

Detection of Email Worm-Infected Machines on the Local Name Servers Using Time Series Analysis

Nikolaos Chatzis and Radu Popescu-Zeletin

Fraunhofer Institute FOKUS,
Kaiserin-Augusta-Allee 31,
10589 Berlin, Germany
{nikolaos.chatzis,radu.popescu-zeletin}@fokus.fraunhofer.de

Abstract: Although email worms have been around for several years, they continue to be a major Internet security threat to network operators and end users. Their high incidence and the damage they cause reveal that conventional network protection systems are inherently incapable of reducing the frequency and impact of email worm outbreaks. In this paper, we study the effect of email worms on the flow-level characteristics of the Domain Name System (DNS) query streams that user machines generate. We present a method that uses unsupervised learning and wavelet-based time series analysis to distinguish between non-infected and email worm-infected user machines on the local name servers. We evaluate the proposed method against a DNS query dataset that consists of 71 email worms, and demonstrate that it is remarkably effective.

Keywords: Domain Name System (DNS), email worm detection, time series analysis, wavelets

1 Introduction

Since the Morris worm incident in 1988, Internet worms have gradually consolidated their position as a major Internet threat. Often, their outbreaks make the headlines because they are linked to extensive economic damage [38]. Recent Internet worms have been capable of stealing security information, destroying key data and, above all, compromising thousands of user machines in a short time. These compromised machines have been typically used as a medium to distribute unsolicited emails (*spam*) and launch distributed denial of service (DDoS) attacks. The unsolicited and/or abusive traffic that compromised 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 security issue in its own right. In this paper, similar to Kienzle et al. [24], the concept *Internet worm* is used to cover every malicious software (*malware*), standalone or file-infecting, that propagates on the Internet independent of whether human interaction is required or not.

Email is an essential communication medium for the Internet-connected society. Increasingly, individuals and organisations depend on it for a growing proportion of their day-to-day communication. As an unavoidable consequence of this dependency, email became, and remains, the main propagation channel for Internet worms [5]. In fact, in recent years, the monthly-updated top-threat lists of all antivirus companies have been almost exclusively populated by worms that use

email as their primary propagation vector (see, for instance, [19,41]). Internet worms of this type stand at the centre of the present study, and are referred to as *email worms* hereafter. This state of affairs has two immediate implications. The first of them is that email worms remain for attackers a very popular means to accomplish their malicious ends. Whereas, the second implication is that the currently deployed worm detection and mitigation systems yield only meagre, if any at all, results against email worms.

The persistent high incidence of email worms poses manifold threats to network operators and end users. From the network operators' perspective, the unsolicited email traffic originating from compromised machines is considered responsible for network congestion and performance deterioration in network infrastructural elements. This is because email worms are, directly and indirectly, associated with the high amount of unwanted email traffic, which, according to the Messaging Anti-Abuse Working Group (MAAWG) [29], accounts for more than 80% of the total email traffic on the Internet. Specifically, during the outbreaks of email worms significant, sudden increases in the global email traffic are observed. Apart from this direct cause-effect relationship, there exists also a substantial indirect relationship between email worms and unsolicited emails. Namely, worm developers and spammers are steadily joining forces by using email worms to hijack networked machines with the objective to turn them into spam-bots. Over the past years, spam-bots have evolved into the principal source of spam [35]. From the end users' point of view, the problem is stabilising, as only a relatively small fraction of that 80% reaches their inbox [14]. Nevertheless, email worms continue to be a serious security problem for end users, as well, because they are the dominant vehicle attackers use to distribute Trojans, spyware and targeted phishing attacks [39].

The existing methods for detecting Internet worms can be categorised into *general* or *behaviour-based (anomaly-based)* depending on whether they are intended to address worms of all types or only those that employ a particular propagation mechanism, respectively. Despite their widespread adoption and use, the general methods, such as signature-based detection, are far from complete solutions against Internet worms. This is because, as it is often pointed out in the literature, they suffer from one or more of three fundamental problems. In particular, they present limited, if at all, effectiveness to identify new, previously unseen worms (*zero-day worms*), worms that alter their code as they spread (*polymorphic worms*), or worms that do not scan the IP address space but misuse legitimate applications to propagate (*topological worms*). Thereby, in recent years, the main research interest has shifted to designing behaviour-based detection

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 216585 (INTERSECTION Project).

methods [13]. As their name implies, these detection methods require specifying some patterns that characterise the expected network behaviour of infected machines. These patterns are commonly called *behavioural signatures*, and are essentially determined by studying the victim selection and propagation mechanisms of captured worms. In connection with this, networked machines whose traffic conforms to predetermined behavioural signatures can be automatically identified and classified as possibly infected.

Naturally, most of the effort in the research and commercial communities regarding specifying behavioural signatures and devising corresponding methods for detecting email worm activity has been devoted to analysing email traffic. As a result, most of the existing methods operate either on the outgoing email servers, which are typically near the infected machines (see, for instance, [15, 45]), or on the network of the potential victims (see, for instance, [3, 37]). The former methods have limited utility because almost all recent email worms come equipped with a built-in email engine to bypass the ever-evolving filtering of outgoing email traffic [44]. Whereas, the latter methods do not target reducing the amount of the unwanted email traffic on the Internet. To address these two limitations, a considerable number of studies have focused on monitoring and searching for valuable correlations in the DNS traffic of email worm-infected machines [4, 17, 28, 32, 33, 42, 43]. These studies share the view that once a user machine becomes infected the application-layer characteristics of its DNS traffic change significantly. Despite their merits in revealing such a connection between email worm infection and DNS traffic, these studies have only modestly contributed to developing deployable detection systems. Their main deficiency is that they concentrate on application layer detection intended to catch the few email worms analysed in each paper rather than on identifying patterns in the DNS traffic that are generalisable to a larger number of worms; and as such have the potential to serve as a strong basis for detecting email worms in the long run.

In the present study, we go a step beyond earlier work, and argue that at the flow level there is still enough information that, if properly exploited, enables detecting accurately machines infected with various email worms. In particular, we demonstrate that the DNS query streams of non-infected user machines share at the flow level many of the same canonical behaviours; whereas, email worms perform similar address harvesting, attack and spreading operations and, thereby, generate DNS traffic that conforms to different common patterns. To accomplish this, we present a method that builds on unsupervised learning and similarity search over time series data. In connection with this, we use the Discrete Wavelet Transform (DWT) in a different way than this often proposed in the literature of Internet traffic analysis. By experimenting with a large number of email worms that appeared in the wild, we show that the network behaviours of non-infected and infected machines remain unaltered in the long run, and that these behaviours are clearly distinguishable from one another. Thereby, we demonstrate that detection of user machines infected with various, even new, previously unseen, email worms at the local name servers is possible with remarkable accuracy.

Our approach offers several advantages over earlier methods. First, processing packets at the flow level, does not involve deep packet inspection. Therefore, the proposed method does not violate user privacy, will still be effective even if email worms use encrypted communication, and has low processing overhead, which is a strong requirement for busy, high-speed networks. Second, the DNS query

streams consist of significantly less data than the input of conventional network intrusion detection systems. This is advantageous because high volumes of input data inevitably degrade the effectiveness of such systems [36]. Third, using unsupervised learning, eliminates the need for expert knowledge, which most detection methods require for setting thresholds and classifying data to train systems but only rarely is readily available in practice. Fourth, the efficiency of an in-network worm detection system increases as the topological proximity between the system and the machines it protects decreases, and name servers are the first link in the chain of Internet connectivity. Fifth, detection at the local name servers, which are located topologically near the infected machines, can contribute to reducing the unwanted traffic traversing the Internet.

The remainder of this paper is organised as follows. Section 2 is intended to familiarise the reader with the background of the current study. In Section 3, we explain our method for detecting email worm-infected machines by flow-level analysis of DNS query streams. In Section 4, we evaluate the proposed method by examining its detection efficacy over various worms. We state our conclusions and discuss plans for future work in Section 5.

2 Background and Related Work

This section is divided into three parts. In the first of them, we take a brief look at the overall architecture and the inner mechanisms of the DNS. In the second part, the life cycle of a typical email worm is described, and it is shown that many operations performed on the infected machines depend heavily on the local name servers. In the third part, we give a critical review of the key findings and implications of the DNS-based methods for detecting email worms that have been suggested to date.

2.1 Domain Name System

The DNS is the largest distributed system in operation, and constitutes a critical infrastructure for the Internet. Its service is to associate many types of information, such as IP addresses or email exchangers, with domain names. Above all, it translates alphanumeric human-friendly host names, like `www.example.com`, into the corresponding numerical machine-friendly IP addresses. This mapping of host names to IP addresses is formally called *name resolution*, and it is crucial for the operation of almost all Internet applications. The DNS is based on the client-server model, and comprises three components: the *domain name space*, the *name servers*, and the clients, which are referred to as *resolvers* in DNS terminology.

The domain name space is organised as a tree structure of domain names. A 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 that is known as a *zone file*, and contains a list of information entries, which are called *resource records (RR)*. Each RR is a five-tuple that maps a domain name to a resource. The five fields that compose a RR are: *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 cached on a name server or a resolver. Type and class take standard mnemonic values that indicate the application and the application family of the RR, respectively. The resource record data is a variable-length field that represents 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 several other RFCs (see, for instance, RFCs 1183, 1706 and 2782).

The name servers are specialised database servers organised in a hierarchical network. There are many configuration options for a name server that, accordingly, determine its functional role in the network. Primarily, a name server can be configured as *master (primary)*, *slave (secondary)*, *caching* or *forwarding (proxy)*. A master name server has internally stored one or more zones, and provides *authoritative responses* for the data in these zones. This means that it answers DNS queries for RRs in these zones by reading its local file system. The critical nature of the DNS requires that at least two name servers support each zone. Hence, a slave name server obtains its zone information from a master, and is the backup if the master fails. The master and slave name servers are referred to as *authoritative name servers*, and the zones in their file systems as *authoritative data*. A caching name server stores locally the RRs it receives from authoritative name servers until their TTLs expire, and responds queries for these RRs with locally-stored data. As its name implies, a forwarding name server forwards all the queries it receives to other name servers and caches the responses for later use.

The vast majority of Internet applications require information from the domain name space to work. This includes the name resolution itself because the name servers resolve domain names in a recursive manner. Therefore, a *resolver* is installed on almost all Internet-connected machines. Its role is to translate the applications' needs to DNS queries, and send these queries to the name servers. Resolvers issue two types of DNS queries: *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 data or a part thereof. Recursive queries ask name servers to do all the necessary work to return complete answers. Most user machines and other networked devices – except for name servers – run a lightweight (*stub*) resolver. Stub resolvers generate recursive queries only, forward them to full resolvers, and resend them if they are not answered in a short time period. By contrast, full resolvers generate iterative queries, as well, and are mainly installed on the name servers. This is, actually, the reason why in almost every network a name server is installed topologically near the user machines, which represents for them the first link in the entire chain of Internet connectivity. This type of name servers stands at the centre of the present study, and is referred to as *local name server* hereafter.

2.2 Email Worm Life Cycle

Increasingly, email worms are developed with the objective to induce financial gain by misusing infected machines rather than merely to spread. Therefore, their operations relate either to their propagation or their malicious activities. Email worms perform these malicious activities upon execution of a 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 machines or other Internet-connected systems; whereas, the latter are designed to hide the malicious activities. Email worm writers have at their disposal a steadily growing number of programs for performing a vast array of destructive operations. Although the exact combinations and the implementation details of these operations can vary significantly from one worm to another, the life cycle of a typical email worm consists of four distinct phases: the *penetration*, *target acquisition*, *payload activation* and *propagation* phases. To increase the virulence and the survival

chances of their worms, worm writers usually program them to repeat all the phases, except for the penetration phase, according to some prearranged patterns or when triggered by a system event.

Email worms circulate via specially crafted emails, which infected machines generate, in either of two possible ways. In particular, they spread either as attachment files, or the infectious emails contain links to compromised Web sites. In the penetration phase, the email worms attempt to install malware on the machines of unsuspecting email users. This involves tricking the email users by means of various social engineering techniques into executing infectious attachment files or clicking fraudulent links. Social engineering techniques provoke end users to make wrong decisions, and they are based on specific attributes of the human decision-making known as *cognitive biases* [30]. Once the malicious code is installed, the 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* subphases. When the email worms are in the reinfection subphase, they attempt to render the infection persistent. To achieve this, they typically perform various auxiliary operations. Common examples of such operations involve manipulating configuration parameters, creating or modifying files, installing additional malware for updating and maintaining the malicious code, and weakening security settings. In the attack subphase, the worms carry out their destructive operations, whose purpose, as previously mentioned, is linked to the intent of worm writers. It is noteworthy that most of the email worms that appeared recently in the wild were designed either to automatically launch DoS or to install remotely controlled malware that has been employed 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 exchangers associated with them. This phase comprises two clearly distinguishable subphases that take place either simultaneously or in succession: the *harvesting* and *name lookup* subphases. In the former, the worms perform various operations to compile lists with as many as possible email addresses that will be targeted for infection. Specifically, they are usually programmed to obtain addresses by searching the address book and files with predefined extensions, guessing by making any possible combination of user and domain names they find in the file system of infected machines, and scanning Web pages. In the name lookup subphase, as its name suggests, the email worms query the local name server on the networks of the infected machines in order to resolve the IP addresses of the email exchangers that deliver emails to the potential victims.

At any time after the target acquisition phase has begun, the propagation phase is initiated. In this phase, the worms send sequentially infectious emails to all the email addresses in the target list. Regarding the sending operation of their worms, worm developers have three options. These are to make them use, in a seemingly legitimate manner, either open email relays on the Internet or the email clients on infected machines, or to arm them with a built-in email engine to render them completely independent of outgoing email servers. The last option offers the advantage that worms can escape detection by systems deployed on the outgoing email servers; thus, it is the one most often used. In addition, to increase the virulence and propagation speed of their worms, worm writers make them capable of spreading over secondary communication channels, such as peer-to-peer and instant messaging applications.

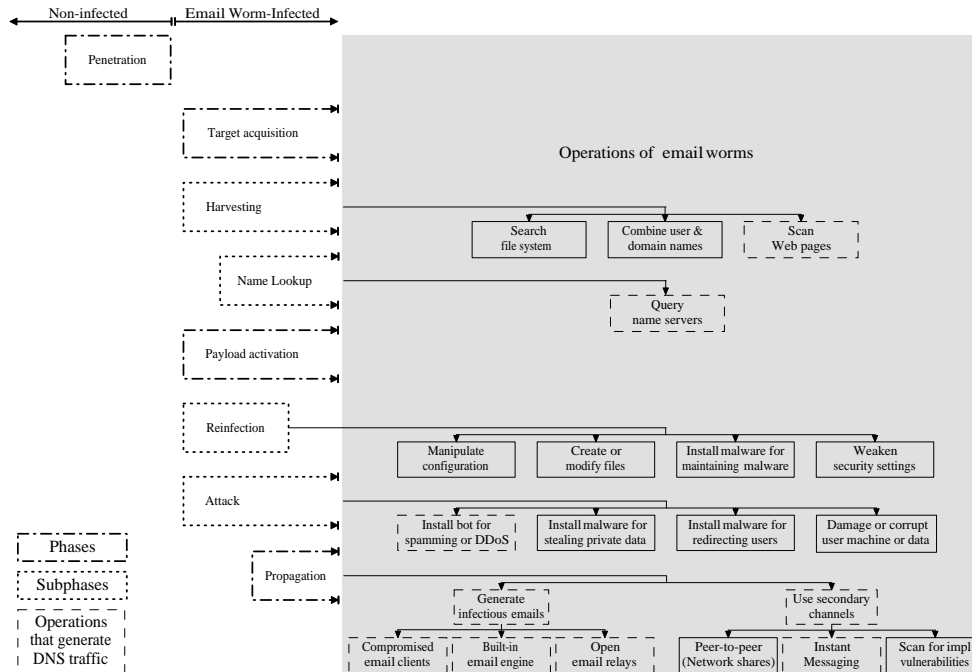


Figure 1: The life cycle of email worms comprises: the penetration, target acquisition, payload activation and propagation phases. The target acquisition and payload activation phases start after the penetration phase is completed, and each of them can be divided into two subphases: the harvesting and name lookup, and the reinfection and attack subphases, respectively. The propagation phase starts at any time after the target acquisition has started. For each phase/subphase the most common operations email worms perform are given (grey highlighted region).

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, during the target acquisition phase, the email worms query the local name servers for the IP addresses of Web pages (harvesting subphase), and email exchangers (name lookup subphase). In addition, as it is detailed in [8], the malicious programs that launch spamming and DDoS attacks (attack phase) are, to a great extent, reliant on the DNS. In Fig. 1, the phases and subphases of the email worm life cycle, their timing relationship, and their most common operations are illustrated. Moreover, the operations that affect the DNS traffic of user machines are marked. The figure is necessarily incomplete because new tactics may arise. However, it is complete with regard to the situation at the time of writing this paper, as it includes all the operations of the email worms that have appeared recently in the wild.

2.3 DNS-Based Email Worm Detection

Wong et al. [43] analyses DNS traffic captured on the local name server of a campus network during the outbreaks of Sobig.F and Mydoom.A. Musashi et al. [32] reports on a complementary study that provides additional information about the same email worms. The authors of this report continue their research and extend their scope by studying Netsky.C [28], Netsky.Q and Mydoom.S [33]. Independent from one another, Whyte et al. [42] and Ishibashi et al. [17] study the DNS activity of machines infected with Netsky.Q. In particular, Whyte et al. analyses a dataset of DNS queries captured on the local name server of an enterprise network. Whereas, Ishibashi et al. examines DNS traffic recorded on the local name servers of a large commercial Internet Service Provider (ISP). Despite their differences with respect to the details of the measured DNS traffic,

all these studies essentially converge toward a common conclusion. This is, namely, that the application-layer characteristics of the DNS traffic that user machines generate change once the machines become infected with email worms.

This conclusion has served, in a straightforward manner, as the basis for most of the work on detecting email worms by monitoring and analysing DNS traffic. As a matter of fact, the methods paired with the analyses in [28,32,33,42] and the one presented in Binsalleeh et al. [4] are based on the same principle. In particular, they inspect the DNS traffic of user machines at the application layer searching for specific volume-based anomalies. Their observation basis is that the volume of queries for email exchangers (MX) RRs, or the relative volumes of queries for pointer (PTR) RRs, MX and address (A) RRs of user machines provide a strong basis for detecting worm activity. Although this holds true for a few specific – mainly outdated – worms studied in each paper, it cannot be generalised for detecting machines infected with various email worms in the long run. In fact, in [10], we show that the number of DNS queries is a rather weak indicator of whether a machine is infected or not. Further, as with any other threshold-based method, these methods rely on expert knowledge for setting suitable threshold values that make it possible to distinguish infected from non-infected machines.

Instead of setting thresholds for the DNS traffic volume, other detection methods, which operate at the application layer, as well, employ more sophisticated procedures [17,47]. In particular, Ishibashi et al. [17] assumes a priori knowledge about some characteristic DNS queries that infected machines generate. Based on this knowledge, the method proposed in this study detects email worm-infected machines by applying Bayesian inference to the DNS queries that local name servers receive. Zhang et al. [47] advances Ishibashi et al. by defining

and computing a score for each networked machine that expresses the probability that it is infected. To achieve this, the method builds decision trees by analysing the DNS queries monitored on the local name servers within a given observation interval. Despite their sound mathematical grounding, these studies have a methodological problem that brings the usefulness of their findings into question. This is that they essentially depend on a priori knowledge about previously unseen worms. Obviously, such knowledge is not available, and if it were it would allow straightforward detection.

Apart from the limitations already pointed out, the studies revised above share also two common intrinsic weaknesses that limit their usefulness further. The first of them is that the detection methods they suggest involve DNS traffic analysis at the application layer. Thereby, they overlook that DNS queries contain sensitive end user information, which imposes many privacy requirements, and thus, may not be suitable candidate for deep packet inspection. Besides this, the fact that they analyse packet payloads, renders them inappropriate for high speed, busy networks. Nevertheless, the second weakness is by far the most significant. This weakness is that the proposals and claims these studies make are supported by weak experimentation due to the lack of realistic benchmarks. In particular, the studies report on experiments against only a few email worms, and do not discuss assessment details using real network topologies. Therefore, despite their positive contribution of defining a correlation between worm infection and DNS traffic, they have only modestly contributed to developing deployable detection systems.

3 Proposed Approach

Our proposed method is based on similarity search over time series that express the DNS activity of networked machines. Given the time series representation, we show that user machines' DNS activity falls into two canonical profiles: *legitimate user* behaviour, and *infected by email worm* behaviour. The basic idea is that machines infected with instances of a single worm – or even of various worms – exhibit similar network behaviour. This is because email worms contain only a certain number of modules that they use to harvest addresses, attack networked machines and propagate. Even when different modules are at hand, they are limited in number. Therefore, it is justified to assume that there will be similarities in the DNS communication patterns of compromised machines. Exploring this principle, we demonstrate that the DNS query streams of non-infected user machines share many of the same canonical behaviours. Likewise, email worm-infected machines generate query streams that share common patterns, so even machines infected with new email worms will have DNS-related behaviours similar to those of machines infected with known email worms. We show that these two behaviours can be reliably distinguished, which implies that email worm activity can be accurately detected at the local name servers.

3.1 Data Preparation

As input, the proposed method uses the complete set of DNS queries that a local name server received within an observation interval, which is referred to as *detection period* hereafter. Each query comprises the time when the server received the query, the requesting host – identified by its IP address – and the requested data. Because we are not interested in application-layer information, we keep for each query only the time and the IP address of the requesting host. To work with query streams of user machines only, we whitelist the servers of

the monitored network that generate legitimate DNS traffic; and we do not take their queries into account. Even on large networks the set of email and other servers that communicate frequently with the local name servers is known and manageable, so listing these machines in this way does not violate the requirement for practicality. We group the remaining queries per requesting user machine. For each user machine, we consider p successive time bins of equal duration, and we count the DNS queries in each time bin. Following this procedure, we produce a set of univariate time series, where each one of them expresses the number of DNS queries of a user machine through time. The set of time series can be expressed as an $n \times p$ time series matrix; n is the number of machines that queried the name server at least once within the detection period.

3.2 Time Series Similarity Search

Formally, a time series is a collection of observations made sequentially through time, which are called *time points* and represent the status of a single variable over time. Similarity search over time series is useful in its own right as a tool for exploratory data analysis, and it is also an important element of many basic data mining tasks, such as indexing, clustering, classification and rule discovery [22]. Intuitively, the similarity between two time series can be simply calculated by using any distance measure. For instance, the Euclidean distance – i.e., L_2 norm – of two p -length time series can be straightforward computed, if each of them is seen as a point in the p -dimensional space, with each time point corresponding to one dimension. This allows us to apply multivariate data mining to time series data directly. However, most similarity-based data mining algorithms perform poorly when applied to time series because of the typical high dimensionality of this type of data. This is because working with each and every time point makes the significance of distance measures questionable [1, 26]. In addition, in high-dimensional spaces the contrast between the nearest and the farthest neighbour becomes increasingly smaller [2]. Therefore, clustering time series using general-purpose clustering algorithms produces low quality results because these algorithms lose their effectiveness as the dimensionality of data increases. Thereby, finding meaningful groups of time series is substantially complex and non-obvious.

To attack this problem, the approach that has received the most research interest involves extracting from each time series a low-dimensional representation, which is called *feature vector*, and using it as input, instead of the time series, to the data mining algorithms. Although a large number of feature vector extraction functions have been proposed in the literature, only a few of them qualify for the present study. The reason is that working with the feature vectors rather than with the original time series results in information loss. Hence, clustering feature vectors could potentially introduce *false dismissals*. False dismissals occur when feature vectors of similar time series appear distant in the feature vector space. For the current work, using a feature vector extraction function that guarantees no false dismissals is a crucial requirement. Faloutsos et al. [12] determines the conditions that a function must satisfy in order to be appropriate for similarity search that introduces no false dismissals. In that study, it is shown that the Discrete Fourier Transform (DFT) meets these conditions. Subsequent work demonstrates that other functions that fulfil the conditions specified in Faloutsos et al. are: the Discrete Wavelet Transform (DWT) [7, 34], the Piecewise Aggregate Approximation (PAA) [21, 46], the Adaptive Piecewise Constant Approximation (APCA) [22], the Piecewise Linear Approximation

(PLA) [11], the Chebyshev Polynomials (CHEB) [6], and the Singular Value Decomposition (SVD) [18, 25].

For the purposes of this study, we employ the Discrete Wavelet Transform (DWT) and refer the interested reader to [9] for a comparative evaluation that shows how the proposed method performs if the other functions listed above are used instead. There are several reasons that motivate using the DWT. The first of them is that the DWT representation of a time series is intrinsically multi-resolution and provides a means to analyse the time series in both the time and frequency domains simultaneously. Second, many coefficients in the DWT representation of most time series encountered in practice are either zero or have very small values, which allows for efficient compression. Third, the DWT is applicable for analysing non-periodic and/or non-stationary signals, and performs well in compressing sparse spike time series. Thereby, we apply the DWT independently to each row of the $n \times p$ time series matrix using Mallat’s algorithm [27]. The Mallat algorithm decomposes a p -length time series, where p must be a power-of-two, in $\log_2 p$ decomposition levels, and is of $O(p)$ time complexity. The number of DWT coefficients produced is equal to the number of time points of the original time series. Therefore, applying the DWT to each row of the time series matrix, gives an $n \times p$ DWT coefficient matrix.

To reduce the dimensionality of the DWT coefficient matrix, we apply a compression method. For compression purposes, the coefficients are often normalised, which means that the coefficients at lower resolutions are weighted more heavily than those at higher resolutions. The normalised coefficients are obtained by multiplying each coefficient by $2^{j/2}$, where j is the decomposition level, at which the coefficient is computed. Retaining the k largest coefficients in terms of absolute normalised value, produces for a given k the optimal feature vector in terms of sum squared error [40]. Another well established compression method keeps the first k coefficients, which capture the low-frequency features of a time series. Mörchen [31] compares these two methods, which are performed independently on each time series, with two methods suited for a set of n time series, and, thereby, applicable to the DWT coefficient matrix directly. The first retains 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. Retaining the first k coefficients or the k columns of the DWT coefficient matrix that have the largest mean element-wise squared value, results in a $n \times k$ feature vector matrix, with $k < p$. By contrast, the other two compression methods produce a $n \times p$ feature vector matrix, with $n \times k$ non-zero elements. Thus, these last two methods have higher storage demands and involve higher bookkeeping overhead for the distance computations during the clustering process. In this paper, we present experimental results only for the compression method that keeps the first k coefficients. We refer the interested reader to [10] for a comparative analysis of how the detection method performs if the other three methods described above are used.

3.3 Feature Vector Clustering

Using clustering, we seek to validate our hypothesis that the DNS query streams of non-infected user machines share similar characteristics, and that these characteristics are dissimilar to those of the DNS traffic that email worm-infected machines generate. To accomplish this, we cluster the rows of the feature vector matrix into two groups. Then, we look at the nature of the two clusters, and examine whether one corresponds to non-infected and the other to worm-infected user

machines. The objective is to demonstrate that the two clusters are reliably distinguishable and pure; namely, that one of them contains only feature vectors of non-infected and the other feature vectors of email worm-infected machines.

We employ hierarchical clustering because it produces good results, as it facilitates the exploration of data at different levels of granularity, and is robust to variations of cluster size and shape. Hierarchical clustering methods require us to specify a dissimilarity measure. We employ as dissimilarity measure the Euclidean distance, since it gives comparable results to more sophisticated distance measures [23]. Hierarchical clustering algorithms are categorised into *agglomerative* and *divisive*. Agglomerative algorithms are bottom-up. They start, namely, with each feature vector in its own cluster, and recursively merge the less dissimilar clusters into a single cluster. By contrast, the divisive algorithms are top-down. This means that they start with all the feature vectors in one root cluster, and subdivide them into smaller clusters until each cluster consists of 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. Because in this work we search for a small number of clusters, we use a divisive algorithm.

The most commonly-cited disadvantage of the divisive algorithms is that they are computationally very demanding. In particular, at their first level, the divisive algorithms consider all divisions of the entire dataset into two non-empty sets. There are $2^{(n-1)} - 1$ possibilities for dividing n feature vectors in two clusters, which renders these algorithms intractable for many practical datasets. However, the DIvisive ANALysis (DIANA) [20] uses a splitting heuristic to limit the number of possible partitions. The DIANA has $O(n^2)$ instead of $O(2^n)$ time complexity. Given its quadratic time on the number of feature vectors, the DIANA scales poorly as the number of feature vectors increases. Yet, it is the clustering algorithm we employ because it is the only divisive method generally available, and for the purposes of this study, it clusters the feature vectors on a typical computer in computing time in the order of milliseconds.

4 Experimental Evaluation

The experimental evaluation of the proposed method that is given below consists of two parts. In the first part, we assess the overall accuracy of the method to identify machines infected with various email worms. Thereby, we show that, indeed, the DNS query streams of user machines fall into two canonical profiles. In the second part, we evaluate the efficacy of the proposed method to detect each email worm independently.

4.1 Datasets and Parameters

Instead of releasing worms in the real network, we set up an isolated computer cluster. In the cluster, we launched 71 out of a total of 164 email worms, which were reported between April 2004 and July 2007 in the monthly-updated top-threat lists of Virus Radar [41] and Viruslist [19]. To ensure typical worm behaviour in the cluster, we copied the file system of a user machine in our research institute’s network to the isolated computers. By reverse code engineering a subset of the worms, we found that their operations are not affected by the portion of the abusive emails that are delivered to the remote email servers or the number of DNS queries resolved. We recorded over a period of eight hours the DNS queries of compromised machines to create – to the best of our knowledge – the largest and most complete

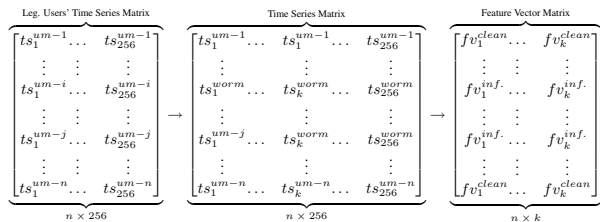


Figure 2: For each dataset-worm pair, we select randomly how many machines get infected, which are these machines; and at which time bins they become infected. Thereby, we produce a time series matrix that contains legitimate and infectious DNS activity. Using the DWT, we decompose the time series to construct the DWT coefficient matrix, which we compress to get the feature vector matrix whose rows we cluster with DIANA.

email worm DNS dataset that has been used to date to evaluate an in-network email worm detection method.

Because our isolated computer cluster has no real users, we merge the DNS query streams of the email worms with legitimate DNS traffic captured at the primary name server of our research institute, which serves daily between 350 and 500 users. We use three DNS log file fragments collected on 27 through 28 March 2006, 30 September through 2 October 2006 and on 29 through 31 January 2007. The fragments consist of 631875, 3702926 and 4183311 DNS queries, respectively. We consider 15-second time bins, and construct from the email worm and the legitimate user DNS query streams time series with length 256. We present here experiments using ten one-hour – i.e., $256 \times 15\text{sec} \approx 1\text{hour}$ – datasets that capture the DNS activity of user machines at different times of the day.

To show that the proposed method identifies accurately user machines infected with various email worms, we assume that only one user machine is infected each time by a different email worm. By contrast, to assess the detection efficacy of our method in detecting individually each email worm, we consider an arbitrary number of machines becoming infected at random times within the detection period. Hence, for this second experiment, we run for each dataset-worm pair three times the random number generator by Haahr [16] to determine how many machines get infected, which are these machines, and at which time bin the infections occur. The number of infected machines and the time bins at which they become infected are in $[1, \frac{5 \times n}{100}]$ and $[1, \frac{6.25 \times 256}{100}]$, respectively; n denotes the number of user machines. Furthermore, we assume that all the user machines have the same probability of being infected. To obtain a sufficient statistical sample, we repeat this procedure 100 times. In both experiments, we decompose the time series in $\log_2 256 = 8$ levels to compute the values of the DWT coefficient matrix, which we, then, compress using different values of k – i.e., 4, 8, 16 and 32. We repeat the experiments 71 times, one for each email worm. The experimental procedure is illustrated in Fig. 2.

4.2 Experimental Results

We examine the members of the two clusters, and discover that the one of them is densely and the other sparsely populated. Since we are dealing with unlabelled data, it is necessary to find a way to determine which cluster contains feature vectors that correspond to legitimate activity and which feature vectors of compromised machines. To do this, we uncover an assumption that intrinsically holds for every

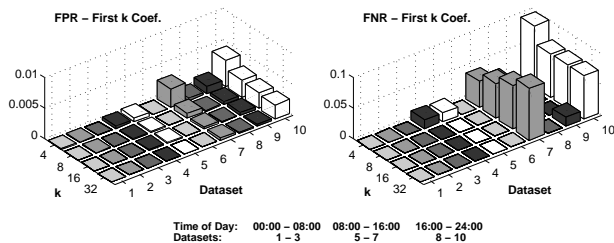


Figure 3: False positive (FPR) and false negative (FNR) rates for detecting various email worms over various datasets. Our method exhibits false positive and false negative rates less than 1% and 7%, respectively, for every dataset and value $k > 4$.

anomaly detection system. This is, namely, that the normal instances account for an overwhelmingly large portion of the dataset in relation to anomalous instances. With respect to this, our method identifies an email worm-infected machine, when its feature vector belongs to the sparsely-populated cluster.

We use the false negative and false positive rates to assess our method. For the first experiment, the two rates refer to detecting one machine being sequentially infected with various email worms, and not to detecting individually each email worm, which will be the case later. To put it differently, false negatives occur when the method fails to detect machines infected with some of the 71 email worms, whereas false positives when non-infected user activity is classified as suspicious. Fig. 3, shows that the method exhibits false positive and false negative rates that remain under 1% and 7%, respectively, for every dataset and feature vector length $k > 4$. This implies that the method can distinguish between legitimate traffic and traffic originating from machines infected with various email worms with remarkable accuracy and negligible false positive rate.

Up to this point we focused on the overall detection of various email worms. In what follows, we attempt a closer look at the detection of each worm and present only results with $k = 8$. For this second experiment, the notions of false positive and false negative rates are different than those considered so far. Specifically, hereafter, false negatives occur when a *specific* email worm instance is not detected, and false positives when the DNS activity of non-infected user machines is misclassified as email worm-infected.

In Fig. 4, we give the mean over 100 iterations for the false positive and false negative rates of our method per worm. As previously mentioned, in each iteration the number of infected computers, which computers are infected, and at which time bin the infection takes place are randomly chosen. The figure shows that our method exhibits mean false positive rate less than 1% for every worm and dataset, which implies that the number of non-infected user machines that our method misclassified as infected is negligible. The mean false negative rate figure shows that our method detects at least one instance of every email worm, and that for more than 75% of all email worms the false negative rate is less than 5%.

5 Conclusions and Future Work

Email worms and the high amount of unsolicited email traffic associated with them remain serious security issues. The methods that have been proposed to date to detect email worm-infected machines and mitigate the potential extent of email worm outbreaks either exhibit limited detection efficacy or do not target reducing the unwanted

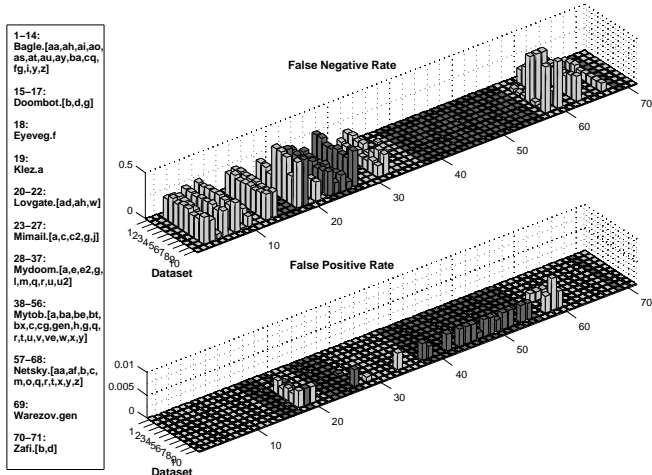


Figure 4: Mean over the 100 iterations false negative and false positive rates independently for each email worm against ten datasets. For every email worm at least one instance is detected, and the false positive rate is negligible.

email traffic traversing the Internet. In this paper, we go a step beyond earlier work and present a method for detecting email worms early as they appear on the local name servers, which are topologically near the infected machines. In connection with this, our method can contribute to limiting the abusive traffic that infected machines send to the Internet. The presented method is based on analysing DNS query streams at flow level by using unsupervised learning and similarity search over time series. Our experimental results demonstrate that the presented method identifies machines infected with various email worms with remarkable accuracy.

Future work calls for analysing DNS data from other networking environments to assess the detection efficacy of our method over DNS data that might have different flow-level characteristics. Moreover, we plan to study containment methods, such as limiting the rate at which local name servers send DNS responses to infected machines, that can be actively applied to slow down the email worm epidemics, once an email worm-infected machine is detected.

Author Biographies

Nikolaos Chatzis is a PhD student at the Technical University Berlin, and is currently finalising his thesis on network security. Since 2006, he has been working as a researcher of the Competence Center NET at the Fraunhofer Institute for Open Communication Systems (FOKUS). His current research activities concentrate on behavioural examination of DNS usage by means of data mining and mathematical modelling with the goal to leverage Internet security by enhancing name servers with proactive and reactive capabilities.

Radu Popescu-Zeletin is professor at the Technical University Berlin and Director of the Fraunhofer Institute for Open Communication Systems (FOKUS), Berlin. Prof. Popescu-Zeletin graduated at the Polytechnical Institute Bucharest, Romania, got his PhD from the University of Bremen, and his habilitation from the Technical University Berlin. For many years he led the research and development department of the BERKOM project of the German Telekom, pilot project for the development of new applications in broadband ISDN environment. He published many papers on distributed computing

systems and applications. He was active in standardization committees and had contributed to the development of telecommunication standards. He is founder of several telecommunication companies and member of advisory boards. He is Senior Member of IEEE, Doctor honoris causa of the Polytechnical Institute Bucharest, Romania and Professor honoris causa of the Catholic University of Campinas, Brasil. He is member of the Romanian Academy as well as bearer of the Public Service Medal of the Republic of Romania.

References

- [1] C. Aggarwal, A. Hinneburg, and D. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *ICDT 2001: Proc. of the 8th Int. Conf. on Database Theory*, LNCS, pages 420–434. Springer, 2001.
- [2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *ICDT 1999: Proc. of the 7th Int. Conf. on Database Theory*, LNCS, pages 217–235. Springer, 1999.
- [3] M. Bhattacharyya, S. Hershkop, and E. Eskin. Met: an experimental system for malicious email tracking. In *NSPW '02: Proc. of the 2002 workshop on New security paradigms*, pages 3–10, New York, NY, USA, 2002. ACM.
- [4] H. Binsalleeh and A. Youssef. An implementation for a worm detection and mitigation system. *Proc. of the 24th Biennial Symposium on Communications*, pages 54–57, 2008.
- [5] M. Braverman. Behavioral modeling of social engineering-based malicious software. In *Virus Bulletin Conf.*, 2006.
- [6] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with chebyshev polynomials. In *SIGMOD '04: Proc. of the 2004 ACM SIGMOD Int. Conf. on Management of Data*, pages 599–610, New York, NY, USA, 2004. ACM.
- [7] K. Chan and A. W. Fu. Efficient time series matching by wavelets. In *ICDE 1999: Proc. of the 15th Int. Conf. on Data Engineering*, pages 126–133, Washington, DC, USA, 1999. IEEE Computer Society.
- [8] N. Chatzis. Motivation for behaviour-based dns security: A taxonomy of dns-related internet threats. In *SECURWARE 2007: Proc. of the Int. Conf. on Emerging Security Information, Systems, and Technologies*, pages 36–41, Los Alamitos, CA, USA, 2007. IEEE Computer Society.
- [9] N. Chatzis and N. Brownlee. Similarity search over dns query streams for email worm detection. In *AINA-09: Proc. of the 23rd Int. Conf. on Advanced Information Networking and Applications*, pages 588–595, Los Alamitos, CA, USA, 2009. IEEE Computer Society.
- [10] N. Chatzis, R. Popescu-Zeletin, and N. Brownlee. Email worm detection by wavelet analysis of dns query streams. In *CICS 2009: IEEE Symposium on Computational Intelligence in Cyber Security*, pages 53–60. IEEE Computational Intelligence Society, 2009.
- [11] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. Indexable pla for efficient similarity search. In *VLDB 2007: Proc. of the 33rd Int. Conf. on Very large data bases*, pages 435–446. VLDB Endowment, 2007.
- [12] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD '94: Proc. of the 1994 ACM SIGMOD Int. Conf. on Management of Data*, pages 419–429. ACM, 1994.

- [13] D. Geer. Behavior-based network security goes mainstream. *Computer*, 39(3):14–17, 2006.
- [14] J. Goodman, G. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Commun. ACM*, 50(2):24–33, 2007.
- [15] A. Gupta and R. Sekar. An approach for detecting self-propagating email using anomaly detection. In *Recent Advances in Intrusion Detection*, LNCS, pages 55–72. Springer, 2003.
- [16] M. Haahr. Random.org: True random number service. <http://www.random.org>.
- [17] 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: Proc. of the 2005 ACM SIGCOMM Workshop on Mining Network Data*, pages 159–164, New York, NY, USA, 2005. ACM.
- [18] K. Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *SIGMOD Rec.*, 27(2):166–176, 1998.
- [19] Kaspersky Lab. Monthly Malware Statistics. <http://www.viruslist.com>.
- [20] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: an introduction to cluster analysis*. Wiley, 1990.
- [21] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inf. Syst.*, 3(3):263–286, 2001.
- [22] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD '01: Proc. of the 2001 ACM SIGMOD Int. Conf. on Management of Data*, pages 151–162, New York, NY, USA, 2001. ACM.
- [23] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining Knowledge Discovery*, 7(4):349–371, 2003.
- [24] D. Kienzle and M. Elder. Recent worms: a survey and trends. In *WORM '03: Proc. of the 2003 ACM workshop on Rapid malware*, pages 1–10, New York, NY, USA, 2003. ACM.
- [25] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD '97: Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of Data*, pages 289–300, New York, NY, USA, 1997. ACM.
- [26] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT 2004: Proc. of the 9th Int. Conf. on Extending Database Technology*, LNCS, pages 106–122. Springer, 2004.
- [27] S. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.
- [28] 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.
- [29] Messaging Anti Abuse Working Group. Email Metrics Reports. <http://www.maawg.org/about/EMR/>.
- [30] K. Mitnick. *CSEPS Course Workbook*. Mitnick Security Publishing, 2004.
- [31] F. Mörchen. Time series feature extraction for data mining using dwt and dft. Technical Report No. 33, Dept. of Maths and CS, Philipps-U. Marburg, 2003.
- [32] Y. Musashi, R. Matsuba, and K. Sugitani. Indirect detection of mass mailing worms-infected pc terminals for learners. In *ICETA 2004: Proc. of the 3rd Int. Conf. on Emerging Telecommunications Technologies and Applications*, pages 233–237, 2004.
- [33] Y. Musashi and K. Rannenber. Detection of mass mailing worm-infected pc terminals by observing dns query access. *IPSJ SIG Notes*, pages 39–44, 2004.
- [34] I. Popivanov and R. Miller. Similarity search over time-series data using wavelets. In *ICDE 2002: Proc. of the 18th Int. Conf. on Data Engineering*, pages 212–221, Washington, DC, USA, 2002. IEEE Computer Society.
- [35] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.
- [36] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland. Characterizing the performance of network intrusion detection sensors. In *Recent Advances in Intrusion Detection*, LNCS, pages 155–172. Springer, 2003.
- [37] 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 *Proc. of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 245–252, Berkeley, CA, USA, 2001. USENIX Association.
- [38] B. Smith. A storm (worm) is brewing. *Computer*, 41(2):20–22, 2008.
- [39] S. Stolfo, S. Hershkop, C. Hu, W. Li, O. Nimeskern, and K. Wang. Behavior-based modeling and its application to email analysis. *ACM Trans. Interet Technol.*, 6(2):187–221, 2006.
- [40] E. Stollnitz, T. Deroose, and D. Salesin. *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
- [41] Virus Radar. Top 10 threats. <http://www.virus-radar.com>.
- [42] 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 CS, 2005.
- [43] C. Wong, S. Bielski, J. McCune, and C. Wang. A study of mass-mailing worms. In *WORM '04: Proc. of the 2004 ACM workshop on Rapid malware*, pages 1–10, New York, NY, USA, 2004. ACM Press.
- [44] M. Xie, H. Yin, and H. Wang. An effective defense against email spam laundering. In *CCS 2006: Proc. of the 13th ACM Conf. on Computer and Communications Security*, pages 179–190, New York, NY, USA, 2006. ACM.
- [45] J. Xiong. Act: attachment chain tracing scheme for email virus detection and control. In *WORM '04: Proc. of the 2004 ACM workshop on Rapid malware*, pages 11–22, New York, NY, USA, 2004. ACM.
- [46] B. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *VLDB 2000: Proc. of the 26th Int. Conf. on Very Large Data Bases*, pages 385–394, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
- [47] J. Zhang, Z.-H. Du, and W. Liu. A behavior-based detection approach to mass-mailing host. *ICMLC 2007: Proc. of the Int. Conf. on Machine Learning and Cybernetics*, 4:2140–2144, 2007.