

Similarity Search over DNS Query Streams for Email Worm Detection

Nikolaos Chatzis
Fraunhofer Institute Fokus
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany
nikolaos.chatzis@fokus.fraunhofer.de

Nevil Brownlee
The University of Auckland
Auckland, New Zealand
nevil@auckland.ac.nz

Abstract

Email worms continue to be a persistent problem, indicating that current approaches against this class of self-propagating malicious code yield rather meagre results. Additionally, these approaches are intrinsically incapable of reducing the high amount of the unwanted email traffic on the Internet because they are deployed in the network of the potential victims. In this work we present a method to detect email worms soon after they appear at the local name server, which is topologically near the infected machines, by analysing at flow level the Domain Name System (DNS) traffic of user machines. Our method uses exact similarity search over time series produced by DNS query streams that user machines generate, and cluster analysis. To evaluate our method, we have constructed and used a DNS query dataset that consists of 71 recent email worms¹, and demonstrate that our method is remarkably effective in detecting email worm activity in the long run. As a secondary result, our work highlights that time series similarity search can be a useful tool for intrusion detection.

1 Introduction

Email is an essential form of communication in today's connected society, and as such, it has become the dominant medium for propagating viruses and worms [1]. In this paper, we use the concept *email worm* to cover every email-borne malicious program. Spreading malicious code using email is an old but still widely used technique because it offers several advantages [2]. It eliminates the need for finding new vulnerabilities, which makes it easy for worm writers to program and release their code. Moreover, the

¹To make all the experimental results we present reproducible, the email worm DNS dataset is publicly available by contacting the authors.

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).

spreading rate of email worms constantly rises, as the number of people that use email communication on a regular basis increases. Additionally, the high incidence [3–5] of email worms indicates that current approaches for mitigating this threat yield only meagre results.

Further, email worm writers and spammers are steadily joining forces to automate delivery of spam by using worms to hijack user machines to turn them into spam-bots. Spam-bots are the principal source of spam [6]. Operators become increasingly concerned as abusive email traffic on the Internet, which currently oscillates above 80% of all email traffic [7], escalates because it causes network congestion, instabilities, and increased resource utilization in infrastructural components. From the end users point of view, the problem is stabilizing, as only a fraction of that 80% actually reaches their inbox [8]. However, email worms remain for them a serious security threat as they are used to deliver Trojans, spyware and targeted phishing attacks.

Current approaches for dealing with email worms are antivirus, antispam packages and rule based filtering at the desktops, mail exchange servers and at the gateways. Antivirus and antispam packages are predominately signature-based, which means that they scan email messages for byte sequences that match known patterns of malicious code. The most well-cited disadvantages of signature-based detection are that it is ineffective against novel worms or worms that alter their code as they propagate, and that developing and distributing signatures requires a great deal of time. Likewise, content-based email filtering has drawbacks [9]. Filtering rules must be continually updated because worm writers can easily confuse filters by using random noise, bad grammar and hidden HTML code. Despite their limitations, these methods are still useful for detecting email worms. However, they go to no lengths to proactively reduce the unwanted email traffic traversing the Internet, which is a serious security concern, because they can only detect abusive email traffic at the receiving end.

In this work, we propose a method for detecting email worms as early as possible, so as to protect users from email

worm attacks, and also to reduce abusive email traffic. We concentrate on the flow-level communication patterns between user machines and the local name server, and show that the DNS traffic of non-infected user machines share many of the same canonical behaviours, whereas email worms rely on spreading methods generating DNS traffic that share different common patterns. To do this, we see DNS query streams that user machines generate as time series, and use exact similarity search and clustering. We argue that exact similarity search, which has attracted increasing interest within the database and data mining community [10–19], can be a useful tool for exploratory data analysis of Internet signals for intrusion detection.

In the rest of this paper, we first discuss related work in Section 2. In Section 3, we present briefly exact similarity search, presenting all the time series representations that are suitable for this task. Then, we explain how we use time series similarity search and clustering for detecting email worms. In Section 4, by experimenting with various email worms that appeared in the wild we search for the best time series representation for our task, and demonstrate that our method is remarkably effective for detecting email worm activity in the long run. Then, we show that it outperforms other methods that look at DNS traffic volume or self-similarity. In Section 5, we discuss methods an email worm writer could employ to evade our detection method and anticipate their effectiveness. We state our conclusions and discuss plans for future work in Section 6.

2 Related Work

In the past, much effort has been devoted to analysing the traffic that email worm-infected user machines generate [20–25]. These studies share the view that once a user machine gets infected its DNS traffic characteristics at the application layer change. Wong et al. [20] analyse DNS traffic captured at the caching name server of a campus network during the outbreak of Sobig.F and Mydoom.A. Musashi et al. [21] report on similar measurements for the same email worms; and they extend their scope in subsequent work by studying Netsky.C [22], Netsky.Q and Mydoom.S [23]. Whyte et al. [24] focus on enterprise networks, and measure the DNS activity of Netsky.Q over ten minute periods. The detection methods presented in [21–24] concentrate on application layer detection based on traffic volume. They suggest that many queries for Mail eXchange (MX) resource records (RR), or the relative numbers of queries for pointer (PTR) RR, MX and address (A) RR of a user machine give a strong basis for detecting email worms. Despite the positive contribution of defining such a correlation between email worm infection and DNS traffic, these studies have only modestly contributed to developing deployable mitigation mechanisms against email worms. Their main limitation is

that they focus on straightforward application layer detection suitable for detecting a few – mainly outdated – email worms studied in each paper, and their observation basis cannot be generalized for detecting various email worms. Moreover, as with any other volume-based method, these methods need an artificial boundary to serve as a threshold to determine whether a user machine is infected or not. Ishibashi et al. [25] propose a method to detect email worm-infected user machines by means of Bayesian inference based on a priori knowledge of email worm signature DNS queries. However, if such knowledge were apparent it would allow straightforward detection. Furthermore, all the aforementioned methods do not take into account that DNS queries carry user sensitive information, and are not suitable for high speed networks because they introduce high processing overhead by analysing packet payloads.

3 Proposed Approach

In this paper, we concentrate on the flow-level characteristics of DNS traffic generated by user machines. We show that non-infected user machines share many of the same canonical behaviours, whereas email worms rely on spreading methods generating DNS traffic that shares different common patterns. We demonstrate that these behaviours are clearly dissimilar, and remain unaltered in the long run, which gives a strong basis for detection of various (even new, unseen) email worms at local name servers. To do this, we employ exact time series similarity search and clustering. In this section, we first discuss the necessary background on exact similarity search over time series data, and then explain our approach.

3.1 Exact Time Series Similarity Search

A time series is a sequence of values of a variable, which are called time points, taken in successive periods of time. When the time points are equally distant the time series is called *univariate*. Time series similarity – hereafter referred to as similarity – search is useful in its own right as a tool for exploratory data analysis, and is also an important element of many data mining applications [15]. The similarity between two time series is typically measured using the Euclidean distance. To compute the pairwise Euclidean distance each time series of length p is seen as a point in the p -dimensional space, with each time point corresponding to one dimension. However, working with all the time points makes it impossible for clustering algorithms to find meaningful groups because in high dimensional spaces the significance of distance metrics becomes questionable [26].

The most promising way to attack this problem is to transform the time series into a representation that facilitates extracting a feature vector. The feature vector is a compressed low-dimensional representation of the time series,

which retains only the important information, and is used as input to the clustering algorithms. Numerous representations have been proposed but only a few of them that do not introduce *false dismissals* are suitable for our framework. False dismissals occur when feature vectors of similar time series appear distant in the feature vector space. Based on whether working with the feature vectors introduces false dismissals or not, similarity search is characterized as *approximate* or *exact*, respectively. That said, we are interested in exact similarity search. Faloutsos et al. [10] proved that a representation is suitable for exact similarity search if it satisfies: $D_{LR}(F(TS_i), F(TS_j)) \leq D_{TS}(TS_i, TS_j)$; where TS_i, TS_j are time series, $F()$ is the feature vector extraction function and D_{TS}, D_{LR} are distance metrics in the time series and feature vector space, respectively. This lemma is known as the *lower bounding lemma*.

In their original work [10], Faloutsos et al. proved that the Discrete Fourier Transform (DFT) representation satisfies the lower bounding lemma. Subsequent work demonstrated that other real-valued representations that satisfy the lower bounding lemma are: the Discrete Wavelet Transform (DWT) [12], the Piecewise Aggregate Approximation (PAA) [13, 14], the Adaptive Piecewise Constant Approximation (APCA) [15], the Piecewise Linear Approximation (PLA) [16], the Chebyshev Polynomials (CP) [17], and the Singular Value Decomposition (SVD) [18, 19]. Finding the best representation is field-specific because all the representations have well-cited strengths and weaknesses that depend on the characteristics of the examined data.

3.2 Time Series Representations

The basic idea of DFT is that any time series of length p can be represented by the superposition of p sine/cosine waves. Each wave is represented by a complex number known as a Fourier coefficient. Each Fourier coefficient has an imaginary and a real part, thus decomposing a time series using DFT produces a $2p$ -dimensional representation. Three methods have been proposed to extract a k -dimensional feature vector, with $k \ll p$. The first method suggests keeping the first $k/2$ Fourier coefficients, which correspond to the low frequency components of the time series. Most real time series have their energy concentrated in their low frequency components, thus discarding high frequency components does not introduce much information loss. Rafiei et al. [11] propose retaining the first and last $k/4$ DFT coefficients because the DFT representation of a time series is symmetric with respect to its middle, thus the first and last $k/4$ coefficients are equally important. Mörchen [27] uses the largest $k/2$ DFT coefficients as feature vector because they carry most energy.

Chan et al. [12] used the DWT to extract a feature vector. Similar to DFT, the DWT approximates a time series by a

superposition of basis functions, which are produced by dilations and translations of a wavelet prototype function. The DWT representation is intrinsically multi-resolution, and the wavelet coefficients are localized in time. The DWT decomposes a time series of length p – a p power-of-two – into $\log_2 p$ subsequent frequency levels, and produces p wavelet coefficients. Retaining the k largest coefficients in absolute normalized value produces for a given k the optimal DWT feature vector in terms of sum squared error. Another well established feature vector extraction method suggests retaining the first k wavelet coefficients, which capture the low frequency information of the time series.

Keogh et al. [13] and Yi et al. [14] independently introduced the PAA representation to search for similar time series. PAA partitions a time series of length p into k equal-length segments, and then considers the k mean values of these segments as the representation of the time series. In contrast to DFT and DWT, PAA produces directly a k -dimensional representation, thus no supplementary feature extraction function is required.

Keogh et al. [15] proposed to relax the requirement of PAA for equal-length segments. The rationale of APCA is to vary the number of segments used to approximate each part of the time series based on its activity. APCA suggests that a time series of length p can be represented by k values, where these values are produced by approximating the time series by $k/2$ subsequent constant value segments of arbitrary length, chosen so that the individual reconstruction error for each part is minimal. The number of segments is $k/2$ to get a k -value representation because APCA requires two values for each segment: one for the constant value and one for the segment length. As PAA does, APCA produces directly a k -dimensional representation.

Chen et al. [16] have used the PLA representation of time series to search for similar time series. PLA segments the time series in $k/2$ equal-length segments and then approximates each segment i with a best-fit linear regression line $y_i = a_i x + b_i$. The time series is then represented by the coefficients a and b of all the segments. Similar, to APCA to produce a k -value representation $k/2$ regression lines are required because PLA requires for each segment i two coefficients: the slope a_i and the y -intercept b_i . Similarly to PAA and APCA, PLA does not need a supplementary feature extraction function.

Cai et al. [17] proposed searching for similar time series using CP. The basic idea is that the polynomial approximation of a time series using first kind Chebyshev polynomials is almost identical to the optimal *minimax polynomial*. The minimax polynomial has the smallest maximum deviation from the original time series. They suggested approximating a time series of length p with the Chebyshev polynomial of degree p , and using the p coefficients of the polynomial as the time series representation. To reduce the dimensions the

first k Chebyshev coefficients are selected as feature vector.

Kanth et al. [18] and Korn et al. [19] used the SVD to search for similar time series. In contrast to the previously presented transformations, SVD operates on a set of time series rather than on each time series independently. Given that each time series with p time points can be seen as a point in a p -dimensional space, SVD examines a set of n time series, and rotates the p axes to maximize variance along the first few dimensions. The formal definition of SVD is $A = U\Sigma V^T$ with A the $n \times p$ time series matrix, U and V orthogonal matrices with dimensions $n \times r$ and $p \times r$ respectively, r the rank of A ; and Σ the $r \times r$ diagonal matrix with elements the singular values of A in descending order. To compress the p -length time series to k -length feature vectors, only the k largest singular values are considered. Thereby, the compressed SVD data are $U_k\Sigma_k$; U_k is derived from U if we zero all elements except for those in the first k columns.

3.3 Detection Method

Once an email worm infects a user machine it compiles a list of email addresses, and then uses its built-in Simple Mail Transfer Protocol (SMTP) engine to send out email messages. To compile the list the worm harvests aggressively the email address book and files with predefined extensions. This involves, guessing addresses by combining commonly used names e.g., `webmaster@` with domains it finds on the infected machine, and searching for email addresses to Web pages visited by the user. Sending out email messages relies on communicating the local name server to find the recipients SMTP server. Intuitively, this implies a change in the characteristics of the DNS traffic of the email-worm infected machine at flow level i.e., DNS queries generated in time. This effect is amplified as the sending process is often repeated (according to some prearranged pattern or when triggered by a system event); also because email worms often spread over secondary communication channels offered by applications that contain rich social information; and finally, because worms make use of DNS information to successfully execute their payload.

In this work, we see the DNS query streams that user machines generate as time series, and show by using exact similarity search and clustering that user machines DNS activity falls into two canonical profiles: *typical user* behaviour determined by legitimate activity, and *infected by email worm* behaviour. As input, our method uses the complete set of DNS queries that a local name server received within an observation interval. Each query consists of the time when the server received the query, the requesting host – identified by its IP address – and the requested data. We retain for each query only the time of the query and the IP address of the requesting host. We group queries per requesting host, we consider p time bins of equal length, and

for each bin we count the DNS queries. Thereby, we produce a univariate p -length time series for each host. The set of time series can be expressed as an $n \times p$ time series matrix; n is the number of user machines that queried the server at least once within the observation interval. We apply to the time series matrix one of the transformations discussed above, and by that means we produce a $n \times k$ feature vector matrix.

Using cluster analysis we infer both the number and nature of distinct underlying populations in the feature vector matrix. Defining the number reveals how many distinct DNS activity classes exist in the examined data, whereas examining each class indicates whether we can distinguish between non-infected and worm-infected activity. Since the best number of clusters is part of the problem, it cannot be defined a priori; therefore, we use hierarchical clustering. Hierarchical clustering methods are categorized into *agglomerative*, which start with each observation in its own cluster and recursively merge the less dissimilar clusters, and *divisive*, which start with all observations in one cluster and subdivide them into smaller clusters until each cluster has only one observation. In this work, we expect that DNS query streams from email worm-infected user machines can be easily separated from the set of DNS query streams generated by legitimate users. Intuitively, this means that we search for a small number of clusters, therefore we use the only divisive method generally available i.e., the well-established DIvisive ANALysis (DIANA) [28]. Although DIANA is of $O(n^2)$ time complexity, we use it because in our framework it achieves clustering on a personal computer in computing time in the order of milliseconds.

Given the output of DIANA, we test if our hypothesis (that the best clustering of the feature vectors is this grouping the feature vectors into two clusters) holds. Criteria that are used to select the hierarchical level on which to base inferences concerning true population differences have been characterized as stopping rules, and numerous such rules have been proposed. Stopping rules assess for a given hierarchical clustering algorithm the clustering structure produced in each hierarchical level by comparing it to the clustering structures that are produced in every other level. Although significant effort has been devoted to assess stopping rules, little in the way of general gold standards exist that are suitable for revealing the optimal number of underlying populations over datasets from diverse fields. Therefore, we use four widely used stopping rules: the Connectivity [29], the Dunn [30], the Silhouette [28] and the Davies-Bouldin [31] indices.

4 Experimental Evaluation

Our experimental evaluation involves three steps. First, using the four stopping rules we examine which represen-

tations validate our hypothesis that DNS query streams fall into two canonical profiles, and if this is the case how many coefficients are needed. The best representation should reveal two canonical profiles with the minimum number of coefficients k . Then, for the representations that qualify the first step we measure their detection accuracy using for each dataset and value k the mean false negative and false positive rates over the 71 worms. Finally, we demonstrate that comparing traffic signals in terms of shape-based similarity captures more information on the signal structure than methods that look at volume and self-similarity.

4.1 Datasets and Parameters

Rather than launching email worms in the real network, we set up an isolated computer cluster, where we launch 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 [3] and Viruslist [4]. To ensure that the worms run in our isolated cluster as if they were infecting machines in the real network, we have copied the file system of a user machine from the real network to the isolated computers. Additionally, we direct all the SMTP traffic of the infected machines to an SMTP server, which we have configured as administrator for a set of domains but never forwards emails to end users. Moreover, by reverse code engineering a subset of the worms, we found that no functionality depends on the portion of emails successfully delivered or on the number of DNS queries resolved. We capture over a period of eight hours the DNS traffic of the infected machines to create – to the best of our knowledge – the largest and most complete email worm DNS dataset that has been used to date to evaluate an in-network email worm detection method. Because our isolated cluster has no real users, we merge the worm query streams with legitimate DNS traffic captured at the primary name server of our research institute. 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. We consider 15-second time bins, and make from the 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.

In this study, we focus on assessing the detection efficacy of our method to detect various worms using different time series representations; therefore, we assume that only one user machine is infected at each time. Although we have conducted experiments with more worm-infected user machines being infected at various times, and the results are very promising, we leave reporting on those results for subsequent work. Therefore, for each dataset we append to the legitimate users’ time series matrix one infectious time series to make the $(n + 1) \times 256$ time series matrix; n is the number of non-infected user machines. We apply DFT,

DWT, PAA, APCA, PLA, CP and SVD to the time series matrix to produce the $(n + 1) \times k$ feature vector matrix. We experiment with different feature vector lengths k – i.e. 4, 8, 16, 32 and 64. Then, we cluster the rows of the feature vector matrix to compute the first ten levels of the cluster hierarchy. The experimental procedure is illustrated in Fig. 1.

4.2 Experimental Results

In Table 1, we show the average – over the ten datasets – percentage of the 71 worms for which the stopping rules indicate that the two-cluster scheme is the best clustering. This implies that the 100 value in the first top left cell of the table means that Connectivity reveals that in average over the datasets the two-cluster scheme is the best clustering for all the email worms examined, when DFT is used and the feature vector length k is four. All four stopping rules uncover a clear dominance of the two-cluster scheme for all the representations. In particular, for APCA, PLA, CP and SVD even for $k = 4$ all the rules indicate that in average more than 95% of the worms the two-cluster scheme is the best describing the underlying populations in the feature vector matrix. For this performance, DWT and PAA require that $k \geq 16$, whereas DFT $k \geq 32$.

We then examine the members of the two clusters, and find that one cluster is densely populated, and the other is sparsely-populated. Since we are dealing with unlabelled data, we need a way to determine which cluster contains data that represent legitimate activity and which contains worm feature vectors. To do this, we uncover an assumption that intrinsically holds for every anomaly detection system. Namely that the normal instances account for an overwhelmingly large portion of the data set in relation to anomalous instances; thereby, the cluster that contains legitimate traffic will contain a much larger number of observations than this of worm-infected machines. That said, our method detects an email worm, when its feature vector belongs to the sparsely-populated cluster. In Fig. 2 we present for DFT, DWT, PAA, APCA, PLA, CP and SVD the mean false positive and false negative rates to detect various worms. We primarily concentrate on APCA, PLA, CP and SVD as candidates for the best representation because they outperform DFT, DWT and PAA in revealing the two-cluster scheme. The figure shows that all representations exhibit comparable false positive rates that account for less than 1% for every dataset and feature vector length k . In particular, SVD performs slightly better than the other methods; however, SVD cannot be considered as a candidate representation for a deployable detection system. First, because it is far more expensive than the other representations i.e., $O(np^2)$ time complexity, using the classic algorithm, which is intractable for even moderately sized sets of time series. Second, because it operates on a set of time series rather than on each time series independently, which

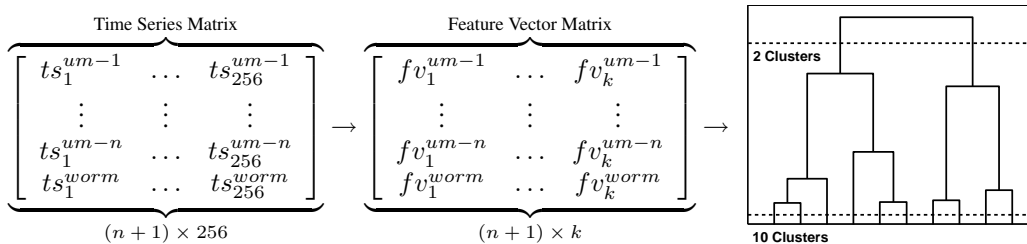


Figure 1. To every legitimate users’ time series matrix , we append one infectious time series; we apply DFT, DWT, PAA, APCA, PLA, CP and SVD, to get the feature vector matrix. Using DIANA we cluster the rows of the feature vector matrix to produce the first ten levels of the cluster hierarchy.

Table 1. The % of email worm instances averaged over the ten datasets for which the stopping rules Connectivity (C.), Dunn (D.), Silhouette (S.) and Davies-Bouldin (DB) indicate that only two distinct underlying populations exist in the feature vector matrix. For APCA, PLA, CP, and SVD even if $k = 4$, all the stopping rules show that for more than 95% of the worms there exist only two groups. For the same result DWT and PAA require that $k \geq 16$; whereas DFT needs that $k \geq 32$.

	$k = 4$				$k = 8$				$k = 16$				$k = 32$				$k = 64$			
	C.	D.	S.	DB	C.	D.	S.	DB	C.	D.	S.	DB	C.	D.	S.	DB	C.	D.	S.	DB
DFT	100	91.7	97.3	90.8	100	96.5	97.6	92.3	100	96.6	98.7	93.4	100	95.8	99.2	95.1	100	95.9	99.3	95.6
DWT	100	93.2	97.5	92.4	100	94.8	99.4	94.2	100	98.7	99.7	95.6	100	96.5	99.9	96.2	100	96.8	99.7	96.6
PAA	100	93.2	97.5	92.4	100	94.8	99.4	94.2	100	98.7	99.7	95.6	100	96.5	99.9	96.2	100	96.8	99.7	96.6
APCA	100	96.8	99.7	95.1	100	98	99.9	95.6	100	99	100	97.9	100	98.6	100	97.5	100	99.2	100	98
PLA	100	97.9	99.9	97.5	100	98	100	97.5	100	98.9	100	97.5	100	98.6	99.9	98.2	100	98.3	100	97.5
CP	100	96.3	99.2	95.5	100	96.3	99.9	96.1	100	96.6	99.7	96.5	100	96.8	98.9	96.6	100	97.9	99.3	97.6
SVD	100	95.9	96.2	95.1	100	94.5	96.1	91.7	100	97.7	98.3	96.1	100	98	98.9	97.6	100	98.2	99.6	98

means that the whole feature vector matrix has to be re-computed each time a machine enters or leaves the network. APCA and PLA outperform CP because their false negative rate for every dataset and $k \geq 8$ are lower than 3%, whereas for Dataset5 and Dataset10 the false negative rate for CP oscillates around 7%. Moreover, CP is of higher complexity i.e., $O((k)p)$; p is the length of time series and k is the feature vector length, than APCA and PLA i.e., $O(p)$. In any case, using APCA, PLA and CP exhibits remarkable accuracy and negligible false positive rate in detecting various email worms.

Finally, we demonstrate in Fig. 2 that comparing traffic signals in terms of shape-based similarity captures more information on the signal structure than methods that look solely at the mean number of queries, or at large spikes or at the self-similarity degree. We consider two volume threshold based methods and a method that looks at the self-similarity of the time series. Let r_{max} be the maximum rate at which a user machine generates DNS queries during a one-hour dataset; the first method – denoted as MaxRate – recognizes as email worm-infected all user machines that that generate DNS queries with rate greater than $0.95 \times r_{max}$. The second volume-based method i.e., MaxValue, suggests detecting as email worm-infected all user machines that generate within a 15-second time bin at least one DNS query spike with value greater than 95% of the maximum spike observed in the one-hour dataset. The self-similarity based method – denoted as MaxHurst– detects as worm-infected all the user machines that generate time

series with self-similarity degree greater than 95% of the most self-similar time series. To measure the degree of self-similarity we estimate the Hurst parameter using the non-parametric estimator presented in [32], which is fast, and robust in the presence of non-stationarities. The figure illustrates that the three methods exhibit false positive rates lower than 2%, but the false negative rates are one order of magnitude higher than those of our method, which implies that significantly less email worms are detected.

5 Evasion of Proposed Method

Although most prevalent email worms are rewrites of previous worms [5], and thus generate similar DNS traffic with their ancestors, and we have considered various worms covering all the most prevalent worm families, in this section we go a step further. We foresee, in order of increasing sophistication, some approaches worm writers might take for evading detection, and anticipate their effectiveness.

Worm writers might try to evade detection with minimal changes such as avoiding querying the local name server, or sending out infected emails using the email client installed on the user machine. For the former a possible defence mechanism is to filter out at the network gateway all outgoing DNS traffic from non-name servers. Whereas, the latter is highly unlikely to (re-)appear in future worms because worm writers originally equipped email worms with an SMTP engine to bypass the ever-evolving detection mechanisms at the outgoing email servers because email worm

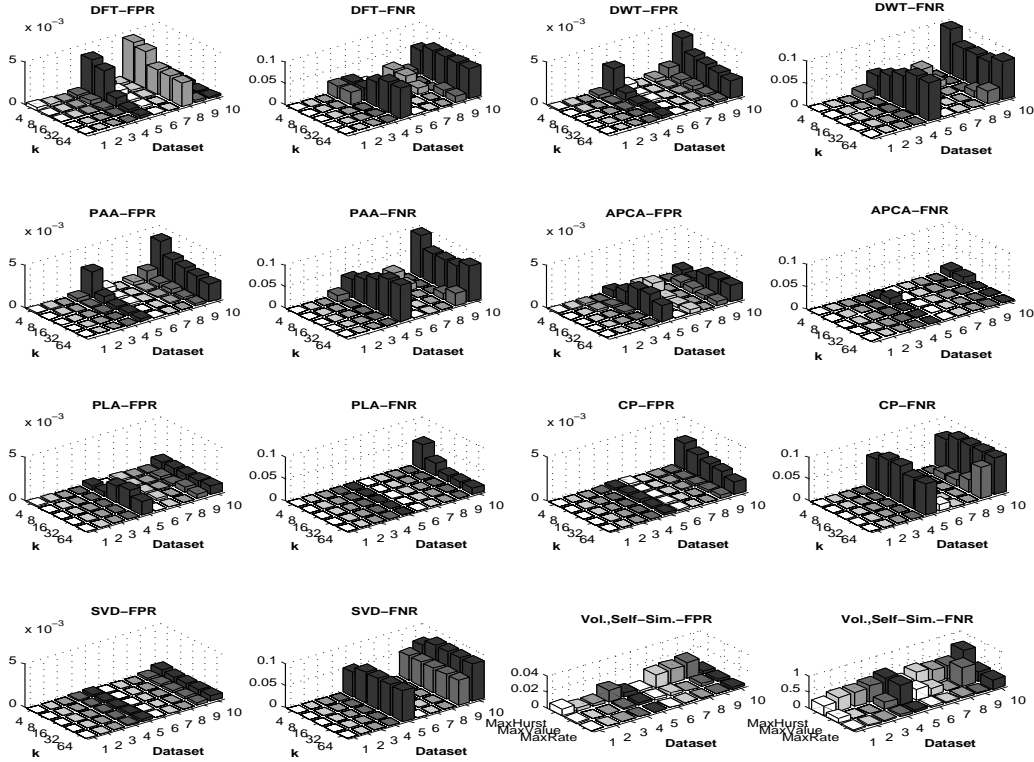


Figure 2. Mean false positive and false negative rates for detecting various email worm instances for DFT, DWT, PAA, APCA, PLA, CP and SVD over 71 worms. All the representations have negligible false positive rate less than 1% for every k . APCA and PLA perform better than the other representations as they have for $k \geq 8$ false negative rate less than 3%. Moreover, our method outperforms methods that look solely at traffic volume (MaxValue and MaxRate) or self-similarity (HurstMax).

propagation is observable on email servers [20, 24]. Furthermore, worm writers might write worms that query the local name server at lower rates, which implies sending DNS queries over a longer period of time. However, as we have shown in Section 4 by comparing our method to VolMaxRate, our method does not rely solely on the DNS query rate; therefore, varying the DNS query rate cannot directly affect the efficacy of our method.

More sophisticated worm writers might change their code to mimic either the behaviour of a legitimate email user e.g., send abusive emails to a subset of the email addresses found on the infected machine or send emails at a slower rate, or try to learn and mimic the typical DNS user behaviour. However, we question whether an email worm that behaves like this can avoid premature death and reach epidemic levels. Related work [2] shows that the behaviour of worms in their early spreading phase is critical to survive. At the same time, the evolution of email worms uncovers that worm writers are increasingly concerned about the hit rate of their worms in their early spreading phase; therefore, they constantly add more functionality i.e., worms use ever-evolving harvesting techniques; furthermore, they use application channels other than email to support their spreading. Intuitively, adding such functionality, which is a trend likely to continue as end-user awareness increases, and de-

tection mechanisms evolve, results in email worm-infected user machine behaviour that increasingly deviates from the normal user behaviour.

6 Conclusion and Future Work

Email worms and the spam associated with them are one of the main operational security issues today because they waste time, money and resources. Systems already deployed for combating email worms yield meagre results in reducing unwanted email traffic traversing the Internet because they detect abusive email traffic on the receiver’s domain. In this work we concentrate on the local name server, close to the email worm-infected user machine, and present a novel remarkable accurate detection method that uses time series similarity search and cluster analysis. We present a comprehensive study of time series representations suitable for exact shape-based similarity, and argue that similarity search over time series data can be a valuable tool for intrusion detection.

Future work calls for presenting thorough experiments assuming different numbers of email worm-infected machines to show that as the number of infected machines increases our method exhibits even better results. Moreover, we are interested in analysing DNS query data from other

networks, and plan to study countermeasures that can be actively applied to thwart the attack, such as blocking or rate limiting DNS responses to the infected user machine.

References

- [1] 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.
- [2] C. Zou, D. Towsley, and W. Gong. Modeling and simulation study of the propagation and defense of internet e-mail worms. *IEEE Trans. Dependable Secur. Comput.*, 4(2):105–118, 2007.
- [3] Virus Radar. Top 10 threats. <http://www.virus-radar.com>.
- [4] Kaspersky Lab. Monthly Malware Statistics. <http://www.viruslist.com>.
- [5] Virus Bulletin. Malware prevalence. <http://www.virusbtn.com>.
- [6] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.
- [7] Messaging Anti Abuse Working Group. Email Metrics Report, 2007. http://www.maaawg.org/about/MAAWG20072Q_Metrics_Report.pdf.
- [8] J. Goodman, G. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Commun. ACM*, 50(2):24–33, 2007.
- [9] V. Cerf. Spam, spim, and spit. *Commun. ACM*, 48(4):39–43, 2005.
- [10] 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.
- [11] D. Rafiei and A. Mendelzon. Efficient retrieval of similar time sequences using dft. In *FODO'98: Proc. of the 5th Int. Conf. on Foundations of Data Organization and Algorithms*, pages 249–257. Kluwer, 1998.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] K. Kanth, D. Agrawal, and A. Singh. Dimensionality reduction for similarity searching in dynamic databases. *SIGMOD Rec.*, 27(2):166–176, 1998.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] R. Matsuba, Y. Musashi, and K. Sugitani. Detection of mass mailing worm-infected ip address by analysis of syslog for dns server. *IPSI SIG*, pages 67–72, 2004.
- [23] Y. Musashi and K. Rannenber. Detection of mass mailing worm-infected pc terminals by observing dns query access. *IPSI SIG Notes*, pages 39–44, 2004.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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.
- [28] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: an introduction to cluster analysis*. Wiley, 1990.
- [29] J. Handl and J. Knowles. Exploiting the trade-off - the benefits of multiple objectives in data clustering. In *Evolutionary Multi-Criterion Optimization*, LNCS, pages 547–560. Springer, 2005.
- [30] J. Dunn. Well separated clusters and optimal fuzzy partitions. *Cybernet.*, 4:95–104, 1974.
- [31] D. Davies and D. Bouldin. A cluster separation measure. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 1(4):224–227, 1979.
- [32] K. Hipel and A. McLeod. *Time series modelling of water resources and environmental systems*. Elsevier, 1994.