As a consequence, these two simple flow-level characteristics are weak indicators of email worm activity. This necessitates devising more sophisticated methods that can better capture the dissimilarities in the DNS query streams of non-infected and infected user machines at the flow level. Thereby, the proposed method uses exact shape-based similarity search over time series. A time series is a collection of observations made sequentially over time, which are formally called time points and represent the status of a variable over time. The variable considered in this work is the number of DNS queries originating from a user machine. A time series made up by p time points is referred to as $p-$length time series hereafter.

The proposed method takes as input the log file fragment or a packet capture file with the DNS queries that a local name server received within a certain time

interval, called detection period hereafter. Each query entry in either of these files consists of the time when the name server received the DNS query, the IP address of the requesting machine and the requested RRs. Since the method works at the flow level, only the query reception time and the IP address of the requesting machine are used from each entry. To take into account only DNS queries originating from user machines, the entries that correspond to servers in the monitored network whose operation depends on the local name server are filtered out. Such servers are, for example, the email servers and legitimate automated mailers, such as library notification systems. Even on large networks the number of these servers is manageable and their IP addresses known, so excluding these entries does not violate the requirement of practicality. The rest query entries are grouped per requesting user machine and, then, split into successive time bins of equal duration. By counting for each user machine the entries that fall into each time bin, a set of time series is produced; each time series describes the number of DNS queries of a user machine over time. The time series are organised into an $n \times p$ time series matrix, where each row represents the DNS query stream of a user machine; n is the number of user machines that queried the local name server of the monitored network at least once within the detection period and p is the number of time bins.

To demonstrate the effectiveness of the proposed method, the largest and most complete dataset of DNS traffic traces of user machines infected with email worms that had been used to the date of writing this thesis to evaluate a new behaviour-based detection method for email worms was constructed. The experimental results that were produced against this dataset and are presented in Section 4.4 along with the discussion in Section 4.5 support three conclusions. The first of them is that the method distinguishes with remarkable accuracy between non-infected and user machines being infected with any of a large variety of recent email worms. The second conclusion is that the method has the potential to detect user machines that become infected with email worms, including zero-day, in the long run. The reason for this is that it bases detection on differences in the flow-level characteristics of DNS query streams of non-infected and infected user machines that have been apparent over the past several years and are unlikely to disappear in the future. As explained in Section 4.5, controlling these characteristics to evade detection, is a challenging task that requires fundamental changes in the code of email worms. The third conclusion is that the method is very effective when operating on time scales significantly shorter than the typical time needed to generate, disseminate, verify, and install attack signatures, which usually ranges from hours to several days; for example, the antivirus vendors took over seven hours on average to generate an attack signature suitable for the email worm Mydoom.M [153].

The principle of the proposed method is illustrated in Fig. 4.2. In the next three sections, the depicted elements of the method and the results of its experimental evaluation are presented in detail. This is followed by an analysis that demonstrates

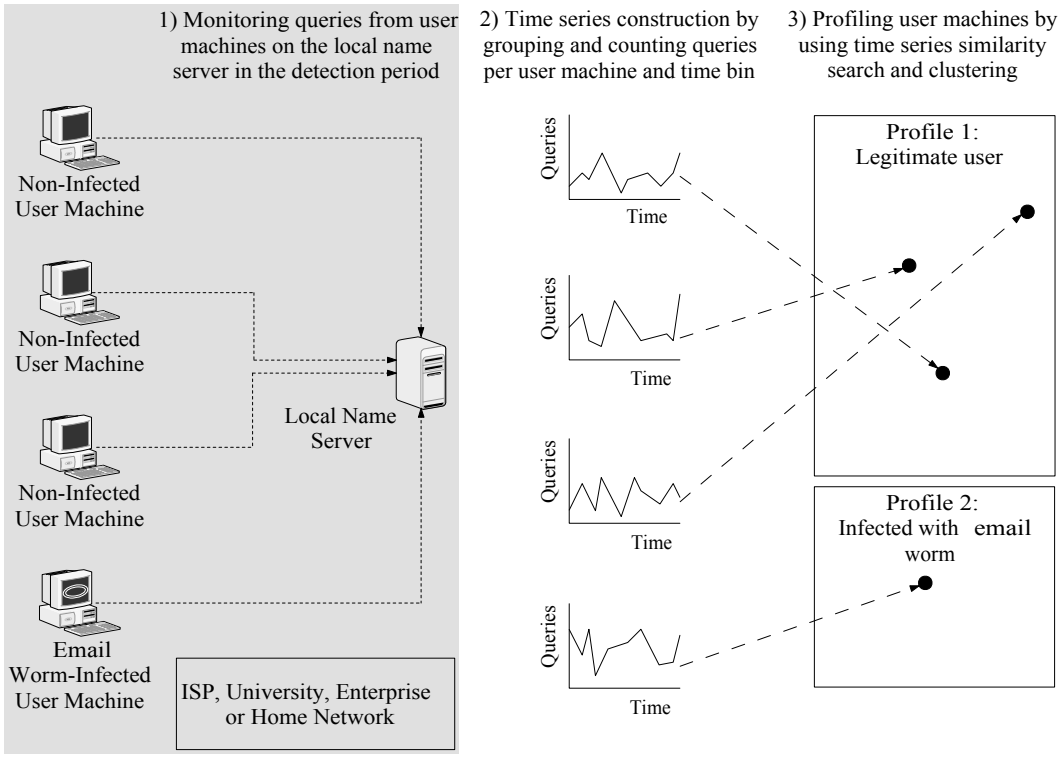

Figure 4.2: The proposed method operates on the DNS queries that a local name server receives from user machines within the detection period (Step 1). It groups these queries per requesting user machine and splits them into time bins of equal duration. For each user machine it counts the queries in every time bin to create a time series (Step 2). Exact shape-based similarity search and clustering are applied to the produced set of time series to assign each requesting user machine into one of two profiles. One profile corresponds to non-infected and the other to user machines that are infected with an email worm (Step 3).

the potential of the method to be effective in the long run. Specifically, in Section 4.2 the necessary background on similarity search over time series is provided, and it is described how the time series of the $n \times p$ time series matrix are transformed and compressed to obtain representations that are used to profile the corresponding user machines. In Section 4.3, it is explained how clustering is applied to these transformed and compressed representations of the time series to assign the user machines into profiles based on the flow-level characteristics of their DNS query streams. In Section 4.4, the experimental results of the method that validate its utility and show that it outperforms methods that base detection on commonly-studied characteristics of Internet traffic signals are discussed. In Section 4.5, several approaches that worm writers might take to evade detection are anticipated and their potential effectiveness is assessed.

4.2 Time Series Similarity Search

Time series constitute a large proportion of the data stored in the databases of various application fields, including medicine, meteorology, astrophysics, geology, multimedia and economics. As a consequence, in recent years, there has been a growing interest in time series data mining. Specifically, a great deal of effort has been devoted to exploring time series with a large number of time points. In connection with this, a topic that has received particular attention is similarity search over time series. The reason for this is that similarity search is useful in its own right as a tool for exploratory data analysis, and it is also an important element of many basic time series data mining tasks, such as indexing, clustering, classification and rule discovery [94]. Despite the large number of studies that have been published on this topic, determining the similarity between the members of a collection of time series continues to be a challenging undertaking. As will become clear in what follows, this is primarily due to the high dimensionality that typically characterises the analysed time series data.

At first glance, it appears reasonable that by using any distance measure, like, for example, the Euclidean distance, one can evaluate the similarity between two time series. The distance between a pair of $p-$length time series can be computed in a straightforward manner if each of them is seen as a point in the $p-$dimensional space, with each time point corresponding to one dimension. Thereby, from a theoretical point of view, performing similarity-based data mining tasks on time series data directly seems to be a promising method for obtaining valuable insight into the data. Nevertheless, in practice, doing so usually yields unreliable results because most data mining algorithms lose their efficiency, effectiveness, or both when applied to time series. This is because, as previously mentioned, the analysed time series usually consist of a large number of time points – thus, are high-dimensional data objects – and the significance of distance measures becomes questionable in high-dimensional spaces [3, 111]. An important implication of this state of affairs for the present study, is that clustering a set of time series using a general-purpose algorithm, produces only marginal results. Furthermore, since very little work has been carried out on devising clustering algorithms that are particularly suited for time series [109], finding meaningful groups in a given set of time series, remains substantially complex and non-obvious.

To attack this problem, the approach that has received the most research attention involves extracting a low-dimensional representation from each time series, and using the representations, instead of the time series, as input to the data mining algorithms. Hereafter, these low-dimensional representations and the function to obtain them from the time series are referred to as feature vectors and feature vector extraction function, respectively. In the last few years, a large number of feature vector extraction functions have been proposed; the interested reader is referred

to Bagnall et al. [10] for a complete review. The operation each of these feature vector extraction functions performs on the time series and the properties of the obtained feature vectors are strongly dependent on the data mining task and the similarity objective that the function was originally intended to serve. Two types of similarity are discussed in the literature: shape-based and structural similarity. Shape-based similarity uses the original raw data to find homomorphic time series, whereas structural similarity describes similarity in the autocorrelation structure [10]. To date, most of the research and practical interest with respect to time series similarity search has been directed at shape-based similarity. Following this direction, the focus in this work is also on shape-based similarity search; thus, the discussion below relates only to shape-based similarity.

Although, as previously pointed out, a large number of feature vector extraction functions exists only a few of them are useful for accurately comparing time series in terms of shape. The reason for this is that extracting feature vectors from the time series inevitably results in information loss. Thereby, the distances between the members of a set of time series necessarily become distorted when the time series are mapped in the feature vector space. Therefore, applying similarity-based data mining algorithms to their feature vectors, can potentially introduce false dismissals. False dismissals occur when the feature vectors of similar time series appear distant in the feature vector space. Depending on whether working with the feature vectors entails false dismissals or not, shape-based similarity search is characterised as approximate or exact, respectively. As explained below, using a feature vector extraction function that guarantees that no false dismissals appear was a crucial requirement for the study presented in this thesis. Therefore, in the remainder of this section, the focus is only on exact shape-based similarity search.

The majority of the pioneering studies on exact shape-based similarity search is linked to the time series data mining task of indexing. Indexing targets finding in a database fast and without false dismissals the most similar time series to a given query time series. To address this issue, Faloutsos et al. [48] introduces the GEneric Multimedia INdexIng (GEMINI), which has significantly influenced the main corpus of recent research on indexing. Given a query time series and a value ϵ, which is referred to as tolerance hereafter, GEMINI is the framework for working with the feature vectors to efficiently and accurately retrieve from the database all the time series that are within distance ϵ from the query time series. The principle of GEMINI is that very dissimilar feature vectors in the feature vector space cannot correspond to time series that are similar in the time series space. Hence, if a feature vector extraction function that does not introduce false dismissals is used to map the time series to feature vectors, a query with tolerance ϵ represents a sphere with radius ϵ in the feature vector space [47]. In connection with this, GEMINI establishes the necessary conditions that a feature vector extraction function should satisfy to be suitable for exact shape-based similarity search.

In more detail, determining whether a feature vector extraction function is suitable for exact shape-based similarity search or not, is according to GEMINI a three-step procedure. In the first step, a distance measure d_{ts} in the time series space needs to selected. In the second step, each one of a given set of time series, or all of them together, is transformed and compressed by means of the feature vector extraction function under consideration, denoted by $f()$ in what follows. Working with the feature vectors, entails no false dismissals if, as third step, it is possible to define a distance measure d_{fv} in the feature vector space that for an arbitrary pair of the time series ts_i, ts_j satisfies:

$$d_{fv}(f(ts_i), f(ts_j)) \leq d_{ts}(ts_i, ts_j) \tag{4.1}$$

In the time series data mining literature, (4.1) is known as the lower bounding lemma. In light of this analysis, the strict definition of a feature vector extraction function that is suitable for exact shape-based similarity search consists of three elements. The first of them is the feature vector extraction function $f()$ itself, for which two options exist. It can be namely either a transformation that extracts the feature vectors from the time series directly or one that generates an intermediate efficiently-compressible time series representation and a corresponding compression method. The terms representation and approximation are used interchangeably to refer to the immediate result obtained by transforming a time series hereafter. The second and third elements are the two distance measures d_{ts} and d_{fv} that, in conjunction with $f()$, satisfy (4.1).

Faloutsos et al. [48] demonstrates that the feature vectors that are extracted by compressing the Discrete Fourier Transform (DFT) of each one of a set of time series satisfy (4.1); both d_{ts} and d_{fv} are the Euclidean distance in that study. By means of GEMINI, many recent studies in the literature propose other feature vector extraction functions and define their corresponding distance measures that satisfy (4.1). In particular, the representations that have been found to be suitable for exact shape-based similarity search (some of them after first being compressed) are the following: the Discrete Wavelet Transform (DWT) [25, 142], the Piecewise Aggregate Approximation (PAA) [93, 205], the Adaptive Piecewise Constant Approximation (APCA) [94], the Piecewise Linear Approximation (PLA) [27], the approximation based on the Chebyshev polynomials of the first kind [22] and the Singular Value Decomposition (SVD) [88, 101]. In Fig. 4.3, the GEMINI framework is illustrated, and the seven feature vector options, which are listed above and presented in detail later in this chapter, are mentioned.

In this work, it is shown that exact shape-based similarity search is a useful tool for distinguishing between normal and anomalous network activity, one that has not been yet adequately explored. To this end, a behaviour-based method that uses exact shape-based similarity search to detect user machines infected with email worms is introduced. The method groups user machines depending on the flow-level

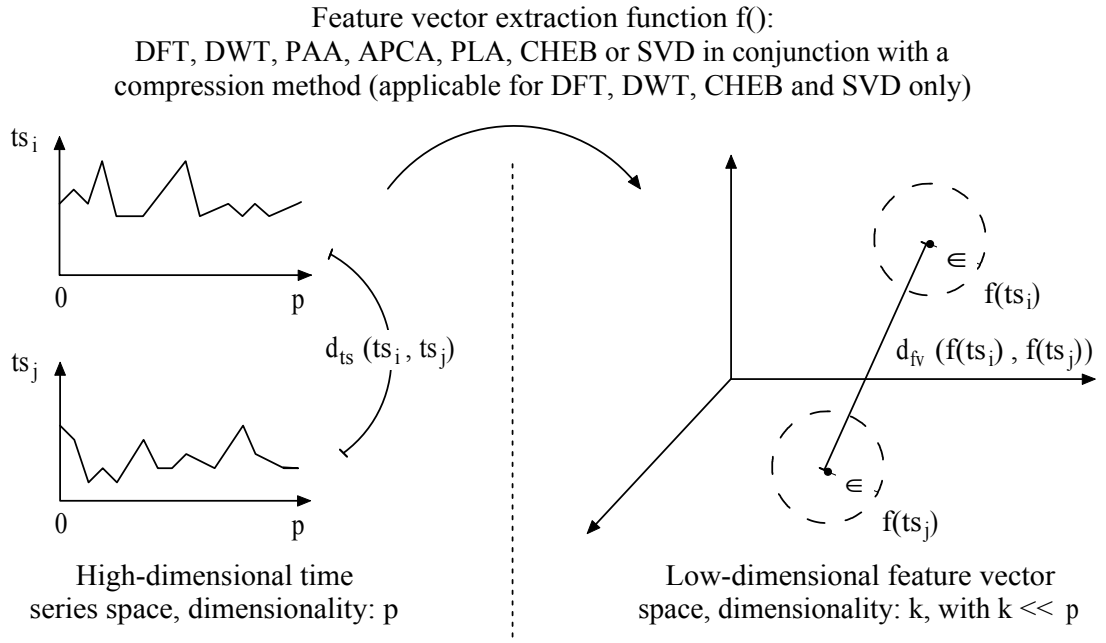

Figure 4.3: The GEMINI is a three-step procedure. In the first step, a distance measure d_{ts} in the p-dimensional time series space has to be selected; p is the number of time points that compose the time series. In the second step, using the feature vector extraction function under consideration $f()$, k-dimensional feature vectors are extracted from the time series, with $k \ll p$ ($k = 3$ in the example shown). In the third step, a distance measure d_{fv} in the feature vector space that for any pair of the time series ts_i, ts_j satisfies the lower bounding lemma has to be defined. Possible options for the feature vectors are: the compressed Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT), approximation based on the Chebyshev polynomials of the first kind (CHEB) and Singular Value Decomposition (SVD) and the Piecewise Aggregate Approximation (PAA), Adaptive Piecewise Constant Approximation (APCA) and Piecewise Linear Approximation (PLA).

characteristics of their DNS query stream. As mentioned in the previous section, the DNS query stream of each one of a set of n user machines is represented by a row in a $n \times p$ time series matrix; p is the number of time points that compose the time series. Thereby, grouping the user machines requires clustering the rows of the time series matrix. As pointed out earlier, applying general-purpose clustering algorithms to a set of time series directly produces low-quality results. This is because clustering, like indexing, is a similarity-based data mining task, and as such it intrinsically relies on a distance measure that quantifies how similar two data objects are. However, as a consequence of the concentration of measure [104], in high-dimensional spaces, such as those typically arising when working with time series, the distance measures become unstable, or meaningless [73].

In light of the discussion above, to address this problem, the time series of the

time series matrix are transformed and compressed to generate a $n \times k$ feature vector matrix, with $k \ll p$. The feature vector matrix is then given (instead of the time series matrix) as input to a general-purpose clustering algorithm. To produce the feature vector matrix, a feature vector extraction function needs to be used. Two time series can be found less or more similar to each other than they really are, if their similarity is calculated by measuring the distance of their corresponding feature vectors. Overestimating the similarity between time series, affects the detection efficacy of the proposed method, and thus it can be tolerated as long as it happens infrequently. By contrast, underestimating the similarity, which basically implies the occurrence of false dismissals, would significantly limit the usefulness of the method. The reason for this is that, if it was permitted, the clustering process, which is detailed in the next section, would discover more DNS query stream profiles than those actually existing. To guarantee that no false dismissals appear, a feature vector extraction function that generates one of the seven approximations listed above has to be used. Thereby, a secondary contribution of this work is that it demonstrates, with a practical example, that the findings of the studies on indexing are also applicable to the context of time series clustering.

Given a set of time series and a data mining task to be accomplished, it is difficult, and often impossible, to determine which feature vector extraction function is the best to use. This is because the functions have intrinsic strengths and weaknesses that depend on the nature of the time series and the data mining task. This has as a result that performing the same data mining task on sets of feature vectors extracted from a set of time series by means of different functions can produce inconsistent results. Apart from this, the relative performance of different data mining tasks carried out on the same set of feature vectors can also vary. Most of the existing studies that propose and evaluate feature vector extraction functions use indexing as basis for experimental validation and comparisons. Furthermore, this work is the first to analyse time series that describe the DNS query streams of Internet-connected machines. Due to this state of affairs, anticipating which feature vector option is best-suited for clustering the time series in hand is not possible. Therefore, within the framework of the present study, all seven options were considered, with the objective to experimentally determine which is the most appropriate for the proposed method. In connection with this, the seven representations previously listed, along with their corresponding distance measures and compression methods, when applicable, are presented in what follows. In addition, it is explained in detail how the $n \times k$ feature vector matrix is derived from the $n \times p$ time series matrix.

4.2.1 Discrete Fourier Transform

The DFT is the first time series representation that was used as basis for extracting feature vectors that are appropriate for exact shape-based similarity search. The DFT

of a $p-$length time series can be written as a superposition of p sine/cosine waves. Each wave can be expressed as a complex number, known as a DFT coefficient. Since complex numbers have a real and an imaginary part, the p DFT coefficients correspond to $2p$ values. Thereby, the DFT of a $p-$length time series is $2p-$dimensional. As a consequence, replacing each time series in the $n \times p$ time series matrix with its DFT, gives a $n \times 2p$ DFT coefficient matrix.

Three methods exist for compressing the $2p-$dimensional DFT obtained from a $p-$length time series to get a $k-$dimensional feature vector, with $k \ll p$. The first method is based on the observation that most of the time series encountered in practice have their energy concentrated in their low-frequency components. Therefore, discarding their high-frequency components, does not introduce much information loss. In connection with this, this method considers only the first $k/2$ DFT coefficients, which capture the low-frequency information of the time series. Rafiei et al. [145] suggests retaining the first and last $k/4$ coefficients instead of the first $k/2$. The principle of this method is that the last $k/4$ coefficients are as important as the first $k/4$. The reason for this is that the DFT of a real-valued time series is symmetric with respect to its middle. Based on Parseval's Theorem, Mörchen [131] proposes to keep the $k/2$ coefficients with the largest absolute values, which, for a given k, are the coefficients that carry most energy.

The compressed representation of a time series that is produced by retaining the first $k/2$ coefficients of its DFT is referred to as DFT-1 hereafter. Similarly, those that result from keeping the largest $k/2$ or the first and last $k/4$ coefficients instead, are referred to as DFT-2 and DFT-3. Agrawal et al. [4], Mörchen [131] and Rafiei et al. [145] demonstrate that, given a set of time series, working with their DFT-1, DFT-2 and DFT-3, respectively, introduces no false dismissals. In these studies, the Euclidean distance is selected for both d_{ts} and d_{fv}. Thereby, it is possible to apply any of the three compression methods discussed above to each row of the $n \times 2p$ DFT coefficient matrix to construct the $n \times k$ feature vector matrix. Although, only one of the DFT-1, DFT-2 and DFT-3 is required for the proposed method, to uncover which of them is the most appropriate, all three are considered later in this chapter in the experimental evaluation section.

4.2.2 Discrete Wavelet Transform

In more recent work, the DWT is used as basis for producing feature vectors that are appropriate for exact shape-based similarity search [25, 200, 86, 142]. In an analogous way to the DFT, the DWT of a time series can be written as a superposition of basis functions. These basis functions, known as wavelets, are generated by dilations and translations of a basic function, known as the mother wavelet. There are several reasons that motivate working with the DWT instead of with the DFT. First, the DWT is more suitable and needs less space than the DFT for representing sparse

spike and non-periodic, non-stationary, or both time series. Second, the time required to compute the DWT of a time series is shorter than that required to compute its DFT. Specifically, using a set of orthonormal wavelets, such as the Haar wavelets, Mallat's pyramid algorithm [116], which is the most well-known and widely-used DWT algorithm, transforms a $p-$length time series in $O(p)$; whereas, the fast Fourier transform algorithm is performed in $O(p\log(p))$ operations. Third, in contrast to the DFT, the DWT is intrinsically multi-resolution, and as such it enables analysing a time series in the time and frequency domains simultaneously. The Mallat algorithm decomposes a $p-$length time series – with p a power-of-two – into p DWT coefficients in $\log_2 p$ levels, where each level corresponds to a frequency band. For the needs of the present study, replacing each row of the $n \times p$ time series matrix with its Haar DWT, generates a $n \times p$ DWT coefficient matrix.

There exist four methods for compressing the $n \times p$ DWT coefficient matrix. Before proceeding with describing these methods, it should be noted that the DWT coefficients are usually normalised to facilitate compression. This involves scaling the coefficients that are computed at the j^{th} decomposition level by a factor of $2^{-j/2}$. Two of the compression methods operate on each DWT in the DWT coefficient matrix independently, and retain the first k and largest k in terms of absolute normalised value coefficients, with $k \ll p$. By contrast, the other two compression methods use as input the entire DWT coefficient matrix. The first of them keeps the k columns of the matrix that have the largest mean element-wise squared value; whereas, the second sets to zero all but the $n \times k$ largest elements of the matrix. One point needs to be highlighted regarding the output of the four compression methods. This is, namely, that keeping the first k coefficients of each DWT or the k columns of the DWT coefficient matrix with the largest mean element-wise squared value, results in a $n \times k$ matrix; by contrast, the other two methods produce a $n \times p$ matrix, with $n \times k$ non-zero elements. Thereby, compared to the former two methods, the latter two require more space and introduce higher bookkeeping overhead when measuring the distances needed to cluster the feature vectors.

The compressed representations of a time series that are produced by retaining the first k and the largest k coefficients of its DWT are referred to as DWT-1 and DWT-2 hereafter. Similarly, those generated by zeroing all the elements of the $n \times p$ DWT coefficient matrix except for those in the k columns with the largest mean element-wise squared value and the $n \times k$ with the largest values are referred to as DWT-3 and DWT-4, respectively. The main contribution of the studies presented in [25, 200, 86, 142] is that they generalise to the DWT the findings regarding the DFT that are reported in [4, 145, 131]. Specifically, these studies show that, given a set of time series, the Euclidean distance between an arbitrary pair of the time series is lower bounded by the Euclidean distance between their corresponding DWT-1, DWT-2, DWT-3 and DWT-4. For the present study, this implies that it is possible to apply any of the four compression methods previously described to the $n \times p$

DWT coefficient matrix to obtain the $n \times k$ feature vector matrix. To find which of the DWT-1, DWT-2, DWT-3 and DWT-4 is the most suitable for the proposed method, experiments using all four of them were conducted and their results are discussed later in this chapter in the experimental evaluation section.

4.2.3 Piecewise Approximations

The PAA, APCA and PLA are piecewise time series approximations that can be used as basis for composing feature vectors that are appropriate for exact shape-based similarity search. Specifically, the PAA, APCA and PLA of a time series are produced by dividing the time series into parts and, then, approximating each part with a line segment. The feature vectors associated with these three approximations are made up by the values of certain properties of the approximation line segments. As a result, the dimensionality (length) of these feature vectors is linked to the number of parts the time series is divided into. Thereby, it can be determined in a straightforward manner; this eliminates the need for a compression method. Within the framework of the present study, replacing each row of the $n \times p$ time series matrix with its $k-$dimensional PAA, APCA or PLA-based feature vector, generates the $n \times k$ feature vector matrix. In the following paragraphs, the procedures to obtain the PAA, APCA and PLA of a time series and the feature vectors associated with them as well as the distance measures that, in conjunction with these feature vectors, satisfy the lower bounding lemma are described in detail.

Keogh et al. [93] and Yi et al. [205] independently present the PAA. Building the PAA of a time series, involves dividing the time series into k equal-length parts and, then, using a constant-value segment to approximate each part. The constant value of each approximation line segment is the mean value of the time points that fall into its corresponding part. The k constant values compose the PAA-based feature vector of the time series. Keogh et al. [24, 94] introduces the APCA that is based on relaxing the requirement of the PAA for equal-length parts. The principle of the APCA is to vary the length of the parts depending on the activity of the time series. Thereby, the APCA of a time series is a set of consecutive constant-value segments of variable length. The lengths of these approximation line segments are chosen in such a way so that the individual reconstruction error of every part is minimal. The constant values and the lengths of the approximation line segments compose the APCA-based feature vector of the time series. Since from each approximation line segment two values are extracted, the time series needs to be approximated with $k/2$ segments to obtain a $k-$dimensional feature vector. Instead of using parts with different lengths, Chen et al. [27] proposes to relax the requirement of the PAA for constant-value segments and introduces the PLA. The PLA of a time series is produced by dividing it into equal-length parts and, then, approximating each part with a best-fit (linear) regression line $y = ax + b$. The coefficients a and b of the

regression lines compose the PLA-based feature vector of the time series. Similarly to the APCA, a PLA with $k/2$ regression line segments is required to construct a $k-$dimensional feature vector. This is because two features are derived from each regression (approximation) line segment: its slope a and its $y-$intercept b.

Given two $p-$length time series, denoted by ts_i and ts_j in what follows, let c_n^i and c_n^j be the features of their $k-$dimensional PAA and APCA-based feature vectors that correspond to the constant value of the n^{th} approximation line segment. Further, let a_n^i, b_n^i and a_n^j, b_n^j be the features of their $k-$dimensional PLA-based feature vector that correspond to the slope and $y-$intercept of the n^{th} approximation line segment. It is worth noting that, for a given value k, n is an integer in the interval $[1, k]$ for the PAA-based feature vector, whereas it is an integer in $\left[1, \frac{k}{2}\right]$ for the APCA and PLA-based feature vectors. Keogh et al. [93], Keogh et al. [24, 94] and Chen et al. [27] show respectively that the distance measures d_{paa} (4.2), d_{apca} (4.3) and d_{pla} (4.4), which measure the similarity between the PAA, APCA and PLA-based feature vectors of ts_i and ts_j lower bound the Euclidean distance between ts_i and ts_j; re_n is the endpoint of the n^{th} approximation line segment of the APCA of ts_i. With regard to these distance measures, two points need to be highlighted. The first of them is that, although Yi et al. [205] demonstrates that the d_{paa}, which is essentially a weighted version of the Euclidean distance (L_2 norm), is generalisable to any L_p norm, the d_{paa} was used for the purposes of the present study. The second point is that computing the d_{apca} requires that the approximation line segments of the APCA of ts_i and ts_j have the same start and endpoints. In connection with this, to work with this distance measure, a special version of the APCA of ts_j, which is obtained by projecting the ts_j onto the APCA of ts_i, has to be produced.

$$d_{paa} = \sqrt{\frac{p}{k}\sum_{l=1}^{k}(c_l^i - c_l^j)^2} \tag{4.2}$$

$$d_{apca} = \sqrt{\sum_{l=1}^{k/2}(re_{l-1} - re_l)(c_l^i - c_l^j)^2} \tag{4.3}$$

$$d_{pla} = \sqrt{\sum_{l=1}^{k/2}\sum_{m=1}^{2p/k}((a_l^i - a_l^j)m + (b_l^i - b_l^j))^2} \tag{4.4}$$

4.2.4 Chebyshev Approximation

Cai et al. [22] presents a procedure for composing feature vectors that involves approximating time series with Chebyshev polynomials, and demonstrates that the obtained feature vectors are appropriate for exact shape-based similarity search. The principle of this procedure is that every time series can be extended to a function

defined on any closed interval $[a, b]$, and as such it can be approximated by a low-degree polynomial. The best choice for this polynomial is the one that, among all the polynomials of the same degree, yields the smallest maximum deviation from the function under consideration. However, the polynomial that has this property, which is known as the minimax polynomial, is usually very difficult to compute. By contrast, the approximations that are based on expanding the function in a series of Chebyshev polynomials of the first kind are very close to that obtained by using the minimax polynomial and are easy to compute [121]. In the next paragraphs, a brief introduction to the approximation of a function by means of Chebyshev polynomials of the first kind is first given, followed by a detailed discussion of how it can be used in the context of exact shape-based similarity search.

The Chebyshev polynomials of the first kind constitute a sequence of orthogonal polynomials that are defined on the closed interval $[-1, 1]$ and denoted by $T_p(t)$. They are the solutions to the Chebyshev differential equation, and they can be obtained from the trigonometric identity (4.5) or the recursive relation (4.6). An implication of their orthogonality – with respect to the weight function $w(t) = (1 - t^2)^{-1/2}$ – is that they form a complete basis set. Thereby, it is possible to express any function $g(t)$ defined on the closed interval $[-1, 1]$ as an infinite linear combination of the $T_p(t)$, as shown in the middle part of (4.7). In addition, given a value p, the partial sum in the right part of (4.7) represents a very good approximation of $g(t)$, which is known as the Chebyshev approximation. The coefficients c_i in (4.7) are called Chebyshev coefficients in what follows, and they can be determined through (4.8). In (4.8), which derives from the Chebyshev-Gauss quadrature [2], t_j denotes the p roots of the $T_p(t)$ that can be computed through (4.9).

$$T_p(t) = \cos(p \cos^{-1}(t)) \qquad t \in [-1, 1] \tag{4.5}$$

$$T_p(t) = \begin{cases} 0 & \text{if } p = 0, \\ t & \text{if } p = 1, \\ 2tT_{p-1}(t) - T_{p-2}(t) & \text{if } p \geq 2. \end{cases} \tag{4.6}$$

$$g(t) = \sum_{i=0}^{\infty} c_i T_i(t) \approx \sum_{i=0}^{p} c_i T_i(t) \tag{4.7}$$

$$c_i = \begin{cases} \frac{1}{p} \sum_{j=1}^{p} g(t_j) & \text{if } i = 0, \\ \frac{2}{p} \sum_{j=1}^{p} g(t_j) T_i(t_j) & \text{if } 1 \leq i \leq p. \end{cases} \tag{4.8}$$

$$t_j = \cos \frac{(j - 0.5)\pi}{p} \qquad \text{with } 1 \leq j \leq p \tag{4.9}$$

The procedure presented in Cai et al. [22] for extracting $k-$dimensional feature vectors, which are appropriate for exact shape-based similarity search, given a set of $p-$length time series, with $k \ll p$, consists of three steps. In the first step, every time series is normalised so that its time points fall into the interval $[-1, 1]$. In the second step, every normalised time series is extended to a weighted step function. In the third step, the Chebyshev approximation with p terms of every weighted step function is defined and the values of the corresponding Chebyshev coefficients are computed. These coefficients constitute the Chebyshev-based representation of each time series, and the first k of them compose the $k-$dimensional feature vector of the time series. This compressed ($k-$dimensional) Chebyshev-based representation is referred to as CHEB hereafter. Applying GEMINI, Cai et al. shows that the Euclidean distance between an arbitrary pair of the time series ts_i, ts_i is lower bounded by the weighted Euclidean distance between their CHEB d_{cheb} (4.10); cc_n^i and cc_n^j denote the n^{th} coefficient of the CHEB of ts_i and ts_j, respectively. For the needs of the present study, replacing each row of the $n \times p$ time series matrix with its $k-$dimensional CHEB, generates the $n \times k$ feature vector matrix.

$$d_{cheb} = \sqrt{\frac{\pi}{2} \sum_{l=1}^{k} (cc_l^i - cc_l^j)^2} \tag{4.10}$$

4.2.5 Singular Value Decomposition

Korn et al. [101] and Kanth et al. [88] use the SVD in the context of exact shape-based similarity search. In contrast to the transformations presented in the previous subsections, the SVD is data-dependent and applicable to a set of time series instead of to a single one. Specifically, given a set of n $p-$length time series, the SVD rotates the axes of the $p-$dimensional time series space so that the variance along the first few dimensions is maximised. The formal definition of the SVD is given in (4.11), A denotes a $n \times p$ time series matrix, U and V two orthogonal matrices with dimensions $n \times r$ and $p \times r$, respectively, and Σ the $r \times r$ diagonal matrix with elements the singular values of matrix A in descending order; r is the rank of matrix A. The matrices V and $U\Sigma$ represent the transform matrix and the transformed data. Within the framework of the present study, performing the SVD on the $n \times p$ time series matrix, produces a $n \times r$ matrix, with r at most $\min(n, p)$.

$$A = U\Sigma V^T \tag{4.11}$$

Obtaining a compressed version of the transformed data with $n \times k$ dimensions, with $k \ll r \leq p$, is possible in a straightforward way. Namely, it involves zeroing the last $r - k$ columns of matrix U and discarding those of matrix Σ before computing their matrix product. With regard to matrix Σ, this implies retaining only its columns

that contain the largest k singular values of matrix A. The resulting matrices are denoted by U_k and Σ_k, and their matrix product $U_k\Sigma_k$ is a $n \times k$ matrix. Korn et al. [101] and Kanth et al. [88] show that the Euclidean distance between any two rows of the matrix $U_k\Sigma_k$ lower bounds the Euclidean distance between their corresponding time series. Thereby, given the $n \times p$ time series matrix and a value k, finding the matrix product $U_k\Sigma_k$, results in the $n \times k$ feature vector matrix.

4.3 Feature Vectors Clustering

By means of a similarity measure, clustering organises a given set of data objects into groups, called clusters, such that the average similarity between data objects belonging to the same cluster is maximised and the average similarity between data objects of different clusters is minimised. The objective of clustering is to determine the partition of the data objects that best resembles their intrinsic grouping; this partition is referred to as optimal partition hereafter. In practice, however, clustering a set of data objects, only rarely produces partitions that are close to its optimal partition. This mainly happens due to reasons that relate to the nature of the data objects in hand. Such reasons are, for example, that the data objects may not form natural groups at all or that the similarity measure used may be inappropriate for capturing the semantic similarity of the data objects. To ensure that clustering a set of data objects, will indeed provide insight into its underlying structure, the clustering process should involve the following four steps that need to be carefully planned and executed: feature vector selection (or extraction), algorithm selection, validation of the results and interpretation of the results [63].

In this work, clustering is used to arrange a set of n user machines into groups, such that each group consists of user machines that generate DNS query streams having similar flow-level characteristics. This essentially requires clustering the rows of the $n \times p$ time series matrix that represent the DNS query streams of the user machines. In the previous section, the reasons that make clustering time series directly, result in meaningless groups and the procedure that should be followed for producing a $n \times k$ feature vector matrix, with $k \ll p$, whose rows should be clustered instead are discussed. This implies that the analysis regarding the first of the four above-listed steps of the clustering process is provided earlier in this chapter. In addition, the interpretation of the clustering results is covered in detail later in this chapter in the experimental evaluation section. In connection with this, the remainder of this section is devoted to discussing the rationale for determining which is the most appropriate clustering algorithm for the purposes of the present study and issues relevant to the validation of the clustering results.

Before looking at these two steps of the clustering process in the subsections that follow, two important points are mentioned hereafter. The first point concerns the fundamental purpose behind clustering the feature vectors that make up the

feature vector matrix and the second the expected result of the clustering process. Specifically, in this work, clustering is used to automatically infer knowledge about the existence, number and nature of intrinsic groups in the analysed feature vectors, and as a consequence in their corresponding time series. In more detail, it is expected that the feature vectors fall into two compact and well-separated clusters and that these clusters are pure in the sense that one comprises only feature vectors of non-infected the other only feature vectors of infected user machines. The implication of this result is that it is feasible to reliably distinguish between non-infected and infected user machines by analysing their DNS query streams at the flow level. This makes it possible to automatically detect user machines that are infected with email worms on the local name server of a monitored network.

4.3.1 Algorithm Selection

Deciding which clustering algorithm to use to group the feature vectors, involves choosing a distance measure and a clustering criterion. The former quantifies the similarity between the feature vectors and the latter assesses the quality of each possible partition. The clustering criteria depend on the values of several input parameters that are used internally in the clustering process to define the optimal partition of the feature vectors. Based on the meaning of these parameters, the clustering algorithms fall into five categories: partitional, hierarchical, density-based, grid-based and model-based [65]. Within the framework of the present study, it was straightforward to determine the most appropriate distance measure and clustering criterion, and thereby to which of the categories listed above the algorithm to be selected should belong. Specifically, with regard to the distance measure, there is only one option for each feature vector type. This is, namely, its corresponding distance measure that lower bounds the Euclidean distance between the time series. Further, in contrast to the algorithms in the other categories, which rely on manually setting the input parameters, based on a priori knowledge about the feature vectors, the hierarchical clustering algorithms determine the required values automatically by examining the feature vectors. As in this work, no a priori knowledge about the feature vectors in the feature vector matrix is assumed (as such knowledge is not available), a hierarchical clustering algorithm is used.

As their name suggests, the hierarchical clustering algorithms generate a hierarchy of partitions, where each partition consists of a different number of clusters. The hierarchy of partitions facilitates the exploration of the feature vectors at various levels of granularity. Thereby, it uncovers more information about the analysed feature vectors than the flat (non-hierarchical) set of clusters that the algorithms in the other categories produce. However, constructing the hierarchy of partitions comes at the expense of more computation; as a result, most of the hierarchical clustering algorithms are computationally very demanding. Based on the order at which they

build the hierarchy, the hierarchical clustering algorithms are further categorised into agglomerative and divisive. The agglomerative algorithms are bottom-up. They start namely with each feature vector in its own cluster and recursively merge the most similar clusters into one cluster. By contrast, the divisive algorithms are top-down. This means that they start with all the feature vectors in one cluster and subdivide them into smaller clusters until each cluster contains only one feature vector. An advantage of the divisive over the agglomerative algorithms occurs when the interest is in searching for large clusters or a small number of clusters [68]. Although no a priori knowledge about the size and number of the clusters of the feature vectors that compose the feature vector matrix is available, the viability of the proposed method depends on easily separating the feature vectors of non-infected from those of infected user machines. Thereby, the method essentially searches for a small number of clusters; thus, a divisive algorithm is used.

As previously mentioned, most of the hierarchical clustering algorithms are computationally very expensive. In particular, the divisive algorithms consider at their first iteration all the possible partitions of the entire set of feature vectors into two non-empty clusters. There exist $2^{n-1} - 1 = O(2^n)$ possibilities for dividing a set with n feature vectors into two clusters; this renders these algorithms intractable for many practical sets of data objects. However, the DIvisive ANAlysis (DIANA) [92] employs a splitting heuristic introduced by MacNaughton-Smith [113] in conjunction with the distance measure between a feature vector and a cluster in (4.12) to limit the number of possible partitions that have to be examined. In (4.12), fv_i and fv_j denote two feature vectors, $d()$ a distance measure on the feature vector space and $|C_k|$ the number of feature vectors in cluster C_k. The DIANA has $O(n^2)$ time complexity instead of $O(2^n)$. Despite the fact that given its quadratic time dependency on the number of feature vectors, the DIANA scales poorly as the number of feature vectors increases, it is the clustering algorithm used in this work. This is because it is the only divisive algorithm generally available, as it is implemented in many statistical analysis packages, and it constitutes the most commonly-used divisive algorithm. Furthermore, within the framework of the present study, it clusters the feature vectors that compose the feature vector matrix in computing time in the order of milliseconds on a typical computer.

$$d(fv_i, C_k) = \begin{cases} \frac{1}{|C_k|-1} \sum\limits_{\substack{fv_j \in C_k, \\ j \neq i}} d(fv_i, fv_j) & \text{if } fv_i \in C_k, \\ \frac{1}{|C_k|} \sum\limits_{fv_j \in C_k} d(fv_i, fv_j) & \text{if } fv_i \notin C_k. \end{cases} \tag{4.12}$$

4.3.2 Results Validation

Since clustering finds groups of feature vectors that are not known a priori, regardless of the clustering algorithm applied to the feature vectors, the clustering results need some kind of validation. In particular, the hierarchical clustering algorithms require an a posteriori decision with respect to the partition that best satisfies or reproduces the underlying structure of the feature vectors. To put it differently, inferring valuable knowledge from the hierarchy of partitions that these algorithms produce, involves using a criterion that can determine the optimal number of clusters. Over the past years, a large number of criteria for specifying the hierarchical level on which to base inferences concerning the true differences between feature vectors have been proposed [64, 126]. These criteria, formally referred to as stopping rules, evaluate for a given hierarchical clustering algorithm and set of feature vectors the partition of each hierarchical level by comparing it to the partitions of every other level. Although significant effort has been devoted to assessing stopping rules, there exist no general gold standards that are capable of revealing the optimal number of clusters over sets of data objects from diverse application fields. Hence, in this work, four well-known and widely-used stopping rules are employed in a complementary way: the Connectivity [66], the Dunn [41], the Silhouette Width [92] and the Davies-Bouldin [39] indices. In the remainder of this section, these stopping rules are briefly presented and their principle is described.

The Connectivity index (CI) measures the degree to which neighbouring feature vectors are placed in the same cluster, and is defined as the average connectedness (C) over the clusters of a partition. The C of a cluster is computed by choosing one of its feature vectors fv_i and examining a predefined number l of its nearest neighbours. If the k^{th} nearest neighbour of fv_i does not belong to the same cluster as fv_i, the value of C is increased by $1/k$. The CI of a partition with n clusters is calculated through (4.13). Thereby, it takes values in the interval $[0, +\infty)$ and smaller values thereof indicate better partitions. The Dunn index (DI) evaluates how compact and well-separated the clusters of a partition are, and it is computed through (4.14); $d(C_i, C_j)$ and $d(C_i)$ denote the distance between any two clusters C_i, C_j of the partition and the diameter of C_i, respectively. The former is the smallest distance between a feature vector in C_i and one in C_j and the latter the largest distance between two feature vectors in C_i. If the set of feature vectors contains compact and well-separated groups, the distances between the clusters are expected to be large and their diameters small. Thereby, the DI takes values in the interval $[0, +\infty)$ and larger values thereof indicate better partitions. The Silhouette Width (SW) is defined as the average silhouette (S) over the clusters of a partition. The S of a cluster expresses the confidence in placing feature vectors in this particular cluster instead of in a neighbouring one. Let $\bar{d}_i^o$ and $\bar{d}_i^c$ be the average distances between a feature vector fv_i in C_i and the rest feature vectors in C_i and those in the closest neighbour cluster to C_i, respectively. Then, the SW of a partition with

n clusters is determined through (4.15). The Davies-Bouldin index (DB) measures the average similarity between each cluster of a partition and its closest neighbour cluster. It is desirable that the clusters are compact and have centres that are distant from each other. In connection with this, the partition that minimises the DB is the best. With the same notations as in the definition of the DI, the DB of a partition with n clusters is calculated through (4.16).

$$\begin{aligned} CI &= \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} \sum_{k=1}^{l} c_{ik}, \\ c_{ik} &= \begin{cases} \frac{1}{k} & \text{if } fv_i, fv_k \text{ in diff. clusters,} \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4.13}$$

$$DI = \min_{i=1,\ldots,n} \left(\min_{j=i+1,\ldots,n} \frac{d(C_i, C_j)}{\max\limits_{k=1,\ldots,n} (d(C_k))} \right) \tag{4.14}$$

$$SW = \sum_{i=1}^{n} S_i = \frac{1}{n} \sum_{i=1}^{n} \frac{\bar{d}_i^c - \bar{d}_i^o}{\max(\bar{d}_i^c, \bar{d}_i^n)} \tag{4.15}$$

$$DB = \frac{1}{n} \sum_{i=1}^{n} \max_{i \neq j} \left(\frac{d(C_i) + d(C_j)}{d(C_i, C_j)} \right) \tag{4.16}$$

4.4 Experimental Evaluation

In this section, the results of the experimental evaluation of the proposed method are presented in two parts. In the first part, results of experiments under the assumption that only one user machine in the monitored network is infected within the detection period are reported. By contrast, the results in the second part concern experiments with many user machines becoming infected at random times within the detection period. The purpose of the first part is threefold. First, it validates using many recent email worms and each one of the time series representations discussed in Section 4.2 independently that, based on the flow-level characteristics of their DNS query streams, the user machines fall into two canonical profiles. Second, it shows that one of these two profiles corresponds to non-infected and the other to infected user machines. Third, it indicates which time series representations make the method perform best in identifying the two profiles and, thereby, in distinguishing between non-infected and infected user machines. This allows reducing the number of choices regarding the time series representations and compression methods involved and focus only on the most appropriate for the method. Based on the findings presented in the first part, the goal of the second part is twofold. First, it reveals which of the time series representations that make the method perform best in the tests of

the first part optimise the efficacy of the method in detecting many infected user machines. Second, it demonstrates that the proposed method outperforms methods that base detection on the volume or self-similarity of the DNS traffic that user machines generate. Apart from the points mentioned above, in both parts, remarks regarding the computational complexity introduced in the method when working with each of the time series representations are pointed out. Before proceeding further, it should be noted that some of the experimental results presented in what follows have been published in [PP1, PP3, PP4, PP5, PP6].

4.4.1 Experimental Setup

To produce the experimental results presented in both parts of this section, the proposed method was tested against 71 out of a total of 164 email worms that were reported between April 2004 and July 2007 in the monthly-updated top-threat lists of Viruslist [91] and Virus Radar [181]. The source code, executable, or both of these email worms were obtained from various public sources (for instance, [137, 184]). Instead of infecting Internet-connected machines, an isolated computer cluster was set up and the email worms were launched in it. To ensure that the email worms run on the isolated computers as if they had infected Internet-connected machines, two actions were taken. First, a number of them was analysed by means of reverse code engineering to find whether their operations depend on the proportion of infectious emails delivered to the remote email servers or that of DNS queries successfully resolved. This analysis uncovered that no such dependencies exist. Hence, the second action, which was to copy the file system of a user machine in the research institute's Fraunhofer FOKUS [50] network to the isolated computers, was enough to guarantee typical email worm activity in the isolated cluster. By recording over a period of eight hours the DNS queries of the infected computers, the largest and most complete dataset of DNS traffic traces of user machines infected with email worms that had been used to the date of writing this thesis to evaluate a new behaviour-based detection method for email worms was constructed.

To assess the detection efficacy of the proposed method, DNS traffic traces of both non-infected and infected user machines were required. Because no end users were working on the computers of the isolated cluster, the DNS query streams that were captured at the local name server of the research institute Fraunhofer FOKUS were merged with those of the infected computers. The way that the DNS query streams of the non-infected and infected user machines were merged together is described later in this section. In the remainder of this paragraph, the focus is solely on providing details about the DNS query streams of the non-infected user machines. In particular, three DNS query log file fragments were considered to produce the experimental results discussed in the following subsections. The log file fragments were written on 27 through 28 March 2006, 30 September through 2 October 2006

and on 29 through 31 January 2007 and comprise 631 875, 3 702 926 and 4 183 311 DNS queries, respectively. Examining these log file fragments, revealed that, depending on the time of the day, the local name server of the monitored network was receiving in each one-hour interval DNS queries from 350 to 500 machines. To take into account only the DNS queries originating from user machines, the servers of the network that were frequently querying the local name server were identified and their queries were not included in the analysis.

For the evaluation of the proposed method, the duration of the detection period was adjusted to one hour. Thereby, the DNS query streams of the non-infected and infected machines were split into one-hour blocks. Following the procedure described in Section 4.1, a set of 256−length time series with time bins corresponding to 15−second intervals was constructed ($256 \times 15\text{sec} \approx 1\text{hour}$) from each block. For simplicity, the time series derived from the DNS query streams of non-infected and infected machines are referred to as legitimate users' and email worm time series, respectively. Each set of legitimate users' time series was organised into a $n \times 256$ matrix, referred to as legitimate users' time series matrix hereafter; n is the number of user machines that queried the local name server of the monitored network within the detection period. To produce the results presented in this section, ten legitimate users' time series matrices representing the DNS query streams of non-infected user machines at different times of the day were considered. The process to obtain from these ten matrices the corresponding time series matrices that were used as input to the method differed for the two experimental parts. In the first part (Section 4.4.2), one randomly-chosen row of each legitimate users' time series matrix was replaced with an email worm time series. In the second part (Section 4.4.3), m rows of each legitimate users' time series matrix were merged with an email worm time series starting from the l^{th} time bin; m and l were taken from two random variables uniformly distributed over the integers in the intervals $\left[1, \frac{5\times n}{100}\right]$ and $\left[1, \frac{6.25\times 256}{100}\right]$, respectively. In both parts of this section, the detection efficacy of the method using four different values for the feature vector length k – 4, 8, 16 and 32 – is discussed. For the reader's convenience, all the parameters mentioned above together with their values are given in Table 4.1.

In both parts, the false negative and positive rates are used to demonstrate and evaluate the detection efficacy of the proposed method. In the context of this work, the false negative rate represents the proportion of the infected user machines that the method misclassifies as non-infected. Likewise, the false positive rate expresses the proportion of the non-infected user machines that the method erroneously identifies as infected. The false positive and negative rates take values in the interval $[0, 1]$, and larger values thereof indicate better detection efficacy. It is important to note, however, that in each experimental part the rates are used to assess a different aspect of the efficacy of the proposed method. Specifically, in the first part, the two rates quantify the efficacy of the method in identifying one user machine that becomes

Table 4.1: The parameters and their values that were used to produce the experimental results of the proposed detection method, which are presented in the rest of this chapter.

Parameter	Value
# of DNS query log fragments	3
# of user machines (n)	$\{350, \ldots, 500\}$
Duration of the detection period	1 h
Duration of time bins	15 s
Time series length	256
# of time series matrices	10
# of email worm-infected machines (m)	$\left\{1, \ldots, \frac{5 \times n}{100}\right\}$
# of email worms	71
Time bin the infection occurs (l)	$\left\{1, \ldots, \frac{6.25 \times 256}{100}\right\}$
Feature vector length (k)	$\{4, 8, 16, 32\}$

infected with various email worms in successive detection periods. In connection with this, the rates essentially uncover whether the method is capable of detecting many different email worms, and are referred to as overall false negative and positive rates hereafter. In the second part, the rates quantify the efficacy of the method in detecting many user machines that become infected with the same email worm within a detection period. In this case, the rates reveal if the method has the potential to detect many different email worm instances, and are referred to as per worm false negative and positive rates in what follows.

4.4.2 Overall Efficacy

To produce the experimental results discussed in this part, a four-step procedure was carried out on each of the ten $n \times 256$ legitimate users' time series matrices. First, one row of each matrix was randomly selected and replaced with an email worm time series to make the $n \times 256$ time series matrix. Second, following the steps described in Sections 4.2.1 to 4.2.5, each time series matrix was transformed and compressed to create its corresponding DFT, DWT, PAA, APCA, PLA, CHEB and SVD-based $n \times k$ feature vector matrices. For the time series representations for which more than one compression methods are available, each compression method was applied independently. Third, the rows of each feature vector matrix were clustered by means of the DIANA to generate the hierarchy of partitions and only the first nine levels thereof (two to ten-cluster schemes) were retained. Fourth, the partitions of the nine levels were given as input to the four stopping rules that indicated which of them is the optimal partition of the feature vectors. This four-step procedure was executed

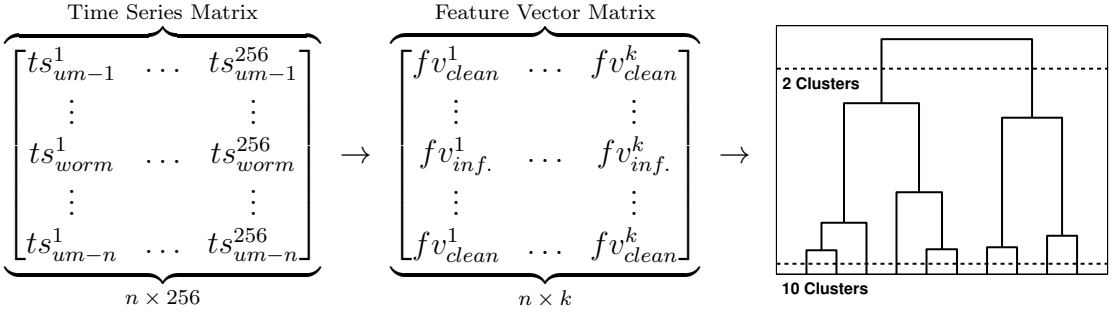

Figure 4.4: One randomly-chosen row of each $n \times 256$ legitimate users' time series matrix is replaced with an email worm time series to produce the time series matrix. This matrix is transformed and compressed to obtain the $n \times k$ DFT, DWT, PAA, APCA, PLA, CHEB and SVD-based feature vector matrices, with k equal to 4, 8, 16 and 32. The rows of these matrices are clustered using the DIANA. The partitions that correspond to the first nine levels of the generated hierarchy of partitions are given as input to the four stopping rules.

71 times, one for each email worm (time series) under consideration. As a result, in total, the experimental procedure was repeated for each email worm: 10 legitimate users' time series matrices $\times$ 11 time series representation and compression method combinations $\times$ 4 feature vector lengths = 440 times. The experimental procedure of this part is illustrated in Fig. 4.4.

The presentation of the experimental results is organised into two subparts. In the first subpart, the output of the four stopping rules is discussed, and it is shown that, regardless of the email worm time series and the time series representation and compression method combination used, the two-cluster scheme is the optimal partition of the feature vectors. The objective of this subpart is to validate the hypothesis that based on the flow-level characteristics of their DNS query streams, the user machines in the monitored network fall into two profiles. In the second subpart, these two profiles are examined by means of the overall false positive and negative rates, and it is pointed out that one of them corresponds to non-infected and the other to infected user machines. The purpose of this subpart is to reveal that the proposed method is remarkably effective in detecting user machines infected with any of a variety of email worms.

The validity of the hypothesis that user machines fall into two profiles, based on the flow-level characteristics of their DNS query streams, was tested for every time series representation and value k combination. For a given combination, the hypothesis holds true if for all or most of its 710 corresponding feature vector matrices (10 legitimate users' time series matrices $\times$ 71 email worms = 710 time series matrices), the optimal partition of the feature vectors consists of two clusters (two-cluster scheme). To find out for which combinations this applies, the first nine levels of the hierarchy of partitions produced by the DIANA at each iteration of

the experimental procedure were given as input to the four stopping rules. The experimental results presented below relate to the output of the stopping rules and serve two purposes. First, they indicate for which time series representations the tested hypothesis was validated. Second, they make it possible to compare these time series representations in terms of how the feature vector length affects the output of the stopping rules. For the purposes of the present study, the best time series representation should make the stopping rules indicate that the two-cluster scheme is the optimal partition using as input the shortest feature vectors ($k = 4$).

The experimental results are summarised in Table 4.2. In the table and the remainder of this chapter, the notations introduced in Section 4.2 for the time series representations are employed. Specifically, the DFT-1, DFT-2 and DFT-3 refer to working with the first $k/2$, largest $k/2$ and first and last $k/4$ coefficients of the DFT of each time series. Further, the DWT-1 and DWT-2 correspond to keeping the first k and largest k coefficients of the DWT of each time series. The DWT-3 and DWT-4 concern retaining from the $n \times p$ DWT coefficient matrix its k columns with the largest mean element-wise squared value and its largest $n \times k$ elements, respectively. Table 4.2 lists for every time series representation and value k combination, the mean (μ) and the standard deviation (σ) over the feature vector matrices of the percentage of the 71 email worms for which the stopping rules indicate that the two-cluster scheme is the optimal partition of the feature vectors. For example, the 100 value in the first top left cell of the table is to be interpreted as follows: the CI indicates that on average over the 710 time series matrices the two-cluster scheme is the optimal partition of the feature vectors in their corresponding feature vector matrices for all ($\mu = 100$, $\sigma = 0$) the email worms under consideration when the first two ($k = 4$) DFT coefficients of the time series compose the feature vectors.

The mean values in Table 4.2 show that the stopping rules validate the tested hypothesis for every time series representation and value k combination, since they uncover a clear dominance of the two-cluster scheme over any other partition. Notably, the CI indicates that the two-cluster scheme is the optimal partition of the feature vectors in all the 31 240 feature vector matrices considered in this part (440 iterations $\times$ 71 email worms = 31 240 matrices). In more detail, working with the DWT-2, DWT-4, PLA, CHEB or SVD and $k = 4$, results in all four stopping rules indicating the two-cluster scheme as the optimal partition for more than 95% of the email worms. Employing the DWT-3 or one of the DWT-1, PAA and APCA, achieves this when $k \geq 8$ and $k \geq 16$, respectively; whereas, employing the DFT-1, DFT-2 or DFT-3, does not yield this result even when $k = 32$. Hence, the DFT-based representations are excluded from the analysis presented hereafter. The standard deviation values in Table 4.2 reveal that clustering the PLA or DWT-4, returns more stable and predictable results than clustering the other time series representations. This is because they suggest that compared to using the other time series representations, using the PLA or DWT-4, makes the output of the stopping rules vary for less than

Table 4.2: The mean (μ) and standard deviation (σ) over 710 feature vector matrices percentage of the email worms for which the Connectivity index (CI), Dunn index (DI), Silhouette Width (SW) and Davies-Bouldin index (DB) indicate that the two-cluster scheme is the optimal partition of the feature vectors, for every time series representation and value k combination. All four stopping rules indicate an absolute dominance of the two-cluster scheme for every combination; thereby, they validate the hypothesis under test.

		$k = 4$				$k = 8$				$k = 16$				$k = 32$			
		CI	DI	SW	DB	CI	DI	SW	DB	CI	DI	SW	DB	CI	DI	SW	DB
DFT-1	μ	100	91.7	97.3	90.8	100	94.1	97.5	91.5	100	94.9	97.9	92.2	100	95.1	98.2	92.9
	σ	0	16	3.5	15.7	0	11.8	3.2	14.6	0	9.8	2.9	13.5	0	9.7	2.7	12.9
DFT-2	μ	100	94.2	97.6	90.8	100	94.9	98.4	92.9	100	95.1	98.7	93.7	100	95.3	98.8	94.1
	σ	0	4.2	2.3	14.2	0	8.4	2.2	12.8	0	8.9	1.9	11.9	0	9.1	1.8	11.4
DFT-3	μ	100	85.5	85.5	74.4	100	89	89.1	79.2	100	91.1	90.5	82.7	100	92.2	91.3	85.5
	σ	0	13.2	13.4	11.5	0	10.4	11	10.7	0	9	9.4	10.3	0	8.2	9	10.3
DWT-1	μ	100	93.2	97.5	92.4	100	94	98.5	93.3	100	95.6	98.9	94.1	100	95.8	99.1	94.6
	σ	0	13.6	3	13.4	0	12.2	2.4	12	0	10.2	2.1	11.3	0	9.9	1.9	10.8
DWT-2	μ	100	98	98.6	95.4	100	98.8	99.1	97.5	100	99.1	99.2	97.3	100	99	99.1	97.4
	σ	0	2.7	2.6	10.4	0	2.1	1.9	7.5	0	1.8	1.7	7.5	0	1.9	1.8	6.7
DWT-3	μ	100	97	98	94.9	100	96.7	98.5	95.4	100	96.5	98.3	95.4	100	96.5	98.3	95.4
	σ	0	6	1.9	9.9	0	7.8	1.9	10.2	0	8.3	2.2	9.8	0	8.2	2.1	9.4
DWT-4	μ	100	97.9	98.9	97.6	100	98	99.2	97.7	100	98	99.2	97.8	100	98.1	99.3	97.9
	σ	0	3.8	2.3	3.8	0	3.7	1.7	3.7	0	3.7	1.5	3.7	0	3.5	1.3	3.6
PAA	μ	100	93.2	97.5	92.4	100	94	98.5	93.3	100	95.6	98.9	94.1	100	95.8	99.1	94.6
	σ	0	13.6	3	13.4	0	12.2	2.4	12	0	10.2	2.1	11.3	0	9.9	1.9	10.8
APCA	μ	100	97	98.5	94.6	100	97.6	99	94.6	100	98	99.3	95.7	100	98.1	99.4	95.2
	σ	0	3.9	2.9	6.4	0	3.3	2.1	6.8	0	3.1	1.8	5.9	0	2.9	1.6	7.4
PLA	μ	100	97.9	99.9	97.5	100	98	99.9	97.5	100	98.3	100	97.5	100	98.3	99.9	97.6
	σ	0	2.5	0.4	3	0	2.7	0.3	3.3	0	2.5	0.3	3.3	0	2.6	0.3	3.2
CHEB	μ	100	96.3	99.2	95.5	100	96.3	99.5	95.8	100	96.4	99.6	96	100	96.5	99.4	96.2
	σ	0	9.2	1.4	10.9	0	9.4	1	10.2	0	9.1	0.9	9.6	0	8.7	1.2	9.1
SVD	μ	100	95.9	96.2	95.1	100	95.2	96.1	93.4	100	96.1	96.9	94.3	100	96.5	97.4	95.1
	σ	0	4	3.3	4.1	0	6.3	3.6	9	0	5.6	3.4	7.9	0	5.2	3.3	7.2

5% of the email worms when different feature vector matrices, with various k values, are analysed. Thereby, the DWT-2 is also not considered in what follows. The reason for this is that, as explained in Section 4.2.2, using the DWT-2 has similar space complexity to using the DWT-4 but the former makes the method perform worse than the latter in uncovering the two-cluster scheme.

The fact that the feature vectors fall into two clusters and thereby that the corresponding DNS query streams into two profiles does not, necessarily, imply that one profile corresponds to non-infected and the other to infected user machines. To show that this is the case, the feature vectors in the two clusters were examined by means of the overall false positive and negative rates for every time series representation and value k combination, and the respective experimental results are

discussed hereafter. Before discussing these results it is worth noting that to calculate the two rates, it was necessary to find a way to determine which cluster should contain legitimate users' and which email worm feature vectors. Counting the feature vectors in the two clusters, revealed that for every time series representation and value k combination, one cluster was densely and the other sparsely populated. Thereby, a hypothesis that intrinsically holds true for every behaviour-based (anomaly-based) detection method was taken into account to resolve this problem. This hypothesis is that the normal instances account for an overwhelmingly large portion of the analysed data objects in relation to the anomalous instances. In connection with this, the proposed method classifies a user machine as infected when the feature vector generated from its DNS query stream is in the sparsely-populated cluster.

Fig. 4.5 presents the overall false positive and negative rates of the proposed method for every time series matrix and value k combination when each of the DWT-1, DWT-3, DWT-4, PAA, APCA, PLA, CHEB and SVD is used. The figure shows that the method is capable of detecting user machines infected with any of a variety of email worms with high accuracy and negligible false alarm rate. In more detail, the overall false positive rate plots reveal that the method misclassified less than 1% of non-infected user machines as infected for every time series representation, time series matrix and value k combination. These plots also demonstrate that working with the DWT-4, PLA or SVD, slightly outperforms working with the other time series representations. The reason for this is that using any of these three time series representations, makes the method exhibit practically constant overall false positive rate, which oscillates near zero for every time series matrix and value k combination. The overall false negative rate plots indicate that the proportion of infected user machines that the method fails to identify is under 7% for every time series representation and time series matrix combination when $k > 4$. Furthermore, these plots reveal that using the PLA, renders the method more effective than using any of the other time series representations. Specifically, the method with the PLA exhibits false negative rate less than 3% for every time series matrix when $k > 4$.

Since the overall detection efficacy of the proposed method with any of these eight time series representations does not differ significantly, a conservative criterion was used to determine which time series representations should be included in the analysis presented in the second part. Specifically, only the time series representations that make the method exhibit overall false negative rate less than 3% for more than 70% of the examined feature vector matrices and every value k are taken into account. Based on this criterion, the DWT-3, DWT-4 and SVD are omitted from the discussion in the remainder of this chapter. One additional reason that motivates omitting the DWT-4 and SVD is that their corresponding feature vector extraction functions take as input a set instead of a single time series. Thereby, working with these time series representations requires recomputing the entire feature vector matrix every time a user machine enters or leaves the monitored network. Moreover, producing the

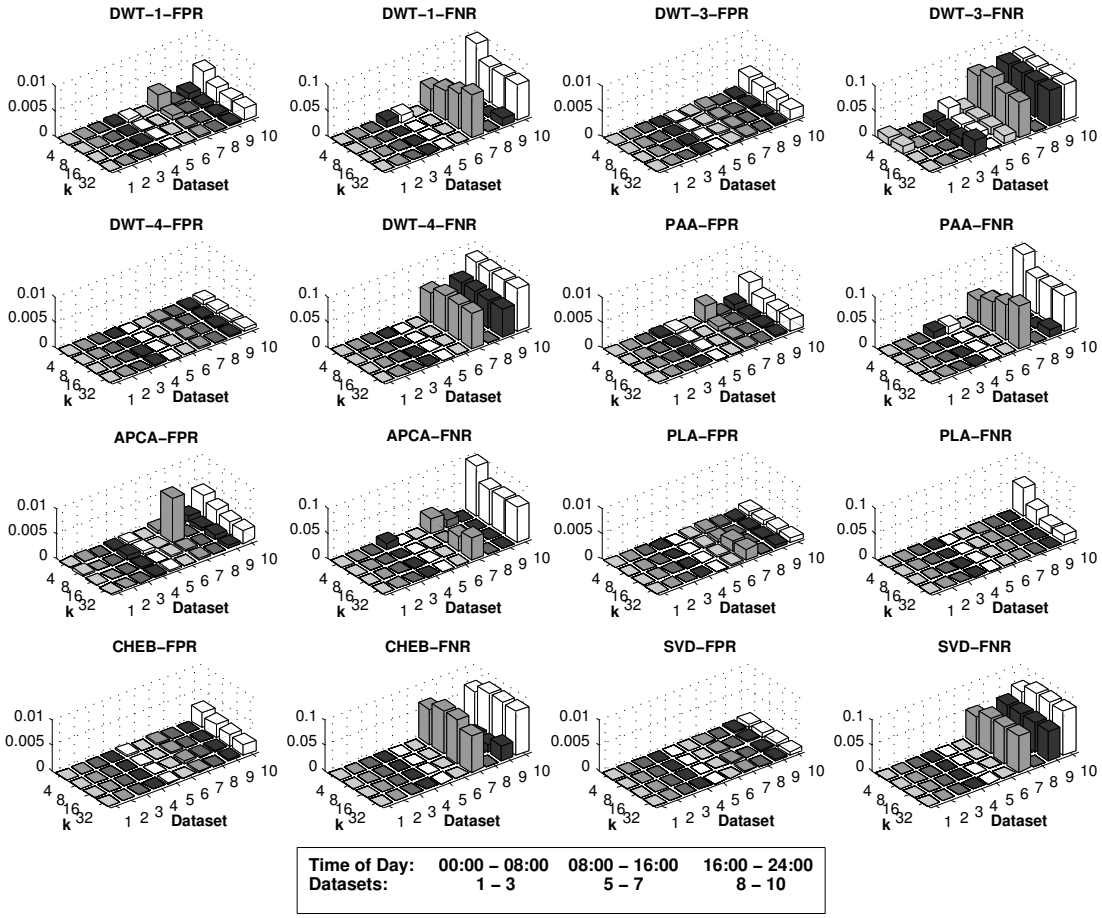

Figure 4.5: Overall false positive (FPR) and negative (FNR) rates of the proposed method against the ten feature vector matrices (datasets) when the DWT-1, DWT-3, DWT-4, PAA, APCA, PLA, CHEB and SVD with k equal to 4, 8, 16 and 32 are used. The FPR of the method with any time series representation is under 1% for every dataset and value k. Regarding the FNR, the method with the PLA is more effective than with the other time series representations, as it exhibits FNR less than 3% against all ten datasets when $k > 4$.

SVD-based feature vector matrix of a $n \times p$ time series matrix, is far more expensive than producing the feature vector matrices associated with the other time series representations. Specifically, the classic SVD algorithm has $O(np^2)$ time complexity; this renders the method intractable even for moderately-sized time series matrices, and as a consequence networks. The DWT-1 is also not considered hereafter because the method with the DWT-1 and the PAA performs identical in terms of overall detection efficacy but the PAA is simpler to implement than the DWT.

4.4.3 Per Worm Efficacy

To produce the experimental results presented in this part, a four-step procedure was performed on each of the ten $n \times 256$ legitimate users' time series matrices. To obtain a sufficient statistical sample, the procedure described hereafter was repeated 100 times for each of these matrices. First, for each matrix-iteration pair, the random number generator provided by Haahr [62] was used three times to determine how many user machines become infected within the detection period, which are these user machines and at which time bin the infection of each of these user machines occurs. Second, the time series matrix was constructed by merging the legitimate users' time series with one email worm time series based on the random numbers generated in the first step. Third, following the steps described in Sections 4.2.3 and 4.2.4, each time series matrix was transformed and compressed to build its corresponding PAA, APCA, PLA and CHEB-based $n \times k$ feature vector matrices. Fourth, the rows of each feature vector matrix were clustered by means of the DIANA to create the hierarchy of partitions and only the first level thereof (two-cluster scheme) was retained. The four-step procedure was executed 71 times, one for each email worm (time series) under consideration. As a result, in total, the experimental procedure was repeated for each email worm: 10 legitimate users' time series matrices $\times$ 100 iterations $\times$ 4 time series representations $\times$ 4 feature vector lengths = 16 000 times. The experimental procedure of this part is illustrated in Fig. 4.6.

The presentation of the experimental results is organised into three subparts. The first subpart analyses by means of the per worm false positive and negative rates the two-cluster scheme partitions of the feature vectors that were produced when the PAA, APCA, PLA and CHEB-based feature vectors were employed. The second subpart complements the analysis of the first subpart by examining independently how the number of infected user machines and the time at which the infections occur affect the two-cluster schemes that DIANA returns. The purpose of these two subparts is twofold. First, they are aimed to provide evidence that the proposed method exhibits remarkable accuracy in distinguishing between non-infected and user machines infected with various email worm instances. Second, they are intended to indicate which is the most appropriate time series representation for the method. In the third subpart, a comparative analysis of the proposed method and methods that base detection on the volume or self-similarity of the DNS traffic that user machines generate is presented. The objective of this subpart is to demonstrate that the proposed method clearly outperforms detection methods that build on taking advantage of commonly-studied characteristics of Internet traffic signals.

Having shown that the proposed method identifies accurately one infected user machine, regardless of the email worm the user machine is infected with, the next step was to assess its efficacy in detecting various email worm instances. To do this, the per worm false positive and negative rates were employed, and the respective experimental results for the PAA, APCA, PLA and CHEB are discussed hereafter.

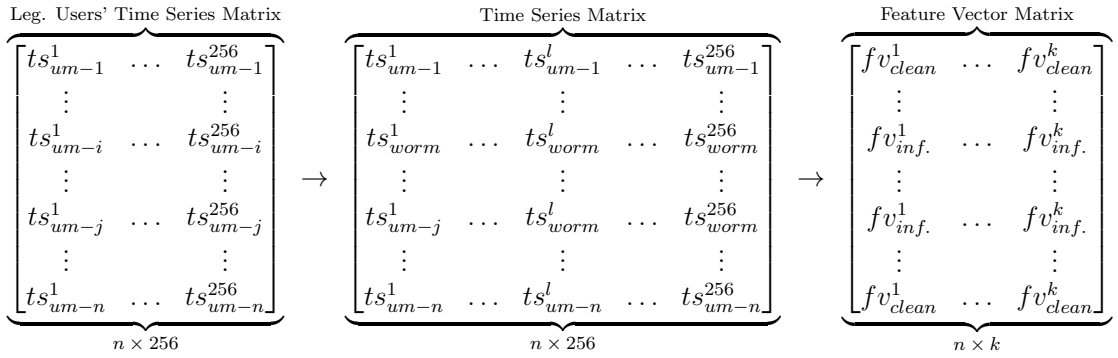

Figure 4.6: For each $n \times 256$ legitimate users' time series matrix, three random numbers that specify how many user machines become infected within a detection period, which are these machines and at which time bins their infection occurs are generated. Using these numbers and one email worm time series, the time series matrix is created. This matrix is transformed and compressed to get the $n \times k$ PAA, APCA, PLA and CHEB-based feature vector matrices, with k equal to 4, 8, 16 and 32. The rows of these matrices are clustered by means of the DIANA and only the first level of the hierarchy of partitions is retained.

Fig. 4.7 and Fig. 4.8 illustrate the mean of the rates over the 1 000 feature vector matrices that correspond to each email worm and value k combination (10 legitimate users' time series matrices $\times$ 100 iterations = 1 000 feature vector matrices). In these figures as well as in every other given later in this chapter, each bar of the depicted plots corresponds to an email worm.

The figures demonstrate that the proposed method with the PAA, the PLA or the CHEB detects various email worm instances with remarkable accuracy and negligible false alarm rate. They also indicate that working with any of the PAA, PLA and CHEB, clearly outperforms working with the APCA, as the former makes the method exhibit mean per worm false positive and negative rates lower than the latter by, at least, one order of magnitude. In connection with this, the APCA is excluded from the analysis presented in the remainder of this chapter. In more detail, the mean per worm false positive rate plots (Fig. 4.7) show that the method classifies only around 1% of the non-infected user machines as infected when any of the PAA, PLA and CHEB with $k = 4$ is used. Moreover, they reveal that the method with the PLA or the CHEB is slightly more effective than with the PAA, as its mean per worm false positive rate remains lower than 1% for every email worm even when $k = 4$. The mean per worm false negative rate plots (Fig. 4.8) show that when any of the PAA, PLA and CHEB with $k < 32$ is employed, the method fails to detect less than 1% of infected user machines for 95% of the email worms under consideration. For the rest 5% of the email worms, the method with the PLA or the CHEB is more effective than with the PAA.

Below, three points that complement the comparative analysis of the PAA,

PLA and CHEB presented above, and contribute to determine which time series representation is the most appropriate for the proposed method are mentioned. The first point is that the feature vector extraction function of the PAA is much easier to implement than those of the PLA and the CHEB. The second point relates to the transformation of the feature vector matrix to obtain the distance matrix, which is provided as input to the DIANA. This is by far the most computationally demanding task of the method and requires less time when the PAA or the CHEB is used. This is because, in contrast to the d_{pla}, the d_{paa} and the d_{cheb} are based on the Euclidean distance for which fast and computationally inexpensive algorithms for computing the distance matrix exist. The third point is that given a $n \times p$ time series matrix, producing its $n \times k$ CHEB-based feature vector matrix, has higher time complexity than producing its PAA or PLA-based one. In particular, the former is of $O((k)p)$ time complexity, whereas the latter of $O(p)$. Based on this analysis, the PLA and CHEB are the most appropriate time series representations for the method. It is noteworthy, however, that the PAA remains a good choice if the focus is on minimising the implementation and computational complexity by sacrificing accuracy to some small extent.

Fig. 4.7 and Fig. 4.8 show that the mean per worm false positive and negative rates have opposite trends as the value of k increases. This expresses in a clear way an inherent trade-off between the false positive and negative rates that is apparent in every statistical test. Specifically, although the main concern in the context of any such test is to discover the values of the parameters involved so that both rates are simultaneously minimised, in practice there is a limit to the degree that this can be accomplished. Thereby, the most appropriate value k for the proposed method is the one that provides the most acceptable balance between its mean per worm false positive and false negative rates. A close look at Fig. 4.7 and Fig. 4.8, uncovers that $k = 8$ is the best choice. This is because it is the smallest value k (shortest feature vector) that minimises the mean per worm false negative rate for every email worm, given that the mean per worm false positive rate remains almost stable under 1% for every email worm and value k combination. In light of the analysis presented above, the experimental results discussed in what follows refer to using either the PLA or the CHEB with $k = 8$.

The experimental results presented above uncover that the proposed method effectively distinguishes between non-infected and infected user machines in the monitored network. However, they provide no insight into how the efficacy of the method depends on the number of user machines that become infected within a detection period or the time (bin) at which the infection of each of these user machines occurs. To fill this void, Fig. 4.9 and Fig. 4.10 illustrate the mean per worm false positive and negative rates of the method as a function of the number of infected user machines and the dispersion of the infection times within the detection period. To compute these rates, the 1 000 PLA and the 1 000 CHEB-based feature

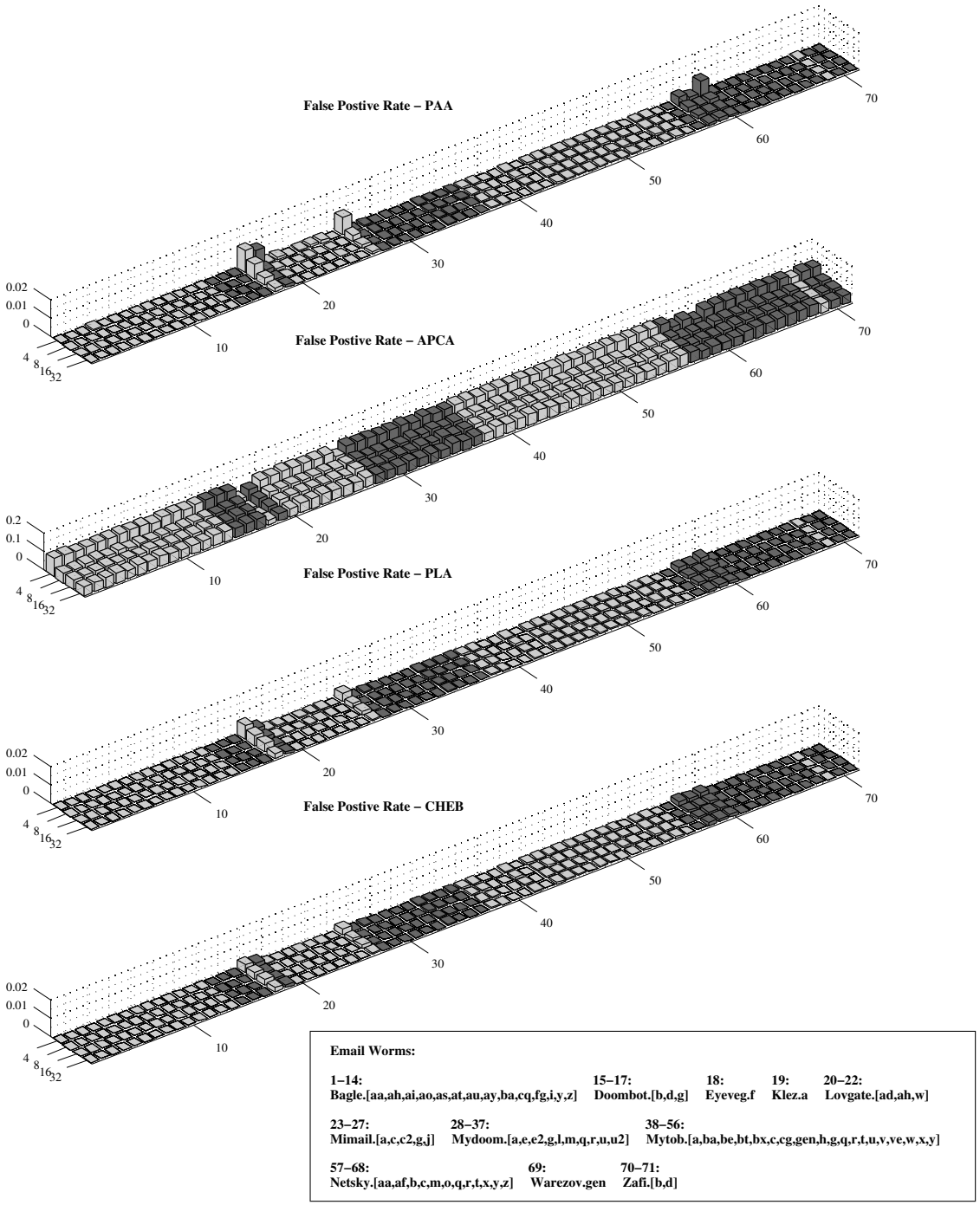

Figure 4.7: Mean over 1 000 (10 legitimate users' time series matrices × 100 iterations) feature vector matrices per worm false positive rate of the proposed method with the PAA, PLA, APCA and CHEB for every email worm (time series) and value k combination. Using the PAA, PLA or CHEB, makes the method exhibit negligible mean per worm false positive rate that in the worst cases reaches 1%; whereas, with the APCA, it performs one order of magnitude worse. The efficacy of the method in detecting various email worm instances increases as the value k increases when the PAA, PLA or CHEB are employed.

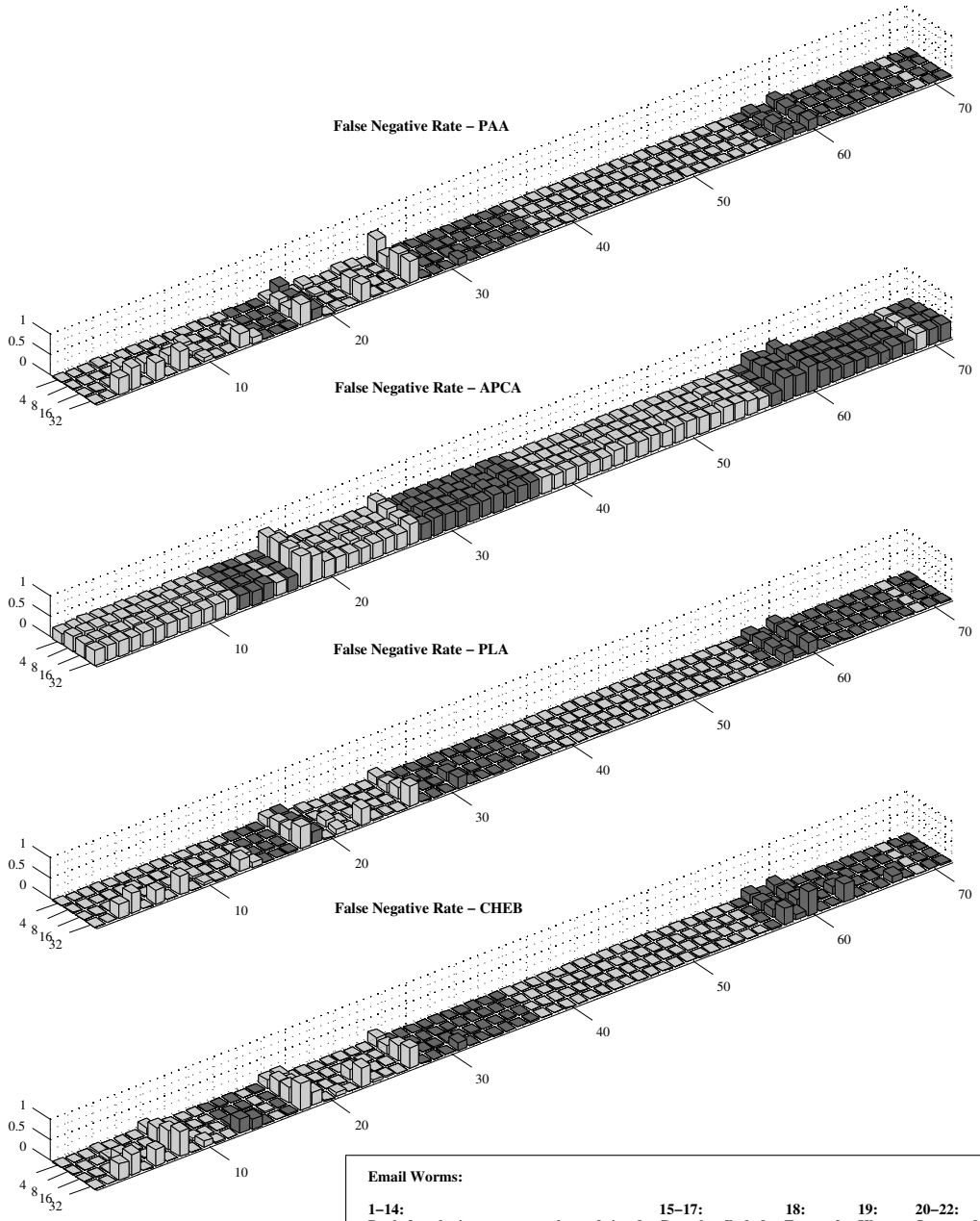

Figure 4.8: Mean over 1 000 (10 legitimate users' time series matrices × 100 iterations) feature vector matrices per worm false negative rate of the proposed method with the PAA, PLA, APCA and CHEB for every email worm (time series) and value k combination. Using the PAA, PLA or CHEB, makes the method more effective than using the APCA by one order of magnitude. For 95% of the email worms, the method with the PAA, PLA or CHEB has mean per worm false negative rate lower than 1% for every combination. The accuracy of the method with the PAA, PLA or CHEB decreases as the value k increases.

vector matrices were grouped independently per the number of email worm feature vectors they contain and per the difference between the time bins of the last and first infection. In Fig. 4.9, the number of infected user machines is given as the percentage of the total number of user machines that queried the local name server of the monitored network within the detection period. In Fig. 4.10, the coordinates on the $y-$axis represent the time that passes between the first and the last infection. For instance, the value 4 on the $y-$axis corresponds to feature vector matrices with difference less than 4 time bins between the first and last infection.

Fig. 4.9 indicates that the efficacy of the proposed method with the PLA or CHEB does not deteriorate significantly when the number of infected user machines increases. This is supported by the fact that the mean per worm false positive and negative rates remain fairly stable under 1% for approximately 95% of the email worms under consideration. For the rest 5% of the email worms, the performance deterioration is negligible with respect to the mean per worm false positive rate, since the proportion of non-infected user machines that are misclassified as infected is less than 1%. By contrast, the mean per worm false negative rate of the method with the PLA or the CHEB gets as high as 30% and 40%, respectively, when 5% of the user machines become infected. This result has two interesting implications. The first of them is that using the PLA, renders the method more effective than using the CHEB; whereas, the second implication is that the method catches at least one instance of every email worm. The latter implication leaves a window open for enhancing the method, in the future, to make it capable of detecting more accurately user machines that become infected with this 5% of the email worms.

Fig. 4.10 shows that the dispersion of the infection times within the detection period has also a negligible negative effect on the efficacy of the proposed method, when the PLA or CHEB with $k = 8$ is used. In more detail, the mean per worm false positive rate plots indicate that the method classifies as infected less than 1% of the non-infected user machines, regardless of the email worm the user machines are infected with and the time that elapses between the first and last infection. The mean per worm false negative rate plots reveal that the times at which the user machines become infected within the detection period affects only slightly the efficacy of the method for 5% of the email worms under consideration. For the rest 95% of the email worms, the method with PLA or CHEB and $k = 8$ exhibits almost constant mean per worm false negative rate that oscillates below 1%.

Hereafter, the efficacy of the proposed method in identifying various email worm instances is compared to that of three threshold-based detection methods that build on taking advantage of commonly-studied characteristics of Internet traffic signals. Two of these methods base detection on the volume and the third on the self-similarity of the DNS traffic that user machines generate. In particular, the volume-based methods classify a user machine as infected if the average number of its DNS queries over all the time bins of the detection period or the number of

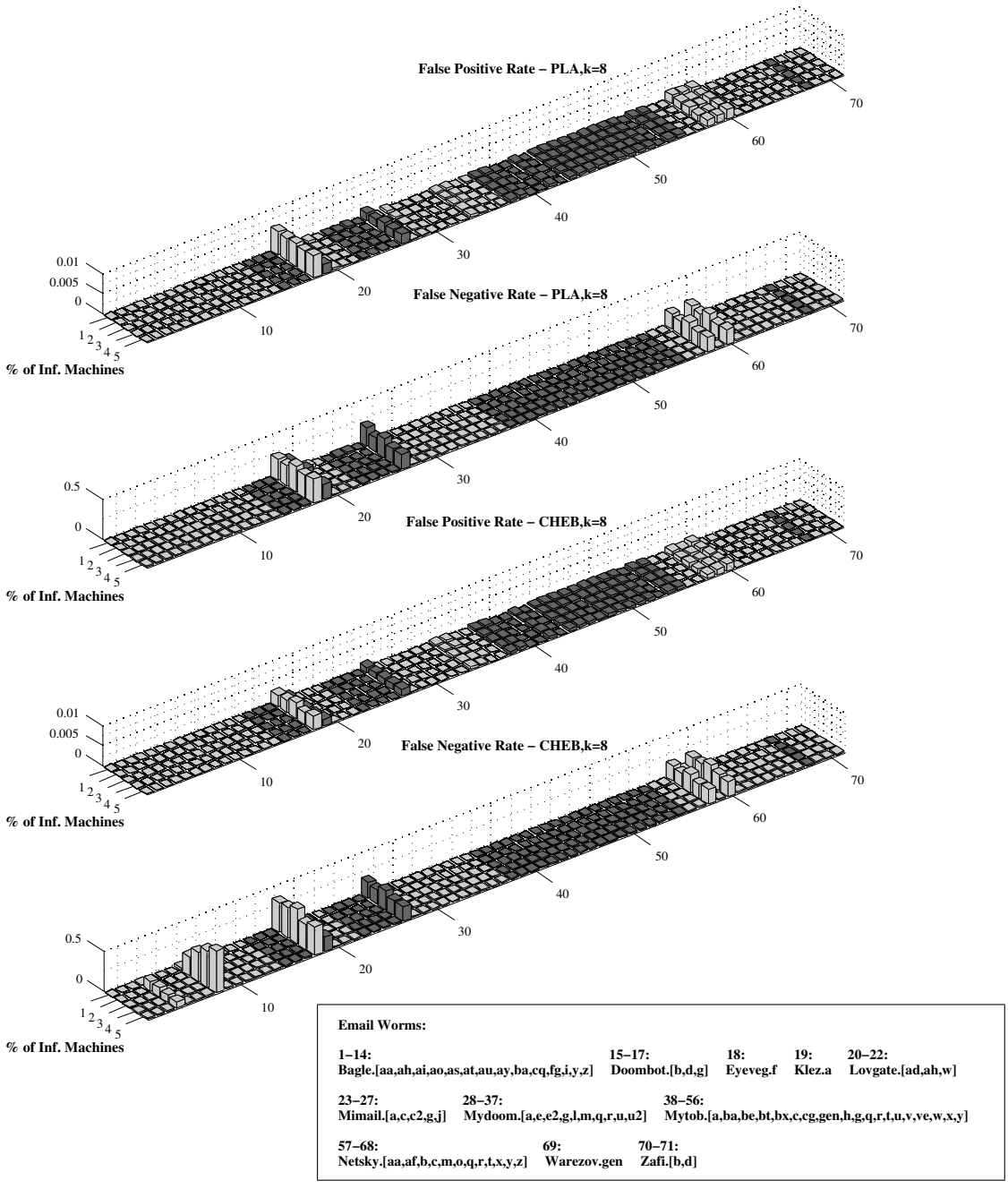

Figure 4.9: Mean per worm false positive and negative rates of the proposed method for every email worm against the percentage of infected user machines within the detection period (1 – 5 %) when the PLA and CHEB with $k = 8$ are employed. The method classifies as infected less than 1% of the non-infected user machines for every email worm regardless of the percentage of infected user machines. For 95% of the email worms, the detection efficacy of the method does not decrease as the number of infected user machines increases.

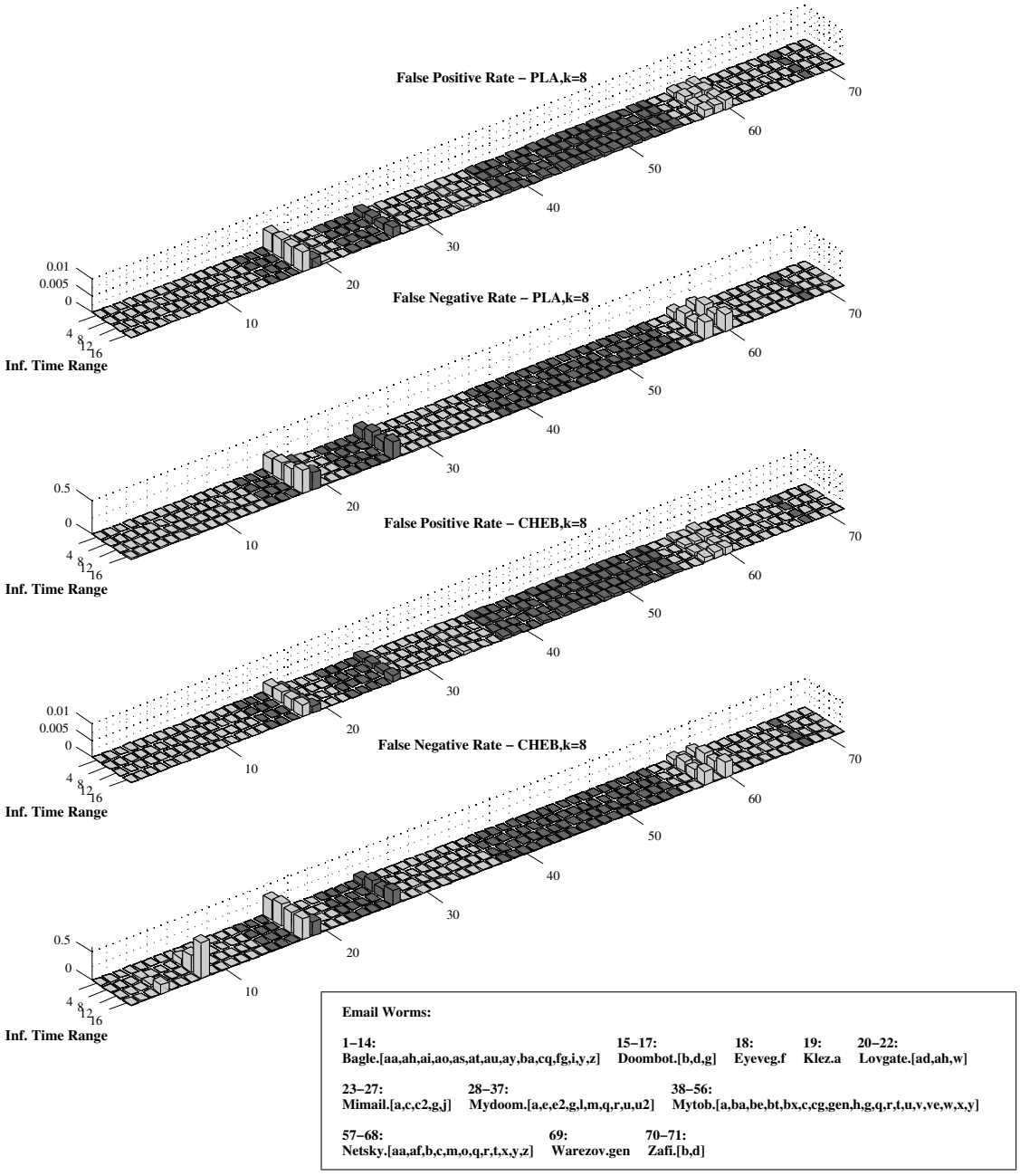

Figure 4.10: Mean per worm false positive and negative rates of the proposed method for every email worm against the time that passes between the first and last infection within the detection period (the coordinates on the $y-$axis represent the time difference in time bins between first and last infection) when the PLA and CHEB with $k = 8$ are employed. The method classifies as infected less than 1% of the non-infected user machines irrespective of the dispersion of infection times within the detection period. For 95% of the email worms, the effectiveness of the method varies negligibly as the dispersion increases.

its DNS queries in one time bin of the detection period exceeds a predetermined threshold value. The third method was inspired by the findings reported in Chong et al. [34], which suggest that the traffic originating from user machines infected with scanning worms is self-similar. Specifically, the third method classifies as infected any user machine whose outgoing DNS traffic within the detection period has degree of self-similarity higher than a predetermined threshold value. To quantify the degree of self-similarity, the Hurst parameter (H) was employed in conjunction with its non-parametric estimator presented in Hipel et al. [70], which is fast and robust in the presence of non-stationarity.

As previously mentioned, each of the three detection methods described above relies on a predetermined threshold value to decide whether a user machine is infected or not. Determining proper threshold values requires knowledge about the typical DNS traffic characteristics of non-infected user machines. Since such knowledge is not available, a three-step data-dependent procedure was followed to determine the threshold values. First, the mean, maximum and H values were computed for each row of each legitimate users' time series matrix. Second, the trimmed means over the mean, maximum and H values, independently, were calculated. The trimmed mean was used because it is a more robust measure of central tendency than the simple mean; the reason for this is that it eliminates the effect of values that depart significantly from the norm. The trimmed means of the mean, maximum and H values were considered to represent the typical values that characterise the DNS traffic that non-infected user machines generate. Third, the threshold value for each detection method was set to the trimmed mean of its corresponding measure plus two trimmed standard deviations.

Fig. 4.11 and Fig. 4.12 illustrate the mean over 100 iterations per worm false positive and negative rates of the proposed method with the PLA and the CHEB when $k = 8$ and those of the three threshold-based methods for the ten legitimate users' time series matrices. The mean per worm false positive rate plots (Fig. 4.11) indicate that the proposed method outperforms the three threshold-based methods. Even in the worst case (Dataset 10), it exhibits false positive rate around 1%, which is one order of magnitude lower that the rates of the threshold-based methods. The mean per worm false negative rate plots (Fig. 4.12) uncover that the proposed method with the PLA fails to detect as many as half of the infected user machines for approximately 5% of the email worms under consideration over two legitimate users' time series matrices. Nevertheless, this performance is still superior to that of the three threshold-based methods. In particular, the two volume-based methods misclassify around 1% of infected user machines for every email worm time series and legitimate users' time series matrix combination; whereas, the self-similarity-based method is totally ineffective because its false negative rate is greater than 75% for more than 33% of the email worms.

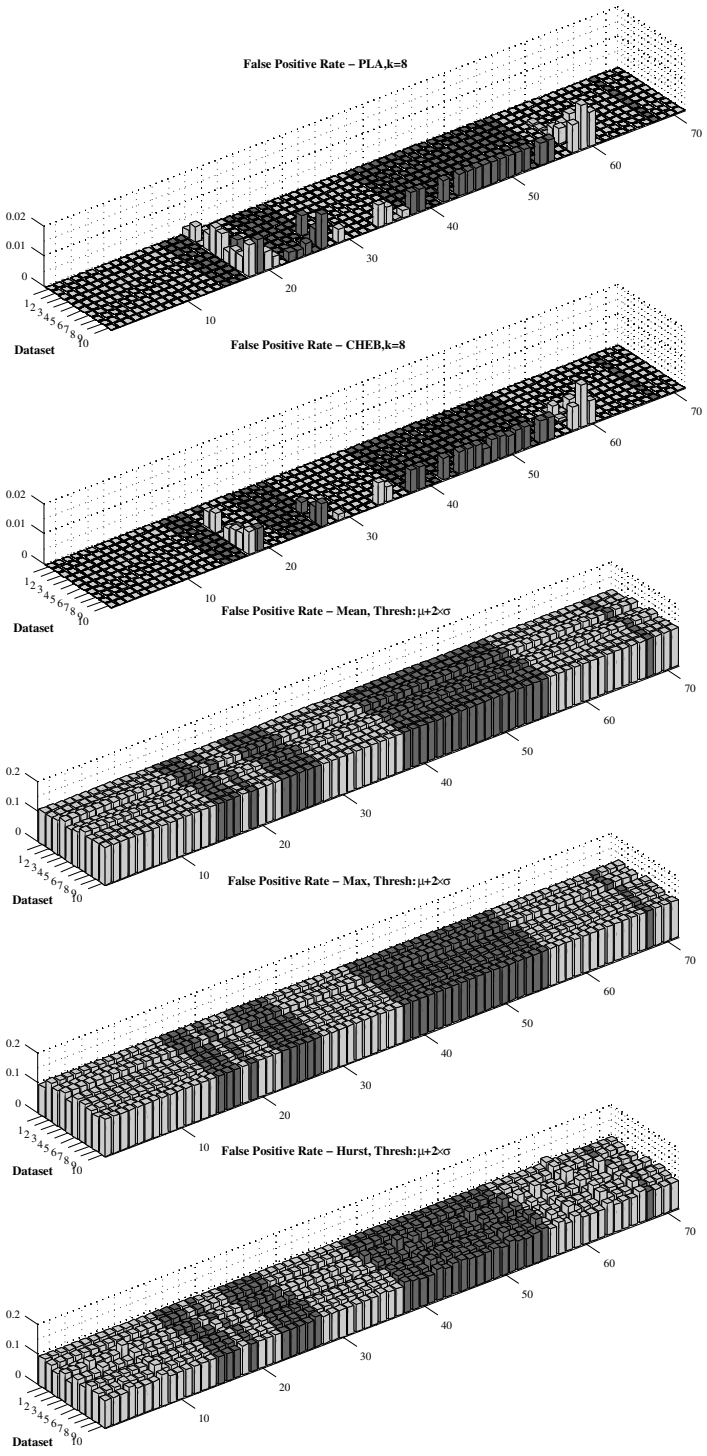

Figure 4.11: Mean over 100 iterations per worm false positive rate of the proposed method with the PLA and the CHEB when k = 8 and of the three threshold-based methods for every email worm (time series) against the ten legitimate users' time series matrices (datasets). The threshold-based methods base detection on the mean rate (Mean), biggest spike (Max) and self-similarity (Hurst) of the DNS traffic that user machines generate within the detection period. The threshold values for these methods were set to $\mu + 2 \times \sigma$. The proposed method outperforms the threshold-based methods by one order of magnitude.

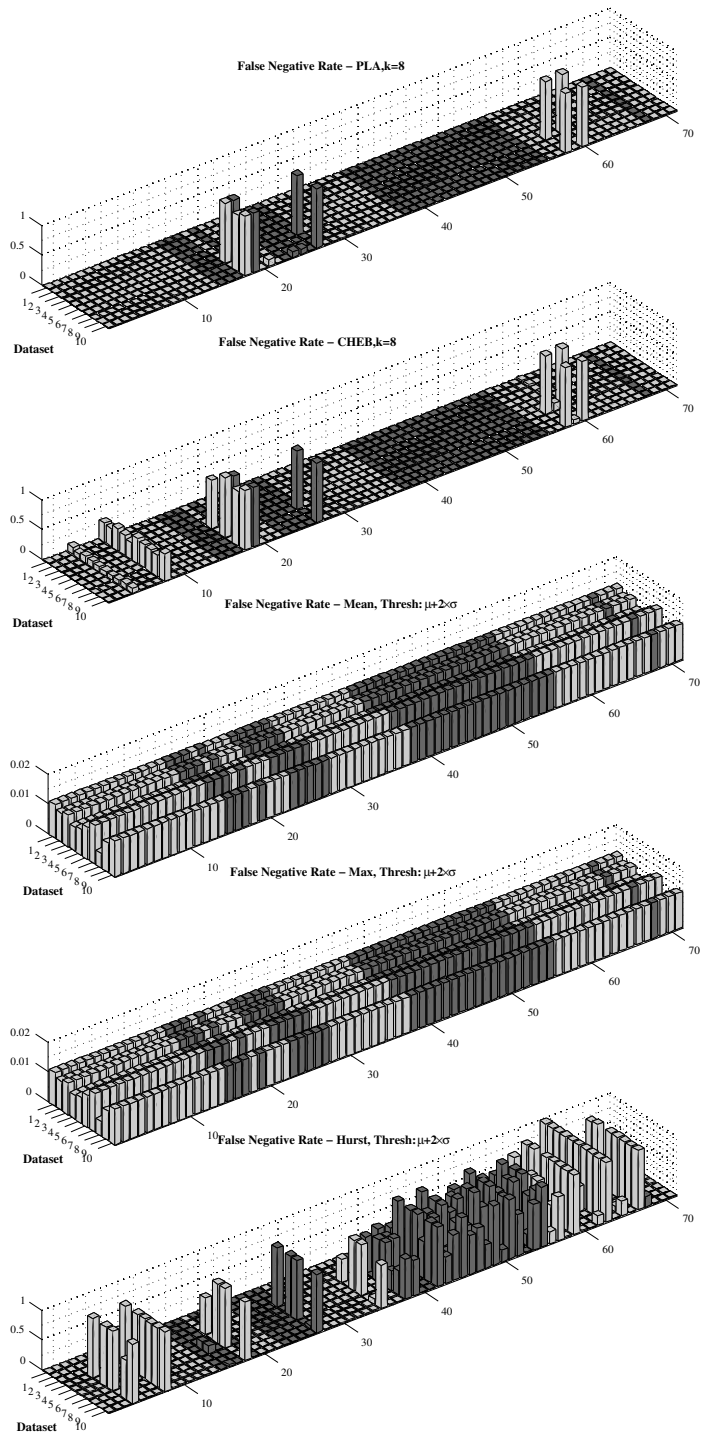

Figure 4.12: Mean over 100 iterations per worm false negative rates of the proposed method with the PLA and the CHEB when k = 8 and of the three threshold-based methods for every email worm (time series) against the ten legitimate users' time series matrices (datasets). The proposed method outperforms the two volume-based methods (Mean and Max), which erroneously classify around 1% of the infected user machines for every email worm against every dataset. The self-similarity method (Hurst) is completely ineffective, since its mean per worm false negative rate is greater than 75% for more than 33% of the email worms.

4.5 Evasion of Proposed Method

A close look at the recent malware prevalence tables, which are publicised by Virus Bulletin [180] and summarised in Section 2.3.1, uncovers that the most prevalent Internet worms are members of a small number of worm families. In fact, new Internet worms are usually rewrites of known ones, and as such they normally perform many similar or identical operations on the Internet-connected machines they compromise to other worms of their family. As a consequence, the flow-level characteristics of the DNS query streams that user machines generate once they become infected with email worms have not changed significantly over the past several years. Thereby, it is reasonable to expect that, as long as this trend in the development of Internet worms continues, these characteristics will remain largely unaltered in the future, as well. Although the proposed method was evaluated against a large number of email worms that appeared recently in the wild and members of all the most prevalent email worm families were considered, its utility is examined from another perspective in this section. Specifically, it is assumed that worm writers are aware of the method and try to program email worms that can evade detection. In the analysis that follows, three approaches worm writers might take to achieve this are described, in order of increasing sophistication, and their effectiveness is anticipated.

The first approach that worm writers might take is to eliminate or minimise the dependency of email worms on the local name servers. To this end, they might consider two options. The first of them is to program email worms to query (instead of the local name servers) recursive name servers that are outside the monitored network, such as name servers deployed on the Internet for public use, open resolvers or name servers under malicious control. The second option is to make email worms send out infectious emails by employing the email clients that run on the infected user machines, which are usually configured to forward emails to the outgoing email servers. A simple and effective countermeasure against email worms using the first option will be to filter out on the gateway of the monitored network all the outgoing queries originating from non-name servers. Email worms using the second option are highly unlikely to (re)appear in the future. This is because, as mentioned in Section 2.3.1, worm writers originally equipped email worms with a built-in email engine so that the infectious emails bypass the outgoing email servers. Apart from this, user machines infected with such email worms most probably will still query the local name servers when the email worms will be in the target acquisition or payload activation phase. Hence, it is reasonable to conclude that the flow-level characteristics of their DNS query streams will differ from those of non-infected user machines, which will make the proposed method classify them as infected.

The second approach worm writers might pursue is to program email worms to issue DNS queries to the local name servers at lower rates. Let alone that implementing this approach will cause a significant decrease in the virulence and

survival chances of email worms, which is confirmed by the simulation results presented in Section 6.5, it cannot guarantee that infected user machines will remain undetected. Evidence supporting this claim is given in Section 4.4.2, where the efficacy of the proposed method is compared to the efficacy of the method that bases detection on the mean number of DNS queries that user machines generate. In more detail, if this approach is taken the time series that will be produced from the DNS query streams of infected user machines within the detection period will have lower mean values than those produced from the DNS query streams of user machines infected with known email worms. In Section 4.4.2, it is shown that the proposed method captures and exploits more information about the analysed time series than merely their mean value. In connection with this, it is justifiable to assume that reducing the rate at which infected user machines query the local name servers will not significantly detract from the usefulness of the method.

The third approach worm writers might consider is to make email worms mimic the patterns of email users with respect to sending out emails or those of non-infected user machines in terms of generating DNS queries. User machines infected with email worms employing the former option will be highly unlikely to go undetected. The reason for this is that they will still query the local name servers differently compared to before becoming infected, when the email worms will be in the target acquisition, payload activation and propagation phases. Developing email worms that use the second option is, so far, an insuperable challenge because no model for the DNS query streams of non-infected user machines exists. However, even if such a model was available and used or email worms were programmed to be capable of learning the patterns of the DNS query stream of each user machine they infect, it is questionable whether email worms behaving this way would avoid premature death and reach epidemic levels. Zou et al. [221] demonstrates that the behaviour of email worms in their early spreading phase, which is the result of the operations they carry out, is critical for their survival. In connection with this, worm writers constantly enrich the operations of email worms so as to improve the infection rate that these achieve in their early spreading phase. Adding operations to email worms, which is a trend likely to continue, as the detection methods also evolve, causes the flow-level characteristics of the DNS query streams that infected user machines produce to increasingly deviate from those of non-infected user machines.

4.6 Summary

In this chapter, a new behaviour-based method for detecting user machines infected with email worms that overcomes the limitations of the available methods summarised in Chapter 3 is presented. The method automatically classifies on the local name servers the requesting user machines as non-infected or infected based on the similarities in the flow-level characteristics of their DNS query streams. To

achieve this, it uses exact shape-based similarity search and clustering over time series produced by counting the number of DNS queries each user machine generates within a certain period of time. Since determining the similarity between time series, and thereby clustering them, is not possible directly, a feature vector is extracted from each time series and the feature vectors are clustered instead. In this work, various feature vector extraction functions and feature vector lengths are considered to determine which is the most appropriate feature vector for the method.

The experimental results of the method support three conclusions. The first of them is that the method can identify with remarkable accuracy and negligible false alarm rate user machines being infected with any of a large variety of recent email worms. The second conclusion is that it outperforms in terms of detection efficacy methods that focus on the volume or self-similarity of the DNS traffic that user machines generate. The third conclusion is that the PLA-based feature vector with length eight is the one that maximises the performance of the method. Further, it is justified to expect that the method will be effective in detecting user machines that become infected with email worms, including zero-day email worms, in the long run. The reason for this is that it bases detection on dissimilarities in the DNS traffic characteristics of non-infected and infected user machines that have been apparent over the past several years and are very unlikely to disappear in the future.

Part III

Mitigation of Internet Worms

Chapter 5

Literature Review

5.1 Mitigation Approaches

There exist the following three approaches for mitigating the potential extent of, and thereby the damage caused during, the outbreaks of Internet worms: prevention, treatment and containment [130]. All three approaches are reactive in nature in the sense that they are intended to take effect after an Internet worm has begun to propagate. In addition, they essentially complement each other because none of them is complete by itself and, as will become clear in what follows, they address the same problem from different aspects. Specifically, they differ in three important ways. The first of them is the amount of time that passes before they can become operative, which is referred to as reaction time hereafter. The second difference is that the prevention and treatment methods are applicable to non-infected and infected machines, respectively; whereas, the containment methods operate on the traffic of possibly or actually infected machines. The third difference of these approaches concerns the effect they are supposed to produce.

Prevention and treatment concentrate on limiting the number of vulnerable machines that known Internet worms can infect or reinfect. To accomplish this, the prevention methods secure non-infected machines so that they are protected if, or when, these Internet worms attempt to infect them. In practice, prevention is usually understood as updating the databases of signature-based detection systems once the attack signatures for emerging Internet worms are generated. The treatment methods clean infected machines and, subsequently, secure them against future infections by the same Internet worms. The typical example of treatment involves removing vulnerabilities from infected machines and, then, updating the attack signature database of the antivirus software that protects them. Instead of concentrating on limiting the number of vulnerable machines that known Internet worms can infect or reinfect, the containment methods slow down the Internet-scale propagation of zero-day, or often even of known, Internet worms to give humans the time to adapt

and apply prevention and treatment. To achieve this, the containment methods prevent possibly or actually infected machines from sending out or restrict the rate at which they send out suspicious traffic.

As mentioned above, prevention and treatment depend, to a large degree, on signature-based detection. This is because the only method to secure an Internet-connected machine from becoming infected with a known Internet worm is to install the relevant attack signature on the signature-based detection system that protects it. Thereby, the available prevention and treatment methods have an inherent weakness that emerges from this dependency. Specifically, they can affect the speed of the epidemics of Internet worms only after the relevant attack signatures have been generated and installed. Nevertheless, as pointed out in Chapter 3, the process from generation to installation of attack signatures entails non-trivial human labour and interactions that normally take significant time. As a consequence, it highly likely, and has often occurred to date, that a zero-day worm succeeds in infecting the entire vulnerable population before these methods can take effect. This state of affairs has led to increasing attention being given to containment, which has the potential to deal with zero-day worms in a more timely manner.

The short reaction time of containment compared to those of prevention and treatment is due the fact that, in contrast to these two approaches, containment is conceptually and operationally related to behaviour-based instead of signature-based detection. As a result, the available containment methods are operative when or take effect shortly after the first observable symptoms of infection are detected in the traffic of a machine in a monitored network. However, although these (behavioural) symptoms might be potentially very reliable, they are not absolute indicators of infection. In addition, at the time when the first observable symptoms are detected there is normally no relevant attack signature to prevent further infections and treat already infected machines. For these two reasons, the available containment methods are network-based and target preventing possibly or actually infected machines from or limiting the rate at which they spread Internet worms further. This usually involves automatically blocking or filtering out traffic that conforms to behavioural signatures associated with Internet worm activity.

5.2 Containment Methods

In Chapter 4, a new behaviour-based detection method for email worms is presented. As stated in the previous section, behaviour-based detection methods serve as basis for designing containment methods. Thereby, a new containment method for email worms, which is closely related to the method presented in Chapter 4, is introduced later in this thesis. However, before proceeding to do this in Chapter 6, in the remainder of this chapter, a thorough, critical review of the existing studies that propose containment methods is given. The objective is to elaborate on two issues.

The first of them is to explain the reasons that render the available containment methods ineffective or inefficient against email worms. The second issue is to highlight the findings of these studies that provide valuable insight into and guidance for designing and assessing the design and performance of the new method for slowing down the Internet-scale epidemics of email worms introduced in Chapter 6.

Apart from making containment methods capable of reacting in a timely manner to the epidemics of Internet worms, the relation between behaviour-based detection and containment has a negative implication, as well. This is namely that the containment methods, for the same reasons as the behaviour-based detection methods, are worm (sub)class-specific. In fact, security researchers and practitioners increasingly argue that devising containment methods that can limit the propagation speed of Internet worms belonging to all, or at least more than one subclass presents insuperable challenges. For instance, Mannan et al. [117] identifies some of the factors that render the containment methods for email worms ineffective against IM worms. Thereby, the methods that are proposed in the literature for containing scanning, IM, P2P and email worms are described separately in what follows.

5.2.1 Methods for Scanning Worms

The number of studies that focus on containment of scanning worms is very limited compared to the number of those that focus on behaviour-based detection of Internet worms belonging to this class. In fact, only a few studies that systematically explore the design space for containment methods for scanning worms can be found in the literature. Apart from these studies, there also exist a few studies that go a step beyond this and report on specific design and performance details of systems that implement such methods. Presumably, the reason for this gap in the literature is associated with the simplicity of the operations that relate to the propagation of scanning worms and the close connection between behaviour-based detection and containment. Specifically, it seems uncomplicated to limit the propagation speed of scanning worms by simply blocking traffic originating from Internet-connected machines whose connection or failed connection rate exceeds static or dynamically adaptable threshold values. However, in practice, designing, developing and configuring systems that can reduce the speed at which infected machines spread scanning worms further, present numerous significant challenges that deserve immediate attention. In connection with this, this subsection is dedicated to discussing previous work that explicitly concentrates on the containment of scanning worms.

As noted above, some of the previous work is devoted to studying the problem of containing scanning worms from a theoretical perspective [130, 220, 199, 213, 30, 143, 110]. The authors of these studies build on the findings of Kephart et al. [95, 96] and use computer simulation to support their claims and proposals. Kephart et al.

demonstrates that the characteristics of the Internet-scale propagation of scanning worms resemble those of biological infectious diseases. Based on this result, it shows that it is possible to adapt the classical deterministic epidemic models to describe the Internet-scale propagation of scanning worms. With the latter as a basis for experimental explorations, Moore et al. [130] identifies the following three dimensions for the design of containment methods: the deployment scenario, the containment strategy and the reaction time. Wong et al. [199] and Zhang et al. [213] concentrate on one of these three dimensions. Specifically, Wong et al. focuses on the deployment scenario and assesses the effectiveness of connection rate limiting on user machines, edge routers and backbone routers. Zhang et al., similarly to Zou et al. [220], suggests blocking the outgoing traffic of possibly or actually infected machines for a short time and evaluates the effect of the reaction time. As a possible way to increase containment efficacy, Chen et al. [30] proposes to quarantine the entire networks in which possibly or actually infected machines reside instead of isolating these machines only. Toward the same goal, Porras et al. [143] describes a containment method that combines the cooperative alert sharing scheme presented in Nojiri et al. [136] and connection rate limiting on the gateway of a monitored network. Liljenstam et al. [110] compares the potential of passive containment methods (generally referred to simply as containment methods), such as blocking or rate limiting connections, to that of proactive ones (generally referred to as prevention and treatment methods), such as patching or removing infected machines.

Apart from the studies presented so far, which mainly aim at identifying design principles, there also exist several studies that discuss implementation and evaluation issues of containment methods [194, 28, 100, 162, 190, 218, 52, 38]. Notably, Williamson [194] is the first study that examines the usefulness of connection rate limiting using real traffic traces. To reduce the propagation speed of scanning worms, it proposes to restrict the maximum allowed rate of connections to unique destinations of Internet-connected machines to 1Hz. This corresponds to permitting one connection to a destination not contacted before per second. Based on the same principle, Chen et al. [28] suggests maintaining the failed connection rate of Internet-connected machines under a predetermined threshold value by dropping a minimum number of connection requests. Kim et al. [100] presents a refinement of the methods described in Williamson and Chen et al. that improves their false alarm rate and shortens their reaction time. Staniford [162] uses the TRW [85, 149] to determine dynamically adaptable threshold values for the connection and failed connection rates of Internet-connected machines. The authors of that study claim that their method can stop infected machines from spreading scanning worms further in as few as ten connection attempts. Weaver et al. [190] introduces a simplified version of the TRW designed to meet the demands of high-performance network hardware. Zou et al. [218] evaluates the effectiveness of a containment method that uses a modified version of Weaver's simplified TRW. Ganesh et al. [52] presents another variation of the TRW, which

is based on the cumulative sum (CUSUM). Instead of employing the TRW, Dantu et al. [38] takes a different approach to contain worms with various scanning rates. The proposed method is based on feedback control theory to dynamically limit the connection rate of possibly or actually infected machines.

Despite the positive contribution of the studies cited above to the design and development of containment methods, their findings are not generalisable to Internet worms that do not probe the IP address space (topological worms). Therefore, they are not directly applicable or adaptable to contain email worms. As for the studies that investigate design issues, this is because the models presented in Kephart et al. are relevant for scanning worms only. Specifically, these models are based on three assumptions that, to a greater or lesser extent, hold true for scanning but not for email worms. The first of them is that Internet worms propagate through direct connections between infected and vulnerable machines. The second assumption is that all the vulnerable machines have equal probabilities of being targeted and becoming infected. The third assumption is that the propagation and infection time are negligible. As pointed out in Section 2.3.2, the propagation of email worms does not rely on direct connections, is heavily influenced by the behaviour of end users and the social information stored on their machines, and takes place over time scales in the order of minutes to hours. In addition, the containment methods proposed in the rest studies previously discussed have a common operational characteristic that renders them ineffective against email worms. Namely, they target limiting the rate at which possibly or actually infected machines connect to other machines. However, machines infected with email worms, in contrast to those infected with scanning worms, do not exhibit connection or failed connection rates that deviate from the typical rates of non-infected machines. As a consequence, if these methods were used to slow down the propagation of email worms, they would produce meagre results. This is because they would impose constraints on the legitimate traffic without significantly, if at all, affecting infectious traffic.

Although the studies discussed in this section do not address email worms, their findings provide valuable directions for designing new containment methods for email worms. In Chapter 6, taking these directions as a starting point, it is shown that applying traffic control mechanisms to the DNS response streams that the local name servers send back to user machines, is a promising approach for effectively slowing down the Internet-scale epidemics of email worms. In connection with this, the shortcomings of the available containment methods that involve controlling email traffic and the potential advantages of controlling DNS traffic instead are analysed later in this chapter. In light of this analysis, the following chapter is devoted to presenting in detail a new containment method for email worms and evaluating its efficacy. The evaluation includes studying the impact of various containment strategies and the effect of the reaction time on the spreading speed of email worms and the traffic of non-infected and infected user machines.

5.2.2 Methods for IM Worms

Only a handful of studies that propose containment methods for IM worms can be found in the literature [159, 196, 118, 183, 76]. Notably, all of them are concerned with adapting the available containment methods for scanning worms to cover IM worms. Specifically, the methods they propose inspect the content of and apply restrictions to the traffic that flows between the machines of IM users and IM servers to control the rate at which infected machines spread IM worms further.

Smith [159] concentrates on reducing the number of vulnerable machines that possibly or actually infected machines can reach. To accomplish this, it suggests temporarily disconnecting the IM users that have the largest contact lists every time an IM worm outbreak is detected. Using the same principle as Williamson [194], Williamson et al. [196] proposes to limit the maximum rate at which IM users are allowed to interact with their contacts. This proposal is based on the fact that IM users typically communicate with a slowly varying subset of IM addresses, whereas IM worms send chat messages to all the IM addresses in the online contact list of the user machines they infect. Mannan et al. [118] presents two refinements of Williamson et al. to relax the constraints the proposed method imposes to legitimate traffic. The first of them involves limiting the rate of URL-embedded chat messages and file transfer requests only and the second enhancing IM servers to issue a CAPTCHA challenge-response test [183] every time a file transfer is requested. Huerta et al. [76] argues that CAPTCHA tests can be easily evaded and introduces a method that traces back user machines that send suspicious URL-embedded chat messages. Just for completeness sake, the method introduced in Hindocha et al. [69], which is highly unlikely to be widely adopted and used, is mentioned here. That study proposes to temporarily shut down the IM server every time an IM worm outbreak is detected, analyse the IM worm during the downtime to generate an attack signature, and force IM users to install the attack signature once the IM server is restarted.

Williamson et al. [196], Mannan et al. [118] and Huerta et al. [76] are essentially based on the same rationale as the containment methods for scanning worms described in the previous subsection. As a consequence, the methods they propose are also intrinsically incapable of containing email worms for similar reasons to those discussed in that subsection. Specifically, the network behaviour of user machines infected with email worms bears no similarities to that of user machines infected with IM worms. Furthermore, to identify and disconnect the IM users that have the largest contact lists, Smith [159] assumes knowledge about the topology of the public client-server IM network and the existence of a central control point (the IM server) that is authorised to block IM traffic. However, in the Internet-scale email network none of these two assumptions holds true. This is because information about the email contacts of each email user is not globally, centrally known. Even if it were, temporarily blocking all the email traffic originating from some user machines, would not be a viable solution. These two factors render Smith's proposals not applicable or adaptable for slowing

down the Internet-scale epidemics of email worms.

5.2.3 Methods for P2P Worms

Three distinct research approaches to devise effective methods for slowing down the Internet-scale epidemics of P2P worms can be identified in the literature. The rationale for the first approach stems from the observation that one important factor that renders P2P worms extremely virulent is that P2P networks typically consist of machines all running the same software. The software monoculture, as this state of affairs has been commonly referred to since Geer et al. [54] was published, makes it possible for P2P worms to rapidly compromise all the machines in a P2P network by exploiting a single vulnerability. The other two research approaches are founded on the same principles as the behaviour-based detection methods for P2P worms that are described in Chapter 3 of this thesis. In particular, they either assume the existence of a set of guardian nodes, which are peers with specialised capabilities that protect the P2P network, or rely on file reputation schemes.

Using computer simulation, Zhou et al. [215] shows that increasing the diversity of the software that runs on peers reduces the potential propagation speed of P2P worms. In connection with this, Freitas et al. [51] introduces the Verme. To limit the speed at which the Internet-scale epidemics of P2P worms progress, the Verme reorganises the P2P network taking into account the platform diversity of peers. The basic assumption of the containment methods presented in [214, 107, 201, 187, 212] is the existence of a set of guardian nodes with various specialised capabilities. Specifically, in Zhou et al. [214], the guardian nodes detect infected peers and broadcast self-certifying alerts, whereas in Li et al. [107] they trace back infected peers. In Xie et al. [201] and Wang et al. [187] the guardian nodes push security updates to all the peers that connect to them and embed patches in frequently downloaded files, respectively. The method proposed in Zhang et al. [212] uses two types of guardian nodes: solders and commanders. The soldiers inspect P2P traffic and send reports to the commanders. The commanders fuse the information they receive from the soldiers and apply prevention and treatment methods to slow down P2P worms. Thommes et al. [172] presents deterministic epidemic models for the propagation of passive P2P worms. Using these models, it shows that file reputation schemes can help to restrict the propagation speed of such worms.

Irrespective of the research approach they pursue, the studies cited above propose methods that are not applicable or adaptable for containing email worms. In fact, email worms remain a serious Internet threat, although the diversity of email clients that run on user machines is significantly higher than that of the software that runs on the peers of a typical P2P file-sharing network. Apparently, the diversification of email clients does not considerably affect the speed at which the Internet-scale epidemics of email worms progress. The reason for this is that almost all email

worms primarily depend on victimising humans to infect user machines and only rarely exploit vulnerabilities. Furthermore, introducing reputation schemes in the Internet-scale email network is associated with a plethora of trust management, administration, computation and network load issues. Hence, containment methods that are based on such schemes are highly unlikely to be widely adopted and used against email worms. The studies that propose methods that rely on guardian nodes present conceptual models without discussing design or implementation issues in depth. Therefore, the generalisability of the proposed methods beyond P2P worms cannot be assessed or predicted.

5.2.4 Methods for Email Worms

Although in recent years the incidence of email worms has been persistent high, only a very limited number of studies that focus on slowing down their Internet-scale epidemics appear in the literature [195, 221, 90]. This is mainly due to two reasons. The first of them is that the primary research interest shifts from detection to containment after a detection method has been devised and proven to be useful. Nevertheless, as pointed out in Chapter 3, the research work on designing methods that reliably distinguish between legitimate and email worm activity in the long run has produced only partial results. The second reason is that many security researchers and practitioners share the view that filtering the email traffic that infected user machines send to the Internet or the emails entering a monitored network is a straightforward and effective way to contain email worms. However, in practice, this is not the case because worm writers increasingly equip email worms with various modules to make them capable of escaping or confusing network-based email filters. These include built-in email engines to bypass outgoing email servers and malware designed to generate infectious emails with fake sender addresses, random noise, bad grammar and hidden HTML code.

Williamson [195] advances the traffic rate limiting method, which its author originally introduced for scanning worms in Williamson [194], to make it effective in containing email worms. Specifically, this study proposes to restrict under a predetermined threshold value the rate at which user machines are allowed to send emails to the Internet. To accomplish this, it suggests filtering the email traffic on the outgoing email servers of each network of the Internet. With the objective of designing a containment method that affects only infectious emails, Kartaltepe et al. [90] proposes to enhance the outgoing email servers to issue CAPTCHA challenge-response tests. Thereby, emails with attachment files are forwarded only after their sender has successfully solved such a test. Using computer simulation, Zou et al. [221] explores the effect of immunisation on the Internet-scale propagation of email worms. In this study, immunisation of a vulnerable machine refers to the act of updating the attack signature database of the antivirus software that protects it (immunisation is

generally referred to as either prevention or treatment depending on if the machine it is applied to is non-infected or infected).

The methods presented in Williamson and Kartaltepe et al. have a common weakness that severely limits their usefulness. This is namely that they are intended to operate on outgoing email servers. The problem with this is that, as previously mentioned, almost all email worms bypass the outgoing email servers. Apart from this significant weakness, Williamson's method has two more weaknesses. The first of them is that it can be easily confused into mistakenly identifying the possibly or actually infected machines because almost all email worms generate infectious emails with fake sender addresses. The second weakness of this method is that it is inherently incapable of dealing with infected machines that send out infectious emails with a rate lower than the predetermined threshold value. By contrast, the findings of Zou et al. serve as a very useful basis for assessing the efficacy of new containment methods for email worms. This is because it is essentially the only study in the literature that is devoted to and provides insight into modelling the Internet-scale propagation of email worms.

5.3 Summary

There exist the following three approaches to mitigate the spread of Internet worms: prevention, treatment and containment. Prevention and treatment aim at reducing the number of machines that are vulnerable to being infected or reinfected, respectively, with known Internet worms. These two approaches are closely associated with signature-based detection, and thereby the prevention and treatment methods become effective only after the necessary attack signatures have been generated and installed. By contrast, containment is associated with behaviour-based detection, and thereby the containment methods take effect promptly after the first observable symptoms of Internet worm activity are detected in the traffic that flows on a monitored network. The containment methods are not intended to reduce the number of vulnerable machines but the propagation speed of Internet worms to give humans the time to adapt and apply prevention and treatment methods. To achieve this, they prevent possibly or actually infected machines from sending or restrict the rate at which they send suspicious traffic to the Internet.

In contrast to the rich literature on the detection of Internet worms, only a few studies that explicitly deal with mitigating the extent of their outbreaks can be found. Most of these studies focus on containing Internet worms and present either design guidelines or methods for restricting the rate at which infected machines spread Internet worms further. Despite the merits of these studies, recent email worms have been capable of compromising a large number of user machines in a short period of time. This indicates that the available containment methods lack the potential to effectively slow down the Internet-scale epidemics of email worms.

This is because the findings of and the methods proposed in the existing studies are either not applicable or adaptable to contain email worms or of limited practical value. This is due to one or more of the following reasons:

- they are tightly associated with the network behaviour that machines infected with Internet worms of one (sub)class exhibit; thus, they cannot be generalised to slow down the propagation of Internet worms belonging to other (sub)classes,
- they depend critically on the topology and properties of the network on which Internet worms of a specific (sub)class spread; thereby, they cannot contain Internet worms that spread on networks with different inherent characteristics,
- they are based on incomplete understanding of the operations of email worms and the network behaviour of infected user machines; as a result, they have limited usefulness in slowing down the Internet-scale epidemics of email worms.

Despite their limitations with respect to extending, adapting or using their findings or the methods they propose to contain email worms, the existing studies make some important contributions. Specifically, they provide valuable guidelines and recommendations that should be considered when designing and evaluating a containment method. Three important points are listed below:

- there exist three key dimensions that need to be considered when designing and assessing the efficacy of a containment method: the deployment scenario, the containment strategy and the reaction time,
- applying in-the-network traffic control mechanisms, such as blocking and rate limiting, to the traffic of possibly or actually infected machines is the most promising approach to contain Internet worms,
- using distributed network sensors that automatically analyse and control the traffic of vulnerable machines in their vicinity is the key enabler for slowing down the Internet-scale propagation of worms.

In this chapter, a comprehensive and critical review of the existing studies on Internet worm mitigation is presented, with particular emphasis being placed on containment. Thereby, the reasons for the absence of an effective method for slowing down the Internet-scale epidemics of email worms and some interesting findings of these studies are discussed. On the basis of this discussion, a new containment method for email worms, which is closely related to the behaviour-based detection method presented in Chapter 4, is introduced in Chapter 6.

Chapter 6

Proposed Mitigation Method

6.1 Overview

In this chapter, a novel method for slowing down the Internet-scale epidemics of email worms, which can be regarded as the direct counterpart in the area of Internet worm containment of the behaviour-based detection method described in Chapter 4, is introduced. The method aims at restricting the rate at which infected user machines spread email worms further. To this end, its modus operandi is based on controlling the flow-level characteristics of the DNS response streams that local name servers sent back to user machines. The method is intended to automatically take effect after a possibly or actually infected user machine has been detected in a monitored network. To demonstrate and evaluate the utility of the method, computer simulation was used. In connection with this, a computational (simulation) model of the set of user machines that compose the Internet-scale email network and the legitimate and illegitimate email and DNS traffic they generate during the outbreaks of email worms was designed and implemented. The simulation results reported in Section 6.4 indicate that the method has the potential to limit the propagation speed of email worms. They also suggest that it can contribute to reducing the illegitimate traffic infected user machines send to the Internet with minimally, if at all, affecting their legitimate traffic. Moreover, the analysis presented in Section 6.5 supports that the method can be useful for containing email worms in the long run.

As discussed in Section 5.2.4, only a handful of studies in the literature deal explicitly with the containment of email worms, and all the methods they introduce operate on the email traffic that user machines generate. As a consequence, none of those methods is particularly relevant to that proposed in this thesis. Compared to those methods and those widely used to filter out unsolicited email traffic entering a network, the proposed one offers the same advantages as the behaviour-based detection method described in Chapter 4. These advantages are detailed in Section 4.1 and emerge from processing at the flow level (instead of at the application level) the

DNS traffic of user machines (instead of their email traffic) on the local name servers that are topologically near possibly or actually infected user machines (instead of on any other network point). Apart from these advantages, the method has several additional interesting properties. One of them is that, as shown later in this chapter in the simulation evaluation section, it can contribute to limiting the illegitimate traffic infected user machines send to the Internet during the outbreaks of email worms. In addition, the method has two features that make its deployment appealing. The first of them is that it can work on each local name server independently without requiring any communication with other servers or services. This implies that it can be deployed incrementally, which is critical to the success of every new Internet security method, as it is not reasonable to expect that any such method will be fully and instantaneously deployed. The second feature is that it has low deployment cost since a local name server is already installed in almost all networks. One last attractive property of the method is that, as demonstrated in Section 6.5, it cannot be easily evaded by small changes in the code of email worms.

The proposed method is based on regulating the flow-level characteristics of the DNS response streams that local name servers return to user machines by utilising a traffic control mechanism. The mechanism controls for a certain time interval, called containment period hereafter, the rate of the DNS response streams, the minimum response time, or both. The response time is the time that elapses between when a local name server receives a DNS query and when it issues the corresponding response. Three traffic control mechanisms are studied in this work: rejecting DNS queries, delaying DNS responses and rate limiting DNS response streams. Rejecting queries by simply dropping them would result in an acute increase in the number of DNS queries that the local name server receives, since resolvers resend all queries not answered within a short period of time. Thereby, in the present study rejecting queries refers to the act of prohibiting access to the name resolution service by forging DNS responses in a way that prevents resolvers from repeating rejected queries. For reasons that will become clear in Sections 6.2 and 6.4, a method that involves rejecting queries is unlikely to be widely adopted and used. Hence, the main focus is on delaying responses and rate limiting response streams, which are more subtle traffic control mechanisms. Further, the method can be run in either of two operational modes: reactively or proactively. When it works in the former mode, the traffic control mechanism is applied only to the DNS response streams sent back to possibly or actually infected user machines. By contrast, when operating in the proactive mode, the method affects all the response streams.

Three approaches can be taken to demonstrate and evaluate the efficacy of a novel containment method: theoretical analysis, empirical analysis and computer simulation. Among them, computer simulation has been, by far, the most commonly used because it offers several advantages over the other two approaches. First, it does not require formulating an accurate and analytically-solvable mathematical

model of the propagation of the targeted worms and the impact of the method under test thereon, which is usually a difficult and often impossible task. Second, it does not involve building costly experimental environments and eliminates the need for analysing Internet worm traffic traces, which are difficult to collect and only rarely, if at all, made publicly available. Third, it makes it possible to explore the potential of a method at scales that cannot be replicated in laboratories, and thereby to gain insight into global effects that cannot be studied by means of empirical analysis (conventional experiments). Fourth, it facilitates examining a range of designs, performing extensive parameter studies and experimenting with mutations of known Internet worms that might appear in the future without any risk in a time and cost efficient manner. For all these reasons, computer simulation was also used for the purposes of the present study. It should be noted, however, that in developing any simulation model, a number of assumptions and simplifications are necessarily made. Therefore, the simulation results presented later in this chapter serve as an illustration of potential and provide qualitative, rather than quantitative, indicators of the effectiveness of the proposed method in the real world.

The principle of the proposed method is illustrated in Fig. 6.1. In the following three sections, the depicted elements of the method, the simulation model that was designed, implemented and used to examine its utility, and the results of the simulation experiments performed are presented. These sections are followed by an analysis regarding the capacity of the method to be effective in containing email worms in the long run. In particular, in Section 6.2 a detailed description of the three traffic control mechanisms and two operational modes that are considered in the framework of the present study is provided. In Section 6.3, the rationale for determining which type of computer simulation is the most appropriate for the purposes of the present study is explained, and the design of the three basic components of the simulation model is described. These components are the Internet-scale email network on which email worms spread and the outgoing legitimate and illegitimate email and DNS traffic of both non-infected and infected user machines during the outbreak of an email worm. In Section 6.4, simulation results that show the utility of the method in slowing down the Internet-scale epidemics of email worms and limiting the illegitimate traffic sent out by infected user machines are reported. In Section 6.5, several approaches worm writers might take to evade containment are anticipated and their potential effectiveness is evaluated.

6.2 Traffic Control Mechanisms

The query-response exchange between user machines and local name servers is based primarily on the User Datagram Protocol (UDP), and the latter provide best effort service (delivery of DNS responses) to the former. This implies that a local name server sends back DNS responses to each user machine with a variable rate and

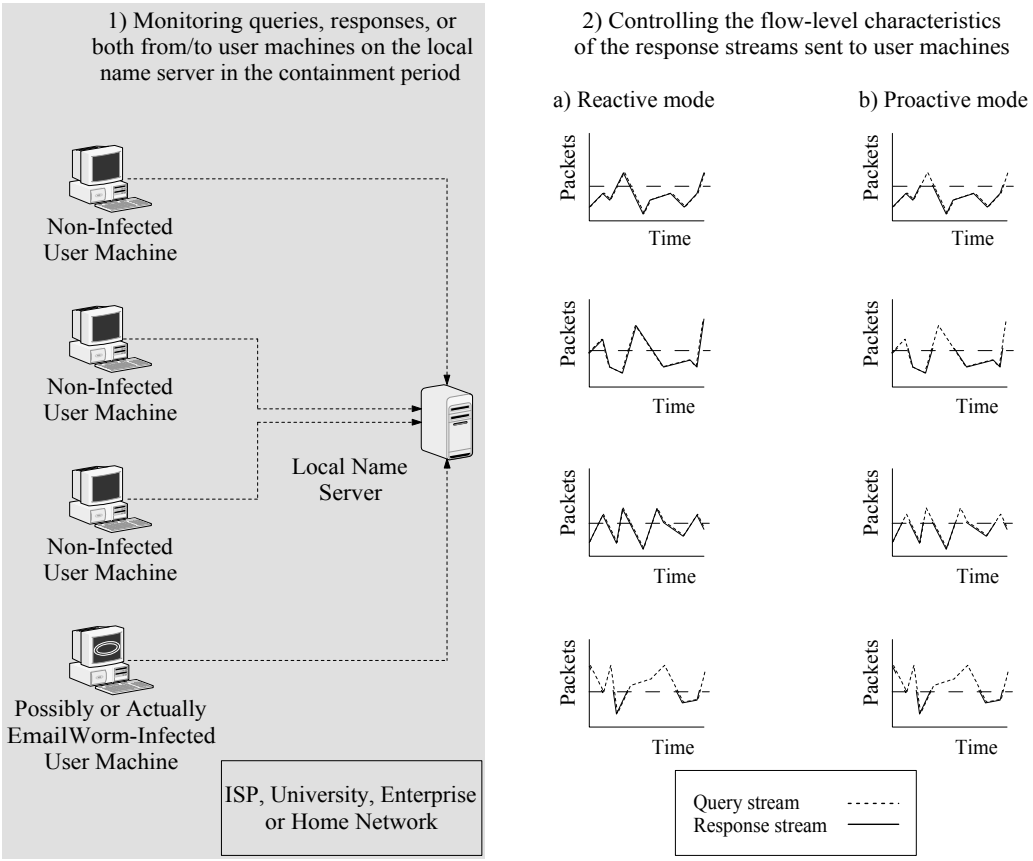

Figure 6.1: The proposed method operates on the DNS responses that a local name server sends back to the user machines within the containment period (Step 1). Using a traffic control mechanism, it regulates the average rate of the response streams, the minimum response times, or both. The efficacy of the method with three such mechanisms is explored: rejecting queries, delaying responses and rate limiting response streams. Moreover, two operational modes for the method are studied: reactive and proactive. In the former mode, the traffic control mechanism affects the DNS response streams sent to possibly or actually infected user machines (Step 2a); whereas, in the latter, all the response streams (Step 2b).

response times that are determined by the traffic load and the time it takes it to resolve each DNS query. By means of a traffic control mechanism, the proposed method restricts the average response rate, the minimum response times, or both, and as a consequence it essentially eliminates the best effort delivery of DNS responses. As previously stated, the potential effectiveness of three traffic control mechanisms was explored for the purposes of the present study: rejecting queries, delaying responses and rate limiting response streams. These three mechanisms as well as the effect they produce on the flow-level characteristics of a DNS response stream are detailed in the following paragraphs. In addition, the containment efficacy that can

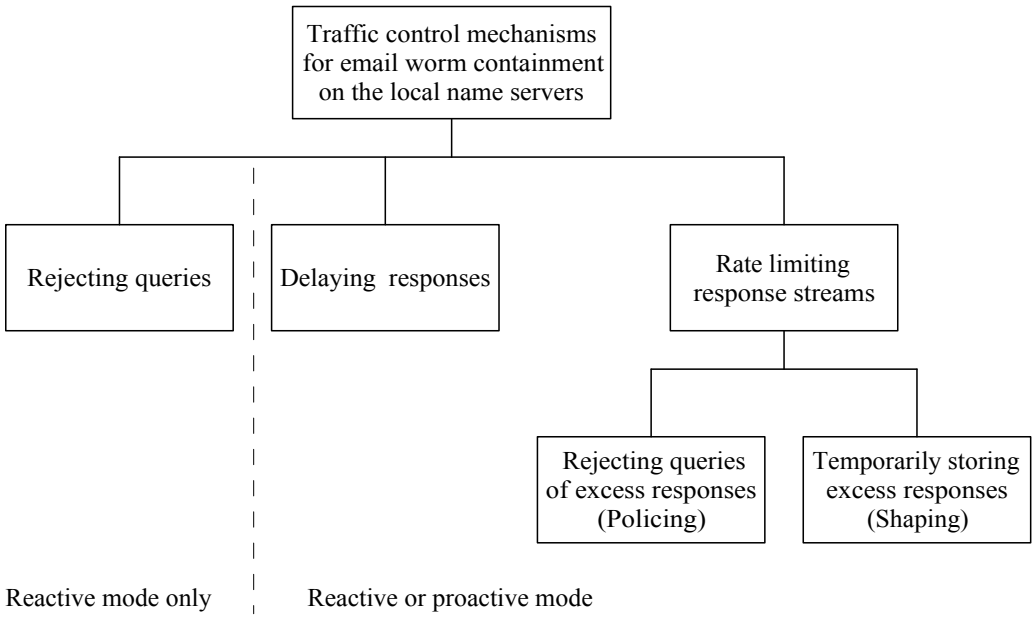

Figure 6.2: The traffic control mechanisms considered in this work to contain email worms by regulating the flow-level characteristics of the response streams that a local name server sends back to user machines. Except for rejecting queries the potential containment efficacy of using each of the mechanisms both reactively and proactively is studied. In the reactive mode, the traffic control mechanism is applied to the response streams of possibly or actually infected user machines, whereas in the reactive mode to all the response streams.

be achieved by running the method both reactively and proactively is examined. In this context, reactive refers to regulating the DNS response streams that are returned to possibly or actually infected user machines only, whereas proactive to regulating all the response streams independent of whether the requesting user machines are infected or not. The rationale for considering the proactive mode is that the method can only take effect after an infected user machine has already generated some DNS queries; hence, it is highly likely that infected user machines will succeed in infecting a number of user machines in their network before reaction is possible. Thereby, in the proactive mode, the local name server essentially switches to an alarm state to maximise the containment effect when a suspiciously behaving user machine is detected in its network. An overview of the traffic control mechanisms described in what follows is given in Fig. 6.2, and for each of them the expected response stream for a sample query stream is shown in Fig. 6.3.

Rejecting the queries that a local name server receives from a user machine within the containment period refers to the act of simply replying to every DNS query with a dummy response so as to prevent the stub resolver that runs on the user machine from reissuing it. Despite its simplicity, rejecting queries is potentially

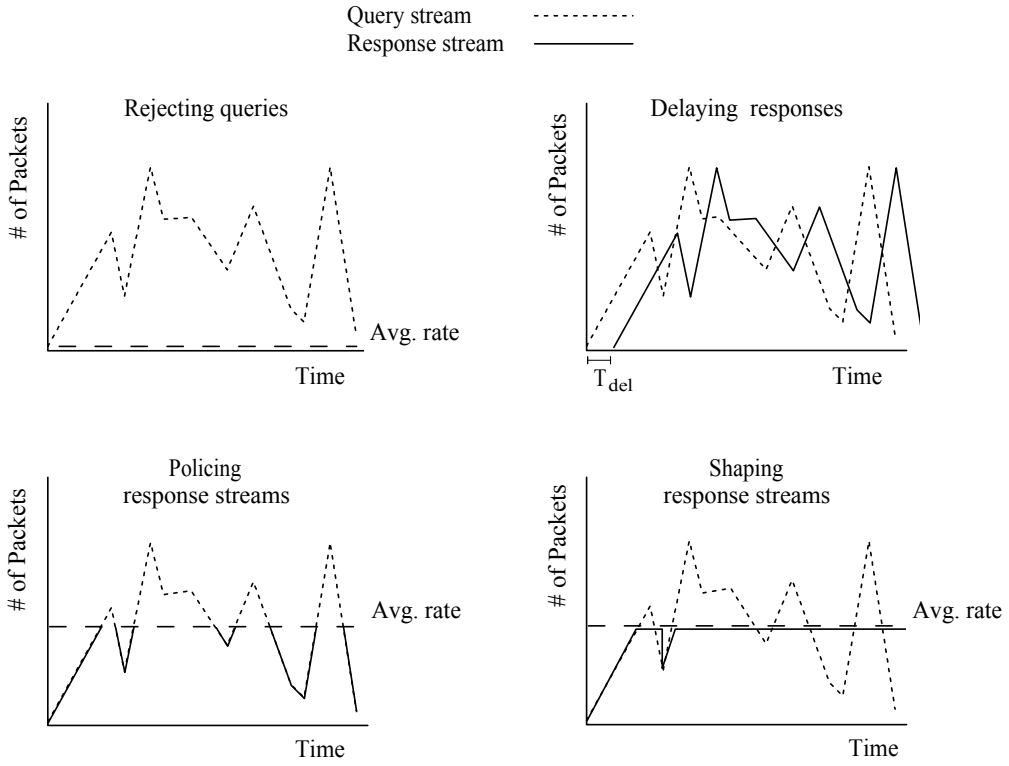

Figure 6.3: The response streams that a local name server sends to a user machine, when the traffic control mechanisms considered in this study are applied to a sample query stream. The dotted and solid lines represent the query and response streams, respectively, and T_{del} denotes the minimum response time, which is relevant only to delaying responses. Rejecting queries (upper left plot) sets the average rate of (valid) responses (Avg. rate) to zero. Delaying responses (upper right plot) affects the minimum response time; whereas, policing and shaping response streams (lower left and right plots) affect the average response rate.

the most effective of the three traffic control mechanisms investigated in this study in reducing the propagation speed of email worms and the degree to which infected user machines pollute the Internet. This is grounded in the fact that it can stop infected user machines from sending illegitimate traffic to the Internet completely because it essentially sets the average rate of (valid) responses to zero. However, an Internet security method that is based on rejecting all queries is highly unlikely to gain wide acceptance and use. The reason for this is that apart from reducing the outgoing illegitimate traffic of infected user machines, it would also severely degrade the quality of Internet applications end users experience. This is because most Internet applications that run on user machines depend on the RRs contained in the responses they receive from the local name servers to work. Thereby, it is mainly for completeness' sake that the efficacy of rejecting queries is examined later in this

chapter in the simulation evaluation section. Nevertheless, for this traffic control mechanism, executing the proposed method in proactive mode is not considered since applying this containment strategy would lead to loss of Internet connectivity to all user machines in the monitored network.

Delaying the responses that a local name server returns to a user machine within the containment period, involves sending out every DNS response only after, at least, a certain time period has elapsed since the reception of the corresponding query. As previously mentioned, this time period is referred to as minimum response time in this thesis, and it is denoted by T_{del} hereafter. The rationale behind defining T_{del} based on the time when a local name server receives a DNS query instead of when it is ready to send out the respective response is to avoid penalising DNS queries that need more than the average resolution time to be processed, such as recursive queries. In contrast to rejecting queries, delaying responses does not result in restricting the access of user machines to the name resolution service altogether but in eliminating the best effort delivery of DNS responses in a subtle manner. In more detail, if a DNS query is resolved in time less than T_{del}, the response is not released immediately but temporarily stored and forwarded to the requesting user machine after T_{del} passes; otherwise, it is emitted directly. Thereby, DNS responses are sent to user machines with an unspecified variable rate, whose average value over the containment period is implicitly determined by the corresponding query rate and T_{del}. The duration of T_{del} has to be carefully selected because delaying responses can negatively impact on the performance of most Internet applications that run on user machines. As a final remark, it is worth noting that applying this traffic control mechanism was inspired by tarpitting [77], a method that combats spam by delaying to forward emails on the outgoing email servers.

Rate limiting the response stream that a local name server returns to a user machine within the containment period, can be thought of as being the orthogonal traffic control mechanism to delaying responses. In more detail, it also results in eliminating the best effort delivery of DNS responses to user machines and not in restricting their access to the name resolution service, like rejecting queries does. However, in contrast to delaying responses, it directly affects the average rate of the DNS response streams instead of the minimum response times. Rate limiting is based on the token bucket model to regulate the average rate of a DNS response stream. According to this model, tokens are added into a bucket of limited size at a certain rate; let B_s denote the size of the bucket, and B_r and B_u be the number of DNS responses and the unit of time that define the rate. If n responses are to be sent back to a user machine in time B_u and at least n tokens exist in the bucket, n tokens are removed from the bucket and all n responses are immediately emitted. By contrast, if $k < n$ tokens are available, only the k out of the n responses whose respective queries were received first are sent out. The remaining $n - k$ responses can be treated in various ways. In the framework of the present study, two ways

were taken into account: queries that corresponded to excess responses were rejected or excess responses were temporarily stored and sent out in a following B_u, when sufficient tokens had been accumulated in the bucket. In the routing literature and the remainder of this chapter, these two variations of this traffic control mechanism are referred to as policing and shaping, respectively [35].

6.3 Simulation Model Design

Every study based on computer simulation consists of three phases: designing a model of the system to be investigated, implementing the model as a computer program and executing the program to produce the output of interest (simulation results). The focus in the remainder of this section is solely on the first phase; whereas, the other two phases are covered later in this chapter in the simulation evaluation section. There are a number of decisions that have to be made when designing a simulation model. Among them, the first and perhaps the most important one is choosing which type of simulation is the most suitable for modelling the system under investigation. Once the best-suited simulation type has been determined, the next step involves modelling the components of the system that are related to the problem in hand. In the following two paragraphs, a brief review of the existing simulation types is provided and the one that was chosen to explore the impact of the proposed method on the Internet-scale propagation of email worms and the illegitimate traffic that user machines generate is highlighted. Then, the three subsections that follow describe in detail the three basic components of the simulation model that was implemented and used for the purposes of the present study. These components are the set of user machines that email worms target for infection, called Internet-scale email network in this thesis, and the legitimate and illegitimate email and DNS traffic that these user machines generate during the outbreak of an email worm.

A computer simulation is characterised as static or dynamic, depending on whether the system under investigation persists unaltered or changes and evolves through time. Since containment methods by definition target slowing down the Internet-scale epidemics of email worms, dynamic simulation appears as the natural choice for exploring the effectiveness of the proposed method. In a dynamic simulation, time is the major independent variable of the simulation model and all other variables are functions thereof; hence, they are referred to as dependent variables hereafter. Based on how the values of the dependent variables change as time progresses, a dynamic simulation is further characterised as continuous or discrete-event. In a continuous simulation, the values of the dependent variables change continuously over time. By contrast, in a discrete-event simulation, changes, referred to as events, occur only at specific points in time. Continuous simulation involves formulating a mathematical model (typically, a system of differential equations) of the system's behaviour that is difficult or impossible to solve analytically and resorting to a

computer to obtain numerical solutions instead. For the needs of the present study, discrete-event simulation was used because developing an accurate mathematical model that describes the Internet-scale propagation of email worms as time evolves presents some, so far, insurmountable challenges [221].

A discrete-event simulation is further characterised as time-stepped or event-stepped, depending on how the simulated time advances. In a time-stepped simulation, time is sequentially incremented by a fixed time step, which corresponds to a physical time interval whose length is determined according to the problem in hand during the design phase of the simulation model. At each time step, the points in time at which each pending event is supposed to occur are examined and the actions associated with the events scheduled for the time step under consideration are performed. Actions of events that are scheduled to occur at the same time step usually take place simultaneously and it is often assumed that they do not influence each other. In an event-stepped simulation, time advances from one event to another in irregular increments of time. When an event has been processed, the occurrence time point of the next event in the simulation is computed and the time is updated to that time point. In the vast majority of studies that are concerned with modelling the propagation of Internet worms and exploring the effectiveness of containment methods (see, for instance, [32, 96, 219, 221] and [61, 130, 186, 203], respectively), time-stepped simulation serves as basis for experimental explorations. Although some early simulation results of the proposed method that were produced by using event-stepped simulation have been published in [PP7], the analysis presented in this thesis is also based on a time-stepped simulation.

6.3.1 Email Network

The study of email networks has attracted much research interest in recent years, and as a result several models have been proposed to describe their topology [134, 42, 221, 203] and temporal changes [188]. In these studies, email networks are modelled as graphs $G = (V, E)$, where each vertex u represents an email user and an edge from vertex i to vertex j implies that the email user i has the email address of the email user j in its address book. $|V|$ denotes the number of email users in the email network and the degree of vertex i, which is defined as the number of edges that are connected to the vertex, represents the email social network of the email user i. The email social network of an email user is the set of email addresses that are stored in its address book. Similarly to the studies listed above, the Internet-scale email network on which email worms spread was also modelled as a graph for the needs of the present study and, without loss of generality, it was assumed to remain unaltered in time. However, several refinements to the existing models had to be made that were necessary to study the Internet-scale propagation of the typical email worm described in Section 2.3.2. In what follows, the findings of the studies that focus on

modelling the topology of email networks are summarised, and the model that was used within the framework of the present study is analysed.

Newman et al. [134] and Ebel et al. [42] explore the topology of university-scale email networks by mining email-related data. Specifically, Newman et al. looks at the topology of a graph produced by examining the address book entries of the email users on a university network at one point in time. That study proposes that it is possible to accurately model the email network as a mixed graph (a graph with directed and undirected edges), whose distribution of vertex degrees decays faster than the power-law degree distribution of other real networks [23]. In addition, that study evaluates the reciprocity of the email network, which is the proportion of directed over the total number of edges. Ebel et al. analyses a graph constructed by inspecting the log file fragment of the email server in a university network written over a certain period of time. Similarly to Newman et al., that study suggests that the email network can be represented as a directed graph. It is noteworthy, however, that the authors of that study conclude that the email network has to be modelled as an undirected scale-free graph when the focus is on studying the Internet-scale epidemics of email worms. A scale-free graph is characterised by power-law distribution of its vertex degrees and has some highly-connected vertices that are central to the graph and keep the graph connected.

Zou et al. [221] and Xiong [203] propose containment methods for email worms and examine the topology of the Internet-scale email network on which email worms spread. In connection with this, their findings are particularly relevant to the present study. Zou et al. suggests that the vertex degrees of the graph representing this email network are heavy-tailed distributed. In that study, due to the absence of a software for generating other graphs whose vertex degrees are heavy-tailed distributed, the Internet-scale email network is modelled as an undirected power-law graph (the vertex degrees of power-law graphs are heavy-tailed distributed). Xiong argues that the average vertex degree in Zou et al. is underestimated because email worms search for email addresses also in locations other than the address book on the user machines they infect. In that study and what follows in this thesis, these additional email addresses are referred to as external to distinguish them from the internal email addresses, which are those stored in the address book. Moreover, in that study, the external email addresses on the machine of an email user i are modelled by adding to vertex i a set of random edges with cardinality obtained from a random variable uniformly distributed over the integers in the interval $[0, D_i]$; D_i denotes the degree of vertex i, which represents the number of internal email addresses. Thereby, the email network model used in Xiong is an undirected power-law graph enhanced with the edges that correspond to the external email addresses. In both Zou et al. and Xiong, the effectiveness of the proposed containment methods is also assessed modelling the Internet-scale email network as a random graph.

Despite the correction to the average vertex degree of the Internet-scale email

network model of Zou et al. that is proposed in Xiong, two further refinements were needed to take into account the ability of email worms to discover email addresses by guessing and scanning Web pages. The guessing efficiency of an email worm depends on its implementation, and thus differs from one email worm to another. A guessing attempt results in either the email address of an email user or an unused email address. The former result is called a correct guess and the latter a false guess hereafter. For the needs of the present study, the number of correct and false guesses of an email worm that infects the user machine of an email user i was taken from two random variables uniformly distributed over the integers in the intervals $[0, W_{cg} \times D_i]$ and $[0, W_{fg} \times D_i]$, respectively; W_{cg} and W_{fg} were introduced to control the correct and false guesses and determine the virulence of the email worm. In particular, large values of W_{cg} and small values of W_{fg} were used to model a highly-virulent email worm that does not waste much time sending infectious emails to (many) unused email addresses. Further, to take into account the email addresses that email worms find by scanning Web pages, a weighting coefficient, denoted by W_{ex}, was introduced in the model of external email addresses of Xiong. In connection with this, the number of external email addresses was obtained from a random variable uniformly distributed over the integers in the interval $[0, W_{ex} \times D_i]$.

In light of the discussion above, for the purposes of the present study the Internet-scale email network on which email worms propagate was modelled as an undirected graph that was assumed to remain unaltered throughout the simulation. This graph was generated by adding to every vertex i of a basic graph three sets of edges with cardinalities in $\{0, \ldots, W_{ex} \times D_i\}$, $\{0, \ldots, W_{cg} \times D_i\}$ and $\{0, \ldots, W_{fg} \times D_i\}$. A power-law graph was used as the basic graph to produce most of the simulation results that are reported in Sections 6.4 and 6.5; however, similarly to Zou et al. and Xiong, the containment efficacy of the proposed method was also evaluated using a random graph as basic graph. The edges in the first and second sets were randomly chosen, modelled the external email addresses and correct guesses, respectively, and connected vertices representing email users. The edges in the third set modelled false guesses and connected vertices representing email users to an empty vertex, which was set to be the recipient of all the infectious emails sent to unused email addresses. The edges attached to vertex i (edges of the basic graph and the union of the three sets) composed the target list that an email worm compiled once it infected the user machine of the email user i. Intuitively, for the vast majority of email users, more than one edges in their target list corresponded to false guesses. The order at which email worms selected edges from the target list was random.

6.3.2 Legitimate Traffic

The legitimate email and DNS traffic that a user machine sends to the Internet was also modelled for the purposes of the present study. In the real world, this traffic is

generated, implicitly or explicitly, by the actions that end users carry out on their machines. Two points should be mentioned with respect to the actions that produce the legitimate outgoing email traffic. The first of them is that, as will become clear in what follows, these actions depend to a large extent on the incoming email traffic. The incoming emails are either legitimate coming from either non-infected or infected user machines or infectious coming from infected user machines. The second point is that these actions determine not only the characteristics of the legitimate outgoing email traffic but also if and when the user machine becomes infected. Sending out legitimate emails is not preceded by querying the local name server. The reason for this is that the legitimate emails are delivered to the outgoing email server, which is responsible to perform all the necessary DNS resolutions to relay them toward their destination. Thereby, the legitimate outgoing email and DNS traffic of a user machine were modelled independent from one another. In particular, as detailed hereafter, the former was modelled indirectly through modelling the actions of an email user, whereas the latter in a straightforward manner.

At each time step of the simulation, an email user i might perform one of the four following actions: check its inbox for new emails, read unread emails, reply to read emails or send an email. Four dependent variables were used to model these actions: the checking time, reading probability, reaction pattern and sending time; these parameters are denoted by T^i_{ch}, P_i, R_i and T^i_s, respectively, in what follows. The checking and sending times represent the time that passes between two successive email checking and sending actions. If at an email checking time step one or more new emails are stored in the inbox of an email user, they are processed (read or ignored) one by one at consecutive time steps until no unprocessed emails are left. The reading probability determines if the email user reads or ignores a new email stored in its inbox. Reading emails, involves opening their attachment files and clicking the links that they contain. The reaction pattern specifies how the email user reacts after reading an unread email. The following three possibilities were considered: the email user might reply to the sender of the email only, or to the sender and additional email users, or not reply at all. These actions are carried out without any delay once the email is read. At an email sending time step, the email user might send an email to either one or many email users of its email social network. In the case of a send-to-many email, the email sending action is completed after an email has been sent to all the recipients. Reading and sending one email, takes one time step; whereas, the email replying action was modelled as a batch process (responses are sent to all recipients at one time step).

For the needs of the present study, checking and sending emails were modelled as independent Poisson processes. This implies that the checking and sending times of each email user are exponentially distributed. Furthermore, it was assumed that the email checking and sending behaviour of each email user is independent of that of the other email users. Thereby, similarly to Zou et al. [221] and Xiong [203], the

mean checking and sending times of an email user i, denoted by $E[T_{ch}^i]$ and $E[T_s^i]$ hereafter, are obtained from two normally distributed random variables $N(\mu_{ch}, \sigma_{ch}^2)$ and $N(\mu_s, \sigma_s^2)$; $E[T_{ch}^i]$ and $E[T_s^i]$ are set to μ_{ch} and μ_s, respectively, if the obtained values are negative. The number of recipients in the case of a send-to-many email and the number of additional recipients in the case of a reply-to-many response, denoted by N_{sr} and N_{rr} in what follows, are taken from a normally distributed random variable $N(\mu_r, {\sigma_r}^2)$; N_{sr} and N_{rr} are set to one if the selected values are negative. In both cases, the recipients were picked at random from the email social network of the email user. The reading probability of each email user was considered to be a constant, because in practice it is determined by the email user's awareness of Internet security risks. The value of this constant is drawn from a normally distributed random variable $N(\mu_p, {\sigma_p}^2)$. The reaction pattern is obtained from a uniformly distributed random variable $U(a_{rp}, b_{rp})$.

For many years, the Poisson process was regarded as the most appropriate model to describe the characteristics of the traffic that a single or a set of Internet-connected machines produce. As previously mentioned, according to this model, packets are generated at random and the times between two successive packets, referred to as interarrival times, are exponentially distributed. The pioneering work of Leland et al. [106], however, shows that the traffic of local area networks is self-similar. As a consequence, the interarrival times are heavy-tailed distributed, and thus cannot be accurately modelled using an exponential distribution. Paxson et al. [141] shows that this finding also applies for traffic captured on wide area networks. The results of these two studies and those of others that build on them (see, for instance, [49, 105]) have led to a tendency within the research community to disregard the Poisson-based models in favour of others that account for self-similarity, such as models based on the Pareto process [59]. Nevertheless, a detailed review of these studies and several recent ones (see, for instance, [89]) reveals that whether traffic is self-similar or not, and thereby better modelled with processes other than the Poisson, depends heavily, among others, on its type, the time period it was captured and the observation scale. In the absence of a study that provides insight into the characteristics of the legitimate DNS traffic that a user machine generates, this traffic was modelled using a Poisson process with rate λ_{lq} and investigating whether it is self-similar or not was left for future work. In connection with this, each non-infected user machine was assumed to generate DNS queries at random and the times between two successive queries to be exponentially distributed with mean $\frac{1}{\lambda_{lq}}$.

6.3.3 Illegitimate Traffic

The outgoing email and DNS traffic of an infected user machine is the superposition of the legitimate traffic, discussed previously, and the illegitimate email and DNS traffic that the email worm generates. For the purposes of the present study, the

illegitimate traffic was modelled indirectly through modelling the operations that email worms carry out when they are in the name lookup subphase (DNS traffic) and propagation phase (email traffic). Moreover, the end user and the email worm were considered to be independent traffic sources. There are two reasons that justify taking this approach. The first of them is that once email worms compromise a user machine, they perform their operations without interacting with the end user in order to avoid raising suspicion. The second reason is that the actions of the end user are not affected by the operations of the email worms because the end user is normally not aware of the infection. In contrast to the legitimate outgoing email and DNS traffic, the email and DNS traffic that email worms produce are correlated, and thus it is not possible to model them separately. The reason for this is that, as described in Section 2.3.2, in the name lookup subphase, the email worms query the local name server for the IP addresses of the email servers that are associated with the email addresses targeted for infection during the propagation phase.

After entering the name lookup subphase, an email worm traverses one by one the email addresses (edges) of the target list it has compiled and issues for each of them a set of DNS queries to the local name server. As mentioned in Section 2.3.2, each of these sets consists of more DNS queries than those needed to resolve the IP address of one email server associated with the email address in hand. In the simulation model, the number of DNS queries in each set, denoted by N_q, is drawn from a normally distributed random variable $N(\mu_q, {\sigma_q}^2)$. A randomly-chosen subset of these queries with cardinality N_{sq} represents those actually needed; N_{sq} was considered to be a constant. Once these queries are responded to, the email worm sends out an infectious email. If the edge is not a false guess, this email is queued in the inbox of the corresponding email user; otherwise, it is forwarded to the empty vertex and no inbox is updated. Once the infectious email is sent, the email worm picks another edge from the target list and repeats the same procedure. Repeating this procedure for every edge in the target list, is referred to as transmission process hereafter. Based on the same rationale as in the case of legitimate DNS traffic discussed earlier, the illegitimate DNS traffic produced during the execution of the transmission process was modelled using a Poisson process with rate λ_{wq}.

Three more points with respect to the email worm model used in this work are stated below. The first point is that the model takes into account the time between when an email worm leaves the penetration phase and when it enters the name lookup subphase. During this time, the email worm harvests and populates the target list with a few, or sometimes all, the email addresses that are targeted for infection in the propagation phase. This time varies from one email worm (instance) to another and typically ranges from a few seconds to a few minutes. However, without loss of generality, for the needs of the present study it was considered as a constant, denoted by T_{inf}. The second point is that the model is suitable for studying both non-reinfecting and reinfecting email worms. A non-reinfecting email worm performs

the transmission process only once and then becomes inactive; whereas, a reinfecting one executes the transmission process periodically remaining inactive for a period of time T_{ri} between two successive executions; T_{ri} was taken to be a constant. When the email worm is inactive, the infected user machine does not generate illegitimate traffic. The third point is that the model is intended to study the efficacy of the proposed method against zero-day email worms; thereby, it was assumed that no prevention or treatment method for immunising user machines exists. This implies that once a user machine gets infected it remains infected thereafter.

6.4 Simulation Evaluation

This section presents the results of the simulation evaluation of the proposed method, and it is organised into two parts. Each part is devoted to one of the two dimensions that, as mentioned in Section 5.2, need to be considered when designing and assessing the effectiveness of any containment method and are not fixed by the design of the proposed method. In particular, the first part focuses on the containment strategy, whereas the second on the reaction time, which is denoted by T_{rct} in what follows. The purpose of the first part is twofold. First, it shows that, regardless of the graph used to model and thereby the topology of the Internet-scale email network on which email worms spread, the method has the potential to slow down the Internet-scale epidemics of email worms and restrict the degree to which infected user machines pollute the Internet. In addition, it demonstrates that the method can achieve this without significantly affecting the legitimate traffic of user machines and, as a consequence, the quality of Internet applications end users experience. Second, it reveals which traffic control mechanisms and in which operational mode make the method perform best. This allows reducing the number of choices regarding the combinations of traffic control mechanisms and operational modes under investigation, and concentrate only on the most appropriate for the method. The most appropriate combinations should produce the maximum effect in terms of limiting the spreading speed of email worms, the illegitimate traffic infected user machines send to the Internet, or both, while having a minimal negative impact on legitimate traffic. Taking into account these combinations only, the goal of the second part is to analyse the influence of the reaction time on the effectiveness of the method.

6.4.1 Simulation Setup

In a time-stepped simulation, the time step can represent any physical time interval, whose length is important only because it determines the accuracy of the simulation with respect to time. In more detail, based on the fact that in a simulation of this type, changes in the values of the dependent variables (events) occur right at discrete time steps, the accuracy increases as the length of the time interval that the time

step represents decreases. For the purposes of the present study, the time step was selected to correspond to roughly one second. This was a rather arbitrary decision made with the aim of keeping the granularity small, given that email worms operate at time scales ranging from a few seconds to a few hours. The underlying goal in doing so was to ensure that the produced results will not deviate significantly from those that would have been generated with a continuous simulation. In order to have the required flexibility to develop the components of the simulation model described in Section 6.3, the general purpose computer language C++ was employed instead of a specialised simulation language. C++ is a fast, flexible, object-oriented language that is freely available, well-documented and highly portable. In connection with this, the values of the stochastic variables of the model were obtained by means of the random number generators provided by the GNU Scientific Library (GSL) [55]. Since most of the dependent variables are stochastic, $N = 100$ iterations were performed for each simulation experiment, and the outputs were averaged to get a set of reliable results. In each iteration, the simulation was started with two randomly-chosen user machines (vertices) becoming initially infected and run for 16 000 time steps.

Unless otherwise stated, the values of the parameter used to produce the simulation results reported in this chapter are given in this paragraph, and listed in Table 6.1 for the reader's convenience. These values are referred to as default values hereafter. The reaction and minimum response times, T_{rct} and T_{del}, were 248 and 5 (time steps), and the parameters of the token bucket model, B_s, B_r and B_u, were set to 2. The buffer associated with shaping response streams had space for only one DNS response. The undirected random and power law graphs had 100 000 vertices, average vertex degree 8, and conformed to the Erdös-Rényi [45] and the Generalized Linear Preference (GLP) [21] models, respectively. Similarly to Zou et al. [221], the power-law exponent of the power-law graph was 1.7. The coefficient W_{ex} was merged with W_{cg} and the ratio of the coefficients W_{cg} to W_{fg} was 0.2. To study the efficacy of the proposed method when applied proactively, the vertices were randomly organised into equally-sized groups of 1 000 members and each group represented a monitored network. The mean checking and sending times of an email user i, $E[T_{ch}^i]$ and $E[T_s^i]$, were drawn from the $N(1\,800, 900^2)$ and $N(1\,200, 1\,200^2)$. Its reading probability P_i was picked once from the $N(0.5, 0.2^2)$, whereas its reaction pattern R_i from the $U(0, 1)$ for each unread email independently; a value in $[0, 0.34)$, $[0.34, 0.67]$ and $(0.67, 1]$ corresponded to replying to the sender only, replying to the sender and additional email users, and not replying. Both the number of recipients for a send-to-many email N_{sr} and that of additional recipients for a reply-to-many response N_{rr} were obtained from the $N(1, 2^2)$. The ratio of the rates of legitimate to illegitimate queries, λ_{lq} to λ_{wq}, was 0.01, and the number of DNS queries an email worm does before sending an email N_q was taken from the $N(10, 3^2)$; $N_{sq} = 4$ of these queries were randomly chosen to represent those really needed. For the simulations reported in this thesis, only reinfecting email worms were considered,

Table 6.1: The parameters and their default values that were used to generate the simulation results of the proposed containment method, which are presented in the rest of this chapter.

Parameter	Value
Physical length of the time step	1 s
# of iterations per simulation experiment	100
Initially infected user machines	2
Simulation period per iteration (in time steps)	16 000
Reaction time (in time steps) (T_{rct})	248
Minimum response time (in time steps, delaying) (T_{del})	5
Size of the token bucket (B_s)	2
# of responses (average rate, rate limiting) (B_r)	2
Unit of time (average rate, rate limiting) (B_u)	2
Size of buffer (shaping response streams)	1
# of vertices (basic graph) ($\|V\|$)	100 000
# of vertices per monit. network (method runs proact.)	1 000
Average vertex degree (basic graph) ($\bar{D}$)	8
Power law exponent of the power-law graph (α)	1.7
Coefficients of add. email addresses (W_{ex},W_{cg} and W_{fg})	$W_{ex} = 0, \frac{W_{cg}}{W_{fg}} = 0.2$
Checking time of email user i (T^i_{ch})	$E[T_{ch}] \sim N(1\,800, 900^2)$
Sending time of email user i (T^i_s)	$E[T_s] \sim N(1\,200, 1\,200^2)$
Reading probability of email user i (P_i)	$P \sim N(0.5, 0.2^2)$
Reaction pattern of email user i (R_i)	$R \sim U(0, 1)$
# of recipients for send-to-many email (N_{sr})	$N_{sr} \sim N(1, 2^2)$
# of add. recipients for reply-to-many responses (N_{rr})	$N_{rr} \sim N(1, 2^2)$
# of email worm queries per email address (N_q)	$N_q \sim N(10, 3^2)$
# of necessary email worm queries per email address (N_{sq})	4
Legitimate and illegitimate query rates (λ_{lq} and λ_{wq})	$\frac{\lambda_{lq}}{\lambda_{wq}} = 0.01$
Time from penetration to transmission process (T_{inf})	0
Time between two executions of the transm. proc. (T_{ri})	0

and the length of the time intervals T_{inf} and T_{ri} were assumed to be negligible.

The numbers of infected user machines, legitimate and illegitimate emails, DNS queries and responses per time step are employed as metrics to demonstrate and assess the containment effectiveness of the proposed method. Notably, the way the values of these metrics change over time when the method was applied as compared to when email worms spread without restriction, referred to as free propagation hereafter, is more interesting than their absolute values. In particular, differences in the time course of the number of infected user machines of these two cases indicate if and to what extent the method can be effective in slowing down the Internet-scale

epidemics of email worms; whereas, differences in the time evolution of the numbers of legitimate and illegitimate emails, DNS queries and responses provide insight into two other important aspects. The first aspect relates to the potential utility of the method in limiting the amount of illegitimate traffic infected user machines send to the Internet. The second aspect concerns the degree to which the method is expected to affect the legitimate traffic that user machines generate and, as a consequence, the quality of Internet applications end users experience. To reduce the effect of outliers on the analysis carried out, the 5% trimmed mean over the values of each metric that were obtained from the 100 iterations of each simulation experiment was calculated for each time step and is considered in what follows. In addition, to enhance the readability of the figures given later in this chapter, the 16 000 trimmed means computed for each metric were downsampled by a factor of 160, and only the remaining 100 values were used to generate the plots presented.

6.4.2 Containment Strategy

To show that the proposed method has the potential to slow down the Internet-scale epidemics of email worms, the number of infected user machines was recorded at every time step of each simulation experiment. Fig. 6.4 illustrates the trend in the value of this metric over time for the free propagation of email worms and when the method runs reactively with each of the traffic control mechanisms considered in this work. The left and right plots correspond to the cases where the power-law and the random graph, respectively, were used to construct the graph modelling the Internet-scale email network on which email worms spread. The figure indicates that the method with rejecting queries or rate limiting response streams can reduce the propagation speed of email worms, whereas with delaying responses it cannot be effective. In more detail, the figure confirms that with this respect rejecting queries makes the method perform best, as with this traffic control mechanism the method even prevents email worms from reaching a significant number of vulnerable user machines. Given that delaying responses does not reject DNS queries and that shaping response streams rejects less DNS queries than policing, since it involves temporarily storing the excess DNS responses, the figure also uncovers that the efficacy of the method depends strongly on the amount of DNS queries it rejects. Thereby, delaying responses is excluded from the analysis presented hereafter.

To demonstrate that, apart from slowing down the Internet-scale propagation of email worms, the proposed method can also restrict the degree to which infected user machines pollute the Internet, the numbers of queries and emails generated at each time step of the simulation experiments were monitored. Fig. 6.5 depicts the time course of the ratios of the numbers of illegitimate to legitimate queries and illegitimate to legitimate emails for the free propagation of email worms and when the method works reactively with rejecting queries and rate limiting response streams.

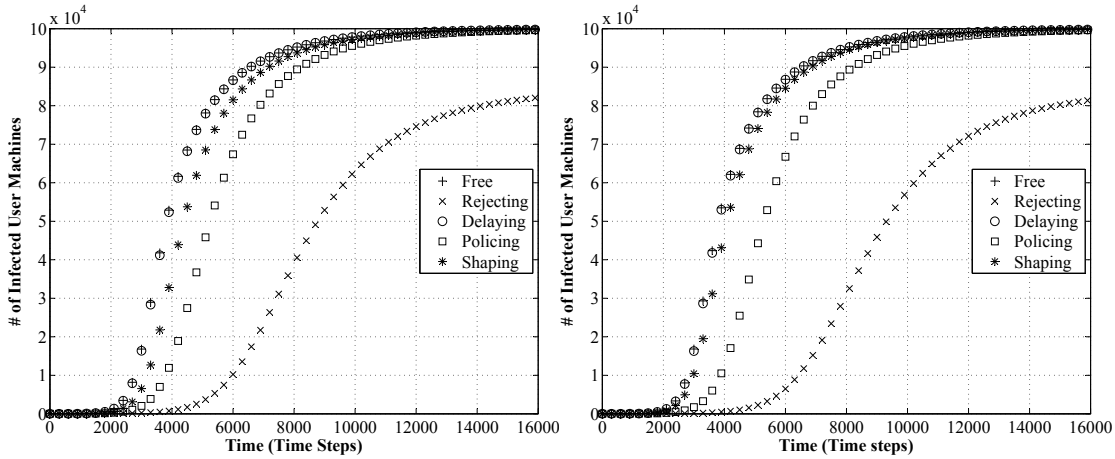

Figure 6.4: Trend in the number of infected user machines over time for the free propagation of email worms and the cases where the method is applied reactively with rejecting queries, delaying responses, policing response streams and shaping response streams when the power-law graph (left plot) and the random graph (right plot) are used as basic graph. When run reactively with rejecting queries, policing or shaping response streams, the method can be effective in slowing down the spread of email worms, whereas with delaying responses it cannot. Its effectiveness increases as the number of queries it rejects increases.

Again, in this figure the left and right plots correspond to using the power-law and the random graph, respectively, as basic graph. The figure reveals that the method with rejecting queries, policing or shaping response streams can be effective in limiting the proportion of illegitimate to legitimate traffic produced by user machines during the outbreaks of email worms. As the numbers of legitimate queries and emails do not vary significantly, neither over time nor from one simulation (experiment or iteration) to another, this implies that the method can reduce the amount of illegitimate traffic generated. With regard to the illegitimate queries, the reduction is linearly related to that in the number of infected user machines; whereas, the reduction that the method causes in the number of infectious emails is proportional to the amount of DNS queries it rejects. Thereby, with this respect the method performs best with rejecting queries and worst with shaping response streams.

To uncover that the proposed method can limit the propagation speed of email worms and the illegitimate traffic infected user machines send to the Internet without significantly influencing their legitimate traffic, the number of responses returned to the user machines at each time step of the simulation experiments was also recorded. Fig. 6.6 depicts the trend in the ratio of the numbers of legitimate responses to queries over time for the free propagation of email worms and when the method with rejecting queries and rate limiting response streams is applied reactively. The figure reveals that the method with policing or shaping response streams has the potential

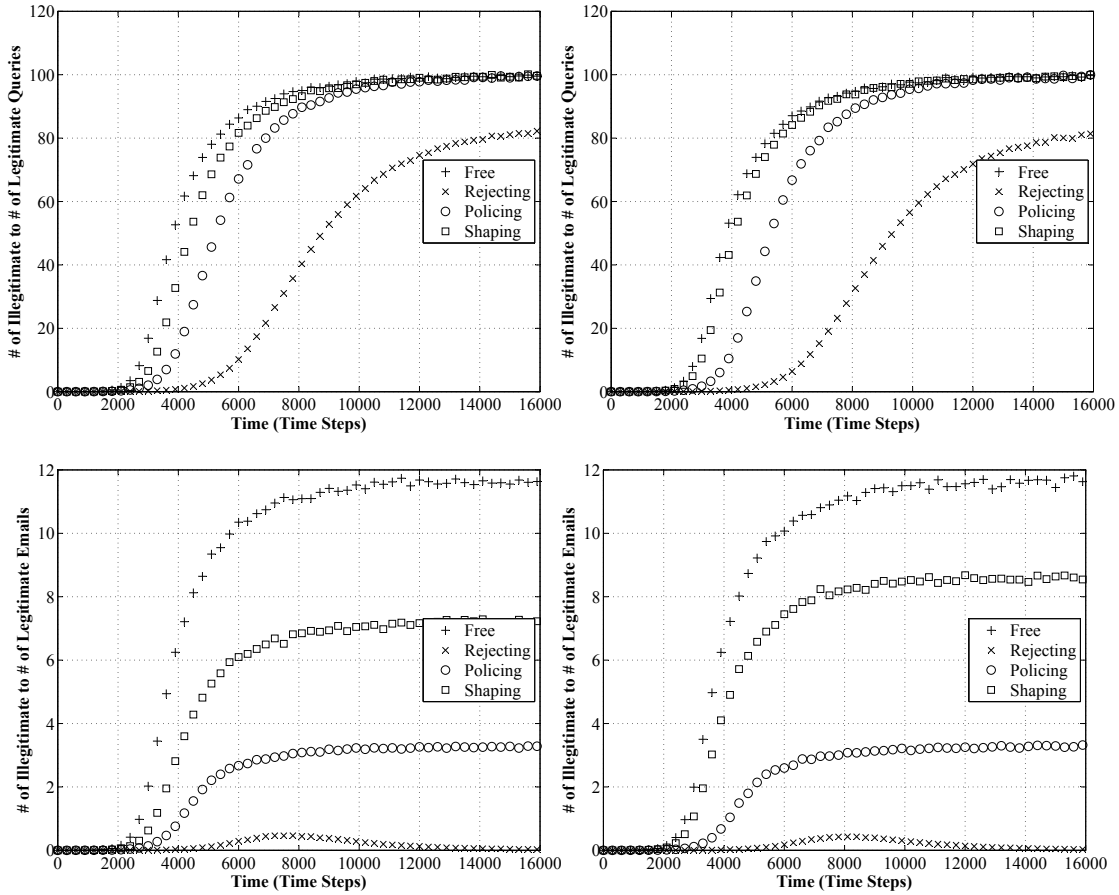

Figure 6.5: Trend in the ratios of the numbers of illegitimate to legitimate queries (upper plots) and illegitimate to legitimate emails (lower plots) over time for the free propagation of email worms and the cases where the method is applied reactively with rejecting queries, policing response streams and shaping response streams when the power-law graph (left plots) and the random graph (right plots) are used as basic graph. When run reactively with any of these traffic control mechanisms, the method can be effective in limiting the amount of illegitimate traffic infected user machines generate during the outbreaks of email worms. It performs best with rejecting queries and worst with shaping response streams.

to contain email worms even when it rejects only a small number of DNS queries, and thereby produces a limited negative impact on the quality of Internet applications end users experience. It also indicates that, as mentioned in Section 6.2, the method with rejecting queries severely affects the legitimate traffic of user machines, and as a consequence it is highly unlikely to meet with wide acceptance and use. In light of these findings, rejecting queries is not considered in the analysis presented in the remainder of this chapter. Furthermore, this figure as well as the other two

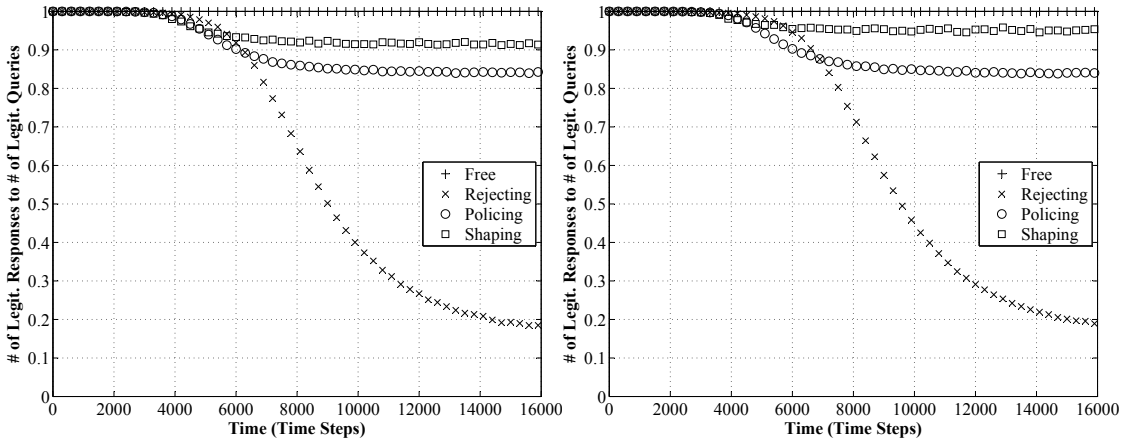

Figure 6.6: Trend in the ratio of the numbers of legitimate responses to queries over time for the free propagation of email worms and the cases where the method is applied reactively with rejecting queries, policing response streams and shaping response streams when the power-law graph (left plot) and the random graph (right plot) are used as basic graph. When run reactively with policing or shaping response streams, the method can be effective in limiting the propagation speed of email worms and the illegitimate traffic infected used machines generate without rejecting a significant amount of legitimate queries. By contrast, with rejecting queries, it is very effective but affects legitimate traffic severely.

described above show that the traffic control mechanisms make the method exhibit the same relative performance regardless of the graph used as basis to build the graph modelling the Internet-scale email network on which email worms spread. In connection with this, only simulation results that were produced using the power-law graph as basic graph are discussed hereafter.

The simulation results reported so far indicate how the proposed method can be expected to perform when employed reactively with each one of the traffic control mechanisms considered in this work. These discussed in what follows concern only the traffic control mechanisms that make the method perform best and show how effective the method can be with each of them when applied reactively compared to when applied proactively. Fig. 6.7 illustrates the time course of the number of infected user machines and the ratios of the numbers of illegitimate to legitimate queries, illegitimate to legitimate emails and legitimate responses to queries for the free propagation of email worms and when the method works reactively and proactively with rate limiting response streams. The plot of the number of infected user machines (upper left plot) suggests that using the method proactively, can yield superior results than when using it reactively in terms of limiting the Internet-scale propagation speed of email worms. This is because when run proactively, the method takes effect against most of the infected user machines without delay (reaction time near or equal to zero) after they become infected. The plots of the ratios of the

numbers of illegitimate to legitimate queries and emails (upper right and lower left plots) indicate that when running proactively, the method can also cause a greater reduction in the amount of illegitimate traffic infected user machines generate during the early stages of the Internet-scale epidemics of email worms. Furthermore, the plot of the ratio of the numbers of legitimate responses to queries (lower right plot) uncovers that when the method operates proactively, it can be significantly more effective than when operating reactively without rejecting more legitimate queries. Notably, this plot also reveals that it is possible to configure the method (by setting the token bucket parameters) in such a way that it does not at all affect the response streams of non-infected user machines when applied proactively.

The final decision about which is the most appropriate traffic control mechanism and operational mode combination for the proposed method can be made only on a case-by-case basis. This is because it depends heavily on the priority a network operator gives to three criteria. The first of them is the size of the effect that the network operator intends to produce on the propagation speed of email worms. The plot of the number of infected user machines of Fig. 6.7 (upper left plot) shows that with respect to this criterion, using the method proactively with policing response streams is clearly the optimal choice. The second criterion is the degree to which the network operator is free or willing to impede the legitimate traffic of user machines and, thereby, the quality of Internet applications end users experience. Fig. 6.6 in conjunction with the plot of the ratio of the numbers of legitimate responses to queries of Fig. 6.7 (lower right plot) indicate that the method with shaping response streams running proactively rejects less legitimate queries than with policing response streams either reactively or proactively. The third criterion is whether the network operator is reluctant to apply the method proactively or not. The plots of the number of infected user machines and the ratios of the numbers of illegitimate to legitimate queries and emails of Fig. 6.7 (upper left and right plots and lower left plot) reveal that policing response streams is the traffic control mechanism that makes the method perform best when used reactively. In light of this analysis, the simulation results that are discussed in the remainder of this chapter demonstrate the efficacy of the method only when it is applied with policing response streams reactively or proactively or shaping response streams proactively.

6.4.3 Reaction Time

The simulation results presented above suggest that the proposed method has the potential to slow down the Internet-scale spread of email worms and restrict the illegitimate traffic infected user machines generate without significantly affecting their legitimate traffic. However, they provide no insight into how the efficacy of the method is influenced by the amount of time that elapses before it takes effect. To fill this gap, the results of simulation experiments that were carried out with the

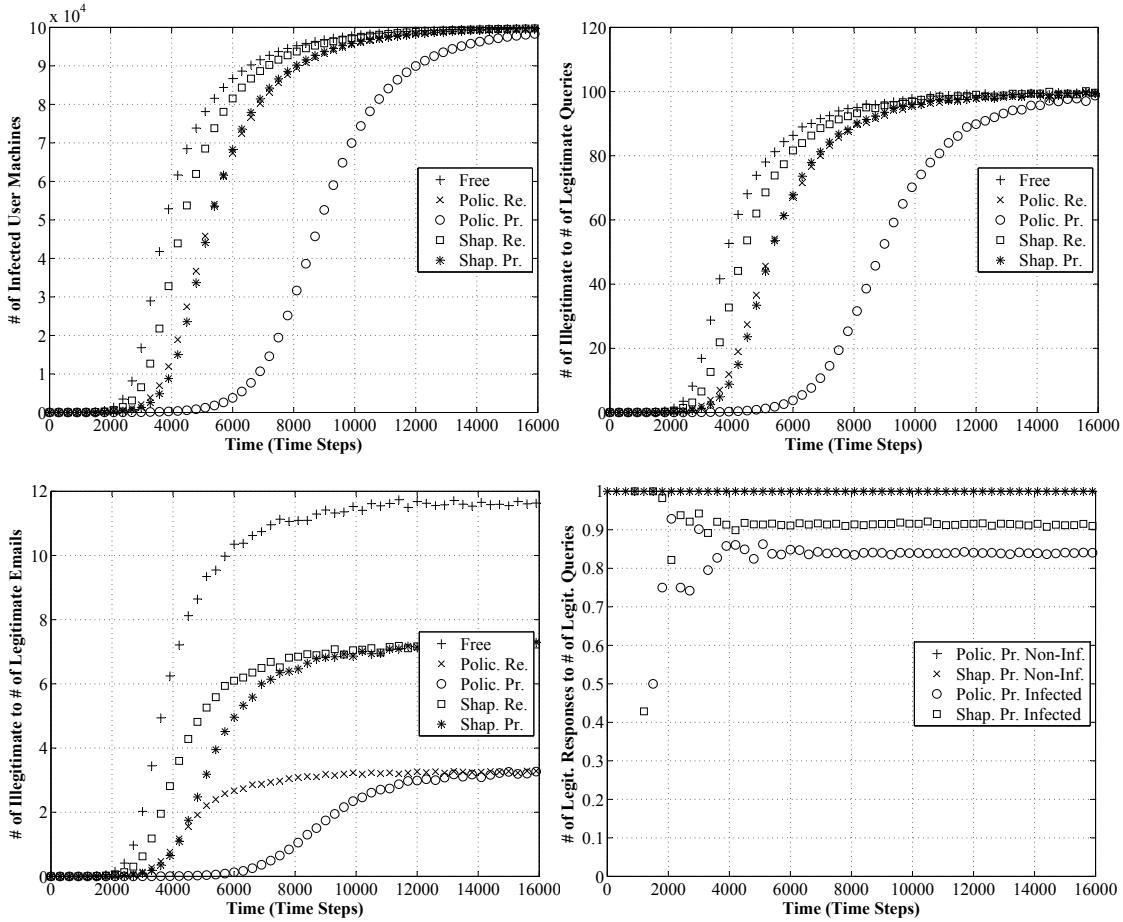

Figure 6.7: Trend in the number of infected user machines (upper left plot) and the ratios of the numbers of illegitimate to legitimate queries (upper right plot), illegitimate to legitimate emails (lower left plot) and legitimate responses to queries (lower right plot) over time for the free propagation of email worms and the cases where the method runs reactively and proactively with policing and shaping response streams. When used proactively, the method can be more effective than when used reactively, even when configured not to affect at all the traffic of non-infected user machines. Based on the strategy of each network operator, the most appropriate traffic control mechanism and operational mode combination is one of policing response streams reactively or proactively or shaping response streams proactively.

reaction time T_{rct} set to values different from its default one listed in Table 6.1 are discussed in this subsection. In this discussion, the length of the reaction time is expressed as a percentage of the amount of illegitimate traffic that an average user machine produces in the course of one execution of the transmission process when it is infected with an average email worm. The term average user machine is used here to refer to the machine of an email user i that is represented by a vertex (of the basic

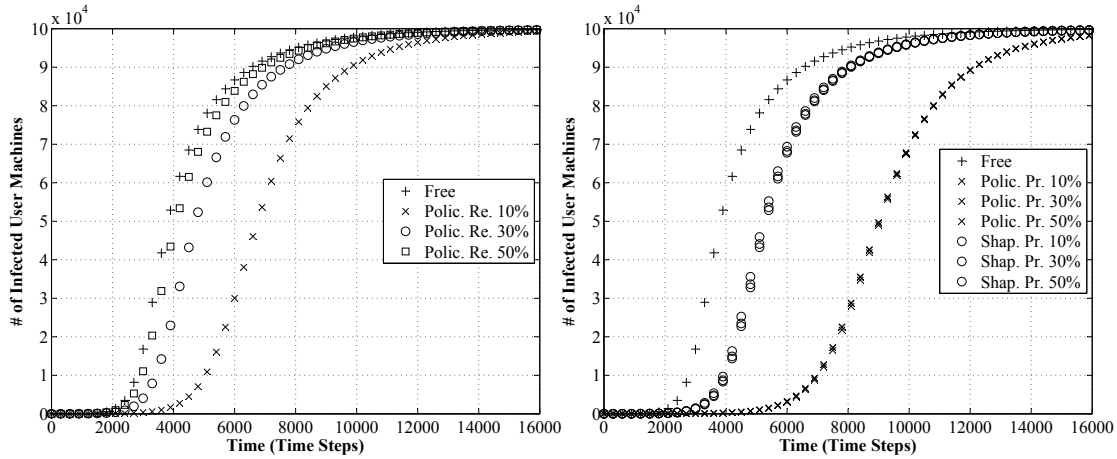

Figure 6.8: Trend in the number of infected user machines over time for the free propagation of email worms and the cases where the method is applied with policing response streams reactively (left plot) and policing and shaping response streams proactively (right plot) when allowing 10, 30 and 50% of the infectious emails produced during the first run of the transmission process of an average email worm on an average user machine to pass before it takes effect. When the method is applied reactively, its effectiveness in slowing down the epidemics of email worms depends on the reaction time. Regardless of the traffic control mechanism used, this dependency can be eliminated by applying the method proactively.

graph) with degree D_i equal to the average vertex degree of the basic graph $\bar{D}$. The term average email worm is used to describe an email worm that performs exactly $\frac{W_{cg} \times D_i}{2}$ and $\frac{W_{fg} \times D_i}{2}$ correct and false guesses, respectively, during one execution of the transmission process. It should be noted that the default value of T_{rct} given in Table 6.1 was chosen to represent the time it takes an average email worm to send one infectious email to 20% of the email addresses in the target list it compiles once it infects an average user machine. The simulation results reported below correspond to setting T_{rct} accordingly so as to allow 10, 30 and 50% of the infectious emails produced during the first run of the transmission process of an average email worm to pass before the method becomes operative.

Fig. 6.8, Fig. 6.9 and Fig. 6.10 depict the time course of the number of infected user machines and the ratios of the numbers of illegitimate to legitimate queries, illegitimate to legitimate emails and legitimate responses to queries for the free propagation of email worms and when the method works with policing response streams reactively and proactively and shaping response streams proactively. The left and right plots of the figures correspond respectively to the cases where the method operates reactively and proactively. The figures show that when the method is employed reactively, its efficacy both in terms of slowing down the Internet-scale epidemics of email worms (Fig. 6.8) and limiting the illegitimate traffic originating

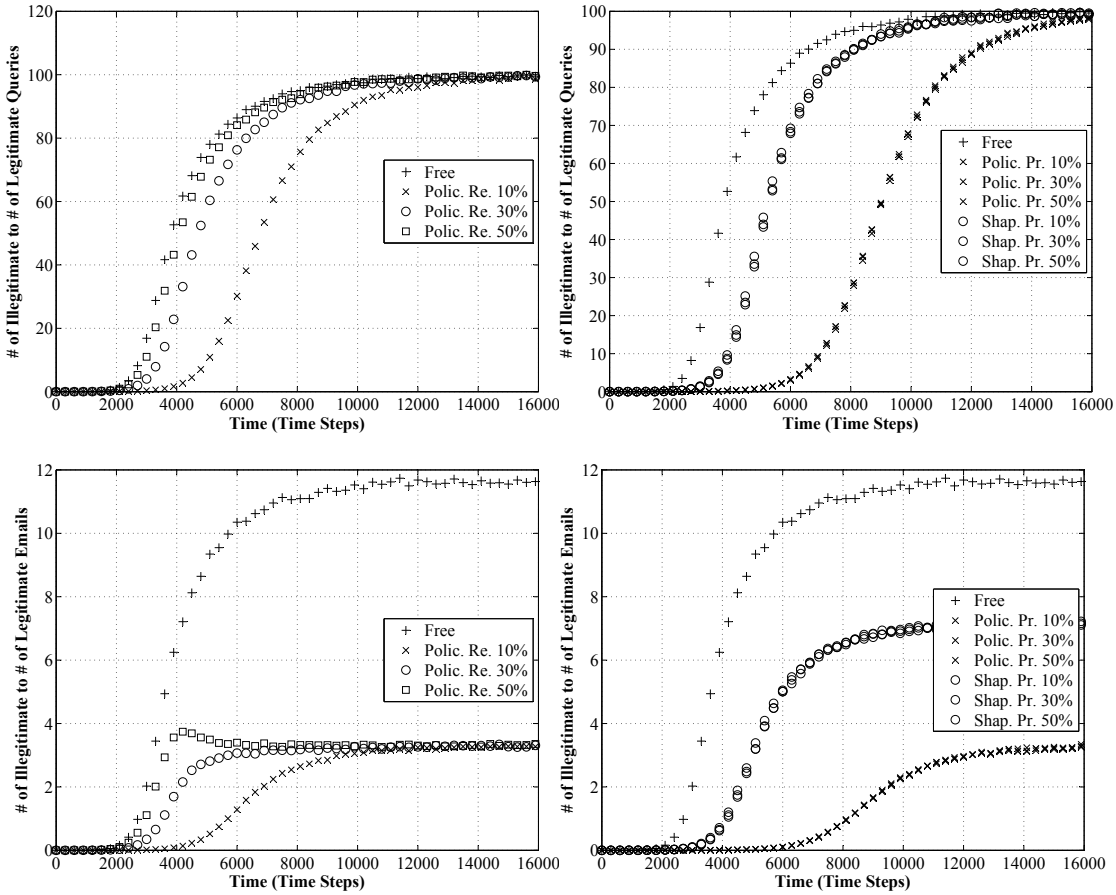

Figure 6.9: Trend in the ratios of the numbers of illegitimate to legitimate queries (upper plots) and illegitimate to legitimate emails (lower plots) over time for the free propagation of email worms and the cases where the method is applied with policing response streams reactively (left plots) and policing and shaping response streams proactively (right plots) when allowing 10, 30 and 50% of the infectious emails produced during the first run of the transmission process of an average email worm on an average user machine to pass before it takes effect. When the method is applied reactively, its effectiveness in limiting the illegitimate traffic generated depends on the reaction time. Regardless of the traffic control mechanism used, this dependency can be eliminated by applying the method proactively.

from infected user machines (Fig. 6.9) drops as the amount of infectious emails sent before it takes effect increases. This uncovers that when the method is applied this way, its performance depends heavily on the reaction time. The figures also indicate that it is possible to eliminate this dependency by applying the method proactively. In fact, when the method is applied proactively, either with policing or shaping response streams, its efficacy does not change with the amount of infectious emails

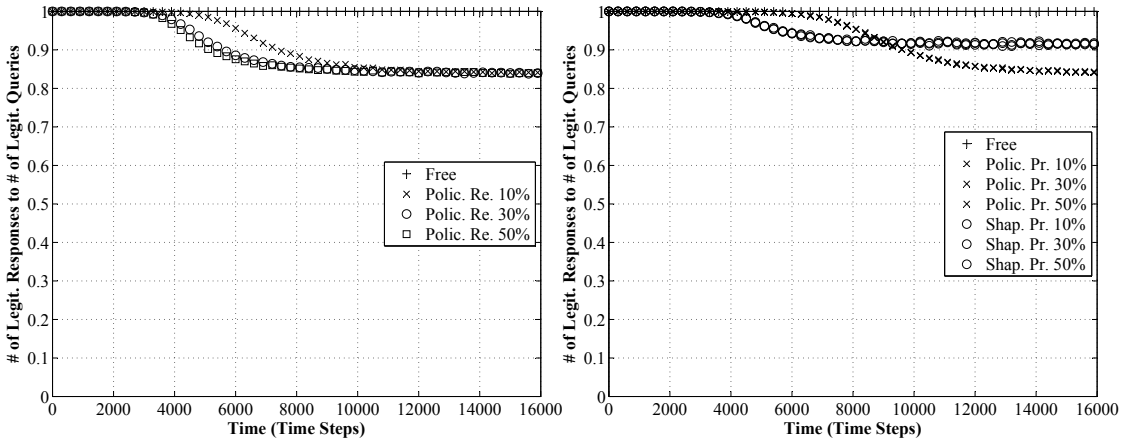

Figure 6.10: Trend in the ratio of the numbers of legitimate responses to queries over time for the free propagation of email worms and the cases where the method is applied with policing response streams reactively (left plot) and policing and shaping response streams proactively (right plot) when allowing 10, 30 and 50% of the infectious emails produced during the first run of the transmission process of an average email worm on an average user machine to pass before it takes effect. When the method runs reactively or proactively, the degree to which it impedes legitimate traffic is largely independent of the reaction time.

that the user machines generate before it becomes operative. The reason for this is that, as pointed out in the previous subsection, when the method runs proactively it takes effect against all, except for the first user machine, that get infected in the monitored network without delay (reaction time near or equal to zero). This finding reveals another advantage, apart from that it produces superior results in terms of containment efficacy, of using the method proactively instead of reactively.

6.5 Evasion of Proposed Method

Having shown that the proposed method has the potential to slow down the Internet-scale epidemics of email worms and restrict the degree to which infected user machines pollute the Internet, the next step was to investigate whether it can be useful in the long run. To this end, several simulation experiments were performed to examine the effectiveness of three approaches worm writers might take to program email worms that will be capable of evading the method, and the results obtained are presented and analysed in this section. The objective of this analysis is to provide insight into the impact that pursuing each of these approaches can have on the Internet-scale propagation of email worms and the efficacy of the method. Each simulation experiment involved setting one of the parameters of the simulation model that relate to the topology of the network on which email worms spread

or the illegitimate outgoing DNS or email traffic of infected user machines to a different value from the one used to produce the results reported in the previous section. The other parameters were kept to their default values given in Table 6.1. Hereafter, the three approaches are described and the time course of the number of infected user machines and the ratios of the numbers of illegitimate to legitimate queries, illegitimate to legitimate emails and legitimate responses to queries for the free propagation and when the method is applied are shown for the corresponding simulation experiments. In the analysis that follows, only the three traffic control mechanism and operational mode combinations that make the method perform best are taken into account. To facilitate comparisons, the trend in the values of the first three metrics over time are plotted again for the free propagation of email worms when all the parameters of the model are at their default values.

The first approach that worm writers might take is to program email worms to generate DNS queries at lower rates. Their purpose in doing so will be to ensure that the illegitimate response streams will not be significantly affected by the traffic control mechanism within the containment period. To explore the implications of implementing this approach for the propagation of email worms and the effectiveness of the proposed method, the rate of illegitimate queries λ_{wq} was set to half the value used to produce the simulation results discussed in the previous section. In connection with this, the ratio of the rates of legitimate to illegitimate queries, λ_{lq} to λ_{wq}, increased from 0.01 to 0.02. Fig. 6.11 indicates that lowering the DNS query rate of email worms, will lead to a significant decrease in the number of infectious emails infected user machines generate per time unit and, as a consequence, the spreading speed of email worms. Although this might increase the chances of email worms to evade detection by methods that look at the volume of DNS queries or emails, it will also increase the time it takes them to reach all the vulnerable user machines and, thereby, the time humans have to adapt and apply prevention and treatment. Moreover, the problems caused by the amount of illegitimate traffic that infected user machines generate will be partially alleviated. In addition, the figure shows that taking this approach will not defeat the method with policing response streams operating reactively or proactively; by contrast, the reduced DNS query stream rate will make the method with shaping response streams perform like with delaying responses. As a matter of fact, the method with policing response streams will be useful for slowing down the Internet-scale epidemics of email worms and limiting the illegitimate traffic infected user machines generate.

The second approach worm writers might take is orthogonal to the one discussed above. Specifically, it involves making email worms generate DNS queries at higher rates. The motivation to consider this approach will be to force infected user machines to send out as many as possible infectious emails before the containment period begins (during the reaction time). To examine the impact of taking this approach on the spread of email worms and the effectiveness of the proposed method, the rate

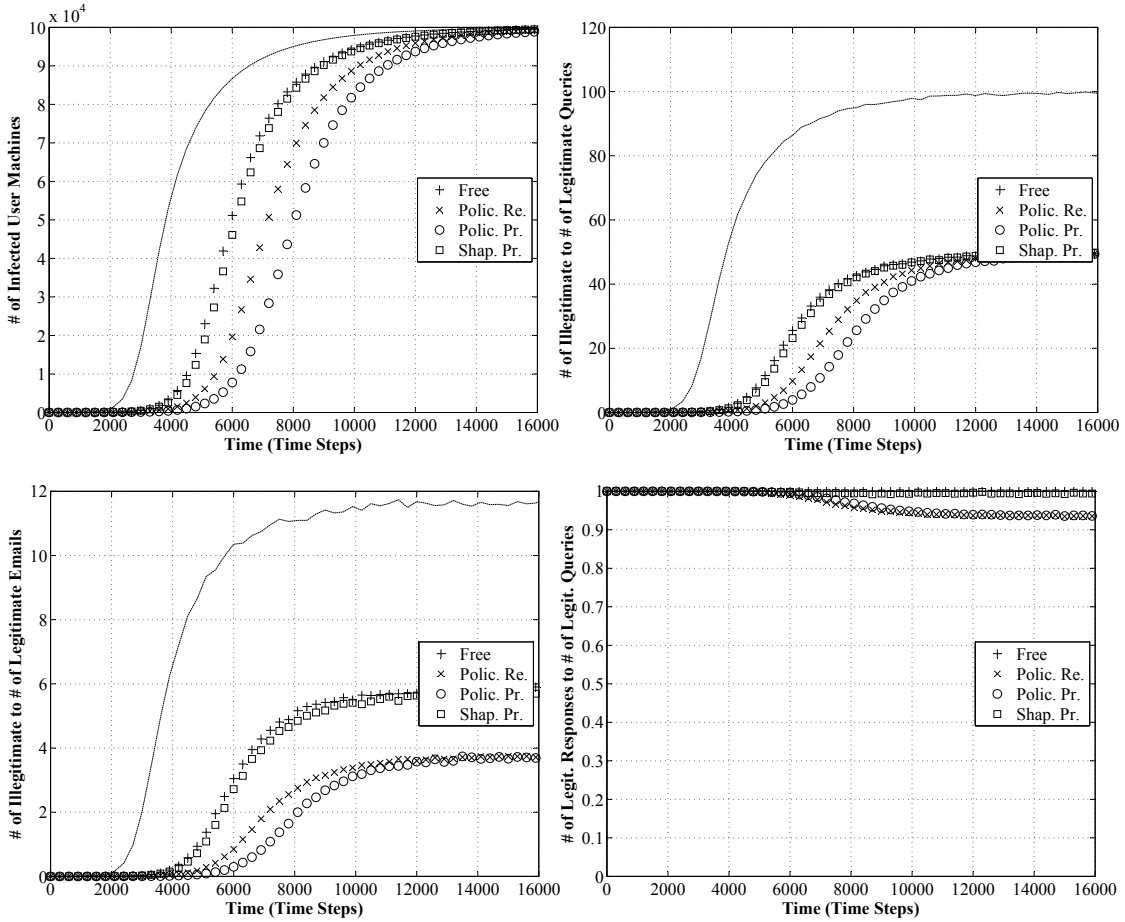

Figure 6.11: Trend in the number of infected user machines (upper left plot) and the ratios of the numbers of illegitimate to legitimate queries (upper right plot), illegitimate to legitimate emails (lower left plot) and legitimate responses to queries (lower right plot) over time for the free propagation of email worms and the cases where the method works with policing response streams reactively and policing and shaping response streams proactively when the rate of illegitimate queries λ_{wq} is set to half its default value. When run with policing response streams reactively or proactively, the method can contain email worms that issue queries at low rates, whereas with shaping response streams proactively it cannot.

of illegitimate queries λ_{wq} was set to twice the value used to generate the simulation results reported in the previous section. Thereby, the ratio of the rates of legitimate to illegitimate queries, λ_{lq} to λ_{wq}, decreased from 0.01 to 0.005. Fig. 6.12 shows that such an increase in the DNS query stream rate of email worms, will cause a considerable raise in the number of illegitimate queries and emails infected user machines produce per time unit. It is worth noting, however, that this raise will

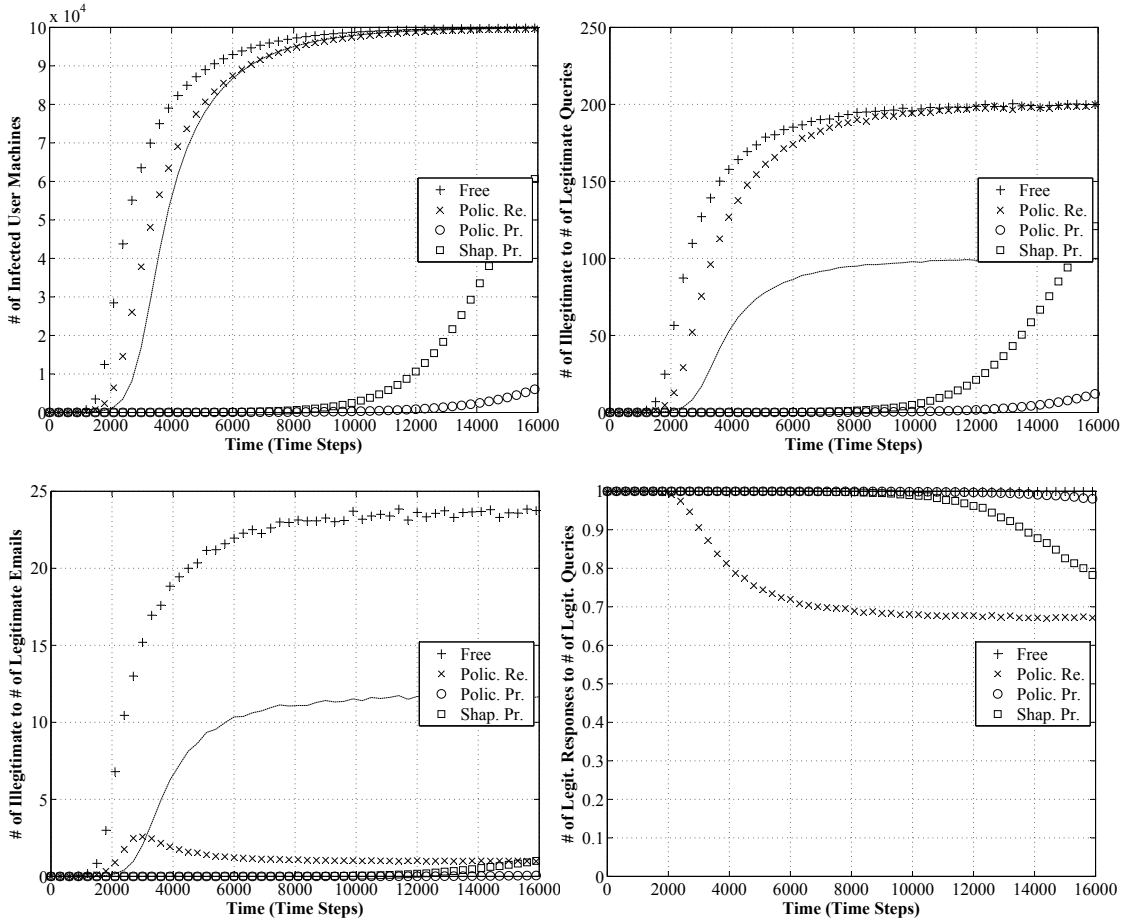

Figure 6.12: Trend in the number of infected user machines (upper left plot) and the ratios of the numbers of illegitimate to legitimate queries (upper right plot), illegitimate to legitimate emails (lower left plot) and legitimate responses to queries (lower right plot) over time for the free propagation of email worms and the cases where the method works with policing response streams reactively and policing and shaping response streams proactively when the rate of illegitimate queries λ_{wq} is set to twice its default value. When run with any of these traffic control mechanism and operational mode combinations, the method has the potential to be effective against email worms that generate queries at high rates.

lead to only a small increase in the speed at which the Internet-scale epidemics of email worms develop. The figure also indicates that pursuing this approach will not detract from the efficacy the proposed method. In more detail, the method with policing or shaping response streams operating proactively will be able to drastically reduce the rate at which infected user machines send out infectious emails and, as a consequence, to significantly limit the propagation speed of email worms. Applying

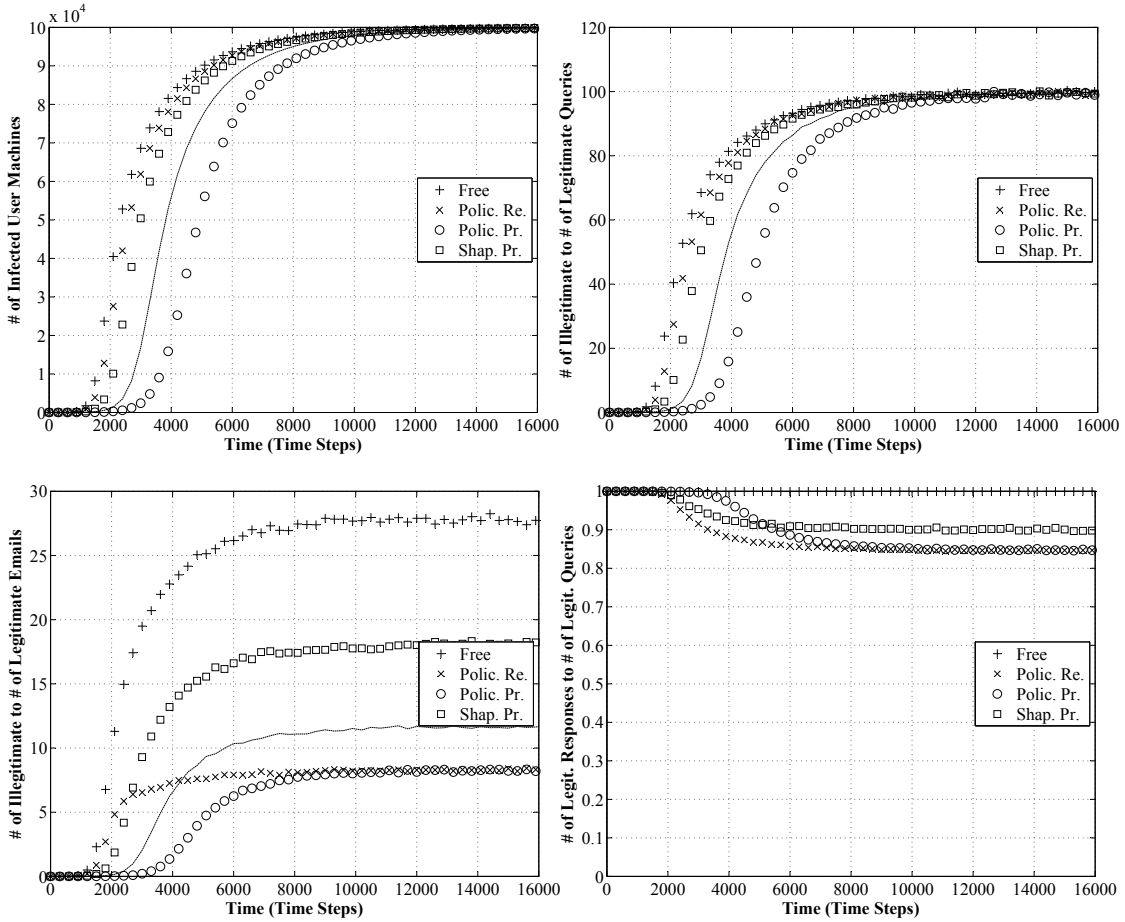

Figure 6.13: Trend in the number of infected user machines (upper left plot) and the ratios of the numbers of illegitimate to legitimate queries (upper right plot), illegitimate to legitimate emails (lower left plot) and legitimate responses to queries (lower right plot) over time for the free propagation of email worms and the cases where the method works with policing response streams reactively and policing and shaping response streams proactively when email worms perform per targeted email address exactly as many queries as needed. When run with any of these traffic control mechanism and operational mode combinations, the method can be effective against email worms programmed to operate in this manner.

the method with policing response streams reactively will achieve worse but still satisfactory results. As a side effect, the method will more severely affect the quality of Internet applications end users experience, as the increased DNS query stream rate will result in more legitimate queries being rejected.

The third approach worm writers might follow is to maximise the efficiency of email worms in terms of the number of infectious emails they produce for a given

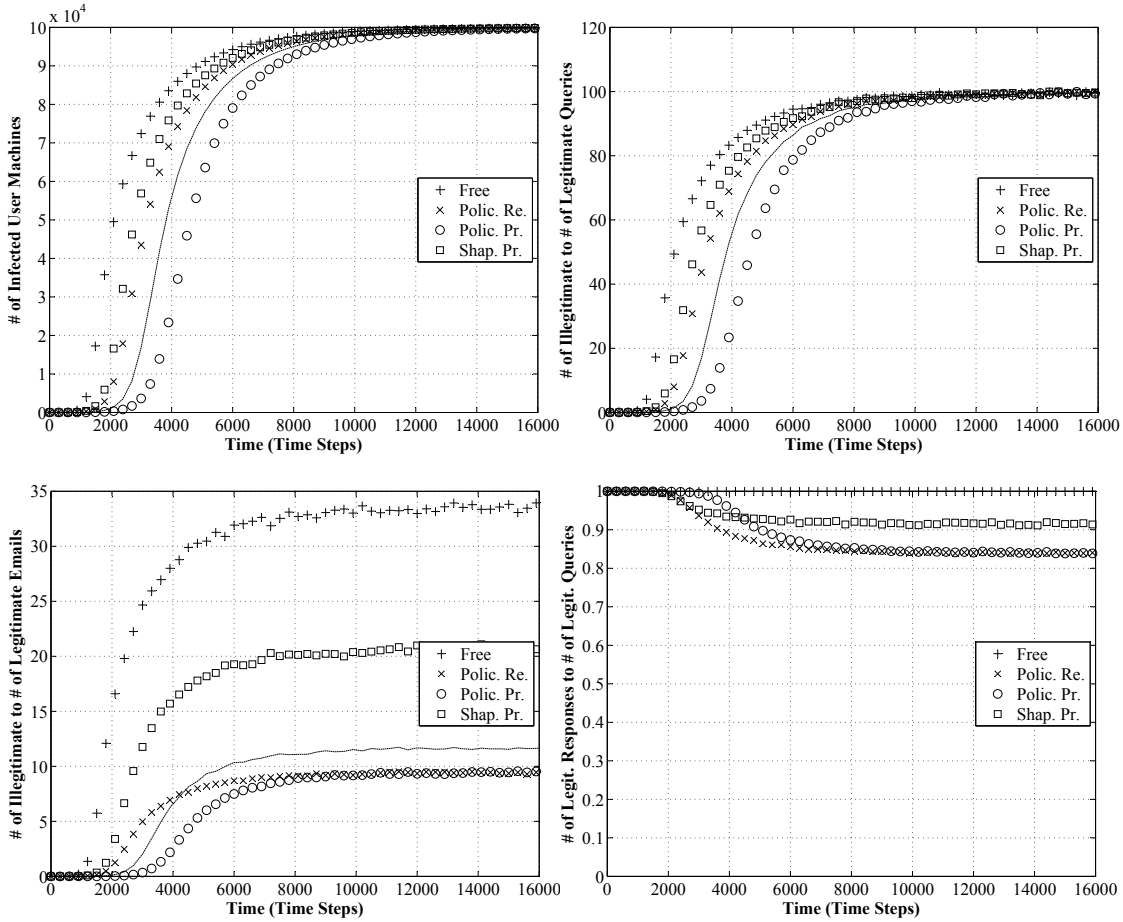

Figure 6.14: Trend in the number of infected user machines (upper left plot) and the ratios of the numbers of illegitimate to legitimate queries (upper right plot), illegitimate to legitimate emails (lower left plot) and legitimate responses to queries (lower right plot) over time for the free propagation of email worms and the cases where the method works with policing response streams reactively and policing and shaping response streams proactively when the email address harvesting efficiency of email worms is improved. When run with any of these traffic control mechanism and operational mode combinations, the method can be effective in limiting the propagation speed of email worms programmed to operate in this manner and the amount of illegitimate traffic that infected user machines generate.

number of DNS queries. To achieve this, they might consider two options. The first of them is to limit the number of DNS queries email worms perform per targeted email address. In light of the analysis in Section 2.3.2, the downside of doing so will be that the infectious emails will have to go through the default email servers, and thus the probability of being filtered out will be higher. The second option is to improve the email address harvesting efficiency of email worms. In practice, there

is a limit to the degree that this can be accomplished because email worms cannot determine before sending out an email whether an email address belongs to an email user (correct guess) or not (false guess). To investigate the implications of taking this approach for the Internet-scale propagation of email worms and the performance of the proposed method, two simulation experiments, one for each option, were carried out. To study the first option, the number of DNS queries that the email worms performed before sending out an email N_q was not drawn from the $N(10, 3^2)$ but assigned to 4, which was assumed to be the number of DNS queries actually needed. To study the second option, the false guessing parameter W_{fg} was set to one fifth of the value used to generate the results presented in Section 6.4; thereby, the ratio W_{cg} to W_{fg} increased to 1. Fig. 6.13 (first option) and Fig. 6.14 (second option) show that increasing the ratio of emails to queries that email worms produce per time unit, will lead to an increase in their spreading speed. Nevertheless, the figures also indicate that even if worm writers pursue this approach, the method with policing response streams proactively or reactively or shaping response streams proactively will continue to be effective in containing email worms.

6.6 Summary

In this chapter, a new containment method for email worms that offers a number of advantages over the available methods discussed in Chapter 5 and those widely adopted and used to filter out unsolicited email traffic entering a network is presented. The method operates on the local name servers and aims at automatically restricting the rate at which infected user machines propagate email worms further. To achieve this, it regulates by means of a traffic control mechanism the flow-level characteristics of the DNS response streams that the local name servers return to user machines within a certain period of time. To evaluate the effectiveness of the method, computer simulations were performed, and the computational model that was designed and developed to this end is described. In this work, three traffic control mechanisms used in two operational modes, reactively and proactively, are considered to determine which of them is the most appropriate for the method.

The simulation results of the method support three conclusions. The first of them is that it has the potential to limit the propagation speed of email worms and the degree to which infected user machines pollute the Internet without significantly affecting legitimate traffic. The second conclusion is that using policing and shaping response streams proactively, eliminates the influence of the reaction time on the performance of the method and maximises its efficacy in terms of the impact it has on the propagation speed of email worms and the legitimate traffic, respectively. On the other hand, the method with policing response streams operating reactively does not impede the traffic of non-infected user machines at all and presents not optimal but still very good performance. The third conclusion is that the method can be

useful for containing email worms, even zero-day, in the long run as it cannot be defeated by minor changes in the code of email worms.

Part IV

Conclusions

Chapter 7

Conclusions

7.1 Key Contributions

This work makes the following contributions to the field of Internet security:

- It looks at Internet security from the DNS viewpoint and uncovers that many Internet threats, such as Internet worms, spamming and DDoS attacks, produce an observable effect on DNS traffic. Thereby, it suggests that analysing and controlling DNS traffic is a very promising approach to effectively detect and mitigate various Internet attacks. To demonstrate the value of this approach, it focuses on combating email worms, which have been a major threat to network operators and end users.

- It proposes a new behaviour-based method that automatically and accurately detects user machines infected with email worms on the local name servers. The method exploits the dissimilarities in the flow-level characteristics of the DNS query streams that user machines generate. To this end, it uses clustering and exact shape-based similarity search over time series produced by counting the queries that the local name servers receive from each user machine within a certain period of time.

- It points out that exact shape-based similarity search over time series, which has received a great deal of attention within the database and data mining community, is a useful tool for distinguishing between normal and anomalous network activity; one that has not been yet adequately explored. In connection with this, it explains and illustrates with a practical example how the findings of the studies that deal with time series indexing can be applied to the context of time series clustering.

- It introduces a new method that slows down the Internet-scale propagation of email worms and reduces the illegitimate traffic email worm-infected user

machines generate. The method operates on the local name servers and affects the best effort delivery of DNS responses to user machines within a certain period of time. To this end, it uses a traffic control mechanism to regulate the flow-level characteristics of the response streams that the local name servers return to user machines.

- It presents a computational model for studying the Internet-scale propagation of email worms and assessing the potential efficacy of email worm containment methods. The model consists of three components: the Internet-scale email network on which email worms spread, the legitimate outgoing email and DNS traffic of a user machine that is produced by the actions of a normal end user, and the illegitimate outgoing email and DNS traffic of a user machine resulting from email worm activity.

- It provides a comprehensive review and a detailed analysis of the strengths and weaknesses of the available methods for detecting and mitigating Internet worms. Particularly, for email worms it also anticipates a number of approaches worm writers might take to increase the virulence and survival chances of their programs. Thereby, it contributes to a better understanding of the dimensions of this Internet threat and the existing solutions, and highlights areas that need further investigation.

7.2 General Discussion

The DNS is a critical infrastructure of the Internet because most of the applications that run on Internet-connected machines depend on the name resolution service it provides to work. Due to its critical nature, the DNS has always been an attractive attack target. In recent years, however, attackers have realised that misusing the DNS comes with more advantages than damaging it or disrupting its service. To achieve their ends, attackers take advantage of vulnerabilities in its design or implementations or simply use the name resolution service. In the last few years, the threat from exploiting vulnerabilities has declined, primarily because the deployment of the DNSSEC is steadily gaining ground and the BIND, the de facto standard name server, is extensively tried and tested. By contrast, many serious threats to network operators and end users, such as Internet worms and botnets, rely on the name resolution service for various destructive and supportive operations, and as a result produce an observable effect on DNS traffic. In this work, email worms are taken as an example to show that this observation opens a promising perspective to detect and mitigate Internet attacks. Thereby, an email worm detection and a containment method that operate on the local name servers are proposed. The methods analyse and control, respectively, the flow-level characteristics of the traffic that local name

servers and user machines exchange. The strengths and limitations of the proposed methods, as pointed out in this thesis, are summarised below.

The proposed detection method overcomes most of the limitations of the available detection methods. First, it is highly automated since it does not require training before it can be used or maintenance during its operation. Thereby, it can identify user machines infected with known as well as zero-day email worms. Second, it is easy to implement and has low overhead because the traffic that local name servers receive from user machines accounts for a small amount of data. Third, it is less computationally demanding than the methods that perform deep packet inspection. Fourth, it does not violate end user privacy. Fifth, it needs a single-point deployment that does not depend on end user participation. Sixth, it runs on each local name server without requiring any communication with other servers or services; hence, it can be deployed incrementally. Seventh, it can detect with remarkable accuracy and negligible false alarm rate user machines that become infected with any of a large variety of recent email worms. Eighth, it has the potential to be useful in the long run as it cannot be evaded by minor changes in the code of email worms. Ninth, it can be effective against polymorphic email worms that might appear in the future because it cannot be defeated by encryption. Tenth, it outperforms in terms of detection efficacy simpler methods that base detection on the volume or self-similarity of the DNS traffic that user machines generate.

With respect to the containment of email worms, the present study fills a void in the literature. This is because the proposed containment method is the first one devised based on a solid understanding of the operations of email worms and the network behaviour of infected user machines. As a consequence, it is the only available method that has the potential to be effective in automatically slowing down the Internet-scale propagation of a large number of email worms. The method is essentially the counterpart of the proposed detection method in the area of Internet worm containment, and as such it has many of the attractive properties mentioned in the previous paragraph. Specifically, it is easy to implement, has low overhead and is less computationally demanding than methods that would operate on packet payloads. In addition, it also needs a single-point deployment that does not depend on end user participation, and it can be deployed in an incremental manner. Apart from these properties, it has three additional interesting properties. First, it has low deployment cost as a local name server is installed in almost every network of the Internet. Second, it can contribute to reducing the illegitimate traffic that infected user machines send to the Internet with minimally, if at all, affecting their legitimate traffic. Third, like the proposed detection method, yet for different reasons, it cannot be easily evaded by small changes in the code of email worms.

Along with the strengths listed above, the proposed methods have also several limitations that emerge from their two basic design characteristics; namely, that they are intended to analyse and control traffic at the flow level and run on the local

name servers. Their first limitation is that they operate only on the traffic that flows between user machines and local name servers. As a consequence, they are blind to illegitimate traffic that infected user machines might exchange with other recursive name servers on the Internet. Although the vast majority of recent email worms depend on the local name servers to carry out some or all their operations and this dependency is not likely to disappear in the future, this leaves a window open for worm writers seeking to evade the methods. A second limitation is that the detection method does not distinguish which DNS queries result from end user and which from email worm activity. Instead, it classifies a user machine as non-infected or infected based on the entire query stream it generates within a certain period of time. In connection with this, the containment method does not affect the illegitimate responses only but treats legitimate and illegitimate responses equally. Therefore, when used, it will inevitably have a small negative impact on the legitimate responses and, thereby, on the quality of Internet applications end users experience. Apart from these limitations that are relevant to both methods, the detection method has two more worth noting. First, it does not identify infected user machines in real time but with a small delay, which is however significantly shorter than the time that passes before the currently-used methods become operative. This is because it processes data in batches, as it needs that an infected user machine has generated some queries before it takes effect. Second, although against most of the email worms considered in this study it was found to be remarkably accurate, it failed completely to detect user machines infected with any from a small subset of them.

7.3 Further Research

This work uncovers that many serious Internet threats, such as Internet worms and botnets, rely on the name resolution service, and as a result affect the DNS traffic traversing the Internet. Based on this observation, email worms are used as an example to demonstrate that analysing and controlling DNS traffic is a promising approach to effectively detect various Internet attacks and mitigate the extent of the damage they cause. In particular, it proposes a detection and a containment method that are intended to run on the local name servers and process the traffic they exchange with user machines. Future work calls for understanding and exploiting the effect that other existing and emerging Internet threats produce on the DNS traffic and developing methods for dealing with them. These methods could work either standalone or complementary to other detection and mitigation methods providing an additional level of protection against these Internet threats. In this direction, studying spambots, which have evolved into the principal source of spam [146] that, in turn, is considered to be the driving force in the economics of botnets [216], would be a natural continuation of the work presented in this thesis. This is because spambot software and email worms share many of the same modules that they use

to find email addresses and send out emails [216]. Hence, it is reasonable to expect that the proposed methods can be also useful for detecting user machines turned into spambots and limiting the rate at which they pollute the Internet.

Although the proposed methods have been shown to be very powerful, they are not a silver bullet for the problem of email worms. In fact, there exist several issues with respect to them that deserve further investigation. These issues relate either to optimising their performance in terms of effectiveness and timeliness or to limiting the options left open to the email worm writers that will try to evade them. The first of these issues concerns studying where on a network is the most appropriate point to implement them and which enhancements are needed to make them capable of dealing with email worms that will not depend on the local name servers at all. In doing so, considering the case of email worms that will query name servers deployed on the Internet for public use, open resolvers or name servers under malicious control and will be able to hide the source of their DNS queries or tunnel their DNS queries through non-DNS traffic, is necessary. The second issue involves exploring possible ways to minimise the detection period without reducing the accuracy of the detection method and examining if and how the method can be refined to work in real time. The third issue regards advancing the detection method so that it can identify user machines infected with any of the few email worms that it currently fails to detect. The fourth issue that is worth looking at is how to increase the effect of the containment method on the illegitimate traffic and restrict that on legitimate traffic. The last issue relates to finding ways to automatically stop the Internet-scale epidemics of email worms instead of just delaying them.

References

[1] I. Abele-Wigert and M. Dunn. An inventory of 20 national and 6 international critical information infrastructure protection policies. *International CIIP Handbook 2006*, 3(1), 2006.

[2] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., 1964.

[3] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional space. In *ICDT'01: Proceedings of the 8th International Conference on Database Theory*, pages 420–434. Springer, 2001.

[4] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *FODO'93: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pages 69–84. Springer, 1993.

[5] R. Aitchison. *Pro DNS and BIND*. Apress, 2005.

[6] P. Akritidis, K. Anagnostakis, and E. Markatos. Efficient content-based detection of zero-day worms. In *ICC'05: Proceedings of the IEEE International Conference on Communications*, pages 837–843. IEEE, 2005.

[7] P. Albitz. *DNS and BIND*. O'Reilly Media, 2001.

[8] Arbor Networks. Worldwide Infrastructure Security Report, Volume III, 2007. www.arbornetworks.com.

[9] Arbor Networks. Worldwide Infrastructure Security Report, Volume IV, 2008. www.arbornetworks.com.

[10] A. Bagnall, C. Ratanamahatana, E. Keogh, S. Lonardi, and G. Janacek. A bit level representation for time series data mining with shape based similarity. *Data Mining Knowledge Discovery*, 13(1):11–40, 2006.

[11] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet motion sensor: a distributed blackhole monitoring system. In *NDSS'05: Proceedings of the Network and Distributed System Security Symposium*. ISOC, 2005.

[12] R. Balzer. Assuring the safety of opening email attachments. In *DISCEX-II: DARPA Information Survivability Conference and Exposition*, pages 257–262. IEEE, 2001.

[13] M. Barreno, B. Nelson, R. Sears, and A. Joseph. User model transfer for email virus detection. In *SysML'06: Proceedings of the 1st Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*. USENIX, 2006.

[14] M. Bauer. Securing DNS and BIND. *Linux Journal*, 2000(78):2, 2000.

[15] V. Berk, G. Bakos, and R. Morris. Designing a framework for active worm detection on global networks. In *IWIA'03: Proceedings of the 1st IEEE International Workshop on Information Assurance*, pages 13–23. IEEE, 2003.

[16] V. Berk, R. Gray, and G. Bakos. Using sensor networks and data fusion for early detection of active worms. In *AeroSense'03: Proceedings of SPIE's 17th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, pages 92–104, 2003.

[17] M. Bhattacharyya, S. Hershkop, and E. Eskin. MET: an experimental system for malicious email tracking. In *NSPW'02: Proceedings of the 2002 Workshop on New Security Paradigms*, pages 3–10. ACM, 2002.

[18] H. Binsalleeh and A. Youssef. An implementation for a worm detection and mitigation system. In *Proceedings of the 24th Biennial Symposium on Communications*, pages 54–57. IEEE, 2008.

[19] B. Boldizsár and V. István. Trap e-mail address for combating e-mail viruses. In *SoftCOM'04: Proceedings of the International Conference on Software, Telecommunications and Computer Networks*, pages 220–224. IEEE, 2004.

[20] M. Braverman. Behavioral modeling of social engineering-based malicious software. In *Virus Bulletin Conference*, 2006.

[21] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *INFOCOM'02: Proceedings of the IEEE Conference on Computer Communications*, pages 638–647. IEEE, 2002.

[22] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *SIGMOD'04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 599–610. ACM, 2004.

[23] D. Chakrabarti and C. Faloutsos. Graph mining: laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):2, 2006.

[24] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[25] K. Chan and Ada W. Fu. Efficient time series matching by wavelets. In *ICDE'99: Proceedings of the 15th International Conference on Data Engineering*, pages 126–133. IEEE, 1999.

[26] G. Chen and R. Gray. Simulating non-scanning worms on peer-to-peer networks. In *InfoScale'06: Proceedings of the 1st International Conference on Scalable Information Systems*, page 29. ACM, 2006.

[27] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. Xu Yu. Indexable PLA for efficient similarity search. In *VLDB'07: Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 435–446. VLDB Endowment, 2007.

[28] S. Chen and Y. Tang. Slowing down Internet worms. In *ICDCS'04: Proceedings of the 24th International Conference on Distributed Computing Systems*, pages 312–319. IEEE, 2004.

[29] S. Chen, X. Wang, L. Liu, and X. Zhang. Wormterminator: an effective containment of unknown and polymorphic fast spreading worms. In *ANCS'06: Proceedings of the 2006 ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, pages 173–182. ACM, 2006.

[30] T. Chen and N. Jamil. Effectiveness of quarantine in worm epidemics. In *ICC'06: Proceedings of the IEEE International Conference on Communications*, pages 2142–2147. IEEE, 2006.

[31] X. Chen and J. Heidemann. Detecting early worm propagation through packet matching. Technical Report ISI-TR-2004-585, USC/Information Sciences Institute, 2004.

[32] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *INFOCOM'03: Proceedings of the IEEE Conference on Computer Communications*, pages 1890–1900. IEEE, 2003.

[33] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, J. Rowe, S. Staniford, R. Yip, and D. Zerkle. The design of grids: a graph-based intrusion detection system. Technical Report CSE-99-2, Department of Computer Science, University of California at Davis, 1999.

[34] K. Chong, H. Song, and S. Noh. Traffic characterization of the Web server attacks of worm viruses. In *ICCS'03: Proceedings of the International Conference on Computational Science*, pages 703–712. Springer, 2003.

[35] CISCO. Policing and shaping overview. http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfpolsh.html.

[36] Common Vulnerabilities and Exposures (CVE). http://cve.mitre.org/.

[37] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levine, and H. Owen. Honeystat: local worm detection using honeypots. In *RAID'04: Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, pages 39–58. Springer, 2004.

[38] R. Dantu, J. Cangussu, and A. Yelimeli. Dynamic control of worm propagation. In *ITCC'04: Proceedings of the International Conference on Information Technology, Coding and Computing*, pages 419–423. IEEE, 2004.

[39] D. Davies and D. Bouldin. A cluster separation measure. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 1(4):224–227, 1979.

[40] DNSSEC: DNS Security Extensions. Securing the Domain Name System, 2007. http://www.dnssec.net.

[41] J. Dunn. Well separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4:95–104, 1974.

[42] H. Ebel, L. Mielsch, and S. Bornholdt. Scale-free topology of e-mail networks. *Physical Review E*, 66:035103, 2002.

[43] D. Ellis, J. Aiken, K. Attwood, and S. Tenaglia. A behavioral approach to worm detection. In *WORM'04: Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pages 43–53. ACM, 2004.

[44] J. Englot and T. Zoli. Critical issues in achieving a resilient transportation infrastructure. In *Symposium on Emerging Developments in Multi-Hazard Engineering*, 2007.

[45] Paul Erdös and Alfréd Rényi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[46] EU COM(2006) 786 EU. Directive on European programme for critical infrastructure protection, 2006. http://eur-lex.europa.eu/LexUriServ/site/en/com/2006/com2006_0786en01.pdf.

[47] C. Faloutsos. *Searching Multimedia Databases by Content*. Kluwer Academic Publishers, 1996.

[48] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD'94: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 419–429. ACM, 1994.

[49] A. Feldmann, A. Gilbert, W. Willinger, and T. Kurtz. The changing nature of network traffic: scaling phenomena. *SIGCOMM Computer Communication Review*, 28(2):5–29, 1998.

[50] Fraunhofer Institute for Open Communication Systems (FOKUS). http://www.fokus.fraunhofer.de/en/fokus/index.html.

[51] F. Freitas, R. Rodrigues, C. Ribeiro, P. Ferreira, and L. Rodrigues. Verme: worm containment in peer-to-peer overlays. In *IPTPS'07: Proceedings of the 6th International Workshop on Peer-to-Peer Systems*, 2007.

[52] A. Ganesh, D. Gunawardena, P. Key, L. Massoulie, and J. Scott. Efficient quarantining of scanning worms: optimal detection and coordination. In *INFOCOM'06: Proceedings of the IEEE Conference on Computer Communications*, pages 1–13. IEEE, 2006.

[53] G. Ganger, G. Economou, and S. Bielski. Self-securing network interfaces: what, why and how. Technical Report CMU-CS-02-144, Computer Science Department, Carnegie Mellon University, 2002.

[54] D. Geer, C. Pfleeger, B. Schneier, J. Quarterman, P. Metzger, R. Bace, and P. Gutmann. Cyberinsecurity: the cost of monopoly. Technical report, Computer & Communications Industry Association, 2004.

[55] GNU Scientific Library (GSL). http://www.gnu.org/software/gsl/.

[56] C. Göldi and R. Hiestand. Scan detection based identification of worm infected hosts. Master's thesis, Swiss Federal Institute of Technology, ETH, Zurich, 2005.

[57] J. Goodman, G. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Communications of the ACM*, 50(2):24–33, 2007.

[58] P. Gopalan, K. Jamieson, P. Mavrommatis, and M. Poletto. Signature metrics for accurate and automated worm detection. In *WORM'06: Proceedings of the ACM Workshop on Recurring Malcode*, pages 65–72. ACM, 2006.

[59] J. Gordon. Pareto process as a model of self-similar packet traffic. In *GLOBECOM'95: Proceedings of the IEEE Global Communications Conference*, pages 2232–2236. IEEE, 1995.

[60] G. Gu, M. Sharif, X. Qin, D. Dagon, W. Lee, and G. Riley. Worm detection, early warning and response based on local victim information. In *ACSAC'04: Proceedings of the 20th Annual Computer Security Applications Conference*, pages 136–145. IEEE, 2004.

[61] A. Gupta and R. Sekar. An approach for detecting self-propagating email using anomaly detection. In *RAID'03: Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pages 55–72. Springer, 2003.

[62] M. Haahr. Random.org: True random number service. http://www.random.org.

[63] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[64] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: part ii. *ACM SIGMOD Record*, 31(3):19–27, 2002.

[65] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2001.

[66] J. Handl and J. Knowles. Exploiting the trade-off - the benefits of multiple objectives in data clustering. In *EMO'05: Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization*, pages 547–560. Springer, 2005.

[67] W. Harrop and G. Armitage. Defining and evaluating greynets (sparse darknets). In *LCN'05: Proceedings of the 30th IEEE Conference on Local Computer Networks*, pages 344–350. IEEE, 2005.

[68] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[69] N. Hindocha and E. Chien. Malicious threats and vulnerabilities in instant messaging. In *Virus Bulletin Conference*, 2003.

[70] K.W. Hipel and A.I McLeod. *Time Series Modelling of Water Resources and Environmental Systems*. Elsevier, 1994.

[71] Homeland Security. Presidential directive 7: critical infrastructure identification, prioritization, and protection, 2003. http://www.dhs.gov/xabout/laws/gc_1214597989952.shtm.

[72] A. Householder and B. King. Securing an Internet name server, 2002. CERT Coordination Center.

[73] C.-M. Hsu and M.-S. Chen. On the design and applicability of distance functions in high-dimensional data space. *IEEE Transactions on Knowledge and Data Engineering*, 21(4):523–536, 2009.

[74] R. Hu and A. Mok. Detecting unknown massive mailing viruses using proactive methods. In *RAID'04: Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, pages 82–101. Springer, 2004.

[75] C.-T. Huang, N. Johnson, J. Janies, and A. Liu. On capturing and containing e-mail worms. In *IPCCC'06: Proceedings of the 21st IEEE International Performance, Computing, and Communications Conference*, pages 257–264. IEEE, 2006.

[76] Y. Huerta and J. Liu. Enhancing malware detection in an IM environment. In *EIT'08: Proceedings of the IEEE International Conference on Electro/Information Technology*, pages 231–236. IEEE, 2008.

[77] T. Hunter, P. Terry, and A. Judge. Distributed tarpitting: impeding spam across multiple servers. In *LISA'03: Proceedings of the 17th USENIX Conference on System Administration*, pages 223–236. USENIX, 2003.

[78] Internet Corporation for Assigned Names and Numbers (ICANN). Domain name hijaking: incidents, threats, risks, and remedial actions, 2005. http://www.icann.org/en/announcements/hijacking-report-12jul05.pdf.

[79] Internet Corporation for Assigned Names and Numbers (ICANN). Factsheet root server attack on 6 February 2007, 2007. http://www.icann.org/announcements/factsheet-dns-attack-08mar07.pdf.

[80] Internet Systems Consortium (ISC). Security Advisories for BIND, 2008. https://www.isc.org/advisories/bind.

[81] K. Ishibashi, T. Toyono, H. Matsuoka, K. Toyama, M. Ishino, C. Yoshimura, T. Ozaki, Y. Sakamoto, and I. Mizukoshi. Measurement of DNS traffic caused by DDoS attacks. In *SAINT'05: Proceedings of the Symposium on Applications and the Internet*, pages 118–121. IEEE, 2005.

[82] K. Ishibashi, T. Toyono, K. Toyama, M. Ishino, H. Ohshima, and I. Mizukoshi. Detecting mass-mailing worm infected hosts by mining DNS traffic data. In *MineNet'05: Proceedings of the 2005 ACM SIGCOMM Workshop on Mining Network Data*, pages 159–164. ACM, 2005.

[83] T. Jiang, W. Kim, K. Lhee, and M. Hong. E-mail worm detection using the analysis of behavior. In *ICDCIT'05: Proceedings of the International Conference*

on Distributed Computing and Internet Technology, pages 348–356. Springer, 2005.

[84] J. Jung, R. Milito, and V. Paxson. On the adaptive real-time detection of fast-propagating network worms. In *DIMVA'07: Proceedings of the International Conference Detection of Intrusions and Malware & Vulnerability Assessment*, pages 175–192. Springer, 2007.

[85] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan. Fast portscan detection using sequential hypothesis testing. *IEEE Security and Privacy*, pages 211–225, 2004.

[86] T. Kahveci and A. Singh. Variable length queries for time series data. In *ICDE'01: Proceedings of the 17th International Conference on Data Engineering*, pages 273–282. IEEE, 2001.

[87] A. Kalafut, A. Acharya, and M. Gupta. A study of malware in peer-to-peer networks. In *IMC'06: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 327–332. ACM, 2006.

[88] K. Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *ACM SIGMOD Record*, 27(2):166–176, 1998.

[89] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido. A nonstationary Poisson view of Internet traffic. In *INFOCOM'04: Proceedings of the IEEE Conference on Computer Communications*, pages 1558–1569. IEEE, 2004.

[90] E. Kartaltepe and S. Xu. Towards blocking outgoing malicious impostor emails. In *WOWMOM'06: Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks*, pages 657–661. IEEE, 2006.

[91] Kaspersky Lab. Monthly Malware Statistics. http://www.viruslist.com.

[92] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.

[93] E. Keogh, K. Chakrabarti, M. Pazzani, and S Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, 3(3):263–286, 2001.

[94] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD'01: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 151–162. ACM, 2001.

[95] J. Kephart and S. White. Directed-graph epidemiological models of computer viruses. *IEEE Security and Privacy*, pages 343–359, 1991.

[96] J. Kephart, S. White, and D. Chess. Computers and epidemiology. *IEEE Spectrum*, 30(5):20–26, 1993.

[97] KFSensor. Advanced Windows Honeypot System. http://www.keyfocus.net/kfsensor/.

[98] D. Kienzle and M. Elder. Recent worms: a survey and trends. In *WORM'03: Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 1–10. ACM, 2003.

[99] C. Kim, S. Lee, and M. Hong. Macroscopic treatment to polymorphic e-mail based viruses. In *ICCSA'04: Proceedings of the Computational Science and Its Applications*, pages 867–876. Springer, 2004.

[100] J. Kim, J. Shim, G. Jung, and K. Choi. Reducing worm detection time and false alarm in virus throttling. In *CIS'05: Proceedings of the International Conference on Computational Intelligence and Security*, pages 297–302. Springer, 2005.

[101] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD'97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 289–300. ACM, 1997.

[102] Neal Krawetz. Anti-honeypot technology. *IEEE Security and Privacy*, 2(1):76–79, 2004.

[103] C. Kreibich and J. Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *SIGCOMM Computer Communication Review*, 34(1):51–56, 2004.

[104] M. Ledoux. *The Concentration of Measure Phenomenon*. American Mathematical Society, 2001.

[105] W. Leland, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of ethernet LAN traffic at the source level. *SIGCOMM Computer Communication Review*, 25(4):100–113, 1995.

[106] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.

[107] Q. Li, Q. Feng, M. Zhang, and L. Hu. PWO: peer-to-peer worm evolution observer. In *ChinaCom'06: Proceedings of the 1st International Conference on Communications and Networking in China*, pages 1–5. IEEE, 2006.

[108] Z. Li, M. Sanghi, Y. Chen, M.-Y. Kao, and B. Chavez. Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience. *IEEE Security and Privacy*, pages 32–47, 2006.

[109] T. Liao. Clustering of time series data – a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[110] M. Liljenstam and D. Nicol. Comparing passive and active worm defenses. In *QEST'04: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems*, pages 18–27. IEEE, 2004.

[111] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT'04: Proceedings of the 9th International Conference on Extending Database Technology*, pages 106–122. Springer, 2004.

[112] Z. Liu, G. Shu, N. Li, and D. Lee. Defending against instant messaging worms. In *GLOBECOM'06: Proceedings of the IEEE Global Communications Conference*, pages 1–6. IEEE, 2006.

[113] P. MacNaughton-Smith, W. Williams, M. Dale, and L. Mockett. Dissimilarity analysis: a new technique of hierarchical sub-division. *Nature*, 202:1034–1035, 1964.

[114] B. Madhusudan and J. Lockwood. Design of a system for real-time worm detection. In *HOTI'04: Proceedings of the 12th Annual IEEE Symposium on High Performance Interconnects*, pages 77–83. IEEE, 2004.

[115] D. Malan and M. Smith. Host-based detection of worms through peer-to-peer cooperation. In *WORM'05: Proceedings of the 2005 ACM Workshop on Rapid Malcode*, pages 72–80. ACM, 2005.

[116] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[117] M. Mannan and P. van Oorschot. Secure public instant messaging: a survey. In *PST'04: Proceedings of the 2nd Annual Conference on Privacy, Security and Trust*, pages 69–77. IEEE, 2004.

[118] M. Mannan and P. van Oorschot. On instant messaging worms, analysis and countermeasures. In *WORM'05: Proceedings of the 2005 ACM Workshop on Rapid Malcode*, pages 2–11. ACM, 2005.

[119] S. Martin. Learning on email behavior to detect novel worm infections. Master's thesis, University of California at Berkeley, 2005.

[120] S. Martin, A. Sewani, B. Nelson, K. Chen, and A. Joseph. Analyzing behavioral features for email classification. In *CEAS'05: Proceedings of the 2nd Conference on Email and Anti-Spam*. IEEE, 2005.

[121] J. Mason and D. Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC, 2003.

[122] D. Massey, A. Mankin, E. Lewis, R. Mundy, and O. Gudmundsson. Public key validation for the DNS security extensions. In *DISCEX-II: DARPA Information Survivability Conference and Exposition*, pages 227–238. IEEE, 2001.

[123] M. Masud, L. Khan, and B. Thuraisingham. Feature based techniques for auto-detection of novel email worms. In *PAKDD'07: Proceedings of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 205–216. Springer, 2007.

[124] R. Matsuba, Y. Musashi, and K. Sugitani. Detection of mass mailing worm-infected IP address by analysis of syslog for DNS server. *IPSJ SIG*, pages 67–72, 2004.

[125] Messaging Anti Abuse Working Group. Email Metrics Reports. http://www.maawg.org/about/EMR/.

[126] G. Miligan and M. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50:159–179, 1985.

[127] K. Mitnick. *CSEPS Course Workbook*. Mitnick Security Publishing, 2004.

[128] D. Moore. Network telescopes: observing small or distant security events. In *SSYM'02: Proceedings of the 11th USENIX Security Symposium*. USENIX, 2002.

[129] D. Moore, C. Shannon, and K. Claffy. Code-red: a case study on the spread and victims of an Internet worm. In *IMW'02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurement*, pages 273–284. ACM, 2002.

[130] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: requirements for containing self-propagating code. In *INFOCOM'03: Proceedings of the IEEE Conference on Computer Communications*, pages 1901–1910. IEEE, 2003.

[131] F. Mörchen. Time series feature extraction for data mining using DWT and DFT. Technical Report No. 33, Dept. of Maths and Computer Science, Philipps-U. Marburg, 2003.

[132] Y. Musashi, R. Matsuba, and K. Sugitani. Indirect detection of mass mailing worms-infected PC terminals for learners. In *ICETA'04: Proceedings of the 3rd International Conference on Emerging Telecommunications Technologies and Applications*, pages 233–237, 2004.

[133] Y. Musashi and K. Rannenberg. Detection of mass mailing worm-infected PC terminals by observing DNS query access. *IPSJ SIG Notes*, pages 39–44, 2004.

[134] M. Newman, S. Forrest, and J. Balthrop. Email networks and the spread of computer viruses. *Physical Review E*, 66(3):035101, 2002.

[135] J. Newsome, B. Karp, and D. Song. Polygraph: automatically generating signatures for polymorphic worms. *IEEE Security and Privacy*, pages 226–241, 2005.

[136] D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. In *DISCEX-III: DARPA Information Survivability Conference and Exposition*, pages 293–302. IEEE, 2003.

[137] Offensive Computing. Malware search. http://www.offensivecomputing.net.

[138] E. Osterweil, M. Ryan, D. Massey, and L. Zhang. Quantifying the operational status of the DNSSEC deployment. In *IMC'08: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 231–242. ACM, 2008.

[139] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[140] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Computer Communication Review*, 31(3):38–47, 2001.

[141] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. *SIGCOMM Computer Communication Review*, 24(4):257–268, 1994.

[142] I. Popivanov and R. Miller. Similarity search over time-series data using wavelets. In *ICDE'02: Proceedings of the 18th International Conference on Data Engineering*, pages 212–221. IEEE, 2002.

[143] P. Porras, L. Briesemeister, K. Skinner, K. Levitt, J. Rowe, and Y.-C. Ting. A hybrid quarantine defense. In *WORM'04: Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pages 73–82. ACM, 2004.

[144] N. Provos and T. Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, 2007.

[145] D. Rafiei and A. Mendelzon. Efficient retrieval of similar time sequences using DFT. In *FODO'98: Proceedings of the 5th International Conference on Foundations of Data Organization and Algorithms*, pages 249–257. Kluwer, 1998.

[146] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Computer Communication Review*, 36(4):291–302, 2006.

[147] M. Roesch. Snort - lightweight intrusion detection for networks. In *LISA'99: Proceedings of the 13th USENIX Conference on System Administration*, pages 229–238. USENIX, 1999.

[148] A. Rowstron and P. Druschel. Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware'01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350. Springer, 2001.

[149] S. Schechter, J. Jung, and A. Berger. Fast detection of scanning worm infections. In *RAID'04: Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, pages 59–81. Springer, 2004.

[150] C. Schuba. Addressing weaknesses in the domain name system protocol. Master's thesis, Department of Computer Sciences, Purdue University, 2000.

[151] M. Schultz, E. Eskin, E. Zadok, M. Bhattacharyya, and S. Stolfo. MEF: malicious email filter - a UNIX mail filter that detects malicious windows executables. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 245–252. USENIX, 2001.

[152] V. Sekar, Y. Xie, M. Reiter, and H. Zhang. A multi-resolution approach for worm detection and containment. In *DSN'06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 189–198. IEEE Computer Society, 2006.

[153] J. Seltzer. Security watch: MyDoom reappears. Security Watch Newsletter, 2005. http://www.pcmag.com/article2/0,1759,1767805,00.asp.

[154] D.-H. Shih, H.-S. Chiang, and C.D. Yen. Classification methods in the detection of new malicious emails. *Information Sciences Informatics and Computer Science*, 172(1-2):241–261, 2005.

[155] S. Shin, J. Jung, and H. Balakrishnan. Malware prevalence in the KaZaA file-sharing network. In *IMC'06: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, pages 333–338. ACM, 2006.

[156] S. Sidiroglou, J. Ioannidis, A. Keromytis, and S. Stolfo. An email worm vaccine architecture. In *ISPEC'05: Proceedings of the 1st International Conference on Information Security Practice and Experience*, pages 97–108. Springer, 2005.

[157] S. Singh, C. Estan, G. Varghese, and S. Savage. The earlybird system for real-time detection of unknown worms. Technical Report CS2003-0761, University of California, San Diego, 2003.

[158] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *OSDI'04: Proceedings of the 6th Symposium on Operating Systems Design & Implementation*, pages 4–20. USENIX, 2004.

[159] R. Smith. Instant messaging as a scale-free network, 2002. http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0206378.

[160] SPECTER. Intrusion Detection System. http://www.specter.com/.

[161] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley, 2002.

[162] S. Staniford. Containment of scanning worms in enterprise networks. Technical report, Silicon Defense, 2003.

[163] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for Internet applications. In *SIGCOMM'01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160. ACM, 2001.

[164] S. Stolfo, S. Hershkop, C. Hu, W. Li, O. Nimeskern, and K. Wang. Behavior-based modeling and its application to email analysis. *ACM Transactions on Internet Technology*, 6(2):187–221, 2006.

[165] A. Studer and C. Wang. Adaptive detection of local scanners. In *ACNS'05: Proceedings of the International Conference on Applied Cryptography and Network Security*, pages 1–17. Springer, 2006.

[166] Symantec. Internet Security Threat Report Trends for January–June 07, Volume XII, 2007. http://www.symantec.com/business/theme.jsp?themeid=threatreport.

[167] M. Taibah, E. Al-Shaer, and R. Boutaba. An architecture for an email worm prevention system. In *Proceedings of the Securecomm and Workshops*, pages 1–9. IEEE, 2006.

[168] Y. Tang and S. Chen. Defending against Internet worms: a signature-based approach. In *INFOCOM'05: Proceedings of the IEEE Conference on Computer Communications*, pages 1384–1394. IEEE, 2005.

[169] Y. Tang and S. Chen. An automated signature-based approach against polymorphic Internet worms. *IEEE Transactions on Parallel and Distributed Systems*, 18(7):879–892, 2007.

[170] The Honeynet Project. Know Your Enemy Whitepapers. http://www.honeynet.org/papers.

[171] The Measurement Factory. DNS Survey: Open Resolvers, 2007. http://www.measurement-factory.com.

[172] R. Thommes and M. Coates. Epidemiological modelling of peer-to-peer viruses and pollution. In *INFOCOM'06: Proceedings of the IEEE Conference on Computer Communications*, pages 1–12. IEEE, 2006.

[173] Trend Micro. Taxonomy of botnet threats, 2006. A Trend Micro White Paper.

[174] US-CERT. Vulnerability Notes Database. http://www.kb.cert.org/vuls/.

[175] M. van Horenbeeck. DNS tunneling. http://www.daemon.be/maarten/dnstunnel.html.

[176] P. van Oorschot, J.-M. Robert, and M. Vargas Martin. A monitoring system for detecting repeated packets with applications to computer worms. *International Journal of Information Security*, 5(3):186–199, 2006.

[177] R. Vaughn and G. Evron. DNS amplification attacks, 2006. http://www.isotf.org/news/DNS-Amplification-Attacks.pdf.

[178] S. Venkataraman, D. Song, P. Gibbons, and A. Blum. New streaming algorithms for fast detection of superspreaders. In *NDSS'05: Proceedings of the Network and Distributed System Security Symposium*, pages 149–166, 2005.

[179] Verisign. Project Titan. http://www.verisign.com.

[180] Virus Bulletin. Malware Prevalence. http://www.virusbtn.com.

[181] Virus Radar. Top 10 Threats. http://www.virus-radar.com.

[182] M. Vojnović and A. Ganesh. On the effectiveness of automatic patching. In *WORM'05: Proceedings of the 2005 ACM Workshop on Rapid Malcode*, pages 41–50. ACM, 2005.

[183] L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.

[184] VX Heavens. Virus Collection. http://www.vx.netlux.org/.

[185] A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast IP networks. In *WETICE'05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies, Infrastructure for Collaborative Enterprise*, pages 172–177. IEEE, 2005.

[186] C. Wang, J. Knight, and M. Elder. On computer viral infection and the effect of immunization. In *ACSAC'00: Proceedings of the 16th Annual Computer Security Applications Conference*, pages 246–256. IEEE, 2000.

[187] F. Wang, Y. Zhang, and J. Ma. Modeling and defending passive worms over unstructured peer-to-peer networks. In *Transactions of Tianjin University*, pages 66–72. Springer/Tianjin University, 2008.

[188] J. Wang and P. De Wilde. Properties of evolving e-mail networks. *Physical Review E*, 70:066121, 2004.

[189] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. In *WORM'03: Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 11–18. ACM, 2003.

[190] N. Weaver, S. Staniford, and V. Paxson. Very fast containment of scanning worms. In *SSYM'04: Proceedings of the 13th USENIX Security Symposium*, pages 29–44. USENIX, 2004.

[191] D. Whyte, E. Kranakis, and P. van Oorschot. DNS-based detection of scanning worms in an enterprise network. In *NDSS'05: Proceedings of the Network and Distributed System Security Symposium*, 2005.

[192] D. Whyte, P. van Oorschot, and E. Kranakis. Addressing malicious SMTP-based mass-mailing activity within an enterprise network. Technical Report TR-05-06, Carleton University, School of Computer Science, 2005.

[193] D. Whyte, P. van Oorschot, and E. Kranakis. Detecting intra-enterprise scanning worms based on address resolution. In *ACSAC'05: Proceedings of the 21st Annual Computer Security Applications Conference*, pages 371–380. IEEE, 2005.

[194] M. Williamson. Throttling viruses: restricting propagation to defeat malicious mobile code. In *ACSAC'02: Proceedings of the 18th Annual Computer Security Applications Conference*, pages 61–68. IEEE, 2002.

[195] M. Williamson. Design, implementation and test of an email virus throttle. In *ACSAC'03: Proceedings of the 19th Annual Computer Security Applications Conference*, pages 76–85. IEEE, 2003.

[196] M. Williamson and A. Parry. Virus throttling for instant messaging. In *Virus Bulletin Conference*, 2004.

[197] C. Wong, S. Bielski, J. McCune, and C. Wang. A study of mass-mailing worms. In *WORM'04: Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pages 1–10. ACM, 2004.

[198] C. Wong, S. Bielski, A. Studer, and C. Wang. Empirical analysis of rate limiting mechanisms. In *RAID'05: Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection*, pages 22–42. Springer, 2005.

[199] C. Wong, C. Wang, D. Song, S. Bielski, and G. Ganger. Dynamic quarantine of Internet worms. In *DSN'04: Proceedings of the International Conference on Dependable Systems and Networks*, page 73. IEEE, 2004.

[200] Y. Wu, D. Agrawal, and A. El Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *CIKM'00: Proceedings of the 9th International Conference on Information and Knowledge Management*, pages 488–495. ACM, 2000.

[201] L. Xie, H. Song, and S. Zhu. On the effectiveness of internal patching against file-sharing worms. In *ACNS'08: Proceedings of the 6th International Conference on Applied Cryptography and Network Security*, pages 1–20. Springer, 2008.

[202] M. Xie, Z. Wu, and H. Wang. HoneyIM: fast detection and suppression of instant messaging malware in enterprise-like networks. In *ACSAC'07: Proceedings of the 23st Annual Computer Security Applications Conference*, pages 64–73. IEEE, 2007.

[203] J. Xiong. ACT: attachment chain tracing scheme for email virus detection and control. In *WORM'04: Proceedings of the 2004 ACM Workshop on Rapid Malcode*, pages 11–22. ACM, 2004.

[204] G. Yan, Z. Xiao, and S. Eidenbenz. Catching instant messaging worms with change-point detection techniques. In *LEET'08: Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–10. USENIX, 2008.

[205] B. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *VLDB'00: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 385–394. Morgan Kaufmann Publishers, Inc., 2000.

[206] I. Yoo. Adaptive firewall model to detect email viruses. In *ICCST'04: Proceedings of the 38th Annual International Carnahan Conference on Security Technology*, pages 197–198. IEEE, 2004.

[207] I. Yoo and U. Ultes-Nitsche. How to predict e-mail viruses under uncertainty. In *IPCCC'04: Proceedings of the 23rd International Conference on Performance Computing and Communications*, pages 675–679. IEEE, 2004.

[208] W. Yu. Analyze the worm-based attack in large scale P2P networks. In *HASE'04: Proceedings of the International Symposium on High-Assurance Systems Engineering*, pages 308–309. IEEE, 2004.

[209] W. Yu, C. Boyer, S. Chellappan, and D. Xuan. Peer-to-peer system-based active worm attacks: modeling and analysis. In *ICC'05: Proceedings of the IEEE International Conference on Communications*, pages 295–300. IEEE, 2005.

[210] W. Yu, X. Wang, D. Xuan, and D. Lee. Effective detection of active worms with varying scan rate. In *Proceedings of the Securecomm and Workshops*, pages 1–10. IEEE, 2006.

[211] J. Zhang, Z.-H. Du, and W. Liu. A behavior-based detection approach to mass-mailing host. In *ICMLC'07: Proceedings of the International Conference on Machine Learning and Cybernetics*, volume 4, pages 2140–2144. IEEE, 2007.

[212] Y. Zhang, Z. Li, Z. Hu, H. Tu, and H. Lin. A P2P e-commerce related network security issue: P2P worm. In *ISECS'08: Proceedings of the International Symposium on Electronic Commerce and Security*, pages 114–117. IEEE, 2008.

[213] Y.-K. Zhang, F.-W. Wang, Y.-Q. Zhang, and J.-F. Ma. Worm propagation modeling and analysis based on quarantine. In *InfoSecu'04: Proceedings of the 3rd International Conference on Information Security*, pages 69–75. ACM, 2004.

[214] L. Zhou, L. Zhang, F. McSherry, N. Immorlica, M. Costa, and S. Chien. A first look at peer-to-peer worms: threats and defenses. In *IPTPS'05: Proceedings of the 4th International Workshop on Peer-to-Peer Systems*, pages 24–35. Springer, 2005.

[215] Y. Zhou, Z.-F. Wu, H. Wang, J. Zhong, Y. Feng, and Z.-Z. Zhu. Breaking monocultures in P2P networks for worm prevention. In *ICMLC'06: Proceedings of the International Conference on Machine Learning and Cybernetics*, pages 2793–2798. IEEE, 2006.

[216] L. Zhuang, J. Dunagan, D. Simon, H. Wang, and J. Tygar. Characterizing botnets from email spam records. In *LEET'08: Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9. USENIX, 2008.

[217] G. Zipf. *Human Behavior and the Principle of Least-Effort.* Addison-Wesley, 1949.

[218] C. Zou, N. Duffield, D. Towsley, and W. Gong. Adaptive defense against various network attacks. *IEEE Journal on Selected Areas in Communications*, 24(10):1877–1888, 2006.

[219] C. Zou, W. Gong, and D. Towsley. Code Red worm propagation modeling and analysis. In *CCS'02: Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 138–147. ACM, 2002.

[220] C. Zou, W. Gong, and D. Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *WORM'03: Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 51–60. ACM, 2003.

[221] C. Zou, D. Towsley, and W. Gong. Modeling and simulation study of the propagation and defense of Internet e-mail worms. *IEEE Transactions on Dependable and Secure Computing*, 4(2):105–118, 2007.

Published Parts

[PP1] N. Chatzis. Mass mailing worm detection by means of situation aware DNS. In *ISADS'07: Proceedings of the International Symposium on Autonomous Decentralized Systems*, pages 279–286. IEEE, 2007.

[PP2] N. Chatzis. Motivation for behaviour-based DNS security: a taxonomy of DNS-related Internet threats. In *SECURWARE'07: Proceedings of the International Conference on Emerging Security Information, Systems, and Technologies*, pages 36–41. IEEE, 2007.

[PP3] N. Chatzis and N. Brownlee. Similarity search over DNS query streams for email worm detection. In *AINA'09: Proceedings of the 23rd International Conference on Advanced Information Networking and Applications*, pages 588–595. IEEE, 2009.

[PP4] N. Chatzis and R. Popescu-Zeletin. Flow level data mining of DNS query streams for email worm detection. In *CISIS'08: Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems*, pages 186–194. Springer, 2008.

[PP5] N. Chatzis and R. Popescu-Zeletin. Detection of email worm-infected machines on the local name servers using time series analysis. *Journal of Information Assurance and Security*, 4(3):292–300, 2009.

[PP6] N. Chatzis, R. Popescu-Zeletin, and N. Brownlee. Email worm detection by wavelet analysis of DNS query streams. In *CICS'09: Proceedings of the IEEE Symposium on Computational Intelligence in Cyber Security*, pages 53–60. IEEE, 2009.

[PP7] N. Chatzis and E. Pujol. Email worm mitigation by controlling the name server response rate. In *SECURWARE'08: Proceedings of the International Conference on Emerging Security Information, Systems, and Technologies*, pages 139–145. IEEE, 2008.