SEVENTH FRAMEWORK PROGRAMME

Information & Communication Technologies
ICT

Cooperation Programme

Nippon-European Cyberdefense-Oriented Multilayer threat Analysis
[†]

Deliverable D2.1: Threat Analysis

Contractual Date of Delivery	November 30th 2014
Actual Date of Delivery	November 30th 2014
Deliverable Dissemination Level	Public
Editors	Adam Kozakiewicz, Romain Fontugne
Contributors	All *NECOMA* partners

The *NECOMA* consortium consists of:

Institut Mines-Telecom	Coordinator	France
ATOS SPAIN SA	Principal Contractor	Spain
FORTH-ICS	Principal Contractor	Greece
NASK	Principal Contractor	Poland
6CURE SAS	Principal Contractor	France
Nara Institute of Science and Technology	Coordinator	Japan
IIJ - Innovation Institute	Principal Contractor	Japan
National Institute of Informatics	Principal Contractor	Japan
Keio University	Principal Contractor	Japan
The University of Tokyo	Principal Contractor	Japan

[†] The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2013-EU-Japan) under grant agreement n° 608533, and the Strategic International Collaborative R&D Promotion Project of the Ministry of Internal Affairs and Communication, Japan.

Contents

1 Introduction

This deliverable is a detailed account of the different threat analysis, rating, and classification mechanisms developed by the members of the *NECOMA* consortium. It also reports the general architecture and design principles of the *NECOMA* threat analysis platform.

The **data aggregation and analysis** (Chapter 2) mechanisms are designed according to the various datasets collected by the *NECOMA* consortium, hence, for each data type and layer considered in the consortium at least one analysis method is studied. These methods are classified into four categories:

- Backbone and telescope traffic analysis (Section 2.1) consists mainly of anomaly detection algorithms to inspect significant traffic collected at the infrastructure layer, namely backbone links and telescopes. Usually based on unsupervised techniques, anomaly detectors identify first evidence of malicious activities.

- Large-scale DNS traffic analysis (Section 2.2) takes advantage of the ubiquitous Internet Domain Name System (DNS) to identify threats. As DNS is commonly utilized by most network applications, the analysis of DNS traffic enables us to systematically monitor and characterize various applications and distributed systems, especially malwares and botnets.

- End-point threat data analysis (Section 2.3) is the detailed examination of data collected from sensitive end systems that are prone to attacks. The developed techniques are usually bound to the mechanisms of specific applications or protocols and they yield concrete results for specific threats.

- Cross-layer threat analysis (Section 2.4) leverages the benefits of multiple datasets by performing transverse analysis of different data types.

Bridging datasets across different layers reveals new insights into threat analysis and is one of *NECOMA* main motivations.

Chapter 3 proposes **rating and classification** mechanisms to automatically sort and prioritize significant threats. As the *NECOMA* platform is expected to report suspicious events at a fast rate, this is a crucial task that permits to uncover threats of prime importance and address them first.

The advances presented in this document provide us with a better understanding of the requirements and constrains of *NECOMA* platform components. Therefore, Chapter 4 presents an updated version of the **architecture and design principles** of the *NECOMA* threat analysis platform, as well as the current hadoop subsystem implementation.

2 Development of threat data aggregation and analysis mechanisms

2.1 Backbone and telescope traffic analysis

Measuring Internet traffic at core infrastructures enables network operators to monitor activities of numerous connected devices and detect network-wide undesirable traffic. To identify the distinctive characteristics of malicious activities from the large amount of collected data, network operators usually rely on detection tools based on anomaly detection techniques. In a nutshell, these tools build a reference model corresponding to the normal traffic behavior and they report traffic deviating from the computed model.

NECOMA is leveraging these techniques to identify malicious activities in the data collected at backbone links and telescopes. The benefits of identifying threats in these datasets is to observe hosts behavior at a coarse scale and monitor remote hosts that are otherwise not accessible to the *NECOMA* consortium. Therefore, we expect to gain a broad understanding of observed malicious activities from our analysis of the backbone and telescope traffic that could be refined by the analysis of data from other layers.

The following presents part of the research achieved towards the analysis of the *NECOMA* backbone and telescope traffic. It includes two unsupervised anomaly detectors for backbone traffic, a MapReduce framework to detect anomalies in real-time, a visualization tools to assist in inspecting the detection results, as well as a taxonomy for telescope traffic.

2.1.1 Backbone Traffic Anomaly Detection with S-transform and Sketches

Overwhelmed by the increasing amount of IP traffic, network operators rely on anomaly detectors to automatically identify harmful events that occur on backbone networks. Unsupervised detection techniques are especially

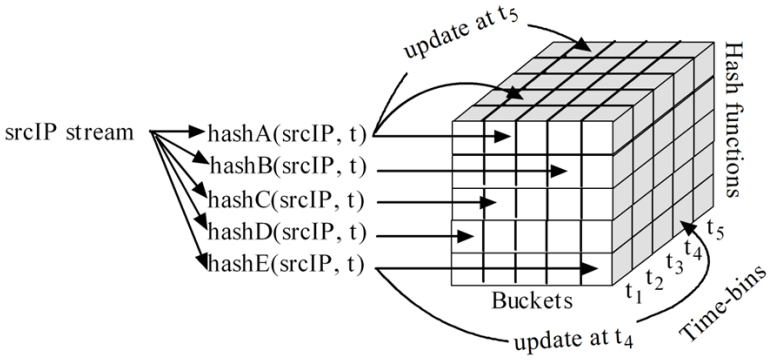


Figure 2.1: Constructing sketches by five hash functions.

attractive as they require no prior knowledge and are easier to deploy. In [73] we propose an anomaly detection method based on *entropy time-series*, *sketches* and the *S-transform*. Sketching traffic entropy time-series allows us to capture fine-grained traffic patterns of sub-streams. Then, S-transform analysis is performed on the sub-stream's entropy to detect anomalous traffic.

2.1.1.1 Sketches and Entropy

IP traffic stream is comprised of packets that have four basic attributes: source IP, destination IP, source port, and destination port. In this stage, an attribute stream, e.g., source IP stream, is divided into sub-streams by hashing. More specifically, all attributes in a time-bin are hashed independently by different hash functions, and stored in a sketch, which is a two-dimensional array: each row is associated to a hash function, and the columns are hash buckets, which store the attributes that have bucket number as hash keys. Figure 2.1 shows five continuous sketches that are being constructed in five time-bins by five hash functions, where the input is the source IP stream, and the number of buckets per hash function is five. Next, we compute the Shannon entropy of attributes in each bucket by $H(X) = -\sum_{n}^{i=0} p_i log_2 p_i$, where p_i is the probability of attribute x_i in the bucket, and it is calculated by the frequency of the attribute x_i divided by the frequency of all attributes in the bucket. The reason we consider the entropy instead of volumes of the attributes, e.g., total number of bytes associated with all attributes in a bucket, is that the entropy provides more fine-grained information of traffic data. Formally, we define *entropy signal* as a vector of the time-varying entropy of a bucket number.

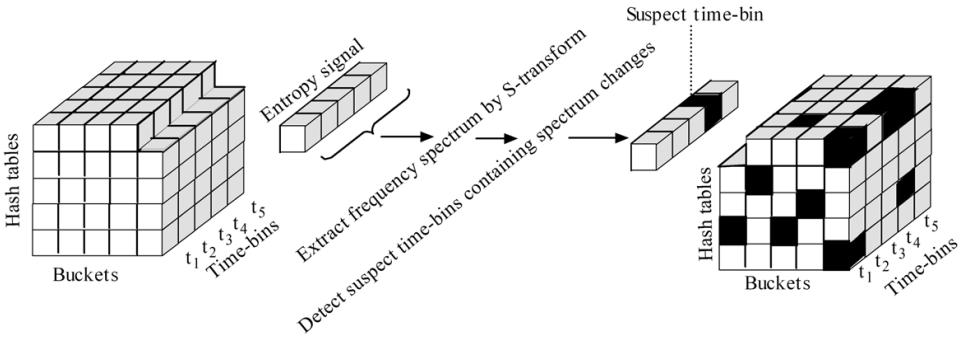


Figure 2.2: Detecting suspicious time-bins from the entropy signal of a sub-stream.

2.1.1.2 Detecting Suspicious Time-bins

This stage detects suspicious time-bins of each sub-stream. Typically, anomalies are defined as events that behave differently from major behavior. In others words, anomalies refer to changes. In this work, we do not detect changes in the entropy signal of a sub-stream but detect changes in the spectrum of entropy signal using S-transform, which is a time-frequency analysis tool like Wavelet transform but produces an output that is easier for analysis and retains absolute phase information of an input signal.

Firstly, the entropy signal is normalized by subtracting its mean value. Secondly, the S-transform converts the normalized signal and produces a matrix indicating frequency spectrum of the signal including time information. The columns of the matrix represent time-bins corresponding to time-bins of the sketches, the rows represent frequencies and the element is frequency amplitude. In order to determine suspicious time-bins, we produce two additional time-series that are obtained by vertically summing all matrix elements in: 1) the upper half, and 2) the lower half of the matrix. Time-bins in the time-series that hold values above a given upper threshold value or below a lower threshold value will be determined as suspicious time-bins of the particular sub-stream. The detection process is illustrated in Figure 2.2, in which the suspicious time-bins are highlighted in black. In order to accomplish this stage, all entropy signals are examined to identify suspicious time-bins.

2.1.1.3 Finding Intrinsic Culprits

The suspicious time-bins contain culprits of anomalies. In order to find these culprits, we combine all the attributes in the suspicious time-bins of each hash function by taking the union. Intrinsic culprits that are hidden among these suspicious attributes are determined by taking the intersection among

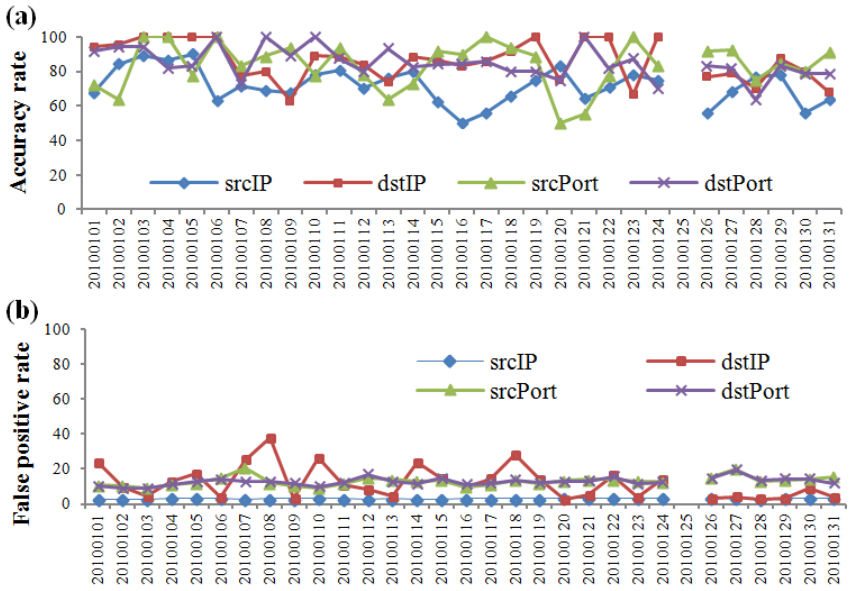


Figure 2.3: Accuracy rate (a) and False positive rate (b) in detecting anomalous source and destination IPs, source and destination ports in MAWI traces of January, 2010.

all the hash functions. The attributes in the intersection results are the intrinsic culprits of the traffic stream and are reported as malicious.

2.1.1.4 Evaluation

We evaluated our method with high-speed backbone traces of the MAWI archive [2] and used the MAWILab labels as benchmark [32]. We set the method parameters as follows: the number of hash functions to construct the sketches is *three*, and the number of hash buckets and time-bin size are set to 64 and *one second* respectively. In addition, two performance metrics were considered, i.e., detection accuracy and false positive rate, which are defined as follows,

- *Accuracy rate*, the ratio between the total number of anomalies that were correctly detected by our method and the total number of anomalies that were classified by the MAWILab.
- *False positive rate*, the total number of normal instances that were incorrectly detected as anomalies by our method and the total number of normal instances in the trace.

Figure 2.3(a) and (b) plot the accuracy and false positive rates in detecting anomalous source and destination IPs, source and destination ports in 30 traces collected in January, 2010. Note that the accuracy and false

positive rates of January 25's trace are not shown because the labels for this trace are unavailable. The plots show that the overall accuracy rate is above 60%, and in some traces, our method succeeds in detecting anomalies with 100% accuracy. The false positive rate in detecting anomalous source IPs is low and stable at about 3%. The false positive rates in detecting anomalous source and destination ports are about 12%. The false positive rate in detecting anomalous destination IPs is ambivalent.

2.1.2 Detection of synchronized sources

When a victim of a DDoS attack tries to protect himself, understanding the nature of the attack is always one of the first steps. Identification of IP addresses being part of the attack can be useful depending on the type of the attack, especially for the attacks that are,

- requiring the use of real IP addresses (e.g. attacks requiring the setup of TCP session to send application layer requests), and
- launched using botnets.

In this work we aim to provide one tool addressing the identification of machines taking part in the attack. The idea is to look at the volume of requests generated by a given source and compare it to the overall volume of observed requests and the hypothesis on which we build is that bots taking part in an attack have a behavior that is closely correlated to other bots, and to the macroscopic evolution of the request volume in a relatively short time window containing data for both before/after attack and during the attack.

We consider intervals of length t, use i as index of interval; and for each contributor c, we denote its contribution during interval i as $r_{c,i}$ and use $r_{c_k,i}$ for the number of requests for k^{th} contributor during the interval i. Now we can obtain the the covariance of contributor c's contribution with respect to the total amount of requests R_i during interval i in the following manner,

$$\begin{aligned}
\operatorname{cov}(c) &= \sum_{i=0}^{n} \left(r_{c,i} - r_c^0\right)\left(R_i - R^0\right), \text{ where} \\
r_c^0 &= \frac{1}{n+1}\sum_{i=0}^{n} r_{c,i}\,, \\
R^0 &= \frac{1}{n+1}\sum_{i=0}^{n} R_i\,, \text{ and} \\
R_i &= \sum_{k=0}^{P} r_{c_k,i}\,.
\end{aligned}$$

This covariance is large for contributors whose behavior deviates from their average behavior at same moments as the global behavior deviates from the global average, the deviations being in the same direction.

The analysis consists of following procedures,

1. identification of suitable analysis window, ideally containing 50/50 ratio of intervals before/after the attack and during the attack,
2. establish the covariance $\mathrm{cov}\,(c)$ for all contributors c,
3. sort the contributors according to the covariance,
4. prune out the contributors one by one starting from the one with highest covariance,
5. stop when the remaining contributors' covariances are below a given threshold L.

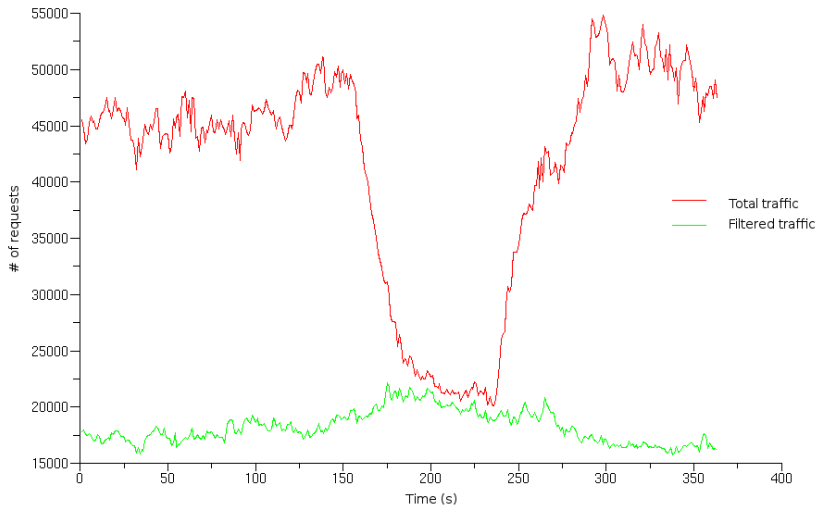


Figure 2.4: Number of requests per second for baseline traffic without bots in green and for total activity including bots in red. The attack is active both in the beginning and in the end of the window.

Figure 2.4 shows an example of the global behavior in terms of requests per second, including normal and bot activity in red. Shown the green line is the baseline activity without the bots, while bots are active in the beginning and end of the analysis window. If we prune out the the individual contributors with the covariances above the threshold L, we end up with the baseline activity. That says, we are able to explain the deviation from the baseline by a set of contributors.

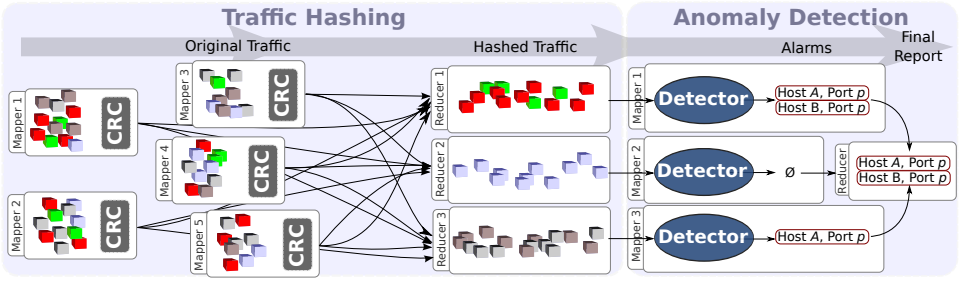
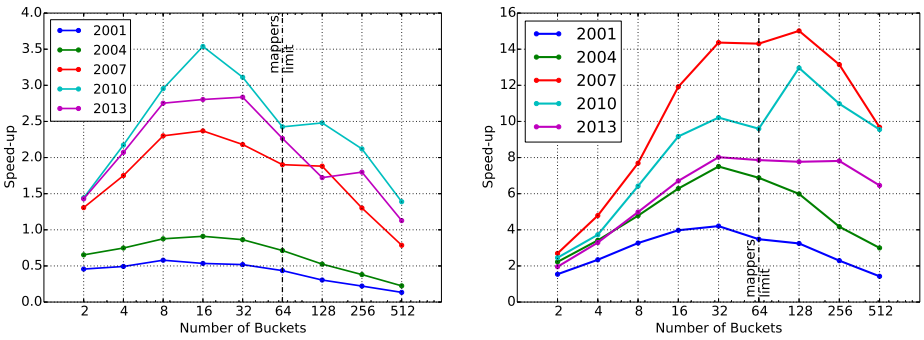


Figure 2.5: Overview of Hashdoop. In this example the original trace contains 5 splits that are hashed in $N = 3$ buckets. Each bucket is independently analyzed by a detector and results of all detectors are summarized in the anomaly report. Note that here traffic is hashed only once for clarity but in fact Hashdoop produces $2N$ buckets for source and destination hashes.

2.1.3 A MapReduce Framework for Anomaly Detection in Backbone Traffic

To cope with the global growth of Internet and obtain the results of anomaly detectors with reasonable latency, the traffic traveling at Internet backbone is usually sampled [14]. However, sampling is inherently detrimental to anomaly detection, as it deliberately discards traffic that may be anomalous.

Motivated by the need of analyzing large datasets, various research communities have developed efficient tools using the MapReduce model [26]. We investigated the benefits of MapReduce to achieve real time anomaly detection with non-sampled traffic, because we observed that available implementations such as Hadoop MapReduce [85] provide high scalability and fault tolerance, which are crucial and desirable features for anomaly detection. However, we found fundamental contradictions between Hadoop data distribution and the requirements of anomaly detectors. As in the MapReduce model data is conceptually record-oriented, Hadoop divides datasets into splits (i.e. subsets of records) and distributes them in the cluster to be independently processed. All the splits of a dataset have the same size in order to synchronize the processing end time of all the nodes. In the case of network traffic, a split usually represents a subset of packets [54], but related packets may spread across different splits, thereby dislocating traffic structures that are essential for anomaly detectors. For example, anomaly detectors can identify DDoS attacks because of the numerous requests sent to the same service. However, detecting this kind of attack with only a subset of the requests is notably more difficult.



(a) Speed-up for the packet count based detector

(b) Speed-up for Astute

Figure 2.6: Packet count based detector and Astute speed-up with Hashdoop.

2.1.3.1 Hashdoop

To overcome this issue, we take advantage of a hash function to divide traffic into splits that preserve the spatial and temporal traffic structures, and a specific scheme termed Hashdoop was proposed in [34], which is a MapReduce framework that splits data with a hash function and processes numerous anomaly detectors in parallel. As shown in Figure 2.5, Hashdoop consists of two steps that are implemented as two distinct MapReduce programs: (1) using hash function the traffic is divided into splits (hereafter called buckets) where both the spatial and temporal properties of the traffic are preserved; (2) anomaly detectors are run with each split of data, and the results collected and reported to network operators.

Traffic hashing. The traffic is hashed twice, once with source IP address as key and once with destination IP address as key, thus assuring that the traffic sent or received by certain host fall in a single bucket. The hash function we employ is the cyclic redundancy check (CRC) algorithm which is commonly used for traffic load balancing [17].

Anomaly detection. Using traffic hashing makes the MapReduce implementation of a network traffic anomaly detector fairly straightforward. The map procedure implements the anomaly detection for a single bucket and results for all buckets are summarized by the reduce procedure. The key advantage of Hashdoop is that virtually any classical anomaly detector can be used for the map procedure.

2.1.3.2 Speed-up

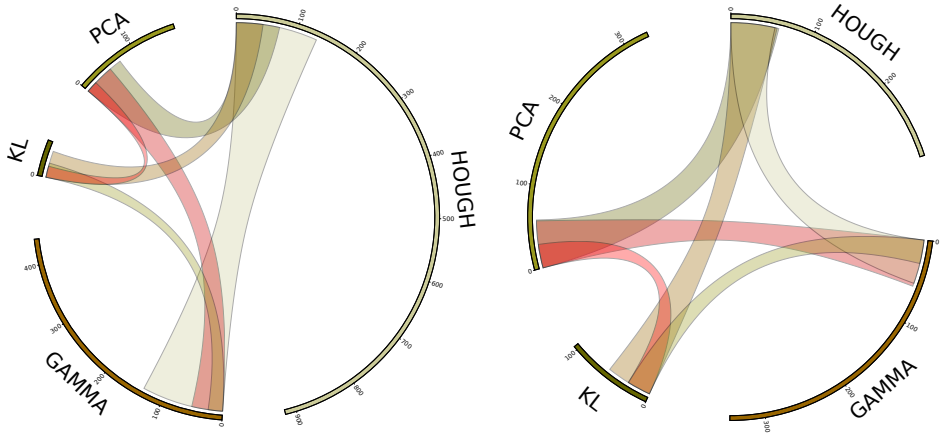
The proposed framework preserves all the advantages of the MapReduce model and can be virtually used with any network traffic anomaly detector. To evaluate its performance, we conducted a set of experiments with traffic measured at a trans-Pacific link (i.e. MAWI archive [2]), and two anomaly detectors were applied: a packet count based detector and a traffic stationarity based detector called Astute [77]. Using a local 6-node cluster, Hashdoop achieves a maximum speed-up of 3.5 with the packet count based detector and 15 with Astute (see Figure 2.6). In other words, the packet count based detector throughput has been improved from 360k packets per second (pps) to 1.27 Mpps, and, for Astute from 25 kpps to 375 kpps. As a result, a 900 seconds trace that was originally analyzed in 1296 seconds with the classical version of Astute could be processed in 216 seconds on Hashdoop. Clearly, this detector is more interesting for real-time anomaly detection. Using Hashdoop we also observed an overall improvement of detection accuracy as anomalies are filtered in splits with less background traffic.

2.1.4 Visual Comparison of Backbone Traffic Anomaly Detectors

Anomaly detection in backbone traffic has received a lot of attention from the research community and led to various proposals relying on diverse statistical methods such as wavelet [8], Kalman filters [78], hash projection [12, 27, 49], Principal Component Analysis (PCA) [46, 52], pattern recognition [33]. Due to this variety of theoretical background, these methods behave differently with traffic fluctuations and parameter tunings, also they report anomalies with diverse characteristics. It is however critical for network administrators to understand the behavior of detectors so they can efficiently inspect the detection results. The current efforts towards this goal are limited to those techniques used for performance evaluation.

Here, our goal is to help network administrators expand their understanding on the behavior of network anomaly detectors. This task is laborious because detector outputs are usually composed of numerous anomalies. It becomes even more complicated when one intends to analyze simultaneously the outputs of several detectors, either to evaluate their performance or to compare their results. To this end, we leverage advanced visualization techniques to assist network administrators in analyzing detection results. The major advantage of using visualization tools is to make a large amount of information concisely represented. In our case, it allows one to assess whether the output of a single detector is similar to a broad consensus among several detectors.

We propose several visualization-based analysis methods based on chord diagrams to provide in-depth analysis of the results obtained from several



(a) Before detector tuning

(b) After detector tuning

Figure 2.7: Intersection of alarm sets between all 4 detectors (unit is equal to 1000 alarms, e.g.: 100 $\rightarrow$ 100000). Detectors' detection results are represented as arcs and links or ribbons connect arcs together in order to display similarities.

detectors [57]. A chord diagram is composed of several arcs located on a circle (refer to Figure 2.7), where the arcs can be labeled and represent the elements to be compared, and links (a.k.a. ribbons) are inside the circle and connect arcs together in order to display similarities.

In our use case, detection results are displayed as arcs, while the links between the arcs represent the similarities between detection results. Also, ribbon width represents the actual number of anomalies detected by both arcs. Note that ribbons are sometimes superimposed but that does *not* mean that ribbons intersect between themselves. In other words, intersections of anomaly set intersection between detectors are not represented by overlaps between ribbons.

It is clear that our proposed approach enables a detailed comparison of detectors outputs. We then applied our proposal to four anomaly detectors used in MAWILab [32], a set of anomalies identified in real backbone traffic traces from the publicly available MAWI dataset. We also analyzed four years of traffic to validate the efficiency of the proposed methods.

Our results show that: (1) the four analyzed detectors exhibit two different behaviors regarding parameter settings; (2) detectors identify different types of anomalies; (3) using chord diagrams we could better tune the parameters of the detectors and maximize their output overlap (see Figure 2.7); and, (4) we improved by approximately 19 percentage points MAWILab results through detectors settings tuning.

2.1.5 Taxonomy of telescope traffic

Providing an opportunity to view and detect remote network security events, darknet [65][87] (a.k.a., network telescope [62]) has drawn much attention of research community. Darknet consists of globally routable but still unused IP address blocks in which no legitimate traffic exists. However, monitoring of those blocks reveals that unexpected packets keep arriving at darknet from a wide range of sources. These unwanted packets destined to darknet are completely non-productive since they originate from worm propagation, (D)DoS attacks, Internet outages, network misconfiguration, or other unsolicited activities. Thus, darknet traffic can be applied for tracking such security related activities in global scale. A good example is the work of Dainotti et al. [24] where they monitor the country-wide Internet outages occurred in Egypt and Lybia in 2011.

Past studies [65][87] on darknet claim that the volume of darknet traffic is not minor and that there is great diversity in this traffic both in terms of address blocks being monitored as well as over time. They also point out that based on different root causes darknet traffic broadly comes from three types of network activities: scanning, backscatter, and misconfiguration. In their descriptions, scanning is largely the result of infected hosts in the Internet attempting to find other vulnerable targets. Backscatter is most often the result of (D)DoS attacks, and misconfiguration generally results from software or hardware errors in network devices. Although this simple categorization was applied to their datasets, it did not give clear definitions of each activity with concrete traffic rules and lacked a further refinement since that traffic classification is just based on TCP flags.

In order to uncover the root causes of darknet traffic, a refined taxonomy of darknet traffic is required. In [55] we aim at filling this gap, first, by proposing a refined taxonomy of darknet traffic on the basis of observations, and second, validating it on real darknet traces of over six years. Our taxonomy mainly defines five types of anomalous events with concrete traffic rules: scanning, one flow, backscatter, IP fragment, and small events (see Table 2.1). The validation results demonstrate that we detect and label anomalous events defined by our taxonomy in more than 99% of source IP addresses, suggesting a extremely low unlabeled source rate. We also obtain some interesting findings on the evolution of different anomalous events especially in 2010, thus shedding light on the overall trends of darknet traffic. We also observed that most source IP addresses are not characterized by one simple event. Instead more than 99% of sources in our traces participate in two or three events with simple attack mechanisms. This taxonomy allow us to automatically characterize threats found in darknet traffic, hence annotates corresponding source IP addresses with pertinent labels.

Table 2.1: A taxonomy of darknet traffic

Event	Category		Traffic Rule
Port Scan	TCP	Heavy	$(\#ipSrc == 1) \cap (\#ipDst == 1) \cap (\#portDst \geq N) \cap (ScanFlagPktRatio \geq R\%) \cap (Avg\ \#Pkt\ per\ portDst > M)$
		Light	$(\#ipSrc == 1) \cap (\#ipDst == 1) \cap (\#portDst \geq N) \cap (ScanFlagPktRatio \geq R\%) \cap (Avg\ \#Pkt\ per\ portDst \leq M)$
	UDP	Heavy	$(\#ipSrc == 1) \cap (\#ipDst == 1) \cap (\#portDst \geq N) \cap (Avg\ \#Pkt\ per\ portDst > M)$
		Light	$(\#ipSrc == 1) \cap (\#ipDst == 1) \cap (\#portDst \geq N) \cap (Avg\ \#Pkt\ per\ portDst \leq M)$
(3) detectors identify different types of anomalies, Network Scan	TCP	Heavy	$(\#ipSrc == 1) \cap (\#portDst == 1) \cap (\#ipDst \geq N) \cap (ScanFlagPktRatio \geq R\%) \cap (Avg\ \#Pkt\ per\ ipDst > M)$
		Light	$(\#ipSrc == 1) \cap (\#portDst == 1) \cap (\#ipDst \geq N) \cap (ScanFlagPktRatio \geq R\%) \cap (Avg\ \#Pkt\ per\ ipDst \leq M)$
	UDP	Heavy	$(\#ipSrc == 1) \cap (\#portDst == 1) \cap (\#ipDst \geq N) \cap (Avg\ \#Pkt\ per\ ipDst > M)$
		Light	$(\#ipSrc == 1) \cap (\#portDst == 1) \cap (\#ipDst \geq N) \cap (Avg\ \#Pkt\ per\ ipDst \leq M)$
	ICMP	Heavy	$(\#ipSrc == 1) \cap (\#ipDst \geq N) \cap ((Type, Code) == (8,0)) \cap (Avg\ \#Pkt\ per\ ipDst > M)$
		Light	$(\#ipSrc == 1) \cap (\#ipDst \geq N) \cap ((Type, Code) == (8,0)) \cap (Avg\ \#Pkt\ per\ ipDst \leq M)$
One Flow	TCP		$(\#ipSrc == 1) \cap (\#ipDst == 1) \cap (\#portDst == 1) \cap (\#Pkt > N_3) \cap (Protocol == TCP)$
	UDP		$(\#ipSrc == 1) \cap (\#ipDst == 1) \cap (\#portDst == 1) \cap (\#Pkt > N_3) \cap (Protocol == UDP)$
Backscatter	TCP		$(\#ipSrc == 1) \cap (\#Pkt \geq 1) \cap (TCP\_Flags \in \{SA \cup A \cup R \cup RA\})$
	ICMP		$(\#ipSrc == 1) \cap (\#Pkt \geq 1) \cap (((Type, Code) == (8,0)) \cup (Type == 3))$
IP Fragment			$(\#ipSrc == 1) \cap (\#fragmentPkt \geq 1)$
Small SYN			$(\#ipSrc == 1) \cap (\#ipDst < N_1) \cap (\#portDst < N_2) \cap (\#Pkt \leq N_3) \cap (TCP\_Flags == S)$
Small UDP			$(\#ipSrc == 1) \cap (\#ipDst < N_1) \cap (\#portDst < N_2) \cap (\#Pkt \leq N_3) \cap (Protocol == UDP)$
Small Ping			$(\#ipSrc == 1) \cap (\#ipDst < N_1) \cap (\#Pkt \leq N_2) \cap ((Type, Code) == (8,0))$
Other			$Other$
Remark: Our parameter settings $\{N = 3, R = 50, M = 3, N_1 = 5, N_2 = 5, N_3 = 15\}$ are empirically decided based on our traces.			

2.2 Large-scale DNS traffic analysis

Domain Name System (DNS) plays a crucial role in the operation of network, as it provides bidirectional translation between domain names and IP addresses. Namely, each internet node has to initiate a name resolution request in order to access a target host. Thus, analyzing DNS traffic may help to reveal clients' intentions. For example, if a client is affected by some malwares, it would send particular DNS queries bound to malicious activities. To date, a lot of researches focus on using DNS traffic to detect infected nodes.

Furthermore, because of the fundamental role of DNS in the Internet, deceiving DNS infrastructure or DDoS attacks against DNS infrastructure will bring destructive impacts on network users. It is therefore important to detect and protect DNS from such kind of attack as well.

In the following, we describe the DNS analysis that have been carried out in the *NECOMA* project. We firstly state the different goals of DNS analysis, we then propose a hadoop-based platform analysis, along with three different DNS analysis techniques developed by the *NECOMA* consortium.

2.2.1 Targets of DNS Analysis

1. DDoS Attack/DNS Amplifier
   DNS Amplifier Attack is one of the serious threats for today's Inter-

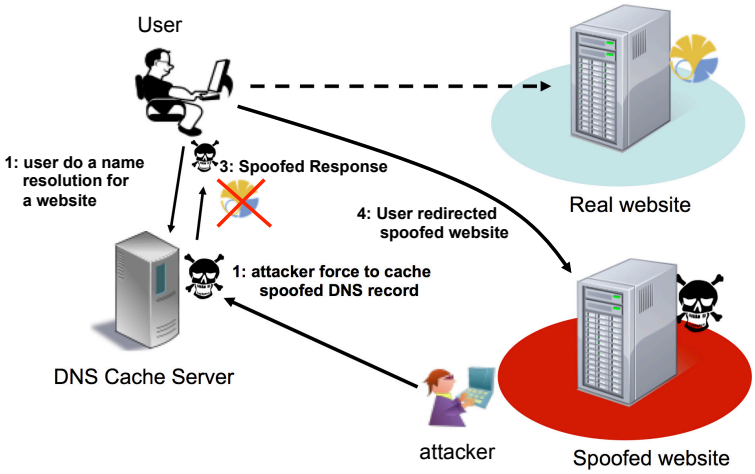


Figure 2.8: Attacks of spoofing DNS packets

net. As DNS protocol uses UDP packets, DNS server will respond the queries without validating source addresses. Also, these attacks exploit DNS cache servers due to two another facts: 1) there are more DNS cache servers than DNS authoritative servers, and 2) these servers reply to recursive queries. Although an ordinary DNS cache server restricts recursive queries from outer network, there are still a lot of DNS cache servers which reply to any recursive queries. Thus, an attacker can choose any domain names from global DNS tree.

Moreover, an attacker sometimes targets the DNS server itself as a victim of DDoS attack. This kind of attack is a little similar to DNS amplifier DDoS attack, but it aims at interfering with DNS server's work itself. If the attacker wants to spoof a DNS cache server, interfering with the corresponding authoritative server is an effective way of increasing the possibility of successful attack, because victims will not be able to receive 'legitimate' DNS Answers.

2. DNS Spoofing
   DNS spoofing is another security threat for DNS service. As each DNS resolver caches the results of his queries and responds from that cache for other queries to anyone who has the same queries. Thus, if an attacker illegally inserts fake answers into DNS resolver, the users of the DNS resolver will receive fake answers, that is, as described in [7], an attacker can forge DNS data and DNS packets and guide the traffic to phishing sites as shown in Figure. 2.8.

   All DNS responses are identified by queried domain name, UDP port number, and DNS ID. Hence, an attacker could deceive DNS clients by

sending a forged DNS answer which is disguised with the same domain name, UDP port number, and DNS ID. In particular, UDP port number and DNS ID are 16-bit integer values, thereby making it difficult to adapt a fake answer to a particular DNS query. In order to deceive DNS clients, an attacker might send a numerous amount of DNS answers which have different DNS IDs and port numbers. Alternatively, an attacker might place a web page which contains links to target domain names. Once a victim opens that page, his browser does pre-fetch these links, and then an attacker could know the timing of sending disguised DNS answer to deceive. However, these attack strategies possibly leave their footprints in DNS traffic log, allowing DNS traffic analysis to detect some attacks.

3. Botnet's Command and Control(C&C) Communication
   Botnets, networks of compromised PCs (bots) controlled by a bot master, are usually the engines behind cyber crimes. They can be used to steal valuable information from business, send spam emails at a large scale, scan networks, attempt to exploit PCs, host malicious websites, and launch various types of cyber attacks. Among those botnets in the wild, ZeuS has grown into the most popular botnet in 2009 [11, 25], and it contains a builder that can generate a bot executable and Web server files for use as the command and control server [29]. The primary goal of ZeuS is to steal credential information [29, 3] by gaining full control over a remote computer.

   According to CERT.pl [20], there are various mutations of ZeuS after the ZeuS version 2.0.8.9 source code leaked. The new mutation introduced both a P2P network and a domain generation algorithm (DGA) to make take down efforts more difficult. Since the earlier version of ZeuS used some predefined URLs to communicate within botnets, it was easier to trace the communication. Instead, the new version employed the P2P network to communicate and to distribute the data. The DGA dynamically produces a large number of random domain names and selects a small subset for actual communication and control use [5]. In addition, ZeuS uses the mechanism as a backup communication channel when the P2P communications are unavailable.

   CERT.pl also contributed a technique to detect randomly generated domains [19]. Figure. 2.9 shows their regular expressions for finding ZeuS DGA. Each name consists of a string with a length of 32 to 48 characters, and one of TLDs: `ru`, `com`, `biz`, `info`, `net`, or `org`.

```
[a-z0-9]{32,48}\.(ru|com|biz|info|org|net)
```

Figure 2.9: DGA detection techniques.

2.2.2 DNS Traffic Analysis Platform with Hadoop Framework

2.2.2.1 Background

DNS detection methods have different goals and features. One of them aims to find out domain names which are used in particular botnet implementations, and yet another method focuses on detecting some malicious attacks to their network. It means a security operator has to use multiple DNS traffic analysis methods to prevent different type of threats. Moreover, each network communication requires a DNS query. Hence, the size of analysed DNS datasets could be huge.

Accordingly, we proposed a platform which could analyze Large-Scale DNS dataset with variety of detecting methods. Our system helps to develop and deploy new DNS analyzing algorithms.

2.2.2.2 Design and Implementation of DNS analysis workbench

In our system we use Hadoop framework and HDFS file system to store numerous DNS traffic data and to process it. This system gathers following datasets.

- DNS query/response on DNS cache server
- DNS query/response on DNS authoritative server
- DNS traffic on backbone network

All datasets are packet capture file(pcap). Each captured datasets are imported into HDFS periodically.

Since the size of dataset can be huge and some detecting methods do same preprocessing, we use intermediate output file to reduce computational resource requirement. Figure 2.10 shows an example of a dataset flow. First, we use simple TTL filter to pick suspicious DNS packets. A DNS answer which has extremely short TTL can be a candidate of DNS fast-fluxing domain name query, because an attacker wants to reflect the change of their C&C server IP addresses rapidly. On the other hand, A DNS answer with extremely long TTL was suspected as DNS cache poisoning because an attacker wants to maximize Time-to-Live of these imitated record cache.

Then, one can use multiple types of detecting methods which focus on discovering DNS fast-flux in these filtered datasets. These detecting methods create independent results.

Finally, one can aggregate these results, and rate each record according to all detecting methods results.

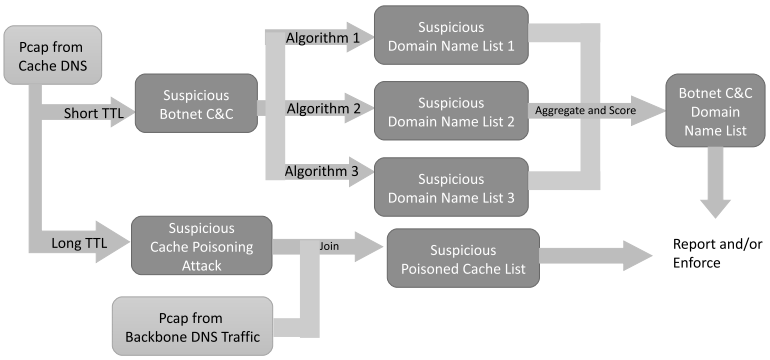


Figure 2.10: Example of Dataset flow

Regular Expression Based ZeuS DGA Detector

The module of ZeuS DGA detector aims at 1) detecting suspicious DNS queries in a hosted network at DNS cache resolver, 2) extracting the IP addresses of the client hosts who attempt to resolve by the suspicious DNS queries, and 3) identifying the flows that were initiated by the DNS queries looking up C&C servers.

We used the following two datasets as shown in Table 2.2 for the detection: 1) pcap trace of DNS queries and responses, and 2) netflow sample data, which contains DNS traffic as well as a data traffic triggered by DNS queries.

Table 2.2: Dataset used for ZeuS DGA detection.

	format	data size (per day)	remark
DNS pcap	as-is	5GB	hadoop-pcap [1]
netflow	CSV	1.2GB	`nfdump`

For the detection, the SQL statements described in figure 2.11 are used. Note that the first query removes queries which contains punycode [48] since the regular expression used by ZeuS DGA also matches the one of punycode.

Thanks to the Hive and Presto-db framework that internally splits optimal jobs into MapReduce framework, our DGA detector of ZeuS bot is really simple.

Result of ZeuS DGA Detector

With the dataset described at Section 2.2.2.2, we analyzed how many hosts are affected by ZeuS bot in a network by counting the number of queries

```
1) select * from dns_pcaps where
   regexp_like (dns_question,
   '[a-z0-9]{32,48}\.(ru|com|biz|info|org|net)')
   AND NOT regexp_like(dns_question, 'xn--');
2) select * from dns_response where dns_answer!='[]';
3) select * from netflow_ut where sa='xxx';
```

Figure 2.11: SQL statements with regular expression to detect ZeuS DGA queries and its affected hosts by ZeuS bot.

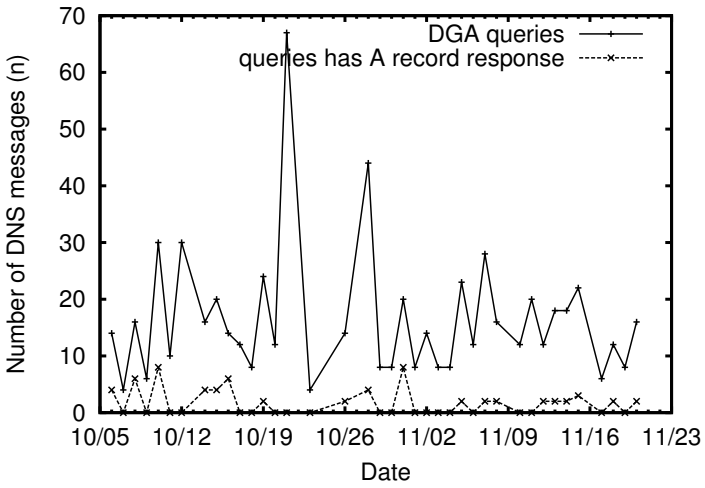


Figure 2.12: Number of queries in a day contains ZeuS DGA name, and A record response.

from the dataset. We collected the `pcap` data for two months and analyzed them with Presto-db.

Figure 2.12 represents the number of queries that contain the suspicious name in the query, and the number of DGA queries response that contain resolved A records. The IP address in the A record is potentially the IP address of C&C server. As you can see, there are a number of ZeuS DGA affected hosts in the network, querying malicious name generated by the botnet program.

We also observed that current regular expression used by the detector gave false positive results. Figure 2.13 lists top 3 frequent names. Apparently the query name `3.1415...` is the false positive entry, which has a normal domain name used by hosted server. We plan to create a white list to avoid such false positives.

```
f528764d624db129b32c21fbca0cb8d6.com.
3.14159265358979323846264338327950288419716939937510582097494459 2.com.
www.paypal.com.verify.securearea.(snip).com.
```

Figure 2.13: Top 3 frequent detected DNS queries.

By using the IP address list obtained by this query, we can identify traffic flows from our netflow dataset communicating with the IP addresses, which might be C&C servers.

2.2.2.3 Regular Expression Based DNS Reflection Attack Detector

This module of DNS reflection attack detector aims at detecting DNS reflection attack which is recently a popular DDoS attack. We measure daily flow volume, which includes the communications with DNS open resolver hosts.

We used the following three datasets (Table 2.3) for the detection: 1) sFlow traffic data collected from internal routers, 2) the list of DNS open resovler host in whole IPv4 address space which is gathered by Takano [79], and 3) the target host list of a campaign.

Table 2.3: Dataset used for DNS Reflection Attack detection.

	format	data size	remark
sFlow	CSV	1GB (per day)	-
DNS open resolver list	CSV	416M	-
Target host list	CSV	22 domains	-

For the detection, we used the following SQL statement (figure 2.14). In this case, we fortunately acquired target host list of the DDoS campaign. This query extracts flows from sFlow datasets which have target host IP addresses as a source IP address and the destination IP address listed in DNS open resolver list. We can easily observe changing the number of suspicious flow in daily.

```
select * from {sflow} join dns_openresolver_list
 on ({sflow}.dt='{date}') and ({sflow}.srcIP
  = dns_openresolver_list.id) and {targetIPaddresses};
```

Figure 2.14: SQL statement with regular expression and join to detect DNS reflection attack.

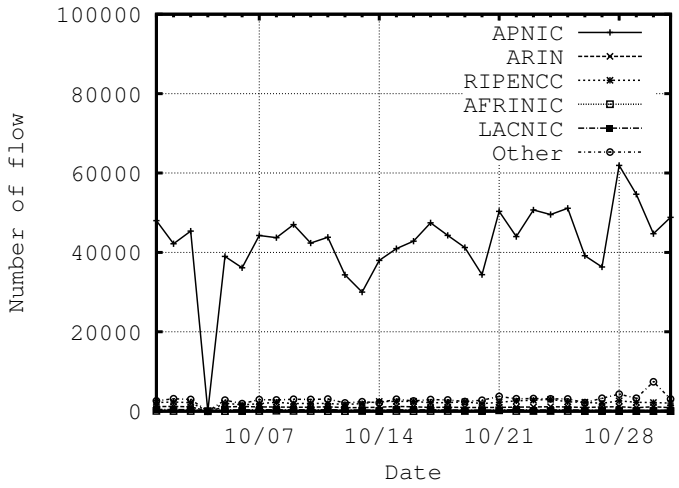


Figure 2.15: Number of flows in a day.

Result of Regular Expression Based DNS Reflection Attack Detector

We analyzed one month sFlow datasets with the attack detector. As a result, we did not observe any trend of the attack campaign. In fact, the campaign was not observed in any other sources. Figure 2.15 shows daily flow volume in an sFlow dataset which is collected on an academic research backbone (sampling ratio is 8192). The graph shows the number of the flows categorized by regional Internet registry. There is such kind of flow in daily traffic. Unfortunately, there are some false positive results in which we label some legitimate DNS queries as malicious. For example, a flow which is going to public open DNS such as Google Public DNS (8.8.8.8) is counted as open resolver communication. However, the flow might be legitimate rather than malicious. As our detector observes only flow volume, it can adapt to detect large scale DDoS campaign based on DNS reflection. In such kind of campaign, the number of open resolver's flow could be increased.

2.2.3 DGA-based botnet detection system

In recent years, botmasters have developed domain generation algorithm (DGA). Attackers generate pseudo-random domains names for C&C server. Each bot periodically executes DGA by receiving a random seed (e.g., the current time) and computes a list of candidate C&C domain names. Finally, each bot sends DNS queries for candidate domain names and gets the IP address of the C&C server. Examples of malware that execute such DGAs are Srizbi[86], Conficker-A/B[71], Conficker-C[72] and Gameover Zeus[4].

In order to detect DGA-based botnets, Antonakakis et al.[6] focus on Non-Existent Domain (NXDomain) responses. Most of the DGA domains that a bot queries would result in NXDomain responses.

We also focus on NXDomain. Our approach is divided into two steps: Extraction and Classification. (1) We receive non-existent domain names that result in *NXDOMAIN* responses from DNS cache servers and DNS auth server, then we extract suspicious domains based on DGA temporal locality. (2) We apply support vector machine (SVM) classifiers to suspicious domains and classify malicious or legitimate domains. Then, we detect hosts that queried to malicious domains as infected hosts.

2.2.3.1 Dataset

We collected DNS traffic captured at recursive DNS resolvers in the University of Tokyo We have observed the period from December 1, 2013 to October 31, 2014. On average per each day, the total number of all DNS queries is 18.2 million and the number of distinct host is 45,000. Then, we also collected DNS traffic captured at a widely queried an upper authoritative nameserver during the same period.

As the ground truth, we performed malware dynamic analysis. To obtain a sample set of malicious domains we executed malware samples in a VM-based malware automated analysis system which only allows legitimate traffic. We executed Zeus Gameover samples and obtained collected 3,000 domains per a day. We also used well-known malicious domains of Conficker[23]. In this data, 18.3 million malicious domains are included together Conficker A / B / C. In addition, by using the top 100,000 popular domains from **alexa.com** as legitimate domains and applying the prefix www. to these, we used the total 200,000 domain.

2.2.3.2 Approach

We extract suspicious domains based on the temporal locality of DGA. There is the temporal locality that domain names which have been generated by DGA are used for only a short time. Hosts that are infected with the same DGA-based bot are likely to query the same non-existent domain names. When many bots generate the same non-existent domain at the same time, the requests to the non-existent domain increase rapidly within a certain period of time. On the other hand, non-existent domains which are queried by user (e.g., typo) are unlikely to be queried at the same time. Other non-existent domains which are queried by anti-virus software for matching blacklist are queried for a long time. Figure 2.16 shows that the number of source IP address that queried to well-known malicious domain generated by Conficker-C and legitimate domain generated by anti-virus software.

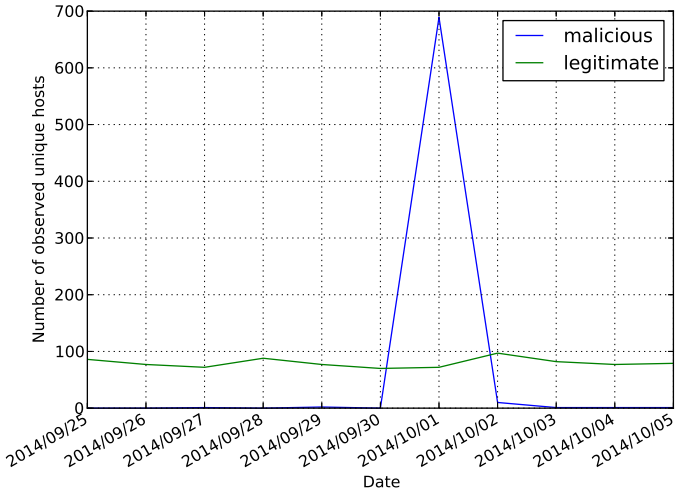


Figure 2.16: The number of hosts which queried each NXDomain during a week

Therefore, we consider non-existent domains which are queried by many hosts in a given period are likely to be DGA domains.

However, there is a bias of non-existent domains for each network. For example, if the web page which is published by the University of Tokyo put the wrong URL, the requests to that non-existent domain increase rapidly in the network of the University of Tokyo . Consequently, there is a lot of noise in the analysis of recursive DNS resolvers. Although we could eliminate the noise in the analysis of an upper authoritative server, we couldn't distinguish between existent domains and non-existent domains. Therefore, we combined recursive DNS resolvers and the upper authoritative DNS server. We extract non-existent domains in recursive DNS resolvers and compute source IP addresses which requested to that non-existent domains in an upper authoritative DNS server. In this way, we extract non-existent domains which were queried by over n hosts within d_{max} days in d_{span} days. These non-existent domains are suspicious. We empirically set $n = 10$, $d_{max} = 2$ and $d_{span} = 7$ in preliminary experiments.

2.2.3.3 Classification

After extracting suspicious non-existent domain names, we apply classification based on the character structure. For each domain, We compute the following structural features.

- The frequency distribution of n-grams(n=2)

Table 2.4: The number of host.

	Dynamic analysis	DNS resolvers	Total
Hosts	24	370,049	370,973

- The entropy of the character distribution(2LD,3LD)
- Domain length(d, 2LD,3LD)
- Domain level
- TLD type

We use support vector machine (SVM) as a classifier. Given a set of training data which belong to malicious or legitimate domain from grunt truth, SVM builds a model that assigns new domains into malicious or legitimate. We apply SVM to suspicious domains obtained from the extraction step. Then, we detect hosts that queried to malicious domains which are assigned as malicious domains by SVM as infected hosts.

2.2.3.4 Results

We evaluated our method against DNS traffic from recursive DNS resolvers and one of widely queried high-level nameserver from October 24, 2014 to October 31, 2014. To evaluate, we mixed domains obtained from recursive DNS resolvers and dynamic analysis. The average number of distinct host per day is 370,049 in recursive DNS resolvers and the number of VM which we ran for dynamic analysis is 24, then the total number of host is 370,973, as shown in Table 2.4 We evaluated how well our method could detect these 24 infected hosts in 370,973 hosts.

Our method detected 30 hosts as infected hosts in all hosts. These 30 hosts included 24 infected hosts. Therefore, our method is proved to detect correctly.

2.2.4 Classification of DNS Erroneous Queries

The domain name system (DNS) is one of the most important service in the Internet, it provides translation between domain names and IP addresses. However, it is reported that the DNS has also been abused for non-legitimate purposes such as spam, botnet controls and distributed denial of service (DDoS) attacks. Therefore, it is necessary to understand abnormal DNS behaviors for preventing malicious activities.

To date, many studies have been devoted to DNS measurement and analysis[84, 18, 39, 45, 64, 5, 90, 35, 41, 70, 15, 37, 10, 44]. DNS errors are caused by un-resolvable DNS queries from local resolvers. We still

have less than enough knowledge about the causes of the DNS errors for reducing them. In addition, a huge number of DNS errors unnecessarily consume network resources as well as those of DNS servers.

Table 2.5: Classification rules for NX Domain errors

No	Rule	Example
1	Used by anti-virus software	waseda.jp.uri.jp1.sophosxl.com
2	Used by anti-spam RBL	1.0.0.0.zen.spamhaus.org
3	Unknown TLD	example.TEst
4	Random words	qebwprbpyy.ac.jp
5	Add “dlv.isc.org”	example.com.dlv.isc.org
6	Configuration words	local, wpad
7	Local name	YUTA-PC
8	(IP Address)+(TLD) or repetition of TLD	192.168.0.11.ac.jp www.waseda.jp.ac.jp
9	RFC 1034 violation	(10.3.1.3).go.jp, ***.com

Thus, we analyze the DNS traffic using passive DNS measurement at an external connection link of an academic backbone network in Japan [47]. In particular, we focus on DNS errors, such as ServFail, Refused, and NX Domain, sent from authoritative nameservers in external networks to local resolvers in the academic network. We report on a number of abnormal phenomena likely caused by malicious and abnormal systems. First, we find that most of ServFail and Refused errors are replies to queries from a small number of resolvers. We also discuss a number of problematic authoritative nameservers that always send back these errors. Second, we propose heuristic classification rules based on observed NX domain names (see Table 2.5). These rules classify NX Domain errors into nine groups and allow us to identify 88.7% of unique domain names observed in our dataset. As a result, we confirm that NX Domain errors are mostly caused by specific anti-virus client software and anti-spam systems that generate many queries for checking if domains are registered on a black-list for legitimate purposes, as well as mis-configurations of servers and end-user machines that query wrong domain names. We also find a set of malicious domains used in spam messages by applying the classification rules to legitimate answer domains. Randomly generated domain names also account for a large fraction of the unique domain names (17.3%) observed in our dataset and allow us to identify hosts infected by malware using domain generation algorithm (DGA).

2.2.5 DNS anomaly detection using PCA

DDoS attacks and especially reflection attacks, that misuse open services on the Internet to indirectly overwhelm victims, have long demonstrated

their nefarious impact on the Internet, at a large scale. Nonetheless, recent research work has indicated that, for example, open DNS resolvers, i.e., which would serve any unauthenticated user, still amount to 29 millions [51]. Therefore, it remains easy to abuse such resolvers to generate a volumetric attack against an unsuspecting victim just by sending many spoofed queries. Our work focuses on detecting anomalies affecting DNS traffic received by DNS resolvers, i.e., DNS queries. Using DNS traffic available within the *NECOMA* consortium, we have carried out several analyses to express patterns among DNS queries. While we are mostly using features readily available within the DNS traffic, we have proposed to add an entropy score computed from the distribution of queries among clients identified by their IP address. This additional feature allows to track anomalous events such as particularly preponderous clients, which may indicate spoofing.

A principal component analysis is performed against a 7-variable vector including DNS query features and the proposed entropy score which has given convincing results for DNS servers of the same type (authoritative, cache). Up to 4 components allow to explain most of the variance of the captured traffic. By using k-means clustering on the PCA-transformed traffic, we were able to group different events, with anomalous ones being the most sparse and distant from the principal component axes. In order to automate and generalize the process for a greater number of clusters, we have used an outlier detection process based on the inter-cluster distance [21].

2.3 End-point threat data analysis

In recent years the commoditization of software marketplaces, applications, addons, extensions and plugins, widely available for anyone, entices and facilitates collection of private information or the propagation of malicious code. Furthermore, new generations of malicious code are increasingly stealthy, powerful and pervasive, thus, there is a clear need for better understanding of threats and resilient-by- design information systems.

Due to the increase of cyber attacks targeting users, the end user protection is included in the *NECOMA* research coverage. It is expected that the information from different layers will give new aspects for identifying phishing attacks. Additionally, we expect that the investigation of the reasons for interpreting websites as trustworthy and executing untrusted programs will help elucidating malicious activities.

The prototypes and ideas described below are part of research activities that aim at designing and developing a framework that will be able to accurately detect various attacks and malicious web based activities with the aim at identifying threats targeted at endpoint interfaces such as web browsers.

2.3.1 Machine Learning-based Phishing Detection

To prevent a user from browsing phishing sites, there are two distinct approaches. One is URL filtering. It detects phishing sites by comparing the URL of a site a user visits with a URL blacklist composed of the URLs of phishing sites. However, it is difficult to build a perfect blacklist due to the rapid increase of phishing sites.

The other approach is a heuristic-based solution. A heuristic is an algorithm to distinguish phishing sites from others based on users' experience, that is, a heuristic checks if a site seems to be a phishing site. A heuristic-based solution employs several heuristics and converts results from each heuristic into a vector. Based on the vector, the heuristic-based solution calculates the likelihood of a site being a phishing site and compares the likelihood with the defined discrimination threshold. Different from URL filtering, a heuristic-based solution has a possibility to identify new phishing sites.

In the analysis, we employed following seven heuristics derived from earlier researches [91, 88].

- Age of Domain
  Checking if the domain was registered more than 12 months ago. If the site has been registered more than 12 months, the heuristic deems it a legitimate site, and otherwise it deems it a phishing site.

- Suspicious URL
  Checking if a URL of the site contains an "at" symbol (@) or a "dash" (-) in the domain name. If so, the heuristics deems it a phishing site because phishing attackers are likely to use these symbols in the domain name of a phishing site. When the at "@" symbol is used in a URL, all text before the @ symbol is ignored and the browser references only the information following the @ symbol as a hostname. Phishing attackers likely abuse this URL scheme: For example, if `http://paypal.com@phishing.com` is used, web browsers would be directed to the `phishing.com`. Even if it seemed like `paypal.com`, web browsers would ignore this.

- Suspicious Links
  Similar to the Suspicious URL heuristic, this one checks if a link on the page contains an "at" symbol or a dash.

- IP Address
  Checking if the domain name of the site is an IP Address. Although legitimate sites rarely link to pages by an IP address, phishers often attract victims to phishing sites by IP address links.

Table 2.6: Definition of True Positive, True Negative, False Positive and False Negative

	Label as Phishing	Label as Legitimate
Actual Phishing	True Positive (TP)	False Negative (FN)
Actual Legitimate	False Positive (FP)	True Negative (TN)

- Dots in URL
  Checking if the URL of the site contains five or more dots. According to [31], dots can be abused for attackers to construct legitimate-looking URLs. One technique is to have a sub domain. Another is to use a redirection script, which to the user may appear like a site hosted at google.com, but in reality will redirect the browser to phishing.com. In both of these examples, either by the inclusion of a URL into an open redirect script or by the use of a number of sub domains, there are a large number of dots in the URL.

- Forms
  Checking if the page contains any web input forms. In the case of CANTINA, it scans the HTML for `<input>` tags that accept text and are accompanied by labels such as "credit card" and "password." If so, the heuristic deems it a phishing site.

- TF-IDF-Final
  This heuristic checks if the site is phishing by employing TF-IDF-Final, which is an extension of the Robust Hyperlinks algorithm [67]. When the heuristic attempts to identify phishing sites, it feeds the mixture word lexical signatures and a domain name of the current web site into Google. If the domain name matches the domain name of the top 30 search results, the web site is labeled legitimate.

We used precision, recall and accuracy as performance metrics for classification. Table 2.6 shows factors for calculating each metric. The precision was calculated by $TP/(TP+FP)$, the recall was $TP/(TP+FN)$. and the accuracy was $(TP+TN)/(TP+TN+FP+FN)$.

Our analysis used roughly 30,000 of phishing sites and the same number of legitimate ones, and employed RandomForest as an classifier of phishing and not. We observed that the precision was 86.06% and the recall was 75.81%, and the accuracy was 81.77%.

2.3.2 Users' behaviour analysis

According to our previous researches [61], experts in identifying phishing tended to assess the credibility of the websites by URL of the websites and

(a) Novice (b) Expert

Figure 2.17: Eye-tracking in phishing website

security information appeared at the browser. Conversely, novices tended to ignore these information assess the credibility by web content. Web content in the phishing sites usually perfect copy of the legitimate ones, and hence, novices falls into phishing.

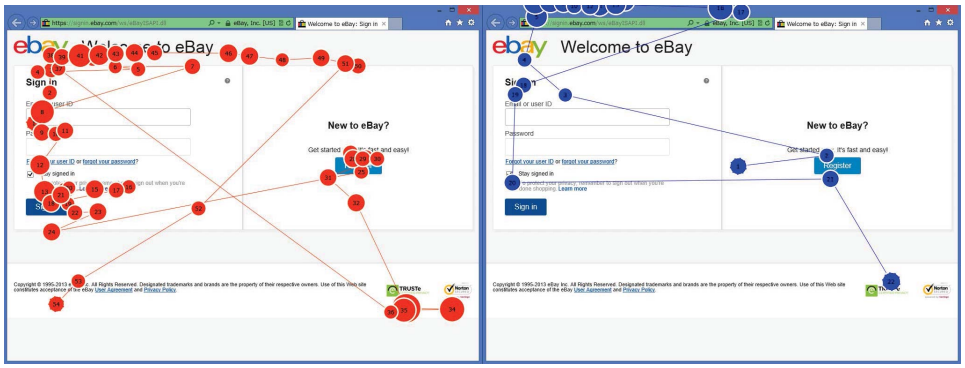
We herein decided to employ eye movements-based observations by following reasons. At first, our motivation is to check if the end users investigated the browser's address bar; eye-tracking is a straightforward way for observing users' behavior. It should be noted that modern web browsers do show the website's URL and security information in the address bar. The second is that monitoring eye movement will not significantly penalize users' convenience. Other tools, such as Brain activity, heart measure, and blood pressure are feasible due to the sensitivity to workload changes, but they tend to require much obtrusiveness for people. Contrastively, Facial expression and Gesture recognition were often affected by Fear of Negative Evaluation [83].

In order to assess if gazing at the address bar is beneficial, we performed a participant-based experiment to monitor an end-user's eye activity. The description of the experiments can be shown in Deliberable D1.3: Endpoint-Layer Threat Datasets.

In this experiment, we recruited 23 participants to observe their eye movement. The volunteers were mainly males in their twenties. It should be noted that we could not collect the information about neither affiliation nor nationality. With their consent, their eye movements were recorded by our prepared eye-tracking device, Tobii TX300 [1]. It required a calibration procedure for each participant.

Fig. 2.17 and 2.18 show typical eye-movement records on both phishing and legitimate website, for a novice and an expert respectively. Circles

[1] http://www.tobii.com



(a) Novice (b) Expert

Figure 2.18: Eye-tracking in legitimate website

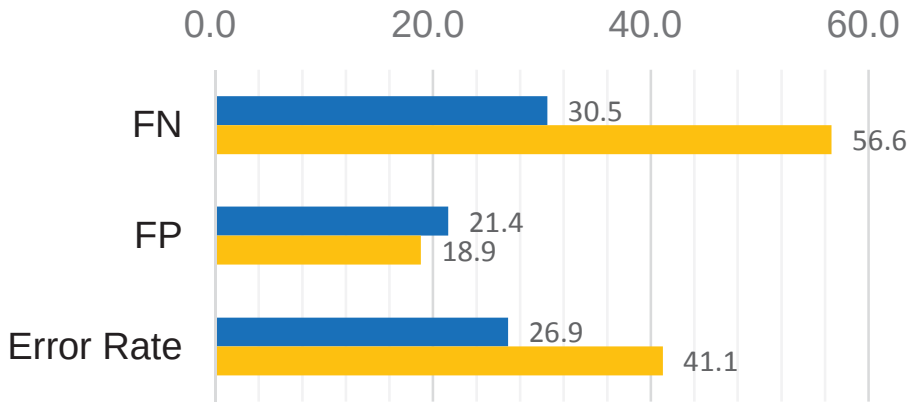


Figure 2.19: The average false positive, false negative and error rate for users that looked (blue) and did not look (orange) at the address bar

denote fixations, and the numbers in the circles denote the order of the fixation. In the phishing case, the novice looked at the web content but ignored the browser's address bar while assessing credibility, as shown in Fig. 2.17a. Since the text and visual in phishing sites are quite similar to the ones in legitimate sites, he failed to label the phishing site correctly. In the legitimate case, he also only paid attention to the web content as shown in Fig. 2.18a. In contrast, an expert tends to evaluate the site's URL and/or the browser's SSL indicator rather than the contents of the web page to judge the credibility of the sites, as shown in Fig. 2.17b and 2.18b.

We then analyzed the detection accuracy of participants who looked at the address bar and those who did not look, respectively. The results were shown in Fig. 2.19, where he blue bar denotes the average rates for the

participants looked at the address bar of the browser, and the orange bar denotes that for the participants did not look at the bar.

Out of the 331 times the bar was gazed, 89 (26.9%) misjudgments were observed. In the phishing websites case, the participants looked at the bar 200 times in total, which occurred 61 (30.5%) false negatives, i.e., labeling phishing as legitimate. In the legitimate websites case, they looked at the bar 131 times in total, which occurred 28 (21.4%) false positives, i.e., labeling legitimate as phishing. In contrast, the average error rate was 41.1% (53 out of 129), the false negative rate was 56.6% (43 out of 76), and the false positive rate was 18.9% (10 out of 53), when participants would ignore the address bar. The average error rate and false negative rate indeed decreased when the address bar was checked, although experimental errors might have occurred due to some possible offsets caused by the eye-tracking calibration procedure. The increase of the false positive rate seems to be marginal. We therefore concluded that checking the browser's address bar is beneficial to end users in making them aware of phishing.

2.3.3 Spam campaign analysis

Spam is a key medium for scams, phishing, illegal advertising, and malware spreading. To send numerous emails in a stealthy manner, spammers take advantage of large networks of compromised hosts also called botnets. As each infected host sends a small number of spam, detecting all members of a botnet is particularly hard.

An effective approach to infer spamming botnets is to identify all spam emails from a same campaign. Thereby, the challenge is shifted to the identification of spams from the same campaign [66]. This task is easier because all spams in a campaign share the same goal, hence, they have common features that permit to distinguish distinct spam campaigns.

We propose a new methodology based on fuzzy hashing to identify spam campaigns [22]. Fuzzy hashing is an effective technique to measure the similarity of two sequences of characters. We implement fuzzy hashing to compare the content of spams and compute a similarity measure. Spam emails from one campaign have a high similarity score among each other and a low score with other emails. Within a campaign spam emails also share other features such as URL or email address. By combining fuzzy hash results and theses common features, we accurately cluster spam emails into campaigns.

The analyzed dataset consists of about 550K spams collected by a few mail accounts over three years. An important contribution of this work is a detailed analysis of long-duration botnet spam campaigns. Figure 2.20 depicts the time evolution of the top 35 spam campaigns that sent the most messages in our dataset, and Table 2.7 shows the main features of the top 15 campaigns. The major findings of our study include:

Table 2.7: Detailed characteristics of the top 15 campaigns

Camp. No	# Spam	# IPs	# Subnets	# MD5	Period(days)	# Titles	# URLs	# Emails
1	8630	8281	7959	7570	467	462	1337	0
2	5640	5107	4683	5304	166	374	5074	0
3	3760	3756	3753	6	80	113	5	0
4	3419	3419	3418	8	91	673	6	0
5	2454	2454	2454	6	278	71	6	0
6	2222	2142	2115	4444	69	169	2261	122
7	1909	1909	1909	1909	186	92	3	0
8	1715	1714	1714	4	152	89	2	0
9	1706	1705	1704	1706	78	191	3	0
10	1276	1276	1276	2	49	298	3	0
11	1031	1030	1030	1031	25	34	5	0
12	917	864	856	829	164	864	1399	10
13	850	225	213	1550	56	483	0	198
14	838	838	838	1	16	432	2	0
15	828	828	827	828	91	37	2	0

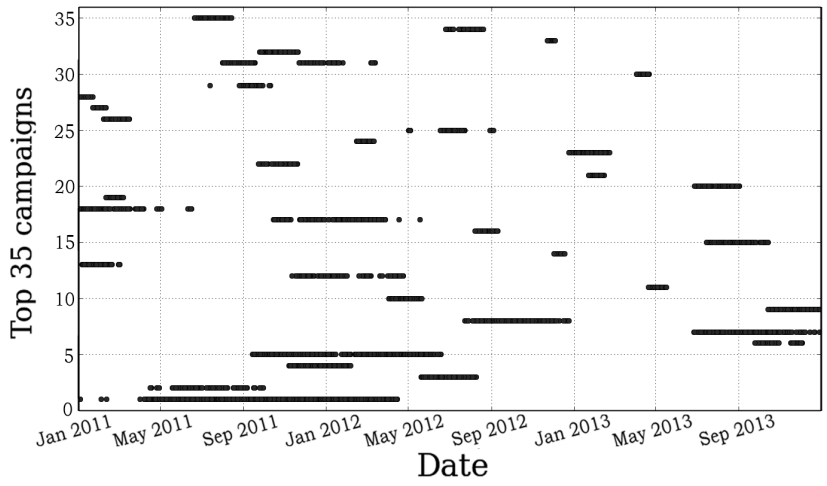


Figure 2.20: Time evolution of top 35 spam campaigns

- Many spam campaigns are constantly appearing, and last for months.
- Different spam campaigns may be initiated from the same botnet.
- Most spam campaigns can be classified into two types; some campaigns are easily detectable as the corresponding spam emails contain common identifiers (e.g. URL), while more sophisticated campaigns generate different contents to avoid simple spam detectors.

Considering that more sophisticated spam campaign appears in future, we believe that tracking spam campaign behavior is an important task for network security. Furthermore, the automated identification of the common identifiers of a spam campaign (e.g. URLs or email addresses), permits to systematically report suspicious Internet resources.

2.3.4 High Precision Phishing Detection

Because phishing attacks rely on impersonating third party services, it is nearly impossible to reach 100% detection accuracy since the malicious web pages resemble legitimate services to a great extent. Traditional classifiers always have up to a few percent of uncertainty leaving the possibility of misclassifying a website. Thus, one of the studies conducted within the *NECOMA* project aimed at investigating techniques that would not be burdened with uncertainty.

2.3.4.1 Initial studies

Because the first kind of information encountered, at the endpoint side, while dealing with phishing attacks is the actual URL or hostname leading to the malicious service, the investigation started by analysing hostnames coming from two publicly available datasets: Alexa serving as legitimate website list (dataset obtained on 26.03.2014) and PhishTank serving as phishing website list (dataset obtained on 16.04.2014). Alexa provides a free CSV download of their top 1,000,000 sites world wide, which is updated daily. PhishTank provides a downloadable database, containing all currently online phishing websites (updated hourly). The aim of the study was to find features that would clearly distinguish the two datasets. An example of such features is shown in Figure 2.21.

Figure 2.21 represents a datasets that consists of 105 269 samples, 99 601 of those being negative samples. The plot shows two features plotted against each other, the number of dots in the host name and the length of the last segment of the host name (e.g. in *google.com* the length of the last segment is 3) . Additionally, the two classes are plotted with different markers, the green filled circles mark legitimate host names where the black outline circles mark phishing websites.

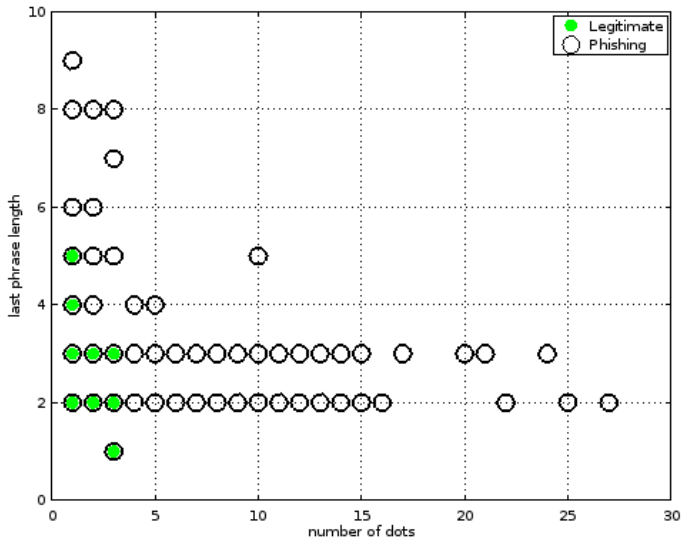


Figure 2.21: Number of Dots vs. Last Host Name Segment Length

The plot clearly shows a tendency in the host names structure where the values of the features for the legitimate websites do not exceed certain values allowing a small portion of the phishing host names to be easily 'cut off'.

This observation allowed us to design a simple machine learning based classifier using as input four most prominent host names features in different combinations. Table 2.8 represents the initial experiment results, where the numbers in the Features column represent respectively: 1 - number of dots in the host names, 2 - last segment length, 3 - second last segment length, 4 - third last segment length.

Clearly, increasing the number of features used for classification increases the recall rate, but it also decreases the precision in cases where it reached 100%. Another fact is that, the recall rate reaches up to 10%, and assuming the worst case scenario, where the number of phishing websites is equal to the number of legitimate websites, if a host name gets classified as a legitimate one, there is around a 55% chance that it actually is legitimate. However this number is an abstract and a very hard to measure assumption since the number of active phishing websites changes very rapidly and is significantly lower then the number of legitimate services.

2.3.4.2 Design Principles

Although the results of the initial investigation were promising, the dataset of phishing websites changes so fast that the results were outdated after a couple of days. This is clearly illustrated by the statistics Phishtaks dash-

Table 2.8: True positives (TP), false positives (FP), false negatives (FN), precision and recall for particular features combinations

Features	TP	FP	FN	Precision	Recall
One Feature					
1	364	0	5304	1	0.0642
2	11	0	5657	1	0.0019
3	3	1	5665	0.75	0.0005
4	58	2	5610	0.9666	0.0102
Two Features					
1, 2	387	0	5281	1	0.0682
1, 3	528	6	5140	0.9887	0.0931
1, 4	478	23	5190	0.9540	0.0843
2, 3	21	2	5647	0.9130	0.0037
2, 4	105	2	5563	0.9813	0.0185
3, 4	380	56	5288	0.8715	0.0670
Three Features					
1, 2, 3	551	5	5117	0.991	0.0972
1, 2, 4	513	23	5155	0.957	0.0905
1, 3, 4	773	164	4895	0.8249	0.1363
2, 3, 4	466	108	5202	0.8118	0.0822
Four Features					
1, 2, 3, 4	814	182	4854	0.8172	0.1436

board[2] where the daily volume of submitted suspicious websites reaches approximately 2 000 entries, reaching 3 000 in highest peaks. Out of all submitted entries around 50% are identified and confirmed phishing web sites.

Verification time of a phishing website ranges from almost 10 hours in January 2014 to over 61 hours in April 2014, as the assessment is performed manually by the PhishTank community. But even this short life span (the websites are blacklisted shortly after verification) allows phishing websites to do significant damage.

The aforementioned facts can be perceived as requirements that would have to be meet in order to facilitate and enhance phishing website detection. To generalize, the requirements can be enumerated as follows:

High performance

Because of the high number of phishing websites occurring daily, the prototype has to be able to cope with the 'worst case scenario'. The 'worst case scenario' would be described by the threshold statistics, that is to analyse 3 000 web sites (highest peaks) in less then 10 hours.

[2] PhishTank statistics - http://www.phishtank.com/stats.php

Full automation
The prototype has to function independently and autonomously. It has to be able to alone gather all the necessary data to make a website classification, furthermore, using that data to make an assessment of the website.

High precision
The prototype has to provide a high precision assessment, meaning that it will not classify legitimate web sites as phishing, providing at the same time highly accurate results.

2.3.4.3 Prototype Design

Taking into account previously specified requirements and initial studies we propose the following prototype design based on real-time machine earning algorithms, along with initial implementation description.

Figure 2.22 illustrates the initial architecture design along with the data flow. Since distributed systems tend to be much more effective and efficient in performing complex tasks then other systems, the prototype consists of four independent modules and an internal database:

Data collection module
This module is responsible for automated data gathering. It can be plugged into an external sources, download the datasets and extract the necessary information placing it in the database. The module distinguishes between phishing and non-phishing datasets, what is later used by the analysis module for learning.

Currently the prototype is able to query PhishTank and Alexa, but both datasets require ad-hoc processing because of various formatting of the information provided.

Feature extraction module
This module reads from the database the data initially provided by the *data collection module*, then, it extracts and stores in the database features that will be later used to train the machine learning algorithms.

Currently the module is divided into two sub-modules, each extracting a different features group: hostname analysis module and HTML analysis module. Both work independently and do not interact with each other.

The hostname analysis module does not need any interaction with the outside network and can extract the features based solely on the hostname of the URL taken from PhishTank and Alexa.

Features extracted by this module are based on the initial studies described before in the document. Negative (currently Alexa) entry with

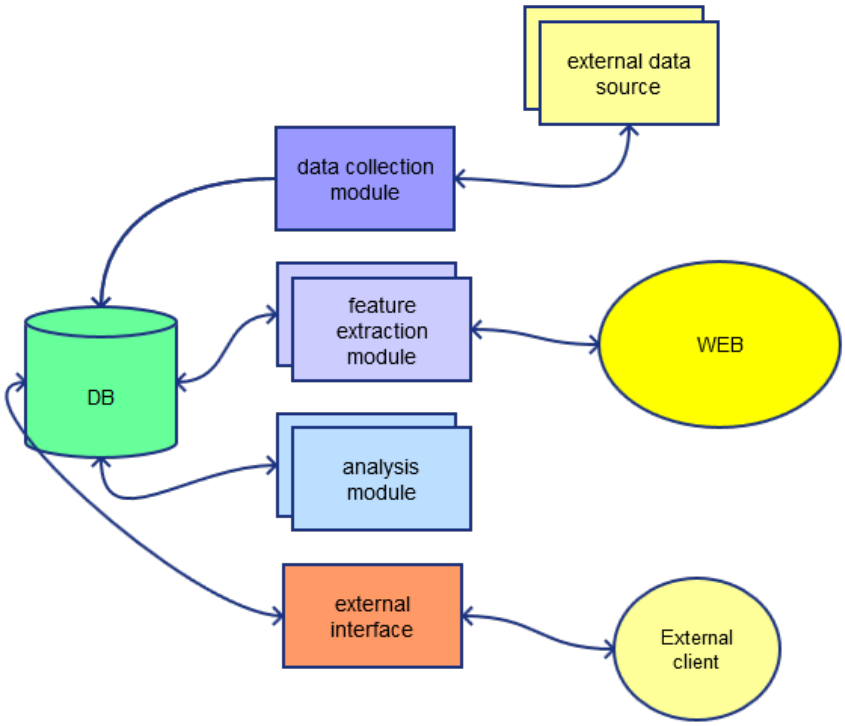


Figure 2.22: Prototypes architecture

the highest number of dots in the hostname is considered the negative threshold. The hostname segments are then extracted based on that.

The HTML analysis module is a web crawler able to query and analyse web pages, and extract relevant features. Currently it is looking for certain keywords as well as things such as the presence of the HTTPS protocol.

All extracted features are placed in the database as parameters of the associated URL.

Analysis module

The analysis module reads from the database the features extracted by the *feature extraction module* and uses those as training datasets to train the classification algorithms.

The classification algorithms are real-time machine learning based algorithms that can be fed a constant stream of data and adjust the decision boundaries accordingly. This gives the algorithm very high

flexibility and the ability to adjust to the changing environment without the necessity to store the training dataset once trained.

The *analysis module* is currently divided the same way the *feature extraction module* is divided: one classification algorithm per feature group.

The parameters obtained during the training are kept in the database for further use.

External interface

In order to make the prototype useful, an external interface is provided for external clients to submit suspicious websites.

The module is able to extract features on its own and implements the classification function of the classification algorithms.

Once a URL is submitted, it extracts the different features groups, and based on the latest trained parameters, classifies the web site. It provides the classification and the estimated accuracy of that classification

The prototype is already partially implemented and initial results show, that, considering only the hostname features group, for 1 000 new URLs submitted to the external interface, the prototype was able to 'cut off' with 100% precision 7% of the malicious web sites in less than 3 seconds.

Thus, we consider the results very promising and foresee, that in combinations with the classifiers mentioned in the other chapters we could maintain the precision at the 100% level while increasing the malicious services detection rate.

2.3.5 Classification of SSL Servers

In today's Internet, the Secure Socket Layer (SSL) and Transport Layer Security (TLS) are the most widely deployed security protocols used when a client and a server desire to securely exchange data over the Internet. SSL/TLS is used in several ways. Online businesses (e.g., online retails) use SSL/TLS to build customers' confidence that their sensitive information will not be compromised during online transactions. Enterprise mail servers utilize SSL/TLS to encrypt messages being transmitted over the Internet or within Intranets.

Unfortunately, as reported by Netcraft in 2012, SSL/TLS can also be used spitefully [63]. Netcraft found a significant number of phishing websites using valid SSL certificates issued by trusted Certificate Authorities (CA), such as Symantec and Comodo. These websites intended to employ HTTPS to convince victims to trust them. Even though they account for a small fraction of phishing attacks, they are eroding trust in SSL/TLS.

Table 2.9: Vulnerability score ranges with security levels to assess an SSL server

Total vulnerability score ($Tscore$) range	Security level
$Tscore < 0.5$	Best
$0.5 \leq Tscore < 1$	Good
$1 \leq Tscore < 1.5$	Average
$1.5 \leq Tscore < 2$	Bad
$Tscore \geq 2$	Worst

To establish an encrypted communication using SSL/TLS, a client and a server perform a handshake. The client requests the SSL certificate from the server. Upon receiving the server's certificate, the client performs certificate verification as follows. It uses the corresponding preloaded CA's public key to verify the authenticity of the digital signature in the server's certificate. It also validates the certificate by checking the certificate's issuance and expiration dates. Finally, it generally verifies that the service for which the certificate has been issued matches the service to which it wishes to connect.

The most popular SSL/TLS clients and servers are web browsers and servers. Typically, when encountering a server's certificate issued by an untrusted CA, an expired certificate, or a mismatched domain, browsers issue a security warning to users. Browsers also provide more security assistance to their users. For example, by clicking on a padlock icon in the browser window, users can see information related to the website's certificate, the certificate's issuer, and the period of validity of the certificate. Furthermore, browsers show a green address bar when a user is connecting to a website using an Extended Validation (EV) certificate. The green bar implies that such a connection is more secure, because CAs use an audited and rigorous entity authentication to issue an EV certificate [81]. However, users must eventually assume the responsibility of trusting an HTTPS website.

We propose three methods for classifying SSL/TLS servers in terms of security: (1) the certificate information-based method relies on heuristics using the identity information of certificates (also called Distinguished Names) and the issuing CA to determine the security level of a server. (2) The protocol version and encryption algorithm-based method identify known security flaws by looking at the protocol version and the encryption algorithm chosen by a server. (3) We also devise a technique to combine results from these two methods and compute a security vulnerability score, $Tscore$, that assesses the reliability of a SSL/TLS server.

Using the proposed methods, we classify a large dataset of real-world SSL/TLS servers that were active from July 2010 to May 2011. In our experiments the vulnerability score, $Tscore$, ranges between 0 and 3.5. Each

Table 2.10: Classification results based on security assessment score

Survey name	#Good servers	#Average servers	#Bad servers	#Worst servers
Jul-2010	4,462,945 (46%)	472,745 (5%)	4,494,127 (46%)	253,371 (3%)
Aug-2010	4,933,374 (42%)	521,230 (5%)	5,307,106 (48%)	283,523 (3%)
Dec-2010	3,720,084 (48%)	308,675 (4%)	3,482,081 (45%)	194,696 (3%)
Apr-2011	2,764,267 (39%)	258,491 (4%)	4,027,915 (56%)	84,200 (1%)
May-2011	1,526,521 (40%)	212,406 (6%)	2,053,710 (54%)	3,800 (<0.5%)

server in the dataset is assigned a security level according to its $Tscore$ and the rules of Table 2.9. The results are shown in Table 2.10, and reveal that no server could attain the "Best" security level in this survey. About 43% of the servers are "Good" servers and about 50% of the servers are "Bad". Less than 7% of the servers have either average or worst security levels. These results (Table 2.10) feature a bimodal distribution where modes represent both good and bad levels of security.

The three proposed methods can be implemented as a supplementary client-side security module (e.g., a web browser plug-in) to aid users in assessing the risk of their SSL/TLS communications. For example, a web browser integrating our classifier could raise a security alarm when the user's encrypted data may easily be compromised due to a weak cipher, or when the user connects to a malicious server. Therefore, we propose 45 features, deemed relevant to security assessment, for future SSL/TLS data collection and analysis.

2.3.6 Classification and Detection of Command and Control Connections through Malware-generated URL Clustering

We propose a behavioral classification system that leverages common HTTP features in malware C&C activity. Current network-based classification systems can still be misled by malware using multiple obfuscation techniques. They monitor all traffic triggered by malware, including C&C connections and other activities such as noise, attacks and connectivity checks, which do not reliably characterize a given malware family. While it is difficult to identify C&C connections during analysis of a single malware, the correlation and clustering of HTTP patterns across a large malware dataset reveals valuable similarities among samples of the same malware families.

Therefore, we process the HTTP traffic triggered by malware executed in a dynamic analysis environment, and we identify URL patterns that characterize its main C&C applications. We propose a two-stage classification of malware HTTP traffic. First, we build URL clusters based on the structure and statistical features of URLs. These clusters characterize malware activities such as spam, adclicks, and connectivity checks. Then we implement a fine-grained, noise-agnostic process that discards noise and builds clusters of malware C&C traffic based on patterns that are shared among samples of the same malware family. To the best of our knowledge, our system is the first to propose an automated network-based solution that isolates noise and builds detection signatures based only on malware C&C connections. Our experiments prove that it reliably characterizes and detects the C&C activity triggered by malware, with a very low false positive rate.

2.3.7 Identification of C&C communication in sandbox data

Execution of malware samples in dynamic analysis environment (sandbox) can provide us with valuable information. Here we propose a new classification system that aims at recognition of previously unknown C&C servers. Our system leverages a fact that similar malware samples show similar network behavior because they share code.

Modern botnets use many evasive techniques such as obfuscation and/or encryption of C&C protocols to avoid payload inspection-based detection. Malware samples also connect not only to their C&C servers but to legitimate servers as well. To disguise their C&C connection, botnets perform various network activities such as connectivity checks, time synchronization or noise generation. Therefore, it is difficult to recognize C&C connection based on single malware sample execution, especially encrypted one. Our system leverages the hypothesis that similar malware samples (e.g. belonging to the same botnet family) not only use similar C&C protocols but also show similarities in other network activities. Based on that assumption and using network traffic features such as transport protocol and sequence of payload lengths, our system builds clusters of samples that behave in the same manner. For samples belonging to one cluster there should be the same sequence of network connections both to legitimate and C&C servers. Using an initial set of both known C&C servers (blacklist) and legitimate ones (whitelist) we are able to identify the sequence of server classes for each cluster and build a network behavior profile. Comparing network behavior of unknown sample to the profiles and choosing closest cluster, we can determine class of each endpoint to which tested sample had connected and potentially discover previously unknown C&C servers.

Many of the system concepts come from similar work described in [28]. However, our assumptions are different and we use different distance computation algorithm used for clustering.

2.4 Cross-layer threat data analysis

Threat analysis is a fairly traveled field as indicated by previous efforts in completed FP7 projects, such as VIS-SENSE or WOMBAT. However, most analyses focus on a single data source or multiple sources of the same kind. Attempts at multi-layer analysis with focus on complex malicious campaigns are mostly missing. The attempts are also scattered, with analytical tools being developed separately for different datasets. While WOMBAT's common data access API went a long way towards enabling multi-dataset analyses, and with VIS-SENSE succesfully exploring the application of visual analytics to the task, the currently available tools do not fully exploit the various kinds of data for operational purposes.

NECOMA focuses on building analytic capability in threat areas not sufficiently covered by existing solutions and integrating the various analyses into a single, consistent tool, with a focus on the security analyst. With its truly international perspective and its range of datasets virtually spanning across all layers, *NECOMA* offers a platform on which malicious activity can be recorded to carry out multiple kinds of analysis. Such platform will not only allow for multi-layer analyses to be carried out over different datasets, but it will also allow researchers from different culture and background to bring with them approaches to elicit knowledge by joining features from different datasets in a cross-layer fashion.

2.4.1 Multi-layer vs Cross-layer

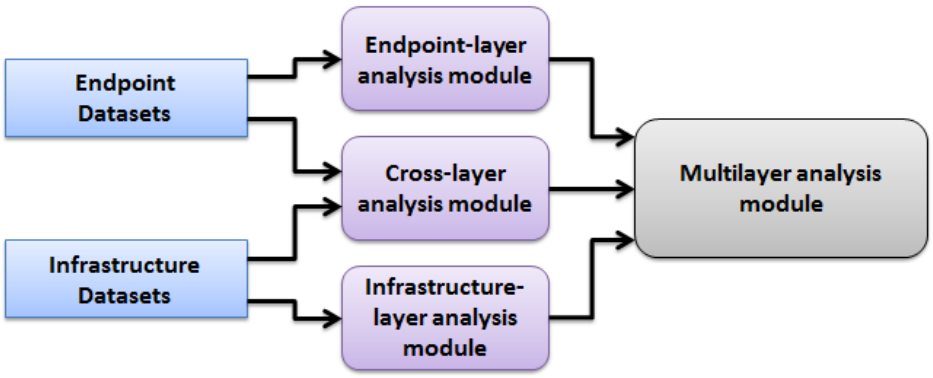


Figure 2.23: Cross-layer and Multi-layer analysis modules relation

As suggested previously, the term *multi-layer* has a broader meaning than *cross-layer*. We may give a brief definition of both terms as follows:

- *multi-layer* pertaining to multiple layers, i.e., endpoint and infrastructure

- *cross-layer* bridging the border between layers, i.e., endpoint and infrastructure (implies multi-layer)

These terms apply to both the threats to be analyzed and the methods used to perform the analysis. As an illustration, we provide a few examples:

- in the context of threats, any attack on multiple layers is a multi-layer attack (e.g., combining phishing with a simultaneous DDoS against the original site), but the term cross-layer attack requires using one layer to attack the other (e.g., using DNS spoofing to redirect to a phishing site or using a drive-by-download to build a botnet and perform a DDoS);

- in the context of threat analysis, processing multiple datasets from different layers always qualifies as multi-layer analysis, even if separate tools and methods are used for each dataset, but a cross-layer analysis means processing multiple datasets from different layers in a single analysis with one or more anchors (such as IP addresses, time, ports) to bridge the link between the two.

In particular, *NECOMA* integrates all analysis modules within the Threat Analysis Platform as described in Chap. 4, which will allow for both multi-layer and cross-layer analyses to take place as illustrated in Fig. 2.23. This platform will allow an analysis to fetch a broader range of information to individual features, to analysis results or raw data in order to perform complex data mining operations. Correlating these pieces of information will be achieved using one or more anchors, bridging datasets, even across layers. In *NECOMA*, a number of these anchors, called *tokens*, have been proposed and integrated into the n6 API.

2.4.2 A Taxonomy of Internet HTTPS Phishing Sites

Many users have been trained to trust websites using HTTPS and thus sophisticated phishers are now setting up HTTPS phishing websites in order to fool victims and extract sensitive information from them. Without knowing how such websites use HTTPS, efforts to combat them may be inefficient or misguided. As a guideline for future efforts, we have created a taxonomy of HTTPS phishing sites based on the list of HTTPS phishing URLs collected within the past two years. The URLs, together with environmental evidences, were manually inspected and categorized. We have found that more than 90% of the HTTPS phishing sites are parasitic, meaning that they do not use their own SSL certificates, but use those of legitimate sites. The three most prevalent categories are online forms, shared SSL domains and hacked domains respectively.

2.4.3 FP-growth based malicious campaign analysis

Frequent pattern mining is a widely used technique for discovering interesting relations between data items in large databases. It is employed today in many application areas including malware detection and Web usage mining. In our research [50] we use the frequent pattern tree structure (FP-tree) – a prefix structure for storing quantitative information about frequent patterns in a database – and the FP-growth algorithm for frequent pattern discovery using a divide-and-conquer strategy, both developed by J. Han et al. [38].

2.4.3.1 Operation of the analysis

The FP-growth algorithm operates in two steps:

1. The FP-tree compact structure is constructed,
2. Frequent patterns are extracted from the FP-tree.

In the first step the patterns occurrence in the input transaction database is counted. Next, infrequent patterns are discarded, frequent patterns are sorted by descending order of their frequency in the database, and the FP-tree structure is built. Common, usually most frequent patterns are shared. Therefore, FP-tree provides high compression close to tree root and can be processed quickly. Recursive growth is applied to extract the frequent patterns. FP-growth starts from the bottom of the tree structure (longest branches), by finding all patterns matching given condition. New tree is created, etc. Recursive growth ends when no patterns meet the condition, and processing continues on the remaining main branches of the original FP-tree.

The Support Vector Machine (SVM) [82] is a supervised learning classification method widely used in data mining research. The concept of SVM is to classify each data sample into one of two categories: positive class denoted by "+1" and negative class denoted by "-1". Thus, the goal is to determine a decision boundary, which divides data into two sets (one for each class), a plane for $n \leq 3$ or hyperplane for $n > 3$. Next, all the measurements on one side of this boundary are classified as belonging to "+1" class and all those on the other side as belonging to "-1" class. The problem is that many such hyperplanes can be determined, and the best one has to be selected. Hence, SVM tries to learn the decision boundary which gives the best generalization. A good separation is achieved by the hyperplane that has the largest distance to the nearest data sample of any class – a wider margin implies the lower generalization error of the classifier. To select the maximum margin hyperplane an optimization problem is formulated and solved for a given training dataset.

The original SVM model – linear classifier – was developed by V.N. Vapnik, [82]. The hyperplane is calculated as a solution of a quadratic optimization problem. However, in many practical applications the datasets are not linearly separable in the data space. To determine the hyperplane the original space is mapped into a much higher-dimensional space, and a kernel function is formulated and minimized. The kernel function $K(x_i, x_j)$ defines the similarity between a given pair of objects. A large value of $K(x_i, x_j)$ indicates that x_i and x_j are similar and a small value indicates that they are dissimilar. Various kernel functions are described in literature [42].

2.4.3.2 Implementation

The architecture of our system for malicious campaigns identification (FP-SVM) is presented in Fig. 2.24. It consists of three main modules: a database collecting malware URLs, a frequent pattern mining module that implements the FP-growth algorithm for frequent patterns discovery and a data classification module that uses the SVM method to classify malware URLs as related or not to a campaign. To produce the SVM classifier model we need

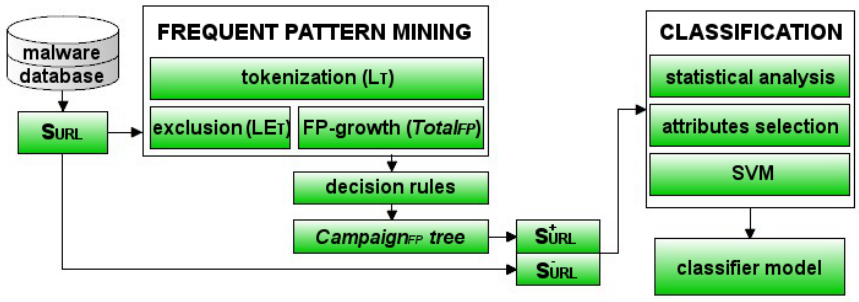


Figure 2.24: The architecture of the FP-SVM system.

a training dataset of URLs that have already been classified into campaigns. Unfortunately access to such databases (if they exist) is restricted. Therefore, the only sensible solution is to define some patterns of URLs related with campaigns based on analysis of malware datasets, and generate a set of samples containing these patterns. It is assumed that all these samples are related with campaigns, and can be used to train the SVM classifier. In our FP-SVM system the FP-growth algorithm is applied to produce the training dataset. A fixed number N of URLs are selected from the database, and the input dataset $S_{URL} = \{URL_1, URL_2, \ldots, URL_N\}$ for frequent pattern mining is created. Each sample from S_{URL} is tokenized. The simple heuristic rules to break up a given URL (stream of text) into shorter strings described in literature [16, 36] are adopted. Each sample is cut using specific characters that are typical for URLs, i.e., ”/”, ”.”, ”?”, ”#”, etc. As a final result of this operation we obtain a set of tokens L_T.

This set of tokens becomes input for further processing such as parsing and typical subsequence mining. The goal is to reduce the size of the dataset consisting of extracted tokens and finally speed up the FP-tree generation. The typical URL's attributes which do not carry valuable information, such as the schemes: "http", "https", domain name parts "www", "org", "com", "waw", etc. and extensions: "exe", "php", "html", "xhtml" are excluded from the set L_T. Once the final set of tokens LE_T is built, the FP-growth algorithm is employed to discover frequent tokens and the FP-tree structure $Total_{FP}$ storing quantitative information about frequent tokens from LE_T is constructed. In this tree each node (besides root) represents an extracted token that is shared by all subtrees consisting of itself and all the nodes beneath it. Each path in the tree shows a set of tokens that co-occur in URLs. Thus two URLs that contain several identical frequent tokens and differ in several infrequent tokens share a common path. The root is the node that has no superior and separates all disjoint sub-trees. The $Total_{FP}$ tree structure is analysed. Simple decision rules are used for data processing. These rules define the characteristics of each URL that is suspected to belong to any campaign.

The final FP-tree structure $Campaign_{FP}$ formed by URLs with these characteristics is created. Next, all URLs from the dataset S_{URL} containing the tokens from the $Campaign_{FP}$ tree are classified to the *positive class* denoted by "+1", and form the set S^{+}_{URL} of URLs forming malicious campaigns. Other URLs from S_{URL} are classified to *negative class* denoted by "-1", and form the dataset S^{-}_{URL} consisting of URLs that are unrelated to any campaign. Both these datasets form a training set of samples that is used to produce the SVM classifier model. The relevant attributes used for URL classification are selected. The commonly used attributes assigned to URLs are: date, time, address (IP, ASN), length of address, domain name, length of domain name, number of subdomains, path name, length of path name, number of subpaths, length of query, number of queries, country code, confidence of code. The selection of adequate attributes is a key feature that guarantees the effective and efficient classification.

Next step of the SVM algorithm is to learn the decision boundary. Four variants of the SVM classifier with linear and nonlinear kernels are implemented in our FP-SVM system. The following nonlinear kernels are provided: polynomial function: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$, radial basis function: $K(x_i, x_j) = exp(\gamma \parallel x_i x_j \parallel^2)$, sigmoid function: $K(x_i, x_j) = tanh(\gamma x_i^T x_j + r)$, where $\gamma > 0$, r, and d denote kernel parameters. Finally, the trained SVM classifier can be employed by malware detection systems to classify malicious URLs taken directly from the Internet as related or unrelated to known malicious campaigns. Note that suspicious, unverified URLs can also be analyzed – if they are found likely to be part of a campaign, their malicious status is confirmed.

The algorithms considered in this section are not limited to URLs. Application on domain names is also possible and potentially interesting. Experiments involving japanese DNS datasets are currently being performed. Since these records belong to the infrastructure layer, the analysis is clearly multilayer. Mixing the two datasets, e.g. by learning on URL sources and applying the classification to DNS sources, is therefore clearly a cross-layer analysis.

2.4.3.3 Performance evaluation

The n6 database collects data taken from various sources, including security organizations, software providers, independent experts, and monitoring systems serviced by CERT Polska. The datasets contain URLs of malicious websites, addresses of infected machines, open DNS resolvers, etc. Most of the data is updated daily. Information about malicious sources is provided by the platform as URL's, domain, IP addresses, names of malware, etc. We have performed a preliminary analysis of 27 700 560 URLs collected in year 2013 and stored in the n6 database. Figure 2.25 depicts the amount of malicious URLs observed per day (in a selected month) and per month. The average number of observed URLs is about 80 000 daily or 2 300 000 monthly.

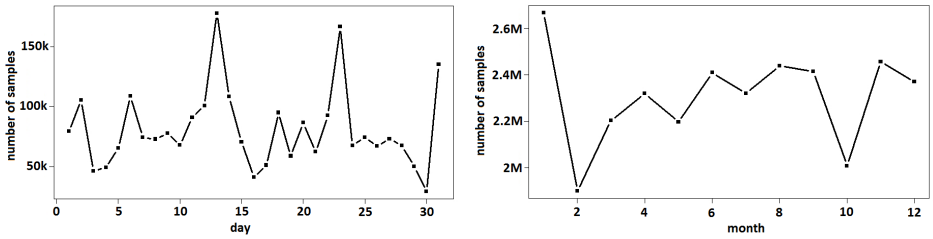


Figure 2.25: Malicious URLs detected during a day and during a month.

The aim of the experiments was to validate the FP-SVM system on the n6 dataset. First, N malicious URLs were selected from the n6 malware database. They formed the S_{URL} training set. Next, frequent pattern analysis was applied and the $Total_{FP}$ tree with nodes representing tokens extracted from URLs from the S_{URL} dataset was constructed. The following rule was used to extract the subtree $Campaign_{FP}$ from the original $Total_{FP}$ tree. We assumed that all URLs containing m common tokens in a sequence were suspected to be related with the same campaign. Hence, short branches with less than m nodes were excluded from the $Total_{FP}$ tree. The $Total_{FP}$ tree and the approach to the $Campaign_{FP}$ tree generation are presented in Fig. 2.26. Table 2.11 presents the results of the application of the FP-growth algorithm and our decision rule with $m = 4$ to the S_{URL}

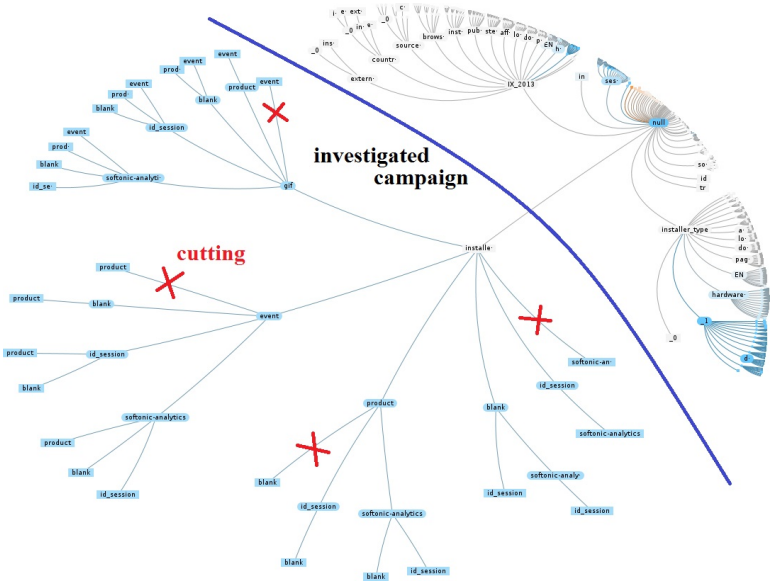


Figure 2.26: Frequent patterns tree and a campaign extraction.

dataset. It contains the number of detected malware campaigns and average number of URL's tokens related with one campaign.

Table 2.11: Identification of malicious campaigns; the S_{URL} dataset.

Number of URLs (number of IP)	500 000 (108 745)
Number of unique URLs (number of IP)	69 906 (5 321)
Number of detected campaigns	72
Average number of related attributes	84 per campaign

Next, the input training set of URLs $S_{URL} = S^{+}_{URL} \cup S^{-}_{URL}$ was used to produce the SVM classifier model. Various sizes of S_{URL} were considered, i.e., $N = n \cdot 100000$, $n = 1, 2, 3, 4, 5$. The decision borders for classification were calculated and validated for each size of S_{URL}. Finally, the trained SVM classifier was applied to malware campaign detection. It was used to classify the dataset consisting of 80 000 malicious URLs (unrelated with URLs from training dataset) into two categories: +1 – URLs related with any campaign, -1 – URLs unrelated with any campaign. Then, the quality of the classification was assessed. The following commonly used criteria were considered: classification accuracy (CA) – ratio of number of correctly identified URLs to the size of the dataset, sensitivity – the proportion of positives that are correctly identified as such, specificity – the proportion of negatives that are correctly identified as such, accuracy of a test (AUC) – the measure that shows how well the test separates the URLs being tested, Precision – how close the separated URLs are to each other, F-measure – another measure of

a test's accuracy. The values of all mentioned criteria obtained for various variants of SVM classifier (providing linear and nonlinear kernel functions) and a training dataset S_{URL} consisting of 250 000 malware URLs are collected in Table 2.12. Figure 2.27 shows the accuracy of the classification for various sizes of the training dataset. In general, the results presented in

Table 2.12: Evaluation of SVM classification; $N = 250000$ URLs.

	Radial	Sigmoid	Polynomial	Linear
CA	0.7035	0.5571	0.5000	0.3495
Sens	0.8750	0.7083	0.4375	0.6042
Spec	0.8202	0.6685	0.8258	0.6011
AUC	0.9250	0.8653	0.8367	0.7823
F1	0.6885	0.4823	0.4200	0.3919
Prec	0.5676	0.3656	0.4038	0.2900

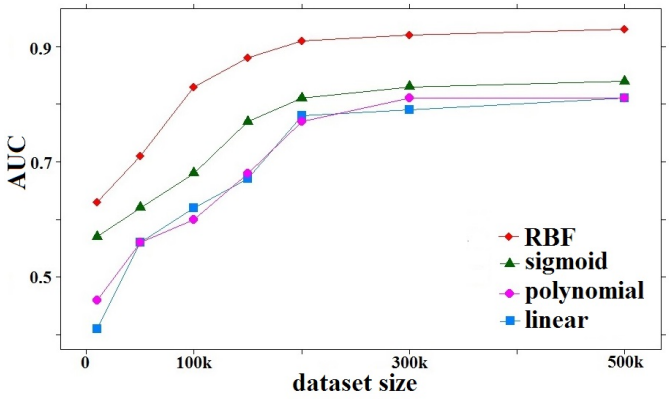


Figure 2.27: Accuracy of classification; various sizes of the training dataset S_{URL}.

Fig. 2.27 and Table 2.12 confirm that the accuracy of classification strongly depends on the dimension and quality of a training dataset. Moreover, it is very important to choose the adequate kernel function. The achieved classification accuracy ranged from about 35% to 70%, accuracy of the test from 78% to 93%, sensitivity from 44% to 88%, and specificity from 60% to 82%. Explicitly, the worst results were obtained for the SVM variant implementing a linear kernel function. The best results – the best values of all criteria – were obtained when employing the radial basis kernel function.

2.4.4 Graph-based malicious campaign analysis

Availability of multiple rich datasets within *NECOMA* is naturally a great opportunity for automatic analyses, extracting valuable information from

a wealth of data far beyond any possibility of effective manual analysis. Significant value can be provided by searching for related information based on an interesting entry in one of the datasets. Automating such a process is potentially a useful part of the platform envisioned in *NECOMA*.

2.4.4.1 Graph representation of the information in NECOMA datasets

One of the results of our work performed within the *NECOMA* project was the identification of a set of common tokens that appear in multiple datasets of different types and even layers. These tokens, such as IPs, MD5 hashes, domain names, can be used to correlate data from different datasets. The graph-based analysis proposed in this section is a simple tool enabling exploration of the entire collection of datasets based on seed tokens: interesting values either identified by other analyses, or obtained in a different way by the user of the system. The goal of the approach is to find as much information related to a given token as possible, while limiting the cost of such operation.

Related information within different datasets can be modeled as a graph, with information from a single dataset represented as a node and shared tokens represented as edges. Thus, for example, an identified spam campaign (dataset A) can share a URL with a phishing site (dataset B), which may share an IP with an exploit server (dataset C) and so on. Depending on the level of linkage between various datasets, a graph constructed in this way can be arbitrarily large, as each new node may introduce several new tokens. In general, the entire collection of data within *NECOMA* can be modeled as a single large graph. Obviously construction of such a graph is neither realistic nor useful, due to its size. However, a well chosen subgraph can be very useful as a source of contextual information. A well designed subgraph can answer many important questions, such as whether an address (either IP or URL) was used to host malicious content, whether a URL is related to a given spam campaign. It can even be used to reconstruct a significant part of the infrastructure of a botnet based on a single C&C server.

Unfortunately building such a graph is an iterative process, where each new token requires a new query to all datasets containing tokens of that type. Given the globally distributed nature of the *NECOMA* system, such operations may be time- and resource-consuming.

2.4.4.2 Search algorithm

The rating mechanisms proposed in chapter 3 enable us to enhance the process, aiming at extracting the most valuable information with reasonable amount of effort. To achieve that we want to focus on queries most likely to produce interesting results, limiting less promising ones.

The starting point of the procedure is a single interesting token. The token can be taken from one of the datasets or a user-provided value. Alternatively, the output of the other analyses within *NECOMA* can be redirected to this analysis to be enriched with related information from other datasets, which may not have been used in the original analysis. Starting with such a root token, a graph of related entries should be built. A recursive, depth-first approach is a simple way to achieve a full graph, but – as mentioned before – this is costly and may result with a graph far too large to be useful, with little or no relation between nodes connected by longer paths. Limiting the depth of search (whether depth-first or breadth-first) will prevent such uncontrolled growth, but does not take into account the different probability of getting a non-empty result, generating unnecessary queries, while at the same time possibly cutting some very promising branches. Our solution creates a graph of tokens and assigns a cost to each query, building the graph by selecting cheapest queries until a preset budget is consumed.

One important characteristic of this approach is that each token is also an identifier of a single node of the graph. Therefore, whenever the same token is encountered again, it will not create a new node; it will be appended to the original node with the new information.

Our solution builds the graph using a growing set T of tokens $t_0, \ldots$, where t_0 is the root token. Each token is of a certain type $\tau(t_k)$, which identifies the set $D(\tau)$ of datasets that contain tokens of this type. For each token we define the following associated information:

- $D_n(t_k)$ – the set of datasets not yet queried about the token,
- $D_v(t_k)$ – the set of datasets already found to contain the token,
- $I(t_k)$ – the set of dataset entries containing the token.

For a simplified description of the algorithm, assume a cost function $\xi(t, d)$ and initial budget C.

We start by setting $T = \{t_0\}$, $D_n(t_0) = D(\tau(t_0))$, $D_v(t_0) = \emptyset$. Now the procedure for building the graph is as follows:

1. Select a pair $(\hat{t}, \hat{d})$ such that $\hat{d} \in D_n(\hat{t})$ and $\forall_{t \in T, d \in D_n(t)} \xi(\hat{t}, \hat{d}) \leq \xi(t, d)$.
2. If $\xi(\hat{t}, \hat{d}) > C$, STOP. Otherwise decrease the budget C by $\xi(\hat{t}, \hat{d})$.
3. Remove $\hat{d}$ from $D_n(\hat{t})$.
4. Query dataset $\hat{d}$. If the token is found:
   (a) Add the obtained information to $I(\hat{t})$.
   (b) Add $\hat{d}$ to $D_v(t_k)$

(c) Extract new tokens $t'_i, i = 1 \ldots$ not yet present in T from the obtained information and – if found – add them to T.

(d) For each new token set $D_n(t'_i) = D(\tau(t'_i))$, $D_v(t'_i) = \emptyset$.

5. Return to step 1.

The effectiveness of this approach depends on the cost function $\xi(t, d)$. To be effective it must take into account the rating results for the different datasets. The basic cost should be proportional to the likelihood of finding the token t in dataset d, which depends both on the characteristics of the dataset itself and the linkage between it and other datasets in $D_v(t)$. Additionally, the cost function should reflect the actual difficulty of the query, delay due to geographical location, etc. Therefore the actual design uses four separate functions:

- f_c is a simple cost function, reflecting the difficulty of obtaining information from a given source for a given token type,
- f_u is the utility function of adding a given token to the graph, involving the different ratings of the dataset (for a cartain token type),
- f_s is the strategy function mapping the values of the above two functions into a choice of token and dataset providing the best expected cost/utility ratio.

The f_u function hints at the existence of a more general function f_U, measuring the utility of the entire graph. In this case f_u is the expected increase of f_U if a certain token is used to query a certain dataset.

The utility and cost functions need to be defined for each dataset separately. Not all tokens of the same general type are equally interesting information – an IP address of a C&C server is clearly different than simply the IP address for a given domain name, even if the domain itself is malicious. In most cases the queries should also be limited to a certain time period, but that is also dataset-dependent.

The proposed analysis has two separate use cases.

In a **interactive mode** it is a tool used to explore the datasets, starting with a manually chosen token. Since in this case the nodes to be expanded can be selected by the user, the utility and strategy functions must include this choice as an important part. Each selection by the user increases the budget, but the further expansion should focus on providing information associated with the selected token. The utility function f_U measures the information content of the graph as the goal of the analysis is to increase the information available to the user.

In **automatic mode** the analysis processes tokens provided by other analyses or taken from some interesting datasets. The utility functions are

different in this case and focus on the chance of identifying clearly malicious behavior. Datasets registering malicious events with low false positives ratio are more important in this case. Once the budget C is expended the utility function f_U of the graph is a base for final decision – if the utility is low, the results are not considered interesting. The generated graph may be stored for a short period of time in case the analyzed token proves interesting for other reasons – in this case the graph is a source of additional information for the user. If the utility is above a preset threshold, the graph is always kept and the user of the platform is alerted that the starting token was identified as part of more complex malicious action. The graph can then be further expanded in interactive mode, using more information-centric utility functions.

The precise tuning of cost, utility and strategy functions require practical experience with multiple datasets and is relegated to the final stage of *NECOMA* development.

3

Development of automated rating and classification mechanisms

3.1 Taxonomy of Anomalies in Backbone Traffic

Anomaly detection techniques for backbone traffic rely on diverse statistical methods such as wavelets [8], Kalman filters [78], hash projection [49, 27, 13], principal component analysis (PCA) [52, 46], and pattern recognition [33]. These techniques however provide little or no information regarding the nature of identified anomalies. Since inherent features of anomalies, such as connection patterns, protocol usages, and traffic volume are essential to evaluate the effect of anomalies on network infrastructure, we propose a taxonomy for network traffic anomalies to classify anomaly detector results.

Our goal here is to assist network administrators to monitor anomalous traffic by providing a framework to classify anomalies into precise and meaningful categories. Previous works address network anomaly classification but usually for a limited and specific set of categories (less than 10). [53, 89, 30, 76, 80]. Detailed taxonomies have also been proposed. These either focus on a specific type of anomaly like distributed denial of service (DDoS) attacks [58] or scans [9], or consider network anomalies in general [69, 59]. These taxonomies provide diverse, non-overlapping and thus incomplete coverage of all known network anomalies. Furthermore, none of these works provide any material that would allow third parties to easily reproduce the results.

3.1.1 Proposed Taxonomy

We propose a network anomaly taxonomy, consisting of a set of anomaly labels (e.g., Denial-of-Service, scan, outage) and corresponding signatures. We build the taxonomy through an iterative process that we bootstrap by ap-

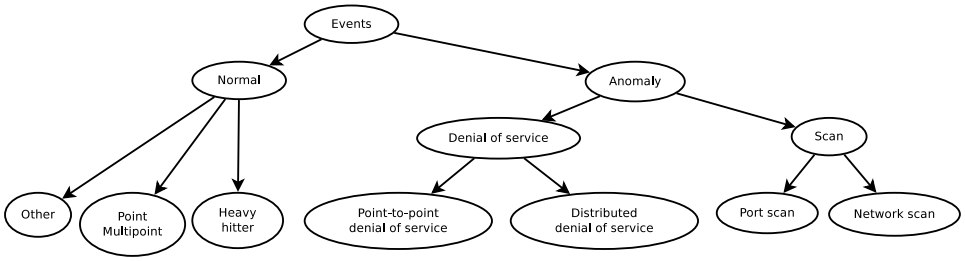


Figure 3.1: Overview of the proposed taxonomy for backbone network traffic anomalies. The complete taxonomy includes more than 100 different labels.

plying expert knowledge on network anomalies. We then iteratively refine our anomaly descriptions by carefully examining events that are flagged by detectors but not classified in the taxonomy. Such events are carefully analyzed, and appropriate signatures are built if an interesting and previously uncharacterized behavior is observed.

Figure 3.1 depicts the overview of the proposed taxonomy. It consists of two main categories: anomalous and normal events. Anomalous events comprise denial-of-service and scans, while normal events include heavy hitter, alpha flows, point-multipoint traffic, and other kinds of events (outages, tunnels, small point-to-point flows). Some events may be considered either legitimate or malicious depending on the context. For example, scans can be research activities [74] or attack precursors. In this work, we follow a conservative and pessimistic approach that considers scans as anomalies.

The taxonomy is built as a tree in which each node contains an anomaly label. The closer a label is to the root of the tree, the more general it is. Each label *may* be associated with a signature. A signature is a set of rules specifying detailed traffic features representing the nature of an event. Our current implementation comprises more than 100 different signatures covering all taxonomies reported in previous works. The proposed signatures are also made publicly available on the Internet [56].

3.1.2 Signature Matching

We assign a *single* label to each anomaly identified in backbone traffic. We first try to match an event with a label belonging to the subtree whose root is the node labeled “anomaly” in Figure 3.1. If there is no match, we repeat this process with the “normal” subtree. If there is still no match, the event is labeled as “unknown”. One event can match several signatures inside one of the two subtrees. In this case the assigned label is the most specialized one, or in other words, the one furthest from the root. Therefore, we ensure that anomalies are annotated with the most accurate labels. The proposed signatures have been evaluated with 6 years of network traffic from the MAWI

repository [56]. Our study using real traffic shows that our results improve on previous classification results by reducing the proportion of unknown events and providing new insights in terms of anomaly occurrence.

3.2 Automated rating and classification of network threat information

Several honeypot designs have been proposed. The two main axes upon which a honeypot is designed is the level of interactivity with the attackers and which side is targeted by the attacks the honeypot will monitor. Concerning the level of interactivity, honeypots can either do simple service emulation (low-interaction), more advanced emulation (medium-interaction) or run real services (high-interaction). As far as the second axis is concerned, we can categorize honeypots as server-side or client-side. The vast majority of honeypots protect the server side. The fundamental difference between the two types is that client-side honeypots search for attackers, instead of waiting to be attacked.

Low-interaction honeypots. Low-interaction honeypots only superficially simulate services. They provide very limited interactivity compared to high-interaction honeypots, but are still useful to lure attackers and gather information at a higher level, e.g., detect network scanning activities. A powerful characteristic of low-interaction honeypots is their ability to enable a single host to claim multiple addresses and run multiple services on each address.

Medium-interaction honeypots. Medium-interaction honeypots also emulate services but, unlike low-interaction honeypots, they do not manage network stacks and protocols themselves. Instead, they bind to sockets and leave the connection management up to the operating system. They mainly focus more on the application-level emulation part, instead of implementing network stacks and protocols as the low-interaction honeypots do.

High-interaction honeypots. High-interaction honeypots, unlike the two previous categories, do not emulate services. On the contrary, they run services in their native environment. Thus, high-interaction honeypots have the advantage that they are real systems; no emulation is used, no fake services run. Therefore, unknown bugs are still present and can be exploited. By being vulnerable to attacks, they can provide useful information on how previously unknown threats emerge and propagate.

3.2.1 The FORTH honeypot network

The FORTH honeypots core is not a centralized farm of honeypots used to gather information in the *NECOMA* project. On the contrary, it is a distributed set of honeyfarms that can collaborate. Inside the core, multiple in-

stances of AMUN low-interaction honeypot are deployed. These honeypots are able of capturing autonomous spreading malware from the internet. To do so, they emulate various existing vulnerabilities. When an attacker tries to exploit an emulated vulnerability, the received payload is analyzed and any contained URL is extracted. Then, the honenypots attempt to download all the malicious binaries, which are stored locally for further analysis.

3.2.2 Automated rating and classification of FORTH honeypots threat traffic

Here we propose an automated rating and classification design that can be applied on the data collected from the network of FORTH honeypots. FORTH honeypots passively wait for attackers to attack them. By default, all traffic destined to FORTH honeypots is malicious or unauthorized as it should not exist in the first place; FORTH Honeypots monitor a portion of a dark (i.e., unused) address space in order to be able to identify possible incoming attacks.

Information collected by the FORTH honeypots is stored in a database. This information is in form of threat incidents. These threat incidents contain all the necessary information of the attacks received by the honeypots, that is the origin of the attacks (e.g. source IP address and port, the communication channels used, the included payloads).

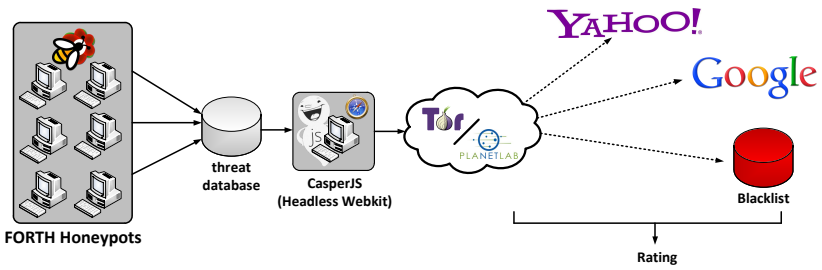


Figure 3.2: A generic overview of the proposed rating system.

An automated rating and classification system on the traffic gathered by FORTH honeypots, could be based on external knowledge, derived from the web (e.g., search engines, blacklists), in order the most prevalent threats to be identified. The input provided to this system could be the top attack sources, gathered by the FORTH honeypots, based on a characteristic, e.g., the attack frequency (i.e., how many times a source IP address tried to communicate with our FORTH honeypots). Then, a web scraper could be used, that will try to find each one of these sources in the various web sources. Such web sources may be various search engined (e.g., Yahoo!, Google, etc.) and various blacklists. An overview of the design of the proposed system is illustrated in Figure 3.2. As we can see, the data derived

from FORTH honeypots' database, that is the top attacking sources based on the current knowledge, will be furthered searched out in various search engines and blacklists. A score will be created based on the number of appearances of such sources in the results of the search engines and in those of the blacklists. This cross-checking could help us to rate and classify the threat traffic of FORTH honeypots.

As shown in the Figure 3.2, different kind of distributed networks, could be used in order to successfully perform multiple search engine queries, as search engines often apply a requests limit to a number of certain queries that a single IP address can made in a time window. If this limit is exceeded, then a measure is triggered; for example Google requires solving a CAPTCHA, in order to be able to continue. We could address this problem by leveraging distributed networks, such as TOR and PLanetLab, and using them as proxies.

3.3 Automated rating of data sources for graph-based analysis

The wealth of datasets available in *NECOMA* are a valuable resource, but at the same time provide an interesting problem, as the usefulness of information differs significantly between different sources. This aspect is most important when the selection of a dataset to query is to be made automatically, as in case of the graph-based analysis described in section 2.4.4. The most effective approach to this problem is to rate the available datasets on various qualities. For obvious reasons preferred rating methods are those that can be effectively automated.

The following section discusses various properties that can be used to rate usefulness of datasets. The choice of properties to decide if a dataset is useful or not obviously depends on the use case. The following text does not deal with this problem, the selection of the right ratings is delegated to the analyses that require dataset ratings. Moreover, the described properties are not normalized, at least in their current form.

This section builds on the work of A. Pinto and K. Maxwell [68].

3.3.1 Rating of datasets with mixed activity

This section refers to datasets containing both benign and malicious activity, with no labels, such as backbone traffic or DNS queries.

Coverage. Given a particular activity of interest (e.g. DNS queries globally, traffic to/from a particular country), determine how much of this activity is captured in the dataset under examination. It can be viewed as an absolute measure of the volume of data. Proposed methods are dataset-specific. For example:

- **backbone traffic** the ratio of the number of IP addresses observed communicating to the number of IP addresses allocated globally,
- **DNS queries to root servers** the ratio of the number of IP addresses observed asking to the estimated number of all DNS resolvers globally,
- **DNS queries to local resolvers** the ratio of the number of IP addresses observed querying to the number of IP addresses allocated globally.

All the values used should be computed over a common, fixed-length period. The rating scales naturally – better ratio is a better rating.

In case of a global platform, such as *NECOMA*, the rating can be computed in two different ways – globally or locally. The global case shows the amount of collected data compared to the total global amount of data of a given type. The local case is limited to the monitored space. The two ratings are very different – e.g. a dataset providing a complete record of all backbone traffic in a single country would obviously have a maximal local rating, but a relatively low global rating. On the other hand, collecting a small sample of traffic on all backbone links would create a dataset with local and global rating equal but low.

Representativeness. This rating is based on country or ASN distribution based on observed IP addresses and is difficult to compute for some datasets. First we compute a reference distribution (based on all other data sets for example) then compare it with the distribution in the examined dataset. The representativeness of a dataset is better if the two distributions are more similar.

Linkage. This very useful rating measures the degree of correlation with other datasets. This property would provide an estimate how useful a particular dataset is in context of analyses that combine multiple datasets. It is very much dependent on other datasets that are already collected – the reference set.

Computing linkage is base on the assumption that all datasets are collections of events, where an event is a collection of tokens (IPs, domains, URLs). The datasets are compared over a chosen time range (e.g. week), the procedure can be repeated for multiple overlapping or non-overlapping intervals. The linkage is the proportion of tokens that the examined dataset has in common with the reference set in the given period. The whole procedure can be repeated to rate all datasets, leaving one out for examination and using the rest as the reference set.

Another useful rating is a linkage map, which rates the amount of common tokens between each pair of datasets separately. Computing this type of rating is relatively simple, but it does require a large amount of data – $O(n^2)$ where n is the number of datasets being compared. This approach is described later, as a *NECOMA*-specific rating.

3.3.2 Rating of threat datasets

These datasets contain only known threats (URLs, IPs, etc) and no benign activity, apart from false positives. Examples of datasets include all datasets currently in n6, spam, etc.

False discovery rate. This rating is usually defined as the ratio of the sum of false positives to the total number of all alarms. Potentially this is a very good measure, however in most cases it is difficult to determine whether an alarm is a false positive. In some cases automatic methods could be developed to correlate a threat dataset with a mixed one to confirm alarms (e.g. perhaps it is possible to identify C&C servers based on backbone traffic characteristics), but accuracy of this method of confirmation would have to be known precisely, which is almost impossible in real-life. Moreover, since attacks can be targeted (watering hole attacks), even a perfect method can not verify all classifications. Due to these difficulties, we will not investigate this possibility further.

Rate. In general this rating measures the amount of data collected by a given dataset. For blacklists this rating is best expressed as the rate of change – how many entries added or removed daily. For sources producing a stream of discrete events, the rating is calculated as the number of unique IPs, domains or URLs per day.

This statistic provides an information on the volume of data but if sources collect information on different threats (e.g. different botnets), this value is not comparable. Even if sources report data on the same threat (e.g. the GameOver Zeus botnet) a source providing more data is not necessarily better – the real value of the dataset depends on the quality of data, measured as the false positives rate. Pinto and Maxwell define this property as "novelty", however they focus on IP blacklists.

Delay. Applicable for sources where it is possible to determine the time between detecting a threat and reporting it. Descriptive statistics can be used to compare delay introduced by various sources. This property can be used to compare almost all sources. Smaller delay indicates a better source.

Representativeness. In principle this rating would work the same as for datasets with mixed activity. Pinto and Maxwell have a similar approach for IP blacklists – they use the term "population" when referring to methods to investigate country distribution in datasets. They use GeoIP database to establish the number of IP addresses in countries and then apply two methods for comparison against distribution in a particular threat dataset that are under examination: exact binomial test and chi-squared proportion test. Authors propose to use this information for trend analysis and data cleanup (not necessarily to rate datasets directly).

A known problem with this statistic as applied to malicious datasets is the skew of data depending on the source, e.g. traffic reaching honeypots in different countries varies greatly.

Linkage. This statistic is exactly the same as in mixed datasets.

Furthermore, there is a key difference between the cited source and the *NECOMA* platform. Pinto and Maxwell correlate all datasets pairwise in similar way, however they use the term "overlap" and focus on getting unique tokens, assuming that availability of the same token in two datasets signifies the same event and is therefore redundant. In *NECOMA* the situation is quite different – most datasets were built independently and describe different phenomena. Therefore in our case high linkage (overlap) is actually desired and allows better, more productive correlation.

3.3.3 *NECOMA*-specific ratings

Cross-dataset linkage. The linkage ratings proposed above are limited through their global scope. For a fixed list of datasets it is possible to define a more practical measure of connection between the different datasets – a direct statistic determining the probability of a given token to appear in two different datasets. The probability can be directly measured a posteriori, by calculating the percentage of tokens from one dataset appearing in the other dataset.

Early results based on URL data from the n6 database and Japanese DNS records show, that the geographical separation of European and Japanese sources is very strong. E.g. very few IP addresses appear in both datasets. On the other hand, the cases where the same tokens appear in such geographically separated datasets are especially interesting, as they represent globally significant phenomena.

User rating. While the automatic ratings can be very valuable in terms of the amount of collected information, there is no universal method of automatic verification of the value of the provided data. Therefore it is crucial that collected data be rated by users of the platform. Manual input is most useful in case of the graph-based analysis, where ratings can be given for information provided by individual nodes in the graph, easily tied to the datasets providing the information.

3.4 Automated rating of external knowledge sources

Automatic collection of associated external knowledge for events identified by the *NECOMA* platform is a relatively simple but important task. The external knowledge collection mechanism developed in WP1 employs existing search engines to locate information relevant to a given token.

The main problem in this approach are the limits set by the operators of search engines. Most effective search engines tend to set limits on automated use, allowing only a certain number of requests per period (hour, day, etc). More queries are allowed in case of paid access to the API.

The *NECOMA* platform is expected to produce new tokens at a rate far exceeding the allowable query rates of most of the existing search engines. Using only free APIs is therefore only possible if external information is only gathered on demand, or newly identified tokens are prioritized and external information is collected only for the most interesting ones.

The alternative to this approach is to purchase paid access to one or more search engines. This approach guarantees the ability to collect external knowledge whenever it is available. However, since the algorithms used by different engines differ, the amount of results obtained for a given token may differ significantly. Therefore selecting the most effective engine is important from the economical point of view.

The natural solution to this problem is to use a running rating of various available sources of external information and use the best search engine as the primary source. The proposed solution is to launch the platform using only free services for external knowledge, limiting the number of queries to stay within the limits (obviously resulting in many ignored tokens). During this period, statistics on result count per token for each search engine would be collected. The search engine providing the most results for a typical token should be selected as the main source of external knowledge.

Since the algorithms used by search engines are constantly updated, the search engine selected using this approach will not be guaranteed to be the most efficient source of information forever.

Instead of focusing on only one search engine, the *NECOMA* platform will use a more dynamic approach. One search engine will be the main source of information – for each newly generated token, a query will be sent. However, the alternative search engines will not be abandoned. For each engine, random tokens will be selected at a rate corresponding to the limit set by the search provider. For each engine statistics of result count will be collected. Whenever an alternative search engine reaches an average result count higher than the selected engine (only for the same tokens) and that result remains stable for several weeks, a change of the selected engine will be suggested.

The proposed approach ensures optimal allocation of resources, as long as the decision about changing the search engine is not automatic, but involves both the rating results and the current pricing of all alternative solutions. Unfortunately access to search engines is not usually covered with a single fee – the payment covers a certain number of queries, usually not large in comparison to the predicted rate of token generation within *NECOMA*. Therefore the actual decision must take into account the pricing model of each engine as well as the result count. It is also possible to combine this with other measures, such as user rating of usefulness of the provided results. It is also possible to introduce separate selection per token type measuring the result count for each token type separately (as in case of the linkage metric for datasets).

4

Initial architecture and design principles of the threat analysis platform

4.1 Initial architecture design

At this stage of the *NECOMA* project we have a clearer view of all the components and modules that will be part of the system. In addition, we also identified initially interfaces that will serve the purpose of information exchange between the *NECOMA* system and external components, endpoints and users.

The current design of the system is an update and extension of the design proposed during the activities related to WP1. With the current achievements and advance on the projects research activities, we are able to define better particular components and outline the system as a whole.

Because of the current complexity of the system and the process flow, the description will follow a step by step explanation of the information flow across the system describing particular components on the way.

Figure 4.1 illustrates the current architecture design, and will be referred to during the process flow description, which begins with data gathering and ends on effective exploitation of the various analyses results. Particular components of the architecture are marked in bold text in the description.

Data and information gathering

The whole process begins with data capturing, at various points by **collection probes**, on different levels of the **infrastructure and endpoint devices**. The complete list of datasets in the *NECOMA* consortium possession, coming from endpoint and infrastructure levels, is available in deliverables D1.2 and D1.3. The descriptions provided in the deliverables include the data formats and sampling methods used in order to capture the data.

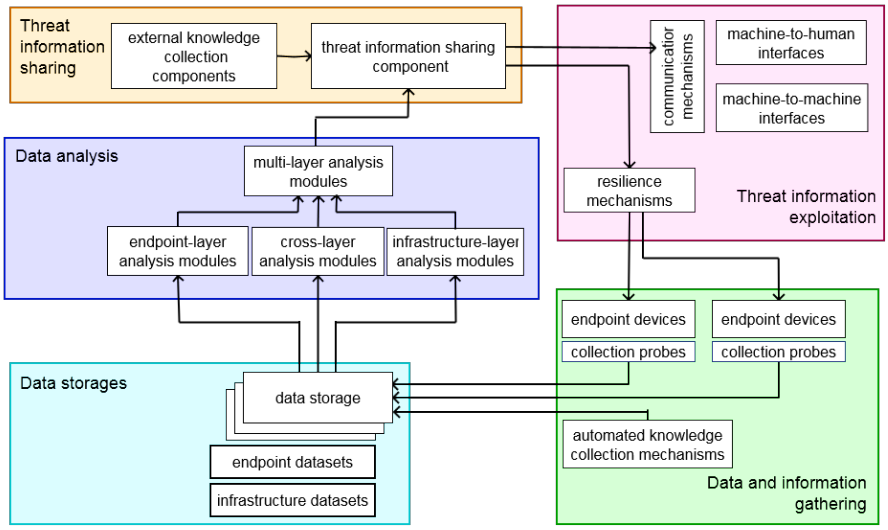


Figure 4.1: Architecture Design

Additionally, the standard probing techniques are enforced by **automatic knowledge gathering mechanisms** which are able to autonomously capture raw data coming from different sources. The raw data collected is stored along with other datasets and later can be shared. Initial designs of such mechanisms were proposed in deliverable D1.1. Currently, this kind of mechanisms are implemented as various types of web crawlers and other components such as Token Seeker.

Data storages

All the raw data collected by all the probes and automated gathering mechanisms is stored in the *NECOMA* systems **data storages** which hold both, **infrastructure and endpoint datasets**. Because of the complexity and volume of data belonging to different datasets, currently each datasets has its own storage. Despite that fact, the data is effectively shared between the analysis modules. One of the ongoing investigations is exploring ways of effective dataset sharing while maintaining proper security and privacy mechanisms, at the same time allowing external clients to make use of the datasets, also allowing them to add their own datasets to the system.

Although the datasets in general do not share a common data storage, some related analyses do share a common storage. Eg. this is the case with the tighlty coupled hadoop-based analyses. Details of this approach are described in the next section.

Data analysis

The analysis modules functioning in the *NECOMA* system can be divided into three main types: **infrastructure-layer analysis modules**, **endpoint-layer analysis modules** and **cross-layer analysis modules**. While the first two modules analyse datasets coming from the corresponding layers, the cross-layer analysis modules perform analysis on correlated data coming from both layers.

The second layer of analysis shown in the diagram, the **multi-layer analysis modules** processes the results of the first-layer analyses – either single-layer or cross-layer. This layer of modules is optional, as the goal of the *NECOMA* platform is to enable such multi-layer analysis to be performed by the user. Therefore in most cases the results are simply forwarded/kept. This is also the place where feedback between analyses can be implemented, eg. most significant tokens found by different analyses can be redirected to the graph-based analysis for further processing.

Detailed descriptions of all the analysis modules, that are included (or plan to be included) in the *NECOMA* system, are provided in this document in Chapter 2.

Threat information sharing

The **threat information sharing component** (TISC) is an abstract concept, where ways for an efficient and effective implementation are under investigation. The component will serve as an orchestrator for exploitation and sharing of the analysis results and obtained threat information. It will keep track of all the available sources of threat information, including **external sources collection component**. The TISC will serve as an intermediate component between the resilience mechanisms and actual threat information databases, retrieving the threat information from the relevant databases by itself and serving it to the clients on request or in real time.

Threat information exploitation

Currently there are two main exploitation means foreseen for the threat data obtained by the analysis modules. The first one is through **resilience mechanisms** that will try maintain acceptable levels of availability of the protected targets, despite disruptions, through reconfiguration e.g: change firewall filtering table (the complete definition of reconfiguration is available in deliverable D1.1 in the introduction chapter). The second way of exploiting the analysis results will be through, so called, **communication mechanisms**, that will be described in detail in deliverable D3.2. The **communication mechanisms** will be effective **machine-to-machine** and **machine-to-human** interfaces for threat data sharing with the aim of reaching a broad au-

dience consisting of users from different fields. **Machine-to-machine** interfaces will enable external users to 'plug in' their devices to the system to be able to, in an automated manner, reach information about threats that might affect the devices. **Machine-to-human** interfaces will provide means for displaying human-readable, easy to understand alerts based on the threats identified by the system.

4.2 An analysis platform based on Hadoop

Some of our analysis modules are designed to be implemented as hadoop applications since they require much computation resources. The high computational requirements enforce a deeper integration of the related modules. The basic design of our hadoop-based analysis platform, which is called MATATABI serving Data storage and Data analysis framework functionality in our initial architecture design (§ 4.1), is described in Deliverable 1.1. In this section, we describe the designed modules of threat analysis based on the MATATABI. This is an instance of Data storage and analysis component of our architecture which tries to tackle a couple of challenges of cross-layer and multi-layer threat analysis. Note that the integration does not limit the ability of the system to function in the basic roles assigned by the architecture – the datasets are still available to the other analyses.

4.2.1 Overview

MATATABI is a system based on Apache Hadoop software to analyze a huge amount of datasets and detect threats. It handles the variety of data sources and provides useful data access methods like SQL-like language with a reasonable processing performance. The detailed example how to implement an analysis module on MATATABI is also described in Deliverable 1.1.

Figure 4.2 illustrates the overview of MATATABI, while it provides Data storage and analysis modules framework for our initial architecture.

Our developed analysis modules are simple scripts or programs that can access data on Hadoop Distributed Filesystem (HDFS) and process it with a familiar program language for various post-processing methods such as a statistical calculation or machine learning, etc.

4.2.2 Analysis modules on MATATABI

In this section, we present the summary of implemented analysis modules on top of MATATABI.

ZeuS DGA detector

The first case is the detection of compromised hosts by the ZeuS botnet in an enterprise network by scanning DNS queries with a particular pattern of

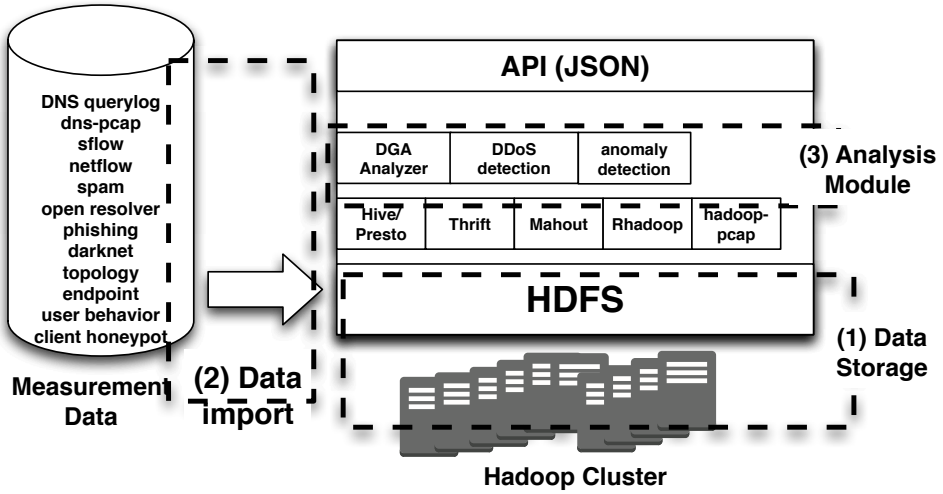


Figure 4.2: Overview of MATATABI. Based on the Hadoop platform, we integrated the data storage with import modules, analysis scripts, and an application programing interface in a single platform.

```
'[a-z0-9]{32,48}.(ru|com|biz|info|org|net)'
```

Figure 4.3: Regular expression of a DGAed domain name.

domain names as shown in figure 4.3. This module detects compromised hosts of ZeuS bot in a managed network, where a host queries suspicious domain names, based on the Domain Generation Algorithm (DGA), is considered a potential compromised host. In the case of proxied query via a DNS forwarder, we looked at traffic information filtered by the IP address of DNS answer records to identify the client IP address.

NTP amplifier detector

This module searches for Network Time Protocol (NTP) servers sending traffic with a particular packet size corresponding to a well-known NTP-amplification attack [75]. It reports the IP addresses of NTP amplifiers, and the IP address and Autonomous System (AS) number of targeted victims.

An additional module for the detector extracts NTP flows at the backbone sampling traffic (i.e., sFlow records) and lists the top ten NTP flows within a given time period.

Anomalous heavy-hitter detector

By using simple statistical tests, this modules detects IP addresses sending or receiving an abnormally high number of packets or bytes, for example, caused by DoS attacks.

Phishing likelihood calculator
This module is an implementation of a previously proposed system [60], which provides a binary detection whether a given URL points to a phishing site or not. The module consists of dataset preparation by crawling contents on pre-known phishing sites provided by PhishTank[1], analyzed by machine learning method with the help of Mahout. The dataset is updated every day since phishing sites changes frequently.

DNS amplification detector
The module tries to detect anomalous DNS traffic, causing amplification attack which fills the link capacity and makes denial of services. It looks at two different datasets, backbone sFlow traffic records and a list of open DNS resolver servers [79], and ranks top 10 speakers of DNS flow which communicates with open resolvers in sFlow datasets.

UDP fragmentation
As realizing an additional way for cache poisoning attack on DNS server based on IP fragmented packets [40], we started to observe how much traffic employed fragmented packets in the backbone traffic. The script simply extracts a record from sFlow dataset and implements a counter-based detection approach.

DNS anomaly detection
This module tries to detect anomalies of DNS response packets by adapting a machine learning method. Various statistical features such as IP addresses, the country code of DNS server, Malware Domain List[2], legitimate domain list, and the AS number of the DNS server are used for the analysis.

SSL scan detector
This module extracts SSL/TLS scans sFlow traffic data, which frequently happened right after the discovery of Heartbleed bug in OpenSSL library. The module simply counts packets destined to a specific port number, and containing the TCP SYN flag.

DNS failure graph analysis
This module tries to find suspicious non-existing domain names and IP addresses that might belong to botnets. The analysis is an implementation of an existing method, DNS failure graphs [43], based on a clustering technique.

4.2.3 Summary

Table 4.1 summarizes all the implemented analysis modules that we have come up with so far (almost for one year). Thanks to the pre-processed data by import module of each dataset and uniform programmability of MATATABI, multiple experiments have been conducted. Furthermore, the

[1] http://www.phishtank.com/
[2] http://www.malwaredomainlist.com/

script are small and easy to implement, with most of the ranging from from 20 to 160 Lines of Code (LoC).

Table 4.1: Analysis modules on MATATABI.

Name	datasets	frequency	LoC (#lines)	remark
ZeuS DGA detector	DNS pcap, netflow	daily	25	hadoop-pcap
UDP fragmentation detector	sflow	daily	48	
Phishing likelihood calculator [60]	Phishing URLs, Phishing content	1-shot	–	Mahout (Random-Forest)
NTP amplifier detector	netflow, sflow	daily	143	pyhive, Maxmind GeoIP
DNS amplifier detector	sflow, open resolver [79]	daily	37	
Anomalous heavy-hitter detector	netflow, sflow	daily	106	pyhive
DNS anomaly detection	DNS pcap, whois, malicious & legitimate domain lists	daily	57	hadoop-pcap, Mahout (Random-Forest)
SSL scan detector	sflow	1-shot	36	
DNS failure graph analysis [43]	DNS pcap	daily	159	pyhive

5 Conclusion

The wealth and diversity of datasets collected by the *NECOMA* project is an invaluable asset enabling researchers to detect and monitor different aspects of malicious activities. However, collecting large amounts of high quality data, while itself an important success, does not translate directly into knowledge of the malicous landscape or ability to react to new threats. Achieving these crucial goals depends on the ability to extract interesting information from the constant flows of raw data, while efficiently analyzing these datasets is an ambitious challenge due to the volume, variety and rate of change of the collected data. To tackle the challenge, this document reports several analysis techniques to independently process *NECOMA* datasets, as well as the cross-layer approaches that leverage the value of existing correlations between multiple datasets of different kinds. The overwhelming number of events extracted by the proposed analysis techniques is then rated and classified to prioritize further operations.

The second challenge addressed by this document is *automated rating and classification of malicious events*. This crucial task is particularly difficult as it is usually dataset-specific, and the pertinent metrics depend on the use case. Thereby, we proposed diverse rating metrics and techniques that are tailored to different applications. In particular, the ratings will not only be available to the external users of the *NECOMA* platform but also can be used internally to prioritize information originating from different sources.

Detailed discussions and implementations of the proposed analysis and rating techniques have shed light on several details of the *NECOMA* architecture design and its current hadoop subsystem implementation. We consequently updated the *NECOMA* initial architecture and achieved an important step towards our definitive architecture design. The outputs of this deliverable also clarified future directions for the development of the remaining modules of the *NECOMA* platform.

Bibliography

[1] Large-scale PCAP Data Analysis Using Apache Hadoop. https://labs.ripe.net/Members/wnagele/large-scale-pcap-data-analysis-using-apache-hadoop. (Accessed 30th January 2014).

[2] MAWI Traffic Archive. http://mawi.wide.ad.jp.

[3] D. Andriesse and H. Bos. An Analysis of the Zeus Peer-to-Peer Protocol. Technical report, IR-CS-74, May 2013.

[4] D. Andriesse, C. Rossow, B. Stone-Gross, D. Plohmann, and H. Bos. Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus. In *Malicious and Unwanted Software: "The Americas" (MALWARE), 2013 8th International Conference on*, pages 116–123, Oct 2013.

[5] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou II, and D. Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *USENIX Security Symposium*, page 16, 2011.

[6] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 491–506, Bellevue, WA, 2012. USENIX.

[7] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833 (Informational), Aug. 2004.

[8] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. IMW '02, pages 71–82.

[9] R. J. Barnett and B. Irwin. Towards a taxonomy of network scanning techniques. SAICSIT '08, pages 1–7.

[10] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *NDSS*, page 17, 2011.

[11] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. M. Youssef, M. Debbabi, and L. Wang. On the analysis of the zeus botnet crimeware toolkit. In *Proceedings of the 8th IEEE Annual Conference on Privacy, Security and Trust*, pages 31–38, Aug. 2010.

[12] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian. Anomaly extraction in backbone networks using association rules. IMC '09, pages 28–34, 2009.

[13] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian. Anomaly extraction in backbone networks using association rules. *IEEE/ACM Transactions on Networking*, 2012.

[14] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina. Impact of packet sampling on anomaly detection metrics. In *IMC'06*, pages 159–164, 2006.

[15] N. Brownlee, K. Claffy, and E. Nemeth. DNS Root/gTLD performance measurements. *USENIX LISA*, pages 241–256, 2001.

[16] P. Calais, D. Pires, D. Neto, W. Meira, C. Hoepers, and K. Steding-Jessen. A campaign-based characterization of spamming strategies. In *CEAS'08*, pages 1–6, 2008.

[17] Z. Cao, Z. Wang, and E. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM'00*, volume 1, pages 332–341, 2000.

[18] S. Castro, D. Wessels, M. Fomenkov, and K. Claffy. A Day at the Root of the Internet. *ACM SIGCOMM Computer Communication Review*, 38(5):41–46, 2008.

[19] CERT.pl. ZeuS - P2P+DGA variant - mapping out and understanding the threat. Available at: https://www.cert.pl/news/4711, 2012.

[20] CERT.pl. ZeuS-P2P monitoring and analysis. Technical report, CERT Polska, 2013.

[21] P. Chan, M. Mahoney, and M. Arshad. Learning rules and clusters for anomaly detection in network traffic. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats*, volume 5 of *Massive Computing*, pages 81–99. Springer US, 2005.

[22] J. Chen, R. Fontugne, A. Kato, and K. Fukuda. Clustering spam campaigns with fuzzy hashing. In *AINTEC'14*, page 8, Chiang Mai, Thailand, Nov 2014.

[23] Conficker Working Group. The conficker working group., June 2014.

[24] A. Dainotti, C. Squarcella, E. Aben, K. Claffy, and M. Chiesa. Analysis of country-wide internet outages caused by censorship. IMC'11, pages 1–18, 2011.

[25] DAMBALLA. Top-10 botnet outbreaks in 2009. Available at: https://blog.damballa.com/archives/569.

[26] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[27] G. Dewaele, K. Fukuda, P. Borgnat, P. Abry, and K. Cho. Extracting hidden anomalies using sketch and non gaussian multiresolution statistical detection procedures. LSAD '07, pages 145–152.

[28] C. J. Dietrich, C. Rossow, and N. Pohlmann. Cocospot: Clustering and recognizing botnet command and control channels using traffic analysis. *Computer Networks*, 57(2):475 – 486, 2013. Botnet Activity: Analysis, Detection and Shutdown.

[29] N. Falliere and E. Chien. Zeus: King of the Bots. Technical report, Symantec, 2009.

[30] G. Fernandes and P. Owezarski. Automated classification of network traffic anomalies. SecureComm '09, pages 91–100.

[31] I. Fette, N. M. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web*, May 2007.

[32] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda. MAWILab : Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *CoNEXT '10*, pages 8:1–8:12, Philadelphia, USA, 2010.

[33] R. Fontugne and K. Fukuda. A Hough-transform-based anomaly detector with an adaptive time interval. *ACM SIGAPP Applied Computing Review*, 2011.

[34] R. Fontugne, J. Mazel, and K. Fukuda. Hashdoop: A mapreduce framework for network anomaly detection. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pages 494–499, April 2014.

[35] K. Fujiwara, A. Sato, and K. Yoshida. DNS Traffic Analysis: Issues of IPv6 and CDN. In *SAINT 2012*, pages 129–137, 2012.

[36] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Zhao. Detecting and characterizing social spam campaigns. In *Proc. of the 10th ACM SIGCOMM conference on Internet measurement*, pages 35–47, 2010.

[37] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An empirical reexamination of global DNS behavior. In *SIGCOMM'13*, pages 267–278, 2013.

[38] Y. Han, Y. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of SIGMOD*, pages 1–12, 2000.

[39] S. Hao, N. Feamster, and R. Pandrangi. Monitoring the initial DNS behavior of malicious domains. In *IMC'11*, pages 269–278, 2011.

[40] A. Herzberg and H. Shulman. Fragmentation considered poisonous, or: One-domain-to-rule-them-all.org. In *Communications and Network Security (CNS), 2013 IEEE Conference on*, pages 224–232, Oct 2013.

[41] K. Ishibashi and K. Sato. Classifying DNS heavy user traffic by using hierarchical aggregate entropy. In *WTC 2012*, pages 1–6, 2012.

[42] T. Jebara. Multi-task feature and kernel selection for svms. In *Proc. of Inter. Conf. on Machine Learning*, pages 55–63, 2004.

[43] N. Jiang, J. Cao, Y. Jin, L. Li, and Z.-L. Zhang. Identifying suspicious activities through dns failure graph analysis. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 144–153, Oct 2010.

[44] N. Jiang, J. Cao, Y. Jin, L. E. Li, and Z.-L. Zhang. Identifying suspicious activities through DNS failure graph analysis. In *ICNP 2010*, pages 144–153, 2010.

[45] A. Kalafut, M. Gupta, C. Cole, L. Chen, and N. Myers. An empirical study of orphan DNS servers in the internet. In *IMC'10*, pages 308–314, 2010.

[46] Y. Kanda, R. Fontugne, K. Fukuda, and T. Sugawara. ADMIRE: Anomaly detection method using entropy-based pca with three-step sketches. *Computer Communications*, 2013.

[47] Y. Kazato, K. Fukuda, and T. Sugawara. Towards classification of dns erroneous queries. In *AINTEC'13*, pages 25–32, 2013.

[48] J. Klensin. Internationalized Domain Names in Applications (IDNA): Protocol. RFC 5891 (Proposed Standard), Aug. 2010.

[49] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: methods, evaluation, and applications. IMC '03, pages 234–247.

[50] M. Kruczkowski and E. Niewiadomska-Szynkiewicz. Support vector machine for malware analysis and classification. In *Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence*, pages 415 – 420, 2014.

[51] M. Kührer, T. Hupperich, C. Rossow, and T. Holz. Exit from hell? reducing the impact of amplification ddos attacks. In *USENIX Security Symposium*, 2014.

[52] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. *ACM SIGCOMM Computer Communication Review*, 2004.

[53] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Computer Communication Review*, 2005.

[54] Y. Lee and Y. Lee. Toward scalable internet traffic measurement and analysis with hadoop. *SIGCOMM Comput. Commun. Rev.*, 43(1):5–13, Jan. 2012.

[55] J. Liu and K. Fukuda. Towards a taxonomy of darknet traffic. In *5th International Workshop on TRaffic Analysis and Characterization (TRAC'14)*, pages 37–43, Aug 2014.

[56] J. Mazel, R. Fontugne, and K. Fukuda. Taxonomy of anomalies in backbone network traffic. In *Proceedings of TRAC 2014*, 2014.

[57] J. Mazel, R. Fontugne, and K. Fukuda. Visual comparison of network anomaly detectors with chord diagrams. In *Proceedings of the 29th Symposium On Applied Computing (SAC '14)*, March 2014.

[58] J. Mirkovic and P. Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 2004.

[59] MITRE Corporation. Common attack pattern enumeration and classification. http://capec.mitre.org/.

[60] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi. An evaluation of machine learning-based methods for detection of phishing sites. In *Advances in Neuro-Information Processing*, volume 5506 of *Lecture Notes in Computer Science*, pages 539–546. Springer Berlin Heidelberg, 2009.

[61] D. Miyamoto, H. Hazeyama, Y. Kadobayashi, and T. Takahashi. Behind HumanBoost: Analysis of Users' Trust Decision Patterns for Identifying Fraudulent Websites. *Journal of Intelligent Learning Systems and Applications*, 4(4):319–329, 2012.

[62] D. Moore, C. Shannon, G. Voelker, and S. Savage. Network telescopes. CAIDA Technical Report, 2004.

[63] Netcraft. Phishing Alerts for SSL Certificate Authorities. Available at: http://news.netcraft.com/archives/2012/08/22/phishing-on-sites-using-ssl-certificates.html.

[64] P. Vixie. AS112 project. http://www.as112.net/.

[65] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. IMC'04, pages 27–40, 2004.

[66] A. Pathak, F. Qian, Y. C. Hu, Z. M. Mao, and S. Ranjan. Botnet spam campaigns can be long lasting: evidence, implications, and analysis. In *SIGMETRICS'09*, pages 13–24, Seattle, WA, Jun 2009.

[67] T. A. Phelps and R. Wilensky. Robust Hyperlinks: Cheap, Everywhere, Now. In *Proceedings of the 8th International Conference on Digital Documents and Electronic Publishing, the 5th International Workshop on the Principles of Digital Document Processing*, Sep. 2000.

[68] A. Pinto and K. Maxwell. Measuring the iq of your threat intelligence feeds. In *DEF CON 22*, 2014.

[69] D. Plonka and P. Barford. Network anomaly confirmation, diagnosis and remediation. CCC '09, pages 128–135.

[70] D. Plonka and P. Barford. Context-aware clustering of DNS query traffic. In *IMC'08*, pages 217–230, 2008.

[71] P. Porras. Inside risks: Reflections on conficker. *Commun. ACM*, 52(10):23–24, Oct. 2009.

[72] P. Porras, H. Sadi, and V. Yegneswaran. A foray into conficker's logic and rendezvous points. In *In USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.

[73] S. Pukkawanna, H. Hazeyama, Y. Kadobayashi, and S. Yamaguchi. Detecting anomalies in massive traffic with sketches. In *Proceedings of The Ninth International Conference on Future Internet Technologies*, CFI '14, pages 14:1–14:2, New York, NY, USA, 2014. ACM.

[74] L. Quan, J. Heidemann, and Y. Pradkin. Trinocular: Understanding internet reliability through adaptive probing. *ACM SIGCOMM Computer Communication Review*, 2013.

[75] C. Rossow. Amplification hell: Revisiting network protocols for ddos abuse. In *NDSS Symposium 2014*, pages 224–232, Feb. 2014.

[76] F. Silveira and C. Diot. URCA: Pulling out anomalies by their root causes. INFOCOM '10, pages 1–9.

[77] F. Silveira, C. Diot, N. Taft, and R. Govindan. Astute: Detecting a different class of traffic anomalies. In *SIGCOMM'10*, pages 267–278, 2010.

[78] A. Soule, K. Salamatian, and N. Taft. Combining filtering and statistical methods for anomaly detection. IMC '05, pages 31–31.

[79] Y. Takano, R. Ando, T. Takahashi, T. Inoue, and S. Uda. A Measurement Study of Open Resolvers and DNS Server Version. In *Internet Conference 2013*. IEICE, 2013.

[80] B. Tellenbach, M. Burkhart, D. Schatzmann, D. Gugelmann, and D. Sornette. Accurate network anomaly classification with generalized entropy metrics. *Computer Networks*, 2011.

[81] The CA/Browser Forum. Guidelines For The Issuance And Management Of Extended Validation Certificates. Available at: https://cabforum.org/wp-content/uploads/EV-Code-Signing-v.1.2.pdf.

[82] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, USA, 1995.

[83] D. Watson and R. Friend. Measurement of social-evaluative anxiety. *Consulting and Clinical Psychology*, 33:448–457, 1969.

[84] D. Wessels and M. Fomenkov. Wow, that's a lot of packets. In *PAM'03*, page 9, 2003.

[85] T. White. *Hadoop: the definitive guide*. O'Reilly, 2012.

[86] J. Wolf. Technical details of srizbis domain generation algorithm, Nov. 2008.

[87] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston. Internet background radiation revisited. IMC'10, pages 62–74, 2010.

[88] G. Xiang, J. Hong, C. Rose, and L. Cranor. CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites. *ACM Transactions on Information and System Security*, 14:21–28, 2011.

[89] K. Xu, Z.-L. Zhang, and S. Bhattacharyya. Internet traffic behavior profiling for network security monitoring. *IEEE/ACM Transactions on Networking*, 2008.

[90] B. Zdrnja, N. Brownlee, and D. Wessels. Passive Monitoring of DNS Anomalies. In *DIMVA'07*, pages 129–139, 2007.

[91] Y. Zhang, J. Hong, and L. Cranor. CANTINA: A Content-Based Approach to Detect Phishing Web Sites. In *Proceedings of the 16th World Wide Web Conference*, May 2007.