

SEVENTH FRAMEWORK PROGRAMME

Information & Communication Technologies
ICT

Cooperation Programme



Nippon-European Cyberdefense-Oriented Multilayer threat Analysis
†

Deliverable D2.2: Threat Analysis Platform

Contractual Date of Delivery	November 30th 2015
Actual Date of Delivery	November 30th 2015
Deliverable Dissemination Level	Public
Editors	Paweł Pawliński, Romain Fontugne
Contributors	All <i>NECOMA</i> 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	7
2	Overview of the threat analysis platform	9
2.1	Final architecture of the analysis platform	9
2.1.1	Endpoint and infrastructure devices	9
2.1.2	Analysis modules and threat information sharing	12
2.1.3	Communication mechanisms and resilience mechanisms	13
2.2	Implementation of the analysis platform	14
2.2.1	Endpoint and infrastructure devices	14
2.2.2	Analysis modules and threat information sharing	15
2.2.3	External interfaces and resilience mechanisms	15
3	Analysis modules	17
3.1	Backbone and telescope traffic analysis	17
3.1.1	NTP amplifier detector	17
3.1.2	Anomalous heavy-hitter detector	19
3.1.3	DNS top speaker analysis	19
3.1.4	SSL scan detector	20
3.1.5	UDP Fragment analyzer	21
3.1.6	Synchronized Sources Detector	22
3.2	Large-scale DNS traffic analysis	24
3.2.1	Botnet C&C domain name detector	24
3.2.2	DNS failure graph analysis	25
3.2.3	DNS cache poisoning detection module	25
3.3	End-point threat data analysis	26
3.3.1	C&C detection in sandbox data	26
3.3.2	SSL server security assessment	29
3.3.3	Malicious URL signature generation	33

3.3.4	High Performance Phishing Detection	36
3.4	Cross-layer threat data analysis	37
3.4.1	Zeus DGA Detector	37
3.4.2	FP-SVM	40
3.4.3	Correlation Graph Analysis	44
4	Dataset rating	51
4.1	Rating components	51
4.1.1	Rate	52
4.1.2	Delay	52
4.1.3	False discovery rate	52
4.1.4	Cross-dataset linkage	53
4.1.5	Representativeness	53
4.1.6	User/utility rating	54
4.2	Evaluation results	54
4.2.1	Rate	56
4.2.2	Delay	56
4.2.3	False discovery rate	56
4.2.4	Linkage	56
4.2.5	Representativeness	57
4.2.6	Utility rating	57
5	Threat metrics	59
5.1	Attack intensity	59
5.1.1	Darknet port scan count	59
5.1.2	Darknet backscatter intensity	60
5.1.3	Spam campaign count	60
5.1.4	Global spam intensity	60
5.1.5	Spam campaign intensity	61
5.1.6	DDoS attack intensity	61
5.2	Report confidence	61
5.2.1	Confidence of alerts generated by multiple backbone anomaly detectors	61
5.2.2	Phishing Likelihood Calculator	62
5.3	Level of sophistication	62
5.3.1	Attack mixture	62
5.4	Impact	63
5.4.1	Addresses involved in an anomaly	63
5.4.2	Addresses involved in scan	64
5.4.3	Addresses involved in spam campaign	64
5.4.4	Distribution of observed suspicious domain queries	64
5.5	Persistence	64
5.5.1	Spam campaign duration	64
5.5.2	Lifetime of observed suspicious domain queries	65

6	Supplementary research	67
6.1	DNS-based Detection of Malicious Cloud Outbound Traffic . .	67
6.1.1	Previous studies	68
6.1.2	Proposal Design	69
6.1.3	Experiments	72
6.1.4	Experience gained	77
6.2	Text Mining Approach for Estimating the Vulnerability Score .	78
6.2.1	Previous study	79
6.2.2	Estimating CVSS base-metrics from CVE documents . .	79
6.2.3	Annual weight assignment	81
6.3	Correlation between Phishing Identification and Eye Movement	83
6.3.1	Previous studies	84
6.3.2	Correlation between phishing identification and eye movement	87
6.3.3	Extraction of implicit intention	88
6.3.4	Estimation of participant's likelihood to be victim . . .	89
6.4	Anti-Phishing Visual Analysis to Mitigate Users' Excessive Trust in SSL/TLS	90
6.4.1	Previous studies	91
6.4.2	HTTPS Phishing Webpages: Building upon a False Sense of Security	92
6.4.3	Detection of Hacked Domains Hosting Phishing Web- pages	95
7	Summary	101

The following deliverable documents the threat analysis platform developed as the main result of workpackage 2 of the NECOMA project. Additionally, the report includes research results of the workpackage which were not reported in Deliverable D2.1. It is the main outcome of task T2.3, although some of the presented work is an outcome of continued research on topics covered by previous tasks in the workpackage.

Chapter 2 describes the final architecture of the NECOMA threat analysis platform, detailing the communication and data sharing solutions used by the consortium.

Chapter 3 documents the analysis modules constituting the platform. The modules directly relate to the descriptions of analysis methods included in Deliverable D2.1, although some changes appeared as the effect of practical experiences with the methods. The descriptions are mostly technical, focusing on the data consumed as input and provided as output, the implementation details of each module and the experience gained during exploitation of the method. Whenever significant changes were made to the method since its description in D2.1, they are described in this chapter as well, unless the modifications are large enough to consider the end result a different method, in which case the details are provided in chapter 6. Note that the correspondence of analysis methods in D2.1 and analysis modules in this chapter is not 1-to-1 – the consortium members were free to combine different methods in one module, drop underperforming analyses or even propose new ones.

Chapter 4 extends the rating and classification mechanisms developed in task T2.2 and presented in D2.1. The rating components that proved most useful in practice are described.

Chapter 5 focuses on the topic of threat metrics. The metrics proposed in this chapter are used to measure different aspects of the current threat

level and the severity of particular attacks. These metrics can also be used to judge the effectiveness of the remediation activities.

Chapter 6 provides the results of research efforts started in the previous tasks, which were not yet ready to be described at the time of delivery of D2.1 or was continued afterwards. These results are not minor enough to be considered mere tuning of the previously described research efforts and warrant a more detailed description.

Finally, the deliverable concludes with a short summary provided in chapter 7.

Overview of the threat analysis platform

2.1 Final architecture of the analysis platform

Since the initial design of the system presented in Deliverable D2.1: *Threat Analysis*, the architecture was finalized to accommodate requirements of the whole consortium. Additionally, the final design of the system evolved in order to address new challenges that were encountered in the course of the project.

The main focus in the new version of the architecture was to facilitate data sharing and collective analysis, enabling more effective multi-layer data correlation as well as better threat information sharing. Users and automated systems outside of the *NECOMA* platform can now interact directly with analysis modules and receive results in a quick manner, at the same time enriching the *NECOMA*'s knowledge base.

Figure 2.1 shows the final architecture design. The structure of the system might be divided into three main areas:

- **Endpoint and infrastructure devices**
- **Analysis modules**
- **Communication mechanisms interfaces and resilience mechanisms**

Each of the items will be explained in the following chapters.

2.1.1 Endpoint and infrastructure devices

The Endpoint and Infrastructure devices are both at the beginning and the end of the processing pipeline in our proposed system. They comprise both sources of raw data, later to be processed, and the assets to be protected or reconfigured when an attack is discovered. In *NECOMA*, we distinguish two main data layers which are then divided into five source categories each.

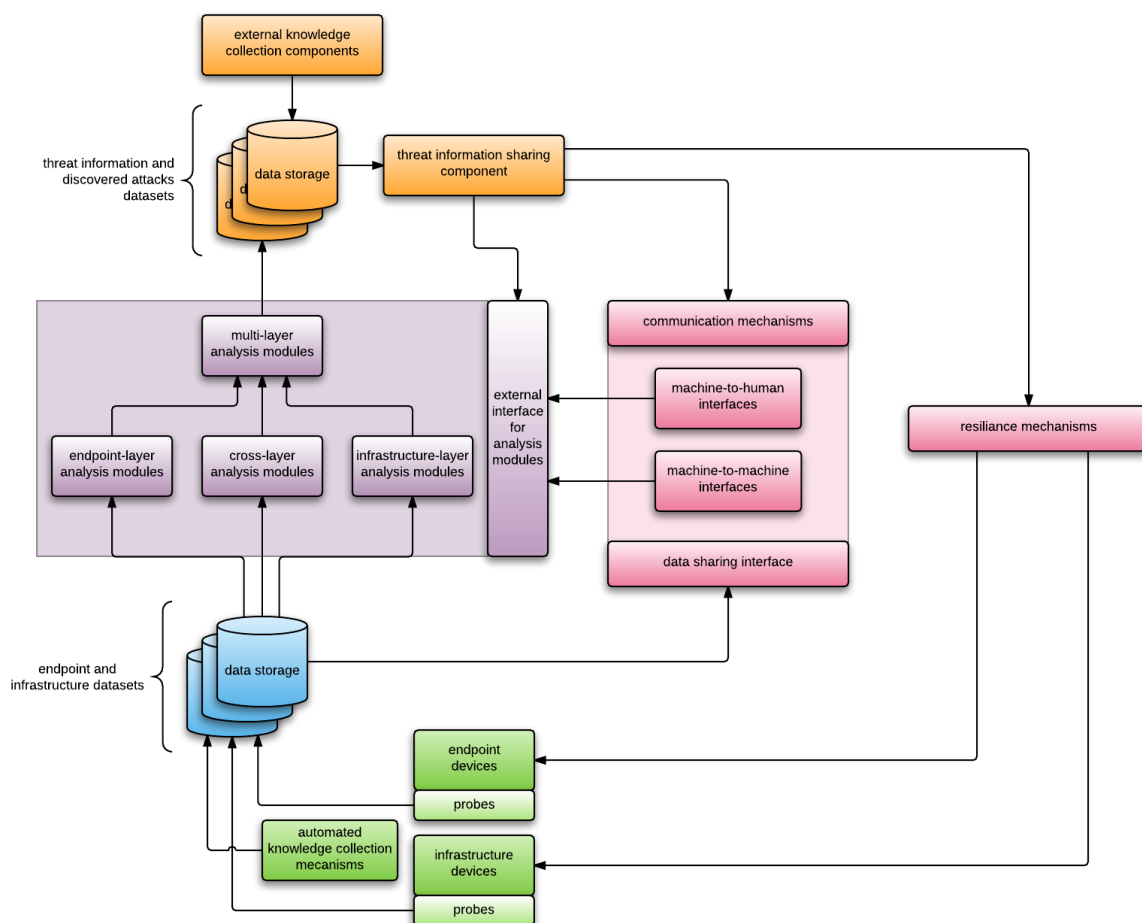


Figure 2.1: The final architecture of *NECOMA*

Table 2.1 lists all the data sources¹ and types which contribute to building the *NECOMA*'s data storage.

Members of the consortium have contributed a total of 34 datasets, out of which 29 are still being constantly expanded through active collection. The consortium is actively working on means to efficiently share the data sets with external actors. As the result of those actions 20 datasets have been made available. Table 2.2 summarises the statistics of datasets in control of the consortium.

Additionally the system's data collection capabilities do encompass Automated Knowledge Collection Mechanisms, which design was described in Deliverable D1.1: *Multilayer threat data collection system design document*. The experiments and initial, working prototypes proved that the datasets

¹ For details about the data sources and data sets please refer to Deliverables D1.2: *Infrastructure-layer threat data sets* and D1.3: *Endpoint-layer threat data sets*.

2.1. FINAL ARCHITECTURE OF THE ANALYSIS PLATFORM

Endpoint Layer
Mail and Messaging Datasets
Web Datasets
User Behaviour Datasets
Sinkhole Datasets
Client Honeypots and Sandbox Datasets
Infrastructure Layer
Traffic datasets
DNS Datasets
Topology Datasets
Telescope Datasets
Early Warning Datasets

Table 2.1: Data sources.

	Endpoint Layer	Infrastructure Layer
Total number of datasets	25	9
Ongoing capture	25	5
Estimated size	23,7 TB	200 GB
Shared (non-private)	15	5

Table 2.2: Data sets statistics.

can be easily enriched by automated mechanisms such as web crawlers, collecting the contents of suspicious web pages, and also the utilisation of search engines for seeking information related to computer security published openly on the Internet².

Research activities in *NECOMA* did also encompass creating a unified, common data storage, that would be a single interface for storing and accessing all the captured, raw information. Although such a design was proposed, the complexity and requirements (including law and regulations apart from technical requirements) made the idea to reach far beyond *NECOMA*'s scope³.

² For details about the designs, please refer to D1.1: *Multilayer threat data collection* system design document, section 4.3.

³ For details about the design please refer to Deliverable D3.2: *Security Information Exchange – Design*, Chapter 4.

2.1.2 Analysis modules and threat information sharing

To take advantage of the huge amounts of captured data, *NECOMA* devoted significant efforts to design and implement advanced analysis modules that are able to produce actionable information on contemporary threats. This information is distributed through External Interfaces and utilized by Resilience Mechanisms, which will be described in the following section.

The analysis modules can be divided into three main categories:

- **Infrastructure-layer analysis modules**
- **Endpoint-layer analysis modules**
- **Cross-layer analysis modules**

The architecture diagram depicts yet another type of module: multi-layer analysis module. This term was created as an abstract node for structuring and correlating outcomes coming from various analysis modules that are not intended to interact. In most cases the analysis results are simply forwarded, but in the context of threats affecting multiple layers (e.g. combining phishing with a simultaneous DDoS against the original site), additional correlation may be performed and results redirected for further processing⁴.

A total of sixteen analysis modules have been implemented within the scope of *NECOMA*. Detailed descriptions of the modules are provided in Chapter 3, while Deliverable D2.1: *Threat Analysis* contains in-depth description of the underlying techniques.

The analysis modules work on the datasets collected by the consortium, although several modules expose input interfaces that enable direct interaction with external actors. For example, external users can submit suspicious URLs to the phishing detection modules in order to assess the credibility of websites. Such scenarios will be covered in the following section.

Another significant component is the *threat data storage*, accessed through the threat information sharing component. Conceptually, the threat data storage component is a single dataset containing output generated by all analysis modules. It holds information about malicious activities, malware and, in general terms, any kind of valuable information that can be extracted from the analysis results and reused by resilience mechanisms or external actors. It consists of multiple databases under the control of *NECOMA*'s consortium members that expose a common interface. Additionally, in order to further enrich produced information, *NECOMA* takes advantage of external threat data sources, such as Phishtank.

To enhance multi-layer analysis, the analysis modules are capable of accessing the threat data storage and taking advantage of the collected threat

⁴ For detailed explanation please refer to Deliverable D1.1: *Security Information Exchange - Design*, Introduction Chapter.

knowledge. This allows for a much broader view of the threat landscape and more effective correlation of analysis results coming from different modules and also external interfaces and sources. This loop further enriches the threat knowledge and may lead to discovering much more sophisticated multi-layer attacks.

In order to facilitate the communication between various interfaces within the system and to enable easy access for external actors, the analysis modules as well as the data storages implement the n6 API⁵. By using the n6 API, *NECOMA* implements a unified way for inter-component communication integrating the numerous components tightly at the same time allowing flexibility in extending the system with new data sets and analysis modules. Furthermore, it provides an easy and documented way for interacting with the analysis modules and datasets.

2.1.3 Communication mechanisms and resilience mechanisms

The last stage in the *NECOMA* processing pipeline are the modules that utilize the threat knowledge produced during analysis. They can be divided into two major groups: *resilience mechanisms* and *communication mechanisms*.

Resilience mechanisms consist of all elements of the protected network, system, or application that can trigger reconfiguration of a device in response to an attack. Two main categories of resilience mechanisms can be distinguished: mechanisms for the endpoint layer and for the infrastructure layer⁶. Both types of defences directly utilize the available threat knowledge and are capable of reconfiguring devices settings in order to mitigate an attack (reactive) or prevent a threat (preventive). *NECOMA* focuses mostly on the reactive measures, although the need for secure by design mechanisms is strongly highlighted and expected as a follow-up result of the project.

Communication mechanisms serve the purpose of information exchange between external actors and the system and, enrichment of the system's knowledge. They facilitate dissemination of information collected in the *NECOMA* platform with the goal of utilizing it outside of the system. Communication mechanisms are also used to provide access to functionality offered by analysis modules to external users.

⁵ For design please refer to Deliverable D1.1: *Security Information Exchange - Design*, chapter 3.2. The implementation tutorial can be found in Deliverable D3.2: *Security Information Exchange - Design*, section 3.

⁶ An extensive design of those mechanisms is available in Deliverable D3.4: *Countermeasure Application - Design*.

2.2 Implementation of the analysis platform

While the previous section introduced the overall design of the *NECOMA* platform, this section describes the implemented prototype – *MATATABI* – which connects all elements of the architecture to provide a complete security information processing pipeline.

MATATABI is built upon the Apache Hadoop framework in order to fulfill some requirements: 1) scalability, 2) real-time analysis, and 3) uniform programmability [43]. The implementation covers the functionalities established by the *NECOMA*'s architecture, including interfaces to external entities such as human analysts or automated systems using results of processing modules. Figure 2.2 provides a high-level overview of the system, each component will be described in the following sections.

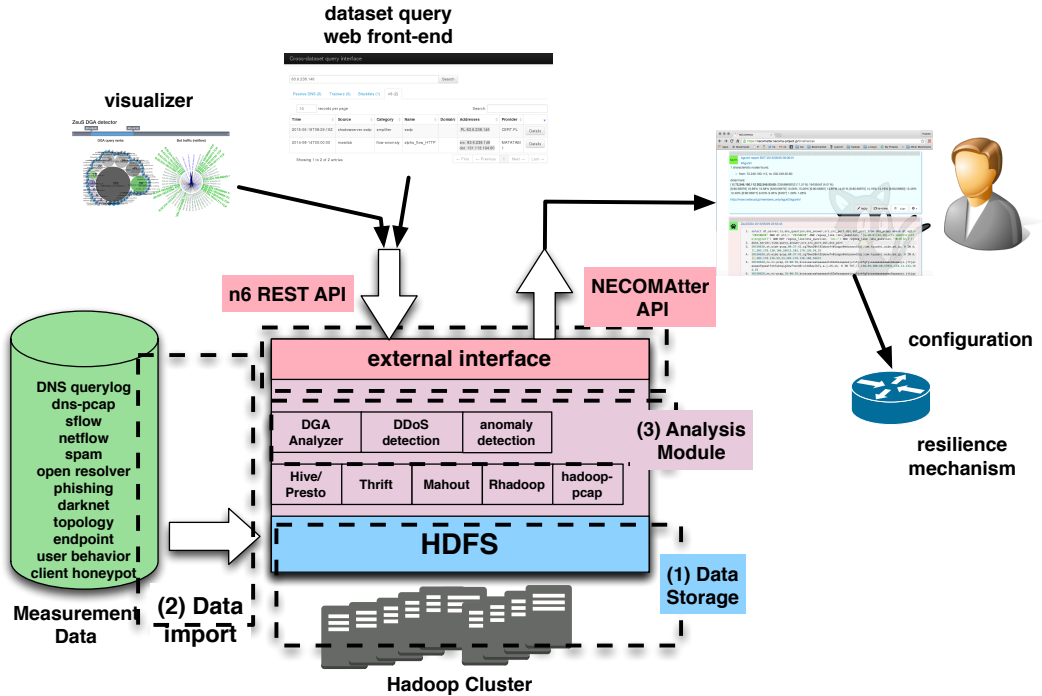


Figure 2.2: Core elements of the *NECOMA* architecture implemented by *MATATABI*: *data probe*, storage, analysis modules, external interfaces. Coloring of elements corresponds to figure 2.1.

2.2.1 Endpoint and infrastructure devices

The data import component of *MATATABI* collects the data at various devices (routers, DNS servers, and crawler) as a *data probe*. Those probes are located at selected measurement points in the infrastructure and store

all collected data in a distributed filesystem, which is a part of a Hadoop instance (*data storage*).

2.2.2 Analysis modules and threat information sharing

Once the data is collected, analysis modules try to look for security threats. MATATABI uses a simple programming model with a powerful computation backend to sift through the huge amount of data of different kinds, which allows to easily implement cross- and multi- layer analysis (*analysis module*).

2.2.3 External interfaces and resilience mechanisms

Results of analyses are accessible through an application programming interface (API) implemented using the n6 SDK (common *machine-to-machine interface*) and through the NECOMatter API together with its associated web front-end (*machine-to-human interface*).

NECOMatter also has the ability to control external entities such as resilience mechanism in the *NECOMA* architecture. DDoS mitigation is one of the use cases: results of analyses are reported as *tweets* through NECOMatter and an application acting as a resilience mechanism executes a command based on the tweeted information, which eventually reconfigures access control lists of Open vSwitches. Since fully automated operation might be too risky in a production environment, reconfiguration can be done by human operators with the help of a machine-to-human interface provided by NECOMatter.

This chapter presents the implemented analysis modules and the experience gained using these modules. The role of each module is concisely described (see Deliverable D2.1 for more details) along with implementation details and the module outcomes. The use of some of these modules have revealed unexpected results, or practical difficulties, that are of prime importance for real deployments. These relevant feedbacks are mentioned along with the modules description.

Modules are listed following the same classification of Deliverable D2.1. Namely, we distinguish four module categories based on the type of data analyzed by the modules:

- Modules designed for traffic captured at backbone networks or Internet telescope (Section 3.1).
- Modules inspecting DNS traffic (Section 3.2).
- Modules analyzing data collected at the edge of the network (Section 3.3).
- Modules with inputs spanning across several types of dataset (Section 3.4).

3.1 Backbone and telescope traffic analysis

3.1.1 NTP amplifier detector

The NTP protocol is commonly used by attackers to initiate important amplification DDoS attacks. The cause of the amplification is due to an NTP command, *monlist*, that sends the list of previously contacted hosts. In older versions of NTP, this command was activated by default, hence, NTP servers

with out-of-date software may potentially contribute to NTP amplification attacks.

This module seeks NTP servers that can potentially contribute to DDoS attacks by looking at servers sending traffic with a particular packet size corresponding to the monlist response packet size.

3.1.1.1 Data flow

This module analyzes IP traffic and reports IP addresses and AS numbers of NTP amplifiers and victims of NTP amplification attacks.

3.1.1.1.1 Input The current implementation takes NetFlow and sFlow data as input. Therefore it is able to analyze all sFlow and NetFlow datasets stored on the MATATABI platform on a daily basis.

3.1.1.1.2 Output The output of this module consists of two tables stored on MATATABI. The first table includes the IP addresses of all NTP amplifiers found in the traffic. This information is crucial for contacting operators to update NTP servers that are out-of-date, and permits to estimate the number of NTP amplifiers available for attackers. The second table is the list of the IP addresses and AS numbers that receive large amount of NTP traffic from the identified amplifiers. This list contains both victims of amplification attacks and scanners looking for NTP amplifiers.

3.1.1.2 Module implementation

The module is implemented in Python, querying traffic data from the Presto database using the pyhive library. The results are stored in the Hive database: for each NTP amplifier we store its IP address, AS number (obtained from the GeoIP API), and the daily number of NTP packets and bytes sent by the amplifier.

3.1.1.3 Experience gained

In practice, this module is able to identify amplifiers in very large datasets in the form of flow reports (*e.g.* NetFlow), hence, it is easily deployable at edge and backbone networks. Using NetFlow format, we had no difficulties to analyze significant datasets captured from core routers or Internet exchange points.

3.1.2 Anomalous heavy-hitter detector

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

The implemented approach tracks the number of packets and bytes sent or received by each IP address and reports IP addresses that experience sudden increases for either quantity. More specifically, the module counts the number of packets and bytes sent/received by each host, and flags all IP addresses whose counters are higher than its average number of bytes/packets plus 3 standard deviations.

3.1.2.1 Data flow

This module computes statistics from IP traffic and reports a list of anomalous IP addresses.

3.1.2.1.1 Input The current implementation takes NetFlow and sFlow data as input. Therefore it is able to analyze all sFlow and NetFlow datasets stored on the MATATABI platform on a daily basis.

3.1.2.1.2 Output The module reports a list of IP addresses that are sending an abnormally high number of packets or bytes. These IP addresses are stored in the Hive database and are available on the MATATABI platform.

3.1.2.2 Module implementation

The current implementation is done in Python reading NetFlow data from the MATATABI platform.

3.1.3 DNS top speaker analysis

The goal of this module is to find hosts which send massive DNS queries. A host which sends large-scale queries is a suspected case of a DNS amplification attack. The module finds such suspicious hosts by analyzing sFlow data sets.

3.1.3.1 Data flow

The module analyzes sampled traffic data such as sFlow and reports suspicious IP addresses with the number of packets. In the analysis, the module counts packets that have a UDP destination port set to 53.

3.1.3.1.1 Input The module analyzes sFlow data stored on MATATABI in the Hive database.

3.1.3.1.2 Output The module outputs the top 10 IP addresses and observed packet counts in one day. The results are posted on NECOMatter, the threat information sharing platform developed in this project. The IP addresses and packet counts are reported in CSV format, so the NECOMatter users can observe the daily results on the timeline of the platform.

3.1.3.2 Module implementation

The module is implemented as a ruby script. The script executes Hive queries on MATATABI to retrieve DNS traffic and stores the results on the same platform.

3.1.3.3 Experience gained

3.1.3.3.1 Scalability We are running the module on our analysis platform (MATATABI) and analyze daily traffic data. The module uses SQL queries for DNS packet counting, hence, it can take advantage of Hive and the MapReduce framework to scale out to very large datasets.

3.1.4 SSL scan detector

This module uncovers SSL/TLS scans from sFlow traffic data. This type of scan has been predominant right after the discovery of the Heartbleed security hole in the OpenSSL library. After the hole has been reported, the number of scans against SSL/TLS servers increased. This module finds the suspicious hosts that send SSL/TLS scans by analyzing traffic data.

3.1.4.1 Data flow

The module simply counts packets destined to a specific port number and containing the TCP SYN flag in traffic data.

3.1.4.1.1 Input The module analyzes sFlow data stored on MATATABI in the Hive database.

3.1.4.1.2 Output The module reports the daily number of packets with the TCP SYN flag set and the destination port number set as 443 (HTTPS). The results are posted on NECOMatter in order to share this information with all the NECOMA consortium members. The IP addresses and packet counts are given in text format, therefore, the NECOMatter users can investigate the daily results on the timeline of the platform.

3.1.4.2 Module implementation

The module is implemented as a ruby script, it retrieves traffic data from MATATABI and stores results in a Hive table on the same platform.

3.1.4.3 Experience gained

3.1.4.3.1 Scalability We are running the module on our analysis platform (MATATABI) and analyze daily traffic data. The module uses a simple SQL for SSL/TLS packet counting and can take advantage of the MapReduce framework for scalability issues.

3.1.5 UDP Fragment analyzer

This module reports the hourly ratio of fragmented UDP packets in sFlow datasets. DNS cache poisoning attacks use spoofed fragmented UDP packets to inject malicious records in DNS cache servers. Consequently, on monitored networks, the attack is seen as an increase of the number of fragmented UDP packets.

3.1.5.1 Data flow

The module analyzes sampled traffic data and reports suspicious IP addresses along with the corresponding number of fragmented packets.

3.1.5.1.1 Input The module analyzes sFlow data stored on MATATABI in the Hive database.

3.1.5.1.2 Output The module outputs the daily ratio of UDP fragmented flows and the absolute number of fragmented flows. The results are posted in CSV format on NECOMatter so these results can be shared and compared with the results of other modules. Thereby, NECOMatter users can monitor the hourly results on the timeline of the platform.

3.1.5.2 Module implementation

The module is implemented as a ruby script. It executes Hive queries on MATATABI to access sFlow data.

3.1.5.3 Experience gained

3.1.5.3.1 Scalability The module uses SQL requests for fragmented packet counting. Therefore, even if the size of collected traffic data is large, the proposed approach is easily computed in parallel using the MapReduce framework.

3.1.6 Synchronized Sources Detector

When a victim of a DDoS attack tries to defend himself, understanding the nature of the attack is one of the first steps to perform. Identification of IP addresses sending harmful traffic is useful especially for attacks requiring the use of real IP addresses (*e.g.*, attacks requiring the setup of a TCP session to send application layer requests), as well as attacks that are launched using botnets.

This module is a tool addressing the identification of machines taking part in an 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. This analysis method is based on the hypothesis that the behavior of a bot taking part in an attack is closely correlated to other bots being part of the same attack, and to the macroscopic evolution of the request volume in a relatively short time window containing data from before and after the attack, as well as, during the attack.

3.1.6.1 Data flow

The module analyses pcap-formatted traffic captures to detect sources with synchronized behavior in terms of request volume as the function of time, and provides the list of source IP addresses exhibiting similar behaviors.

3.1.6.1.1 Input The pcap formatted data captures can be obtained through any source. Captures need to contain at least the IP header to be usable, *i.e.* full packet capture is not required. The module can handle gzip compressed captures as well.

3.1.6.1.2 Output The analysis results consist of a list of IP addresses considered to take part in the synchronized activity. Natively, the output is formatted as IDMEF alerts and written to a file.

3.1.6.2 Module implementation

The analysis module is implemented in Perl. Several parameters for adjusting the analysis can be passed as arguments, for example:

- the pcap files to analyze;
- the size of time bins *i.e.* the length of time intervals used to construct the time series for analysis;
- the optional Berkeley Packet Filter used to select a subset of packets to process; and

- the optional time window to be analyzed (if smaller than the time spanned by the captures).

3.1.6.3 Experience gained

In this section, we discuss the experience gained with this analysis module, supporting tools, and their potential usages.

3.1.6.3.1 Scalability The analysis module poses essentially three types of challenges for scalability: space and I/O requirements related to input data, and CPU usage related to its processing.

Disk space The traffic captures kept in storage need to cover, at least, the considered time window for the duration of the analysis. As such, the minimal requirements for disk space are not important. In practice, captures are often maintained for a longer period (1) as the time required for analyzing one window varies according to the overall load of the analysis host and to avoid automated file rotation to remove old captures before they have been analyzed; and (2) in order to be able to come back to the data and to process periods of interest with other tools or manually as well.

Disk I/O Moving the packet captures around (the initial capture from the network, the copy of capture files from one host to another, and/or reading the captures for analysis) is limited by the storage medium's I/O capacity. The operations are sequential (vs random), meaning the advantage of SSD storage over rotational disks is less spectacular than in some other applications.

Overall we often use a strategy where the traffic is initially captured on a separate host and already separated to different files according to time and file size criteria, as well as other criteria meaningful to the user (e.g. grouping by destination addresses sharing the same function such as Web servers or the service provided). This initial capture can be done on a RAM disk in host memory. These captures are then transferred as soon as they are completed to another host executing the actual analysis and eventual long-term storage.

CPU The analysis itself requires CPU cycles. As the analysis is typically done over a time window that spans tens of minutes and the time window is shifted forward to less than the time window width, the capture files are analyzed more than once, adding to the overall CPU usage. On the other hand, the separation of traffic captures into different files mentioned above allows the parallelization of different analysis runs, which makes it feasible to keep up with the rate of traffic at data center entrance.

3.2 Large-scale DNS traffic analysis

3.2.1 Botnet C&C domain name detector

This module detects malicious domain names used by botnets. The use of Domain Generation Algorithm (DGA) in botnets is a common technique to mitigate C&C server take-downs, hence each botnet may employ a different implementation of DGA. The goal of this module is to classify domain names using machine learning methods and identify domains generated by botnets.

3.2.1.1 Data flow

The module is analyzing DNS traffic captured at DNS caching servers and DNS root name servers. The key assumption for this module is that a group of generated domain names has a particular pattern for the domain name structure, client distribution, and temporal locality. This module discriminates malicious domain names from legitimate ones using these metrics and Support Vector Machine (SVM).

3.2.1.1.1 Input This module analyzes DNS traffic data from one DNS cache server and one DNS root name server. The captured data is stored on MATATABI in a Hive database. This module also uses well-known blacklists of malicious domain names and well-known whitelists of legitimate domain names for the training phase of the machine learning.

3.2.1.1.2 Output This module generates a list of malicious domain names and IP addresses that queries the identified domain names. The list of IP addresses indicates potentially infected hosts (i.e. bots) which use the DNS caching server. This module also presents the distribution of IP addresses which send malicious queries to a root name server. It classifies and counts observed DNS queries' IP addresses by AS and country.

3.2.1.2 Module implementation

The current implementation is written in Python and retrieves data from a Hive database.

3.2.1.3 Experience gained

3.2.1.3.1 Applicability This module uses (1) well-known C&C blacklist, (2) well-known whitelist, and (3) Root/TLD name server traffic for learning. (1) and (2) are publicly available on the Internet. However, (3) is usually not available for outsiders, because DNS data contains sensitive data related to users privacy.

Therefore, to deploy this module in a real environment, root/TLD name server operators have to analyze their own data and share only the identified suspicious DNS names with the community.

3.2.2 DNS failure graph analysis

The goal of this module is to find hosts sharing similar DNS failed resolutions. This synchronous behavior of hosts trying to access unknown domains reveals botnets based on Domain Generation Algorithm (DGA). Thereby, the module retrieves DNS errors from the DNS datasets and seek for host communities that share the same errors.

3.2.2.1 Data flow

The module analyzes DNS data captured by the *NECOMA* consortium and reports groups of IP addresses with similar DNS errors.

3.2.2.1.1 Input The analyzed traffic consists of DNS data captured at authoritative and cache servers available on MATATABI in the Hive database.

3.2.2.1.2 Output The results of the implemented module are stored in the same Hive database. The reported IP addresses represent potential bots that can be further investigated or reported to network operators.

3.2.2.2 Module implementation

The current implementation is written in Python, it accesses the MATATABI database using the pyhive library.

3.2.2.3 Experience gained

This module monitors failed DNS queries, that are mainly queries to non-existing domains. In our experiments, we found that certain bots generate a humongous number of failed queries, which is detrimental to the performance of this module. Sampling failed queries, however, permits to overcome this issue.

3.2.3 DNS cache poisoning detection module

This module detects DNS cache poisoning attacks that inject fraudulent DNS records into DNS cache servers using malformed DNS replies.

DNS cache poisoning attack is achieved by sending a malformed DNS reply packet to the recipient before a regular reply. The 16-bit transaction ID (DNS TXID) in the response packet must match the one of the sent query,

and the destination port of the response must also match the port used to send the query.

Estimating the DNS TXID is particularly difficult for an attacker. Therefore the attacker usually sends numerous DNS responses for one DNS query. This analysis module detects this disparity between queries and responses.

3.2.3.1 Data flow

The module counts the number of sent DNS queries and received DNS responses every ten minutes.

3.2.3.1.1 Input This module uses traffic originating from port 53 captured on a network border gateway. The captured data is stored on MATATABI in Hive database.

3.2.3.1.2 Output This module outputs two lists: attacked domain names, and receiver IP addresses. The list of attacked domain names denotes the domain names that an attacker is trying to compromise. The list of receiver IP addresses indicates potential victims of the DNS cache poisoning attack.

3.2.3.2 Module implementation

The current implementation is written as a shell script with several hive queries.

3.2.3.3 Experience gained

3.2.3.3.1 Scalability This module inspects DNS traffic data in a small time period, and only counts DNS packets using simple regular-expressions based on observed queries. The MATATABI platform is designed to extract and count data in fixed periods from a huge dataset, hence, the module in its present form is able to analyze data from enterprise environments like a data center or an Internet service provider.

3.3 End-point threat data analysis

3.3.1 C&C detection in sandbox data

The module detects new, previously unidentified C&C server addresses using network traffic dumps recorded during execution of malware samples, as described in Deliverable D2.1 (section 2.3.7 and Appendix A.3).

The method works on network behaviors, listing individual connections (highly aggregated) in order of appearance in the network dumps. IP addresses of previously known C&C servers and well known benign servers are

marked. Malware samples are clustered based on similarity of behavior – the comparison uses dynamic time warping (DTW) to match corresponding flows even in presence of obfuscating traffic. For each cluster a network behavior profile is built – like the network behaviors, it is a list of flows, but information about each flow is enriched with a sequence of outgoing packet sizes (averaged over the set of corresponding flows) and a weight showing how stable these sizes are in the observed samples.

The set of clusters with network behavior profiles is then used to process data from new malware samples. The network behavior of each sample is compared with the profiles (again using DTW) and if a match is found, the correspondence between flows in the sample and in the profile is used to identify C&C connections. The destination IP addresses of such connections are very likely to be C&C servers as well.

3.3.1.1 Data flow

3.3.1.1.1 Input The module uses the following data sources:

- C&C addresses – taken from the n6 platform.
- Benign addresses – local, periodically updated list generated from alexa.com top domains data.
- Malware traffic dumps – pcap files generated by the sandboxes operated by NASK/CERT Polska, downloaded periodically to local storage.

3.3.1.1.2 Output The output of the analysis is a list of discovered C&C IP addresses, represented textually as dotted decimals. This data is currently displayed and/or dumped to a text file for manual verification. Access using the n6 API is planned but currently not available as the results are not yet trusted enough to warrant automatic reaction – false positive rate is high enough that manual verification is necessary.

Additional outputs include the lists of IP addresses identified as benign and many statistics about the analysis process – data useful in research and tuning of the module, but not actionable information. As such it is not and will not be accessible through the n6 API.

3.3.1.2 Module implementation

The module is implemented in Python and consists of two main parts – a library and the main program. The library provides functions for building the initial data structures from input streams, performing the learning phase, analyzing new samples using the clusters generated in the learning phase and many additional tools. The main program is responsible for accessing

the data sources, performing the different tasks in the right order and providing the output.

The current implementation can be used either for single-run analysis of available data or for continuous operation. The continuous mode is implemented through periodic re-learning. The new learning set is built from the previous one by discarding a fixed percentage of oldest samples and the oldest samples from the largest clusters (limiting cluster size) and including new samples which in the detection phase were classified as outliers or joined the smallest, low-quality clusters. Each iteration uses fresh data from the n6 platform as C&C addresses. The benign addresses are updated periodically (less frequently). New samples are pulled from the sandboxes periodically, much more often than the re-learning. Samples older than a pre-set threshold are removed to keep the storage requirements low.

3.3.1.3 Progress since initial research

The progress was stunted for some time by the loss of the main researcher involved in the design and implementation of the method. Research of modifications increasing the effectiveness of the method and – most importantly – reducing the false positive rate is now being conducted and will be continued outside of *NECOMA* as the method is considered valuable. Modifications including adding to the flow descriptions the port numbers and some limited information about the flow's content are currently being tested, but the module in actual use is generally the same as described in the previous deliverable of Workpackage 2, although the implementation was debugged and optimized. Of course this applies only to the actual analysis – the mundane management code, including I/O, etc. was heavily reworked.

3.3.1.4 Experience gained

3.3.1.4.1 Scalability The scalability of the approach depends heavily on the size of the learning set of network traffic dumps.

The learning process requires a lot of memory and processing power – as more individual flows appear, the growth is quadratic. In practice, a small server can work well with learning sets of roughly 1000 network traffic dumps. Methods to overcome this limitation through performing separate analysis of subsets of data and combining the results have been proposed but not verified as the current stream of data is not large enough to warrant such effort. Still, since larger learning sets theoretically offer better results, the effort might be worthwhile for large sandbox farms.

The speed of the detection process depends on the amount of clusters generated during learning and generally drops with learning set size. Given the current capability of the NASK / CERT Polska sandbox system, the module is fully capable of processing all recorded samples as they become avail-

able even with the largest learning dataset possible to process on the server used for this task. The detection process scales much better than the learning phase – the computational cost is lower and each sample is processed separately, allowing efficient parallelization.

In summary, the module is currently efficient enough to process all available data – the bottleneck is the small size of the sandbox system. Also, the method scales well enough to be applicable even in large sandbox systems, at least in the detection phase.

3.3.1.4.2 Value added The method provides good, usable output. Under the current load it provides a stream of mostly correctly identified addresses small enough to verify manually. The verified results are immediately actionable and valuable in battling botnet activity. However, making the results directly actionable requires further lowering of the false positive rate.

3.3.2 SSL server security assessment

SSL/TLS is one of the most widely used cryptographic protocols to provide secure communications over the Internet. Its scope covers a broad range of applications from HTTP transactions to email exchange to instant messaging to VoIP. Among these applications, *HTTPS* (HTTP over SSL, as it was known) is the most prominent. Despite the obvious benefit offered by proposing HTTPS connections to users, websites seemed reluctant to propose SSL/TLS certificates for some time, prompting the Electronic Frontier Foundation to actively campaign for its adoption and deployment in 2010, through their SSL Observatory¹ initiative, and even providing a browser extension to enforce SSL connections when available.

On the other hand, implementations of SSL/TLS stack have been quite prone to bugs, as evidenced by a series of vulnerabilities² discovered these recent years, prompting forced updates from one version of the protocol to another, endangering whole infrastructures with issues of backward compatibility.

This analysis module aims at accounting for the status of the SSL/TLS deployment in terms of deemed *secure* and *unsecure* versions of the protocols, or cryptographic suites, as well as taking interest in the amount of tangible information available in certificates displayed by web hosts to identify a website owner. Two different analyses targeting, on one hand, the TLS/SSL configuration (version and ciphersuites), and on the other hand, the server certificate, allow to derive security scores, on which we can assess the overall security of the communications provided by a web host, and

¹<https://www.eff.org/observatory>

²<https://tools.ietf.org/html/rfc7457>

Table 3.1: Seemingly meaningless values of each standard DN attribute

C (Country)	O (Organization)	OU (Organizational Unit)	S (State/province)	CN (Common Name)
XY, NON-STRING-VALUE, single double quotation	SomeState, Someprovince, SomeOrgani- zation, MyCompany	single double quotation, Single dot, SomeState, Someprovince, SomeOrgani- zationUnit, Division, section	SomeState, Someprovince, Some_State, Select one, Default, default	localhost. localdomain, 127.0.0.1

classifying these hosts into groups ranging from secure to unsecure servers. Ultimately, clustering these groups according to such parameters allow to summarize the overall status of the SSL/TLS deployment, and allow for a more tolerant detection of unsecure configurations.

3.3.2.1 Data flow

This module assesses the security of a set of HTTPS servers based on the collected responses to HTTPS connection requests, *i.e.*, the supported SSL/TLS versions and ciphersuites, as well as the server certificate.

3.3.2.1.1 Input The SSL dataset described in Deliverable D1.1 is taken as input through an n6 interface. For each collected information, it does provide both server configuration information (protocol versions and supported ciphersuites) and server certificate information. Any SSL/TLS dataset presenting the same structure may be used, especially if they are accessible through the n6 interface.

3.3.2.1.2 Output Two scores are computed for each analyzed HTTPS server. One is a security assessment score based on the server configuration and the other one is based on the relevance of the server certificate information. The scores are returned in a JSON message, through the n6 API.

3.3.2.2 Module implementation

This analysis module focuses on the classification of SSL servers in terms of security using the information embedded in the `Server Hello` and `Certificate` messages sent by the servers during the SSL handshake.

3.3.2.2.1 Certificate Information This analysis module makes the following assumptions. First, a reliable certificate (e.g., a certificate signed by a trusted CA) tends to include all standard Distinguished Name (DN) attributes with qualified values. Conversely, an unreliable certificate tends to have sloppy DNs. Second, a certificate issued by a known compromised CA or a CA offering incredibly-cheap/free certificates is likely to be a risky certificate. Third, a self-signed certificate is not inherently trusted. Finally, we assume that most malicious servers (e.g., a phishing host) tend to hold unreliable or risky certificates due to their negligence. Based on the information in [12] and these assumptions, we propose the following certificate-based indicators to discriminate risky SSL servers from seemingly harmless SSL servers.

- Indicator 1: at least one standard DN attribute is not presented in the certificate.
- Indicator 2: at least one standard DN attribute value is not a qualified value.
- Indicator 3: the value of the O/OU attribute contains `self-signed`, `127.0.0.1`, any compromised CA name, or any name of CAs/resellers issuing low-priced or free certificates.

The list of compromised CAs is based on collected disclosures regarding CAs that have been compromised and alleged risky CAs. Notable examples include Comodo and DigiNotar. Below are listed string keywords containing names of some compromised CAs and CAs/resellers offering low-priced/free SSL certificates identified:

- Compromised CAs: Comodo, DigiNotar, and GoDaddy.
- CAs/resellers offering certificates with low-priced costs: Namecheap, RapidSSL, fxdomain, hostingdude, and cheap-domainnames.
- CAs/resellers offering free certificates: StartSSL, StartCom, and CAcert.

As for implementation, we also considered replacing Indicator 2 with the following two representative indicators for cases where memory usage and/or dedicated analysis time are constrained.

- Representative indicator 1: the CN's value is not in domain name format.
- Representative indicator 2: at least one standard DN attribute (not including SerialNumber) contains one of the following values.

Table 3.2: Protocol version and encryption algorithm pairs for SSL server assessment

Protocol version	Secure algorithm	Risky algorithm	Insecure algorithm
SSLv3	None	3DES_CBC, IDEA_CBC	DES_CBC, RC2_CBC, RC4
TLSv1.0	None	3DES_CBC, AES_CBC, IDEA_CBC	DES_CBC, RC2_CBC, RC4
TLSv1.1	3DES_CBC, AES_CBC, IDEA_CBC	None	DES_CBC, RC2_CBC, RC4
TLSv1.2	3DES_CBC, AES_CBC, AES_CCM, AES_GCM, Camellia_CBC, Camellia_GCM	None	RC4

- an empty or an implied empty value (e.g., “”, “ ”, NONE, None, none, BLANK, blank, X(s), ?(s), or -(s)),
- a default value (e.g., S=“SomeState”) or a seemingly meaningless value (see Table 3.1).

3.3.2.2.2 Protocol Version and Encryption Algorithm A key assumption we make is that the protocol version and the encryption algorithm chosen by an SSL server are suitable parameters to assess the security level of the communication as well as the SSL server. From a security standpoint, application data sent over a weak protocol version (e.g., SSLv2) or encrypted with a weak encryption algorithm (e.g., RC4) is endangered due to known security flaws of these protocols. On the other hand, if the data was sent using a known-good protocol version or a strong encryption algorithm that does not suffer from any known security vulnerabilities, the client can assume that the communication is safe. To assess an SSL server, we analyzed the encryption algorithms supported by each SSL protocol version and assessed their efficiency based on publicly discovered flaws.

Table 3.2 categorizes the implementation of different encryption algorithms by different versions of SSL with respect to their security. A secure server is one that selects a known-strong protocol version and encryption algorithm. The strongest protocol version at the moment is TLSv1.2. Some known-secure encryption algorithms are the Advanced Encryption Standard

(AES) standardized by the US National Institute of Standards and Technology (NIST) in 2001 and Camellia developed later by Mitsubishi and Nippon Telegraph and Telephone (NTT). We consider AES and Camellia as secure encryption algorithms because AES has yet to be compromised by any attacker and Camellia has been proven as strong as AES [40]. Furthermore, to break these algorithms, technically an attacker requires an enormous number of years, e.g., 5×10^{21} years for 128-bit AES encryption [22]. On the other hand, a risky server is one that selects a risky encryption algorithm and SSL protocol version that may be vulnerable to attacks. CBC ciphers are considered risky because they can be broken by the BEAST (Browser Exploit Against SSL) attack [15]. However, if the user's application is designed to defeat the BEAST attack, the attack can be mitigated. Table 3.2 also indicates that if a server uses a newer version of SSL, the security of that server increases because the number of implementation flaws has decreased. For example, a server that chooses 3DES_CBC with TLSv1.1 is described as secure because TLSv1.1 has fixed issues regarding the BEAST attack. Finally, an insecure server is one that selects any publicly known weak encryption algorithm. For example, DES has been defeated by brute-force and differential [16] attacks, and RC2 or RC4 have been defeated by related-key attack [17]. Most importantly, a server that selects old-fashioned SSLv2 with any encryption algorithm is immediately considered as insecure because SSLv2 is flawed in a variety of ways [3]. For example, it has a weak Message Authentication Code (MAC) construction that uses MD5 with a secret prefix, making it vulnerable to length extension attacks [14].

3.3.2.2.3 Score Computation The analysis module embeds two functions that requests information from the SSL dataset described in Deliverable D1.1. When computing the security of an SSL server configuration, a request similar to the following is issued:

```
/ssl_info.json?ip_address=W.X.Y.Z&timestamp=YYYY-MM-DDTHH:MM:SSZ
```

The contents of the JSON response are parsed and fed to a Python function that will compute a security score based on the protocol version information and ciphersuites extracted. The score is returned as a JSON object.

Similarly, the security assessment score based on the certificate information takes as input the JSON response to the following request:

```
/get_certificate.json?ip_address=W.X.Y.Z
```

3.3.3 Malicious URL signature generation

Botnets still constitute one of the most severe threats on the Internet, and can be found in many different kinds [23], let it be the architecture, communication protocols or constituents. For years, IRC-based centralized botnets

were the norm, but more and more botnets are using the HTTP(S) protocol in order to make it difficult for IDSes to distinguish their traffic from legitimate HTTP transactions. Furthermore, a new trend has appeared where vulnerable web applications are used to transform web servers into bots, as evidenced by the notorious example of the `itsoknoproblembro`³ DDoS tool, which exploited vulnerabilities of content management systems such as WordPress and Joomla.

This module analyzes URLs generated by bots in order to extract patterns able to identify botnet traffic. The intuition is that bots will necessarily attempt to contact their command-and-control (C2C) servers, when Internet connectivity is available. Obviously, legitimate HTTP requests may be issued in order to obfuscate their C2C communication in a lot of noise. However, assuming bots of the same kind would communicate according to a similar protocol, we should be able to observe similar patterns from one bot to another, in terms of the structure of the URL (the path) and the semantics (the querystring, in particular, the key-value pairs).

3.3.3.1 Data flow

This module clusters a set of URLs using a 2-step clustering process, based on the similarity of the path and querystring of these URLs, with the ability to reduce the impact of noise. The resulting clusters allow to generate a signature able to match most of the clustered URLs. These signatures can later be used by an IDS or an IPS to raise an alert about botnet activity or to block suspected botnet traffic, respectively.

3.3.3.1.1 Input Any URL dataset can be used as input as long as the path and querystring are available. On the other hand, domain names can be ignored as they are not used in the clustering process.

3.3.3.1.2 Output Once clusters have been computed, the last stage of the analysis module is to compute the quality of each cluster, as well as generate a signature (*i.e.*, a regular expression) matching most URLs in the cluster. The list of signatures and the associated cluster quality is returned. The cluster quality can be considered as a level of confidence associated to the cluster signature. The output format is a text file, but other formats such as CSV and JSON can be considered and would not require too much alteration to the analysis module. The results are accessible through the Web-based user interface, but a REST API could easily be deployed for machine-to-machine communication.

³<http://www.infosecurity-magazine.com/news/dissection-of-itsoknoproblembro-the-ddos-tool/>

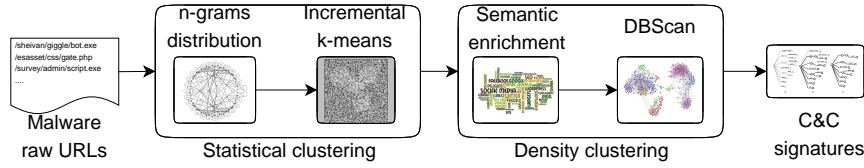


Figure 3.1: Architecture and workflow of the 2-stage URL patterns signature generation

3.3.3.2 Module implementation

The overall architecture of the analysis module is illustrated in Figure 3.1. It leverages the fact that C&C traffic holds common patterns and features that are shared among variants of the same malware family. It can be separated from noise which usually includes random patterns that depend on the running environment and analysis time of each malware. Therefore, the module operates a two-stage clustering process. First, it splits the initial dataset into smaller clusters based on common URL statistical features. URLs often include patterns (e.g., /images/, /adi/, /generate_204/) and keywords (e.g., .php, .exe, .gif) that refer to the nature and type of resources accessible on the remote server. The module leverages the distribution of characters within URLs in order to group together malware URLs that include similar or redundant patterns. It builds a features vector that captures the characters distribution within the URL. The URL path and parameters are handled differently because they have different structure and semantics. Statistical clustering is implemented in order to put these URLs into the same clusters. Although malware may still use obfuscation or add random URL patterns to avoid detection, statistical clustering mostly leverages patterns that are shared among samples of the same malware family.

Statistical clustering provides coarse clusters where URLs only share similar statistical patterns. Therefore, the analysis module applies a second, density-based clustering that builds fine-grained clusters where URLs share similar structure and semantics, and that characterize common C&C applications. It implements a semantic enrichment process where it adds meta-data that characterize the type and semantic of URL parameters. The density-based clustering uses semantic meta-data in order to build fine clusters where URLs are associated with the same C&C applications. Remaining malware URLs are discarded into *noise* clusters that are no longer considered for signature extraction. The signature module uses generalized suffix trees [7] to extract a malware detection signature associated with each dense cluster, and that characterizes a common malware C&C application.

The tool is implemented in Python for URL parsing and manipulation as well as clustering and signature generation, Bash for automation and command-line integration, and Octave/MATLAB for visual rendition of the

clusters. Additionally, a Web-based user interface has been developed using `bootstrap`, `ember.js` and `socket.io`.

3.3.4 High Performance Phishing Detection

The aim of this analysis module is to discover malicious web sites based on the analysis of the hostnames. The module is based on a real-time machine-learning approach capable to adapt itself to changing environments.

3.3.4.1 Data flow

The module works basically by analyzing string representations of hostnames.

3.3.4.1.1 Input Since the module requires training datasets to establish the analysis parameters, the input includes a reliable source of malicious and legitimate hostnames. For this purpose the module queries, in fixed intervals, Phishtank (as source of malicious hostnames) and Alexa (as source of legitimate hostnames). The module converts the hostnames into feature vectors.

The module exposes a REST API and once trained allows submission of suspicious hostnames in string format.

3.3.4.1.2 Output The API response to the submitted hostnames is a n6-formatted object containing the classification result along with a confidence score. For the classification, it follows the metrics described in Sect. 5.2.2. Details about the format and accessibility of the results are described in Deliverable D3.3: *Security Information Exchange – Results*, chapter 3.4.

3.3.4.2 Module implementation

The whole prototype is implemented in Python with Mongo DB attached as a database holding information about the current status of the module configuration.

3.3.4.3 Progress since initial research

Initially the module was a standard, off-line, machine-learning-based module. But the rapidly changing environment of phishing attacks as well as the expansion of legitimate hostnames namespaces required a solution that would efficiently adapt to those changing variables maintaining effective

¹For details about the API please refer to Deliverable D3.3: *Security Information Exchange – Results*, chapter 3.4

malicious hostnames discovery. Consequently, the module evolved to a more advanced online-learning approach, which allows modification of the system parameters in real time while feeding a stream of training data.

3.3.4.4 Experience gained

3.3.4.4.1 Scalability The solution has relatively high performance and delivers real-time analysis on a commodity computer. It has very low requirements in terms of storage as it only needs the configuration parameters to carry out the analysis. The drawback is that, as with any machine-learning solution, the module is very dependent on the quantity and quality of the training datasets.

3.3.4.4.2 Value added The module works as an excellent malicious hostnames pre-filter. Although the detection rate is quite low, the precision rate is very high. It is possible to increase the detection rate by lowering the precision, but with the detection rate at around 8% the module performs with 100% precision.

3.4 Cross-layer threat data analysis

3.4.1 Zeus DGA Detector

This module detects hosts compromised by the ZeuS malware, by looking at hosts querying suspicious domain names or hosts using Domain Generation Algorithm (DGA). In the case of a 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.

3.4.1.1 Data flow

The module inspects DNS pcap dataset stored on the MATATABI platform, by searching for DNS queries that satisfy the specific regular expression filter depicted in figure 3.2.

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

Figure 3.2: Regular expression of a DGAed domain name.

Then it records IP addresses included in the corresponding DNS answers, if any. Those IP addresses are likely to be command and control servers (C&C servers).

Then, the module looks into the backbone traffic datasets (i.e., NetFlow datasets) to identify the clients IP address generating the suspicious DNS queries.

Additionally, a visualization tool generates a graphic based on detected DNS query names and IP addresses (see Figure 3.3). These results are also stored on the MATATABI platform, and reported in a NECOMatter timeline in order to share the text-based information to other parties.

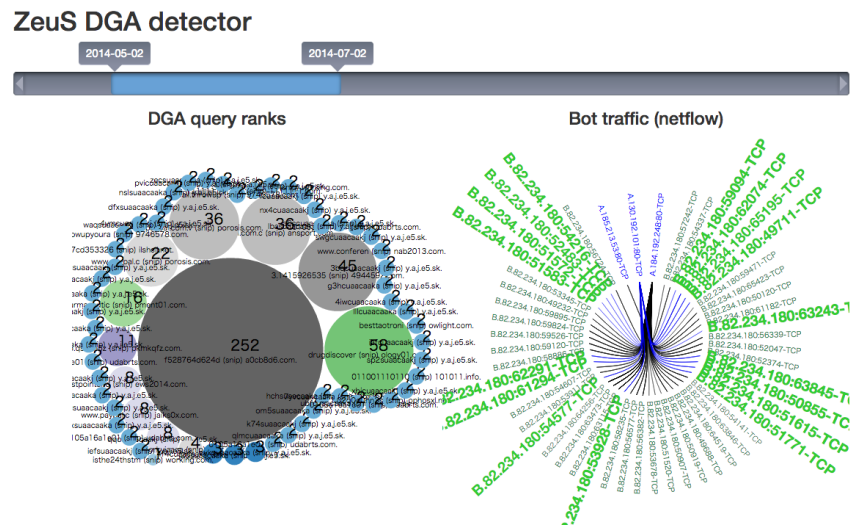


Figure 3.3: An overview of detected compromised hosts associated with the traffic to the C&C servers. A bubble chart on the left represents the rank of queried names matched with the regular expression (figure 3.2), and the chord diagram on the right represents the traffic flows associated with the DNS queries.

3.4.1.1.1 Input The input for this analysis module is listed below.

- traffic of DNS cache servers
- NetFlow traffic data from backbone networks

3.4.1.1.2 Output The module generates the following information:

- a list of suspicious DNS queries
- a list of flows associated with the suspicious DNS queries.

The second information is especially useful to identify the compromised hosts using the DNS cache server of interests.

The above information is accessible by two different ways.

- via the web based visualizer (figure 3.3)
- via n6 based API (figure 3.4)

```
{
  "category": "bots",
  "name": "zeus",
  "proto": "tcp",
  "address": [
    {
      "ip": "1.2.3.4",
      "dir": "src"
    },
    {
      "ip": "4.3.2.1",
      "dir": "dst"
    }
  ],
  "time": "2015-07-13T08:25:33",
  "dport": 443,
  "sport": 63974,
  "until": "2015-07-13T08:25:33"
},
```

Figure 3.4: An example output of n6 API.

3.4.1.2 Module implementation

The module is implemented in a shell script, accessing datasets on our MATATABI platform by a set of presto SQL queries.

3.4.1.3 Experience gained

3.4.1.3.1 Scalability Considering the amount/size of input datasets on our analysis platform (MATATABI), the simplicity of the analysis module implementation, and the execution interval of an analysis, the module scalability is good even if the size of datasets is growing bigger everyday. The analysis is a simple query with a regular expression, which does not affect the overall load of the analysis platform, with a single analysis in a day.

It is worth mentioning that the current visualization (figure 3.3) with a bubble chart and a chord diagram is not suitable for a huge amount of

data points: if the user specifies a long duration, more than 6 months for instance, it does not clearly illustrate the information of interest. This can be alleviated by specifying a reasonable duration using the appropriate slide bar in the web page.

3.4.2 FP-SVM

This module investigates the use of frequent pattern mining and supervised learning classification for malicious campaign identification. The detection process is based on the analysis of URLs (Uniform Resource Locators). Because malicious URLs are generated by tools which employ the same obfuscation mechanisms, for a given campaign, we assume that usually attackers keep some parts of the URL static, while other parts are changed systematically and in an automated fashion.

We have designed and developed the module for malware campaigns identification. The module utilizes FP-tree data structure of tokenized malicious URLs to form a training dataset for learning Support Vector Machine (SVM) classifier. Finally, the trained SVM is used for on-line classification of new malware URLs into related or unrelated with malicious campaigns.

3.4.2.1 Data flow

3.4.2.1.1 Input In our experiments, we analyzed mainly the malurl dataset retrieved with the n6 API, but any dataset containing URL addresses or domain names can be used.

3.4.2.1.2 Output The result of the analysis is similar to the malurl dataset but with additional class attribute, as the purpose of the analysis is to assign class value based on URL analysis. This module results are stored in CSV format with the attributes presented in Table 3.3

3.4.2.2 Module implementation

The architecture of the module for malicious campaigns identification is presented in Fig. 3.5. It consists of three main elements: a database collecting malicious data, 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 malicious datasets containing URLs as related or not to a campaign. To produce the SVM classifier a training dataset collecting malicious data (precisely URLs) that have already been classified into campaigns is required. In the FP-SVM module 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 $S_{URL} = \{URL_1, URL_2, \dots, URL_N\}$ for frequent pattern mining is created.

Table 3.3: Resultant dataset description.

Attribute	Description
<i>Time</i>	Time of occurrence of a incident in Unix Timestamp format.
<i>ID</i>	ID of an incident
<i>Source</i>	Source name that detected an event.
<i>Category</i>	Category of a incident: 1-malicious URL, 2-spam, 3-phishing, 4-botnet, 5-C&C servers.
<i>Name</i>	Name of an incident
<i>md5</i>	md5 shortcut
<i>IP address</i>	IP address in byte-decimal format.
<i>URL</i>	Uniform Resource Locator
<i>fqdn</i>	Domain name
<i>ASN</i>	Autonomous System Number.
<i>cc</i>	Country code compatible with ISO 3166-1 alpha-2.
<i>Details</i>	Additional information
<i>Class</i>	Value of class label

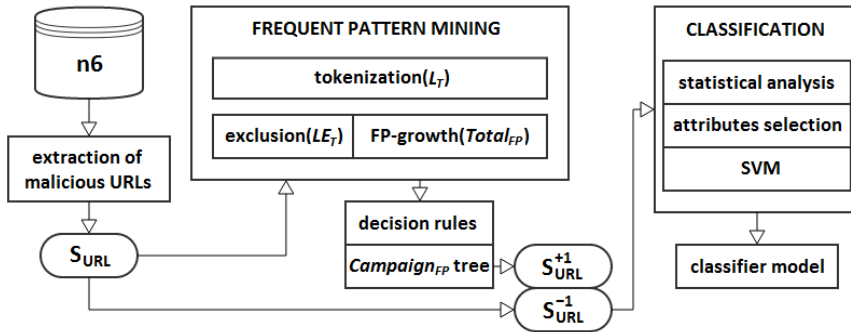


Figure 3.5: The architecture of the FP-SVM module.

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 are adopted. Each sample is cut by a specific characters that are typical for URLs, i.e., “/”, “.”, “?”, “#”. 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 infor-

mation, 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.

The following attributes have been selected in the FP-SVM module: date, time, source, category, address (IP, ASN), country code. 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 the FP-SVM module. The following nonlinear kernels are provided: polynomial function: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$, radial basis function: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, 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. The classification model has to be continuously updated. The data about new Internet security threats should be used in training process.

3.4.2.3 Experience gained

3.4.2.3.1 Value added The aim of the experiments was to validate the FP-SVM module 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

3.4. CROSS-LAYER THREAT DATA ANALYSIS

from the $Total_{FP}$ tree. The $Total_{FP}$ tree and the approach to the $Campaign_{FP}$ tree generation.

Table 3.4 presents the results of the application of the FP-growth algorithm and our decision rule with $m = 4$ to the S_{URL} dataset.

Table 3.4: Identification of malicious campaigns; the S_{URL} dataset.

Data related with any campaigns ($\{+1\}$)	62%
Data unrelated with any campaigns ($\{-1\}$)	38%
No. of samples	100 000

It was used to classify the dataset consisting of 100 000 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. The accuracy of the test is measured by AUC (Area Under ROC Curve – the measure that shows how well the test separates the URLs being tested) and another popular measure – F-measure. The value of precision shows how close the separated URLs are to each other. 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 3.5. Figure 3.6 shows the accuracy of the classification for various sizes of the training dataset. In general,

Table 3.5: Evaluation of SVM classification; $N = 100000$ URLs.

Criterion	Value
\overline{Sens}	0.72
\overline{Spec}	0.81
\overline{CA}	0.78
\overline{AUC}	0.82

the results presented in Fig. 3.6 and Table 3.5 confirm that the accuracy of classification strongly depends on the size and quality of a training dataset. Moreover, it is very important to choose the adequate kernel function. 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. It is worth to mention that due to our assumption that the same obfuscation mechanisms is used to the malicious URLs generation, and usually intruders keep

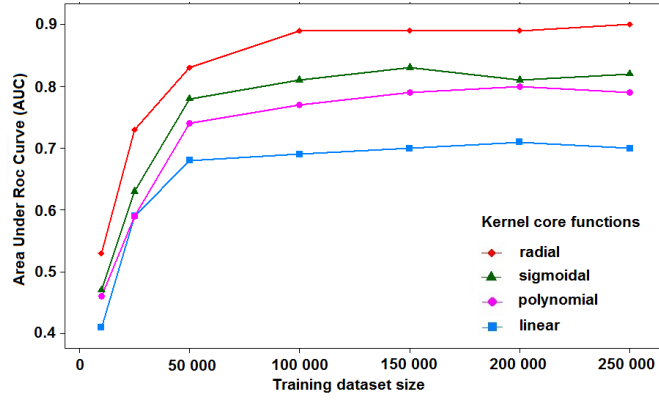


Figure 3.6: Area Under ROC Curve values for various sizes of the training dataset S_{URL} .

some parts of the URL static we are able to identify the URLs related to a given campaign. In our approach we use the most frequent tokens (the static parts of malicious URLs) in the classification process.

3.4.3 Correlation Graph Analysis

This module implements a technique for automatic identification of a malicious campaigns based on the cross-layer analysis of different datasets containing data related to various types of malicious activities. The goal of the method is to compare and detect a correlation among datasets containing malware data based on relationships between selected, relevant attributes related to collected data. As a final result, we generate a correlation graph from all occurrences of values of the selected attribute based on a comprehensive analysis of data from all accessible databases containing malware data. Next, the value of a rating function is calculated for each vertex corresponding to a value of a given attribute. Finally, the correlation graph and rating values assigned to the vertices are used for automatic identification of malicious web campaigns.

3.4.3.1 Data flow

3.4.3.1.1 Input The analysis uses the malurl, phish, spam, bots and cnc datasets from the n6 platform.

3.4.3.1.2 Output The result of the analysis is an additional class attribute for the malurl dataset, identifying correlation between malurl and others datasets. The output is currently provided in csv format, pending inclusion in the original n6 dataset. The content of the resulting file is presented in the Table 3.6

Table 3.6: Output fields of the correlation graph analysis

Attribute	Description
<i>Time</i>	Time of occurrence of an incident in Unix Timestamp format.
<i>ID</i>	ID of the incident
<i>Source</i>	Name of the source that detected the event.
<i>Category</i>	Category of the incident: 1-malicious URL, 2-spam, 3-phishing, 4-botnet, 5-C&C server.
<i>Name</i>	Name of the incident
<i>md5</i>	md5 hash
<i>IP address</i>	IP address in byte-decimal format
<i>URL</i>	Uniform Resource Locator
<i>fqdn</i>	Domain name
<i>ASN</i>	Autonomous System Number
<i>cc</i>	Country code compatible with ISO 3166-1 alpha-2
<i>Details</i>	Additional information
<i>Class</i>	Value of the class label

3.4.3.2 Module implementation

The correlation graph is generated based on the cross-layer analysis of S_b -bots, S_s -spam, S_c -cnc, S_u -malurl and S_p -phish datasets. It shows the relationships among these datasets based on values of the selected attribute. All values of attributes collected in table 3.6 can be extracted from the datasets. The correlation graph is formed by vertices and edges. The vertices represent all detected values of a selected attribute. One of the following attributes can be assigned to vertices: IP address (ip), country code (cc), autonomous system number (asn). One attribute must be chosen as the starting point of the identification process. The vertices can differ in size and colour. We assign a unique colour to each incident type, and at the same time to each dataset (S_b , S_s , S_c , S_u , S_p). The size of each vertex depends on the number of occurrences of a given value of the attribute assigned to this vertex. Hence, the repeated pattern attribute value results in a big size of the corresponding vertex. The edges show the relationships between vertices – two connected vertices belong to the same domain.

The algorithm of the correlation graph generation is based on a comprehensive multiple malware datasets analysis and will be presented on an example where the selected attribute is the IP address (ip). One cycle of the

algorithm performed for a given IP address, i.e., $ip = IP^k$ where k denotes the number of iteration, is presented below.

Step 1

Create dataset S^k

$$S^k = \{sample_1(IP^k), \dots, sample_i(IP^k), \dots\}$$

Extract all samples containing IP^k from the database $S = S_b \cup S_c \cup S_u \cup S_p$.

Step 2

Count value of the rating function $Rf(ip)$ for $ip = IP^k$.

Step 3

Create dataset Sd^k

$$Sd^k = \{domain_1(IP^k), \dots, domain_L(IP^k)\},$$

L - number of domains.

Extract all domain names assigned to all samples from the set S^k .

Step 4

Remove S^k from S ,

$$k = k + 1$$

Goto Step 1

We start from the query generation – the initial value of the IP address $ip = IP^k$, $k = 0$. All data samples related to this IP address are extracted from the database $S = S_b \cup S_c \cup S_s \cup S_u \cup S_p$ and form the set $S^k = \{sample_1(IP^k), \dots, sample_i(IP^k), \dots\}$. A value of the rating function $Rf(ip)$ is assigned to IP^k address. This value depends on the frequency of the occurrence of this IP in the whole database with malicious software. It is higher for more frequent IP addresses. In this example we assumed that the rating assigned to each k -th IP address is equal to the number N^k of occurrences of this IP in the whole dataset.

Next, all domain names assigned to all samples from S^k are extracted and form a set containing domain names

$$Sd^k = \{domain_1(IP^k), \dots, domain_l(IP^k), \dots, domain_L(IP^k)\}.$$

Each $domain_l$ from this set points to other IP addresses that are related to this domain. We select another IP address from the set of samples with the domain $domain_l$ and increase the iteration number, $k = k + 1$. Again, the set S^k with data related to the new IP address is formed, and the rating function $Rf(ip)$ is calculated for the new address (IP^k), etc. The vertices of the graph represent the IP addresses extracted from the database S .

3.4.3.3 Experience gained

3.4.3.3.1 Value added We performed the comprehensive analysis of data from S_b , S_s , S_c , S_u and S_p datasets extracted from the n6 database. The qualitative correlation graph with vertices corresponding to IP addresses was created based on the data analysis, Fig. 3.7. The graph provides information about number of correlated vertices (IP addresses), their interconnections and values of rating functions $Rf(ip)$ related to all vertices. The interconnection between two vertices means that these two different IP addresses are related with the same domain name. For data from a one month interval we obtained the correlation graph $G = (V, E)$ formed by 100 vertices corresponding to a given IP address.

Next, we performed a detailed analysis of all graph vertices. We checked the occurrence of each IP address in all datasets considered (S_b , S_s , S_c , S_u , S_p). We divided all vertices into two groups: a multi-type and a single-type. Each multi-type vertex corresponds to such IP address, which is related with at least two types of malicious activity is assigned by dominated type colour i.e., S_b – botnet (yellow), S_s – spam (pink) S_u – malicious URL (red), S_p – phishing (blue), S_c – command & control servers (green). The single-type vertex corresponds to IP address, which is related with only malurl dataset. It is shown in black. Finally, we divided all vertices (all extracted IP ad-

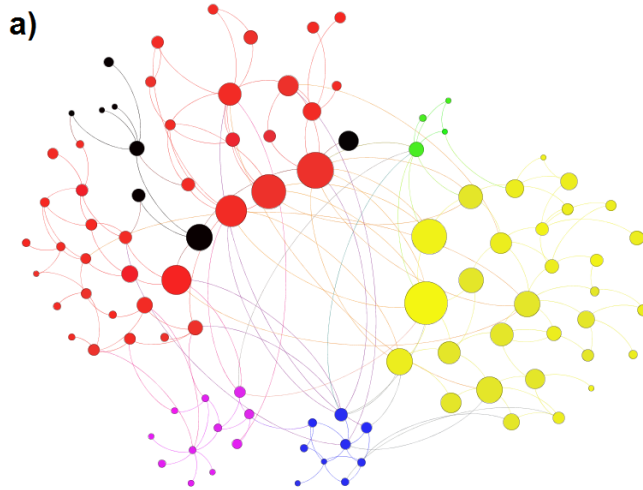


Figure 3.7: The quantitative correlation graph for malicious datasets: S_b , S_s , S_c , S_u , S_p

resses) into five clusters related to five types of attacks, i.e., botnet, spam, malicious URL, C&C servers and phishing. The division was made according to the following rule: all vertices corresponding to IP addresses that mainly occurred in S_b were classified to botnet cluster (botnet campaign), etc. It can be observed that black vertices corresponding to the IP addresses related

to multiple attacks dominate all other vertices – the malicious activity was detected by various systems and accompanied incidents of different types.

The degrees of vertices of the graph are shown in Fig. 3.8). The average degree is 3.720.

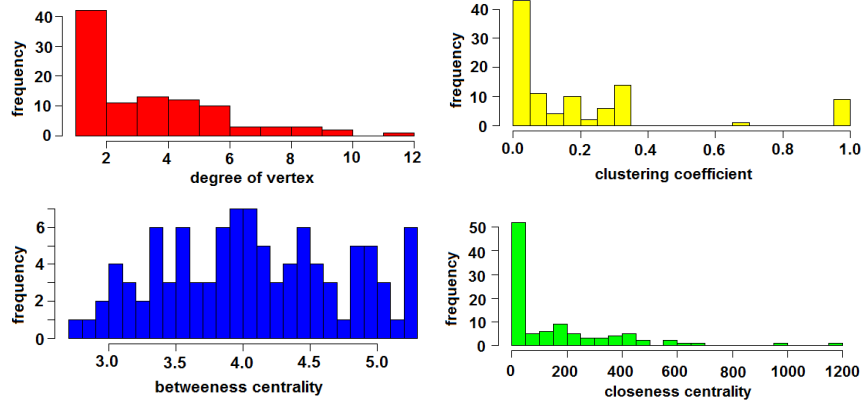


Figure 3.8: The degrees of vertices (the graph from Fig. 3.7).

To provide a more detailed analysis of the presented correlation graph we calculated two common measures in the graph theory: clustering coefficient and betweenness centrality. The results are presented in Figure 3.8. They can be used to support the decision process during mitigation of the attack.

The local clustering coefficient $C_i \in [0, 1]$ is used to measure the clustering in a given graph. This coefficient for an undirected graph is defined as follows:

$$C_i = \frac{2|\{e_{jk} : v_j, v_k \in V_i, e_{jk} \in E\}|}{K_i(K_i - 1)}, \quad (3.1)$$

where $V_i = \{v_j : e_{ij} \in E \wedge e_{ji} \in E\}$ is a set of neighbours of the vertex v_i and K_i the number of neighbours. $C_i = 0$ denotes that there is no clustering, $C_i = 1$ denotes maximal clustering, which happens when the network consists of disjoint cliques. The average clustering coefficient is the mean of all local clustering coefficients:

$$\bar{C} = \frac{1}{n} \sum_{i=1}^n C_i, \quad (3.2)$$

where n denotes the number of vertices. \bar{C} is used to measure the overall level of clustering in a given graph. Fig. 3.8 shows the distribution of the local clustering coefficients for vertices of the graph. The average clustering coefficient is equal to 2.851. A high value of the average clustering coefficient suggests that the groups of strongly connected vertices can be

detected, and the vertices belonging to the same group probably represent attributes of malicious instances from the same campaign.

Another common measure is the betweenness centrality of a vertex v_i . It indicates the vertex's v_i centrality in a graph. It is defined as follows:

$$B_i = \sum_k \sum_{j>k} \frac{g_{jk}^i}{g_{jk}}, \quad (3.3)$$

where g_{jk} denotes the total number of shortest paths linking the vertices v_j and v_k g_{jk}^i is the number of these paths that pass through the vertex v_i .

Figure 3.8 shows the distribution of the values of betweenness centrality of vertices of the correlation graph. The number of shortest paths for a subgraph with diameter equal to 5 hops and a radius equal to 3 hops was equal to 9900. The average path length was equal to 4.078. A vertex with high betweenness centrality clearly plays an important role in the attack.

Effective cross-layer analysis requires access to good quality data sources that can be combined to create a complete, multifaceted picture of contemporary threats. From the perspective of a consumer of the data, for example an automated cross-layer analysis module, the number of potentially useful sources is large, however they all come with a cost. Upfront costs include integration effort to connect a new dataset with existing tools it, which often must occur before the actual utility of the source can be established. Equally important are the operational costs, since each query against the dataset incurs some cost: they might be translated into actual monetary cost (as is the case with many commercial services with query limits) but also consume increase the processing time which is crucial for near-realtime or large-scale applications. Dataset rating can augment two decision processes:

- On the organizational scale: choosing which data sources should be integrated.
- On the scale of a single investigation performed by an analyst or a single automated analysis module: choosing data sources optimal for a particular query.

The proposed method of rating was described in Deliverable D2.1 (section 3.3) and this chapter describes the updated set of rating components and results that were obtained by applying them to 45 datasets available in the n6 platform operated by CERT Polska (part of NASK), which constitutes more than a half of datasets active in 2015.

4.1 Rating components

To experimentally verify the utility of the previously proposed rating method, rating components defined for threat datasets (D2.1, section 3.3.2) were

computed for multiple datasets stored in the n6 platform. This section describes the technical aspects of the rating method, including any differences from the methods proposed initially.

4.1.1 Rate

Rate of a source is defined as the number of unique IP addresses or domains reported per day.

Investigated datasets were divided into two groups: ones that provide information on suspicious IP addresses and ones that provide URLs and domains. This distinction was made since all datasets in n6 are automatically enriched — the original information is extended through correlation with external data sources, including DNS. By determining the original type of information provided by the source, we are able to evaluate the property of the source itself, without changes introduced during processing.

4.1.2 Delay

n6 always stores the original unprocessed data received from a source, along with the time that it was received by the system. If the data source provides sufficiently precise timestamps, it is possible to compute the time between the *detection* of a suspicious event and the time it was *received* by n6.

This time includes the processing latency introduced by the producer of information and the latency related to the transport mechanism used to fetch the data. For example, for mechanisms based on polling, the interval between queries will have a big influence on the resulting delay.

The value of delay for the whole source is a mean of delays for individual events in the evaluated period.

4.1.3 False discovery rate

In order to obtain an estimate of false positive ratio, IP addresses and domains reported by data sources were compared to predetermined lists of addresses that can be assumed to be “safe”, i.e. not performing malicious actions and not hosting malware. The following lists were created:

- Domain whitelist: top 258 websites according to traffic statistics published by Alexa[1] list, augmented with several additional entries that can correspond to well-known benign domains.
- Domain greylist: whitelisted domains or subdomains of the whitelisted domains that cannot be assumed to be benign at all times, e.g. ones that host arbitrary user-supplied content or ones associated with adware or other suspicious software. The greylist has precedence over the whitelist.

- IP whitelist: one thousand IPv4 addresses and networks associated with popular services — website hosting, DNS, internal infrastructure of the largest cloud providers and 14 non-routable networks (“bo-gons”). Popular networks were identified based on network statistics in the NASK internal network and the WHOIS database.

For sources providing domains and URLs, the false discovery rate is defined as the number of domains reported by a source that appear on the whitelist and do not appear on the greylist divided by the total number of reported domains. For sources providing IPs, the false discovery rate is equal to the number of addresses appearing on the whitelist divided by the total number of reported domains. In both cases the examined sets consist of all unique domains or IPs for the whole evaluated period are investigated.

Presented method of estimating the number of false positives is based on the assumption that it is rare for high-profile services (e.g. Google search engine) to be involved in a malicious activity on a scale that would be detected and reported by the available sources. The resulting rate corresponds to the lower bound of the false discovery rate, since this approach does not attempt to validate reports concerning less known addresses.

4.1.4 Cross-dataset linkage

Linkage between datasets can be used to estimate the usefulness of a particular source for correlation using common tokens. This metric can also reveal which datasets describe different aspects of the same threat.

Cross-dataset linkage is defined as the ratio of events from the given source that share at least one IP address with an event from another source, as long as the other event belongs to a different category. Links within a single category were omitted because they correspond to the simple *overlap* between sources — such events in general describe the same aspect of a threat. For example, a scanner might be detected by multiple sensors but having multiple reports about the same activity does not provide more context for an analyst. Therefore the focus is put on links between categories, since they can reveal additional actionable information. In the previous example, if the source of the scan is linked to a malware infection, it is much easier to understand the observed activity.

In order to make the results more representative, linkage was computed for each source in n6 that was active in the evaluated period.

4.1.5 Representativeness

The initial proposition for this rating component was based on comparing country and ASN distribution of the evaluated dataset against a “reference” distribution computed for a collection of other selected datasets. This approach was not found useful in practice for datasets available in n6.

Instead of comparing distributions, the Wilcoxon's DM[46] index (deviation from the mode) was used for the purpose of distinguishing between datasets focused on a single country and ones with a global scope. This is a relevant issue, since often third-party data providers limit the scope of the shared data. Particularly in the case of n6, a significant number of the datasets are limited to IP addresses located in Poland only.

The DM index is defined as follows:

$$1 - \frac{\sum_{i=1}^k (f_m - f_i)}{N(K - 1)}$$

where f_m is the frequency of the most common category, K is the number of categories, f_i is the frequency of the i -th category, N is the total number of elements.

The examined categorical variable was the country associated with an event reported by the given source in the evaluated period. Relationship between events and countries is performed via standard geolocation mechanisms that are used by n6 on all of the incoming data.

4.1.6 User/utility rating

Including feedback from users was proposed as a method of rating the *utility* of a dataset in real-world conditions. Unfortunately, collecting a sufficient amount of human feedback is difficult and the approach to rating had to be modified.

The alternative method of establishing utility of a data source is to examine if it contains events relevant for analysts working on investigating threats. A dataset containing “relevant” information was created by extracting logs from a system for querying and correlating data from multiple sources, which is used internally in CERT Polska. This yielded 2134 unique IPs and domains that were manually queried by analysts in the course of one year.

These queries were issued again for datasets available in n6. Value of the utility rating is computed as the ratio of queries that returned at least a single match for the given dataset to the total number of queries.

4.2 Evaluation results

Table on the next page presents a summary of all rating components computed for n6 datasets. Data source names are anonymized and prefixed with category name (for sources that provide events from a single category only), except for 7 datasets provided directly by NASK.

4.2. EVALUATION RESULTS

source	type	rate	delay	fpr	link	dm	cc	utility
arakis.gov-scan	ip	3504.2	24.2h	.0001	.144	.617	US	.0314
arakis.gov-snort	ip	4219.9	18.5h	.0000	.185	.813	US	.0230
arakis.pl-scan	ip	294.3	20.7h	.0000	.223	.737	CN	.0122
arakis.pl-snort	ip	157.2	19.2h	.0000	.251	.842	US	.0103
cert-pl.cuckoo	domain	187.4	-	.0512	.448	-	-	.0122
cert-pl.sinkhole	ip	595428.1	831s	.0000	.002	.804	EG	.0342
cert-pl.zeus-p2p	ip	11463.2	0s	.0000	.001	-	-	.0005
amplifier.1	ip	16954.2	37.7h	.0000	.036	.017	PL	.0019
amplifier.2	ip	33738.5	35.7h	.0001	.032	.003	PL	.0014
amplifier.3	ip	59104.4	36.7h	.0000	.077	.002	PL	.0070
amplifier.4	ip	491.7	28.1h	.0000	.027	.000	PL	.0000
amplifier.5	ip	42882.2	33.9h	.0000	.066	.001	PL	.0061
amplifier.6	ip	19992.7	28.6h	.0000	.094	.001	PL	.0056
amplifier.7	ip	762.3	31.3h	.0000	.023	.000	PL	.0000
bots.4	ip	602.5	14.7h	.0000	.008	.482	JP	.0005
bots.5	ip	89.5	16.8h	.0000	.147	.112	PL	.0000
bots.6	ip	1852428.9	613s	.0002	.001	.856	IN	.0440
bots.8	ip	9311.9	711s	.0000	.115	.001	PL	.0178
bots.9	ip	19956.3	25.9h	.0000	.133	.001	PL	.0136
bots.10	ip	5424.4	28.3h	.0000	.100	.001	PL	.0061
bots.11	ip	1471.2	26.7h	.0000	.132	.002	PL	.0037
bots.12	ip	20772.3	12.1h	.0000	.119	.001	PL	.0258
bots.13	ip	109.2	15.1h	.0000	.104	.006	PL	.0028
bots.14	ip	3204.9	0.9h	.0000	.112	.006	PL	.0056
malurl.1	domain	23493.0	-	.0313	.088	.541	US	.3594
malurl.2	domain	680.9	-	.0003	.118	.550	US	.0197
malurl.5	domain	19.6	-	.0056	.111	.092	US	.0019
malurl.8	domain	159.8	-	.0000	.168	.000	PL	.0066
malurl.9	domain	637.7	-	.0005	.105	.662	US	.0173
malurl.10	domain	1.7	-	.0303	.100	.562	US	.0009
mixed.1	ip	192.3	-	.0024	-	.068	PL	.0145
mixed.2	ip	140.2	31.3h	.0000	-	.001	PL	.0075
other.12	domain	23.5	25.1h	.0000	.411	.007	PL	.0080
other.6	domain	3.6	-	.0000	.800	.469	US	.0108
phish.1	domain	2128.5	-	.0014	.607	.576	US	.0567
phish.2	domain	139.4	-	.0012	.554	.342	US	.0272
phish.3	domain	1090.8	-	.0028	.635	.469	US	.0651
spam.1	domain	11.3	26.2h	.0000	.996	.500	PL	.0014
vulnerable.1	ip	2829.8	38.9h	.0000	.085	.055	PL	.0000
vulnerable.2	ip	401.1	35.0h	.0000	.075	.036	PL	.0000
vulnerable.3	ip	283.4	35.4h	.0000	.067	.023	PL	.0000
vulnerable.5	ip	203.5	35.9h	.0000	.201	.032	PL	.0000
vulnerable.6	ip	5043.0	33.9h	.0000	.056	.005	PL	.0000
vulnerable.7	ip	11208.3	34.6h	.0000	.385	.001	PL	.0000
vulnerable.8	ip	7.5	42.4h	.0000	.654	.000	PL	.0000

4.2.1 Rate

Evaluated period: 2015-07-01 – 2015-07-20, except for *cert-pl.cuckoo* and *cert-pl.zeus-p2p*: 2014-01-01 – 2014-01-10.

Amount of data provided by individual sources varies greatly, even within a single category. The biggest data providers are sinkholes, which were observed to report up to 1.9 million of unique IP addresses per day. Compared to IP addresses, a much lower number of domain names are reported, which is not surprising given the different roles these two indicators play in attacks.

4.2.2 Delay

Evaluated period: 2015-07-01 – 2015-07-20, except for *cert-pl.zeus-p2p*: 2014-01-01 – 2014-01-10. Since not every source provide exact timestamps for its data, it was not possible to determine delay in many cases.

It can be observed that the majority of the sources send the data with a significant delay, exceeding 24 hours. Sources using stream transport, e.g. *cert-pl.sinkhole*, *cert-pl.zeus-p2p* can achieve low delay, even less than a second. However the biggest problem are sources that do not provide precise time of reported events at all – this occurs mostly for domain datasets.

4.2.3 False discovery rate

Evaluated period: 2015-07-01 – 2015-07-20, except for *cert-pl.cuckoo* and *cert-pl.zeus-p2p*: 2014-01-01 – 2014-01-10. Column name: **fpr**.

This rating component is tightly related to trustworthiness of a source. Most sources, especially ones providing IP address, manage to avoid obvious false positives. The main problem can be seen with domain sources, although in some cases it can be explained by the collection method. For example, *cert-pl.cuckoo* reports all domains contacted by malware, without attempt to filter out internet connectivity checks or other non-malicious connections.

4.2.4 Linkage

Evaluated period: 2015-07-01 – 2015-07-20. Not defined for sources with mixed categories. Column name: **link**.

In general, sources providing data on vulnerable or misconfigured (in case of amplifiers) infrastructure connect to many events from other sources. This effect might be related to the fact, that vulnerable infrastructure, e.g. home routers, might often be abused by criminals for multiple purposes or coincide with malware infections. Visualization of linkage between sources is presented in the figure [4.1](#).

4.2.5 Representativeness

Evaluated period: 2015-07-01 – 2015-07-20. Column name: **dm** (Wilcox's DM index), **cc** (most frequent country).

Proposed methods allows to easily distinguish between datasets filtered for a particular country. The index value below 0.1 is a strong indicator that the provider performs filtering on a per-country basis.

4.2.6 Utility rating

Evaluated period: 2013-01-01 – 2015-11-30.

This component might be considered a strong indicator of dataset quality. It allows to identify which datasets were most relevant for tasks related to incident response and malware research. In particular phishing datasets appear in results more often than other categories.

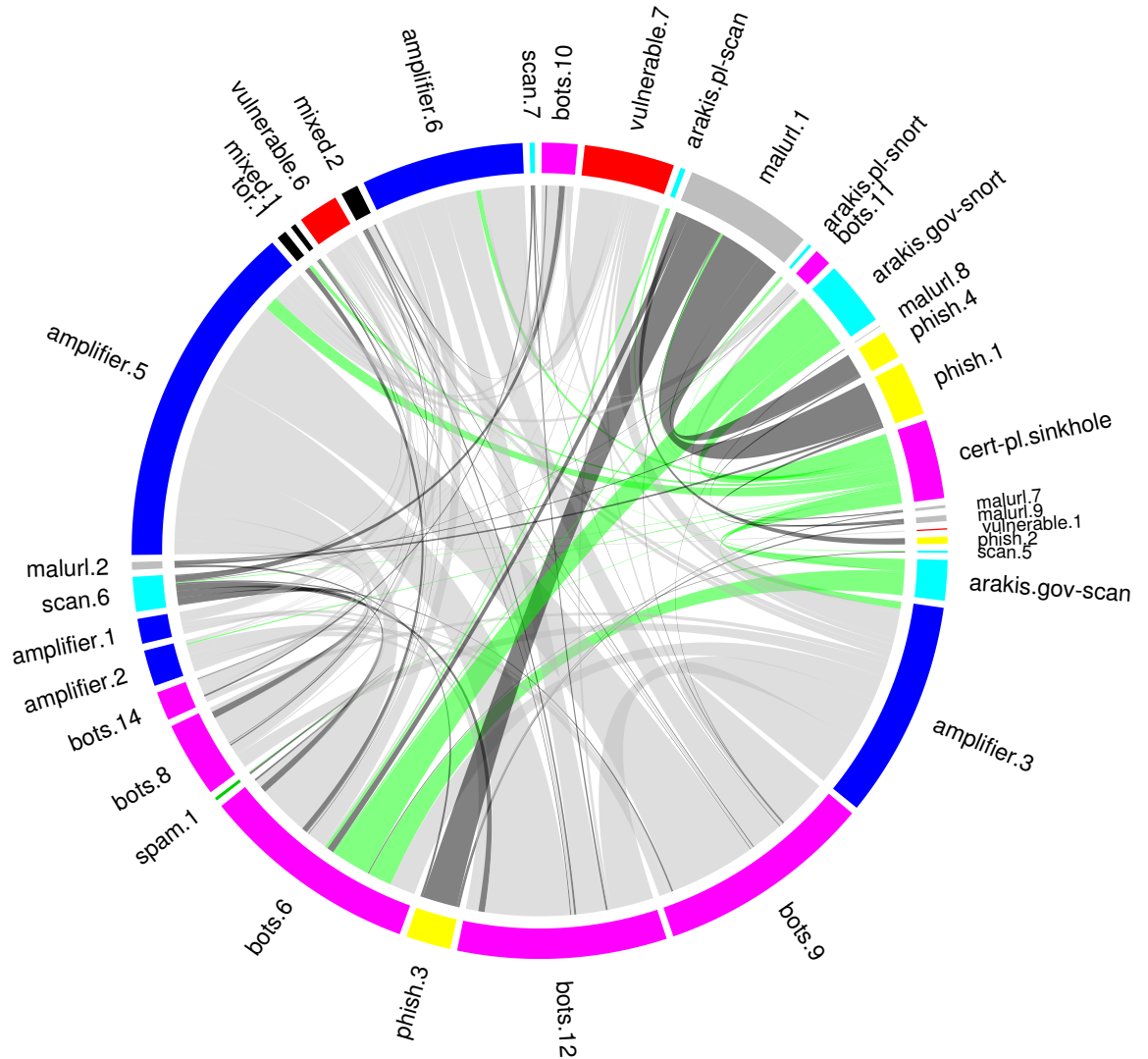


Figure 4.1: Visualization of linkage between sources in n6. Degree of an arc is proportional to the number of events in a source that were linked to events of a different category. Circle segment color corresponds to the category of a source (pink for bots, light blue for scanning, etc.). Relationships between sources are coded in their color as follows: **green**: all links with sources produced by CERT Polska (as opposed to third-party data feeds), **dark grey**: other links that do not connect to *amplifier* or *vulnerable* categories, **light grey**: remaining links.

Analysis of multiple large-scale that datasets was the core of work done in workpackage 2. By leveraging the gained experience, it was possible to propose 14 threat metrics that can be applied to represent both the severity of particular attacks and the global threat level. Proposed threat metrics describe concrete technical properties and were grouped into five related categories:

- Metrics describing *attack intensity* can be used to express the scale of attacks observed (section 5.1).
- Some of the metrics attempt to address the issue of establishing the accuracy of an incident report, i.e. the *report confidence* (section 5.2).
- A metric to express the *level of sophistication* of a DDoS attacker was introduced (section 5.3).
- Observing activity of potentially infected machines provide information on *impact* of a threat (section 5.4).
- Finally, *persistence* of threats can be measured as well (section 5.5).

5.1 Attack intensity

5.1.1 Darknet port scan count

The *darknet port scan count* measures the amount of port scan events observed during a certain period of time, showing a global activity of scans. It is normalized by the size of darknet metric. Formally, the *darknet port scan count* is defined as the number of port scan events $\#scans(D)$ in a darknet address space D (IPs), during a unit of time, Δt :

$$DPSC = \frac{\#scans(D)}{\Delta t}.$$

For example, DPSC is 74424/day (12.4% of the all events) in a darknet traffic data measured at a /18 network in Oct 2015, and 1478/day (2%) in Oct 2006.

5.1.2 Darknet backscatter intensity

The *darknet backscatter intensity* approximates a global level of backscatter activities triggered by (D)DoS events. Formally, the *darknet backscatter intensity* is defined as the number of backscatter events $\#backscatters(D)$ in a darknet address D , during a unit of time, Δt :

$$DBI = \frac{\#backscatters(D)}{\Delta t}.$$

DBI is 9141/day (1.5% of the all events) in a darknet traffic data measured at a /18 network in Oct 2015 and 6200/day (11.4%) in Oct 2006.

5.1.3 Spam campaign count

The *spam campaign count* measures the amount of campaigns that are concurrently executed during a certain period of time. Monitoring concurrent spam campaigns conveys both the number of spammers and their level of activity. Formally, the *spam campaign count* is defined as the number of monitored spam campaigns, $\#campaigns$, during a unit of time, Δt :

$$SCC = \frac{\#campaigns}{\Delta t}.$$

In the current implementation a spam campaign is defined as a set of similar emails sent in a limited period of time. As described in D2.1 Section 2.3.3, we identify these spam campaigns using fuzzy hashing and we found spam campaigns lasting several months.

An average SCC is 118 ± 29 /day in a spam data set collected in Jan 2012.

5.1.4 Global spam intensity

The *global spam intensity* metric summaries the number of spam emails received on a single email address over time. Thereby, this metric depicts the intensity of spam attacks from the point of view of the targets. Formally, the *global spam intensity* is defined as the number of spam emails, $\#spams(E)$, sent to a unique email address E , during a unit of time, Δt :

$$GSI = \frac{\#spams(E)}{\Delta t}.$$

An average GSI is 602 ± 76 /day in Jan 2012.

5.1.5 Spam campaign intensity

The *spam campaign intensity* metric quantify the level of aggressiveness of a spam campaign in terms of email volume. Formally, this metric is the number of spam emails, $\#spams(C)$, sent for a certain campaign C , during a unit of time, Δt :

$$SCI = \frac{\#spams(C)}{\Delta t}.$$

An average SCI is 4.1 ± 0.5 /day per campaign over top 100 campaigns on Jan 2012.

5.1.6 DDoS attack intensity

DDoS *attack intensity* is a commonly recognized metric for characterizing the capability of DDoS attacks, in particular volumetric ones. It generally refers to the amount of traffic per time unit coming to the attack target or victim. As the traffic can be classified into different granularities, *e.g.*, bytes, packets, or even HTTP requests for web servers, the corresponding metrics can be bits-per-second, packets-per-second, and so forth. It can be simply represented as follows,

$$DAI = \frac{\#traffic(V)}{\Delta t}.$$

where $\#traffic(V)$ is the traffic going to victim V , Δt is the time unit. According to Arbor networks, the largest reported attack in 2014 has already surpassed 400 Gb/s.

5.2 Report confidence

5.2.1 Confidence of alerts generated by multiple backbone anomaly detectors

One strategy to raise the confidence of backbone anomaly detection is to combine multiple anomaly detection modules based on different theoretical backgrounds. Each anomaly detector outputs a binary value telling if the traffic is anomalous or not. We can estimate the report confidence for each vote by running the corresponding detector with several parameter sets and measuring the variability of its output (*i.e.*, parameter sensitivity). In the following we refer to a set of alarms reporting the same anomaly and detected by different detectors with multiple configurations as a community.

The confidence score φ of a detector d for a community c is defined as: $\varphi_d(c) = \phi_d(c)/T_d$ where T_d is the total number of configurations with the detector d and $\phi_d(c)$ is the number of these configurations that reports at least one alarm belonging to the community c . The confidence score is

a continuous value that ranges $[0,1]$, 0 means that the detector does not report alarm for this community whereas 1 means that all configurations of the detector identify the community.

Now, we define three types of report confidence (average, minimum, and maximum) relative to a given community c in a value $\mu(c)$ as follows:

- Average report confidence: $\mu_a(c) = \frac{1}{L} \sum_{i=1}^L \varphi_i(c)$ where L is the number of detectors.
- Minimum report confidence: $\mu_{min}(c) = \min_i(\varphi_i(c))$
- Maximum report confidence: $\mu_{max}(c) = \max_i(\varphi_i(c))$

Consequently, we reliably consider a given community c as anomalous if $\mu(c) > \mu_{th}$, $\mu_{th} = 0.5$.

5.2.2 Phishing Likelihood Calculator

To deal with phishing attacks, a heuristics-based detection method has begun to garner attention. 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.

This approach was used to develop the Phishing Likelihood Calculator (PLC), that provides the likelihood of the phishing on output: 0 means legitimate, 1 is phishing, and 0.5 is a threshold. If PLC outputs 0.6 for the likelihood of a website A, and 0.9 for a website B, both websites would be labeled as phishing. However, the confidence level might be differ between the website A and B; the website B is more likely to a phishing site rather than the website A.

PLC was designed to provide the confidence rating described above to a policy decision point. The implementation utilized n6 formats and its confidence level as follows:

The confidence level was categorized into *low*, *medium*, and *high*. In the case of the phishing sites, confidence level *high* means the likelihood is close to 1, determinately phishing.

5.3 Level of sophistication

5.3.1 Attack mixture

The *attack mixture* measures the sophistication of a combined attack by counting the number of different attack techniques used simultaneously

```

{
  "address": [
    {
      "ip": "93.184.216.34",
      "asn": 15133
    }
  ],
  "category" : "phish",
  "source": "plc",
  "url": "http://www.example.com",
  "confidence": "high"
}

```

Figure 5.1: Response message from the analysis module to the PDP

against a target, *i.e.*, a single host or even a subnetwork:

$$AM = \#attack_types$$

By different attack techniques is meant the fact that attack flows can be broken down by categories (*flood, amplification/reflection, state-exhaustion, application-layer*), commonly shared among DDoS mitigation product vendors. These categories of DDoS attacks may even be refined into finer-grained categories, if applicable.

Annual threat reports often consider the combination of different attack types to be a sign of attack sophistication [4, 6, 24, 37]. Their reports usually include some illustrating examples where different attack techniques are used simultaneously in order to confuse the target or to bypass part of the countermeasures. According to the above-mentioned metric, a non-sophisticated attack would score 1, while DDoS attacks involving more than one type of attack flows would necessary be graded with $AM > 1$.

5.4 Impact

5.4.1 Addresses involved in an anomaly

The number of addresses identified in an anomaly reveals either the population of IP addresses that are affected by the anomaly or the source of the anomalous traffic. This information is essential to evaluate the impact of the anomaly.

In practice the definition of anomaly is closely related to the implementation of the anomaly detector. Our implementation relies on different detectors (see D2.1) thus the meaning of the identified anomalies can vary.

Nonetheless, the impact of the anomaly is generally given by the number of involved IP addresses.

5.4.2 Addresses involved in scan

The number of unique IP addresses appeared in darknet scan events indicates a global level of scanning activity. A scan event is defined as a source IP address sending probes at least N darknet IP addresses during a given time unit. The limitation of this metric is that source IP addresses can be forged by originators, though the definition of scans ignores random isolated probes. For example, the number of IP addresses involved in scan ($N=5$) is 1418 and 34789 at a /18 darknet for 24 hours in Oct 2006 and 2015, respectively.

5.4.3 Addresses involved in spam campaign

Similarly to anomalies, the number of addresses found in spam campaigns provides a good indicator of the campaign impact. Here we differentiate two types of addresses, the destination email addresses reveal the coverage of the campaign, while the number of source IP addresses convey the resources employed to initiate the campaign. Both quantities are crucial to understand the extend of spam campaigns. For example, the biggest spam campaign appeared in a spam data in 2012 consists of 2009 unique source IP addresses.

5.4.4 Distribution of observed suspicious domain queries

The *Distribution of observed suspicious domain queries* metric represents the distribution of IP addresses which query malicious domain name. Each malicious domain names are classified by malicious behavior, thus this metric indicates scale of a particular spam/phishing/botnet campaign in the global Internet.

5.5 Persistence

5.5.1 Spam campaign duration

The metric proposed to measure the intensity of spam campaigns are fundamental to uncover aggressive campaigns sending large amount of emails in a short period of time. Stealthy spam campaigns on the contrary emit spam emails at a lower rate but last significantly longer. Consequently, the duration of spam campaigns reveal another crucial characteristic of spam

campaigns. For example, the longest duration observed in the top 100 campaigns in 2011-2013 is about 450 days, though the majority of them are 50-100 days.

5.5.2 Lifetime of observed suspicious domain queries

The *Lifetime of observed suspicious domain queries* metric represents the lifetime of a specific malicious domain names from the perspective of requests issued by affected users. It is defined as the amount of time during which clients query the given domain and the domain is part of a malware infrastructure, i.e. returned DNS records point to C&C servers, phishing websites, etc. This metric cannot be applied to domains associated with compromised legitimate sites, where the malicious content is installed in a “parasitic” manner.

The threat analysis research conducted by members of the NECOMA consortium is mainly described in Deliverable D2.1. Research that has been carried on after the deliverable is, however, reported in this chapter.

- Section 6.1 reports the implementation of sensors meant to detect malicious cloud outbound traffic, *i.e.* cloud traffic originating from infected virtual machines and targeting external hosts. DNS data is captured at the boundary of the cloud infrastructure and analyzed to produce a list of suspicious domains, IP addresses and a score associated to them.
- Section 6.2 investigates machine learning and text mining approaches to systematically parse Common Vulnerability and Exposures (CVE) descriptions and rate the impact of the reported vulnerability.
- Section 6.3 studies the relationship between web users eye movements and their competence to identify phishing websites.
- Section 6.4 presents a study on HTTPS phishing websites and proposes a detection method based on the visual characteristics of phishing websites.

6.1 DNS-based Detection of Malicious Cloud Outbound Traffic

This Section presents the results of the implementation of sensors meant to detect malicious cloud outbound traffic. The intent is to detect malicious traffic originating from within a cloud infrastructure and targeting external hosts, later to be mitigated. For testing purposes, traffic from real DNS servers was captured and then analyzed later with the implemented sensor.

The DNS servers where the traffic probes were placed, were part of a real enterprise cloud infrastructure for which a simplified architecture outline can be seen in Fig. 6.1.

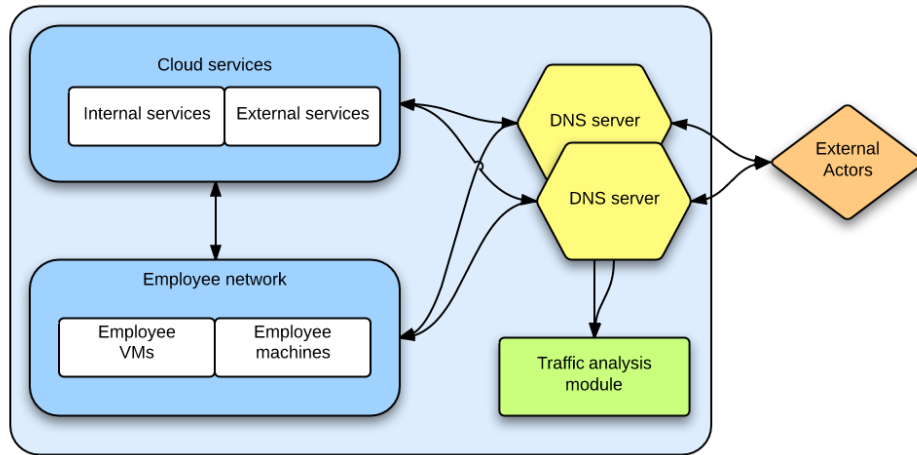


Figure 6.1: Enterprise cloud architecture used to capture test traffic.

The main points of interest are the two DNS servers through which all the DNS traffic goes, whether it is outgoing or incoming. The cloud also contains numerous devices, which can be divided into two main categories: the employee network and the cloud services. The employee network is an enterprise network encompassing all the employee machines, as well as, their personal, exposed VMs. The cloud services involve servers and services that are exposed to external users, as well as enterprise services meant for internal usage only. Although many of the services are restricted to internal users (employees), they can be accessed from outside the cloud network.

6.1.1 Previous studies

Mitigating malicious traffic in real time is challenging for companies targeted by distributed denial-of-service attacks, but the task gets even more complicated when the attack comes from legitimate services that cannot be easily filtered, such as cloud service providers. This kind of threat requires the cloud service providers to take a more direct approach to detecting and mitigating compromises that turn their hosted systems into an attack source. The problem is that any sort of filtering could cause a disruption of the customers' services, which is an unacceptable risk for the cloud providers. So far, there has been very few, if any attempts to combat such threats in this particular scenario. Thus, the approach we undertook rather leverages the

best techniques already proposed, regarding different kinds of investigations (among others [8, 20, 42, 47, 26, 25]) meant for other environments, and tailors them to this setting. Also many of the proposed techniques were not implemented in a standalone prototype yet. Therefore, we not only face the challenge of dealing with an unusual scenario, but also attempt to best combine a set of untested (in real environment) techniques.

6.1.2 Proposal Design

The goal is to effectively capture, analyse and detect any kind of traffic that might indicate that the DDoS attack is being carried out from within the cloud and is targeted at an external party. To reach the goal we designed traffic analysis sensors that are deployed on outbound cloud communication traffic nodes.

The traffic analysis sensor consists of a set of analysis modules that analyse two types of outbound traffic:

- DNS traffic, looking for certain patterns and features that lead to the identification of Fast Flux Service Networks (FFSN), and in the end, domains and IP addresses that could potentially belong to a botnet used for malicious purposes such as DDoS attacks, malware distributions, etc.
- NetFlow records of traffic passing through a router at the border of the cloud infrastructure, looking for botnets that behave similarly to a P2P network within the cloud infrastructure.

6.1.2.1 DNS traffic analysis module

Each module focuses on the analyses of certain features of the DNS data, and produces a list of suspicious domains, IP addresses and a score associated with them. Afterwards, an orchestration component implements an algorithm that takes into account the output score of each of the modules and computes the resulting likelihood associated with the domains and IP addresses. Besides the DNS data, which is the main source for the component, the component also takes public available blacklists and whitelists as input.

The analysis for suspicious activities detection is done in 5 groups, depending on the type of behaviour the sensor will test:

Time based analysis group

This group will search for patterns regarding the timestamp of the different queries and responses to the servers. The drawback of this group is that it needs to analyse the traffic of several days (no less than 3) in order to work properly. One of the functionalities is the

analysis of the temporal distribution of timestamps for the queried domains over a period of time. In an anomalous behaviour, the domains are queried a lot for a short period of time, and after that, never queried again. In a normal behaviour, time intervals where domains are queried are more equally distributed along the period of time of the experiment. Several other ways of detection are applied, such as identifying sudden changes over time of the number of requests for a domain.

TTL based analysis group

This group will search suspicious behaviour regarding the TTL (Time to Live) field in the request. Lower values are used for benign servers to hold a high availability type of service; unfortunately, it is often used by attackers to create disposable domain names for malware to be more resistant to blacklisting. In an anomalous behaviour, the FFSN (Fast-flux Service Network) use a low TTL combined with a constantly growing DNS answers list (i.e., distinct IP addresses), whereas in a normal behaviour, the TTL is set to higher values. Also malicious domains tend to have a more scattered pattern of TTL values and change constantly over time.

Domain name based analysis group

Attackers bypass domain blacklisting tools by creating new domains automatically using DGAs (Domain Generation Algorithm). These generators usually have a pattern that our tools will search to determine whether the domain is suspicious or not. Additionally the tool also checks whether the responding domain names are blacklisted or not, by using Google safe browsing API. The list of domains and IP addresses is checked against the Google Safe Browsing database of known malware and phishing sites.

DNS answer based analysis group

Domains like Google balance the load of their servers by resolving a different IP address every time the domain is queried in a round robin fashion. Attackers, however, use this technique to resolve malicious domains to compromised computers all over the world, so this group will search for spatial inconsistencies in the queries (resolved IP addresses in different countries).

DDoS Amplification attack group

This tool checks if there has been an attempt to launch an Amplification DDoS attack. To achieve the amplification effect, the attacker issues a DNS request that he knows will prompt a very large response, taking advantage of the DNS protocol extension EDNS0 [44]. The attack uses a poorly configured DNS server. The DNS attacks exploit

name servers that allow open recursion. Recursion is a method of processing a DNS request in which a name server performs the request for a client by asking the authoritative name server for the name record. In DNS attacks, each attacking host uses the targeted name server's IP address as its source IP address rather than its own. The effect of spoofing IP addresses in this manner is that responses to DNS requests will be returned to the target rather than the spoofing hosts.

6.1.2.2 NetFlow traffic analysis module

The analysis of NetFlow data aims at identifying botnets by discovering anomalous behaviour in the network traffic. These observations may lead, for instance, to identify the hosts in the network that are part of a botnet, but also to the identification of a compromised network device and the C&C server that is sending commands to it.

Design architecture

Figure 6.2 depicts an overview of the main elements of a NetFlow-based sensor for botnet detection.

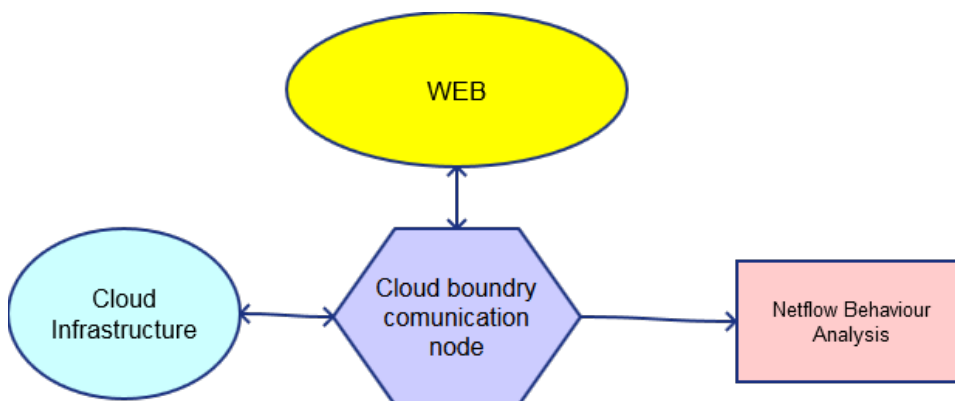


Figure 6.2: Architecture design

The analysis module is receiving the NetFlow data generated by the communication node located at the border of a cloud. This communication node might be a switch or a router that is mediating the incoming/outgoing traffic between the cloud internal hosts and the Internet. The NetFlow data is processed by the NetFlow Behaviour Analysis Module to detect anomalous behaviour that may lead to conclude the cloud infrastructure is being used by a C&C server and that the network device has been compromised.

Besides the analysis of the network behaviour represented by the NetFlow captured data, the sensor uses a list of known malicious domains, IP addresses and DNS servers in order to identify connections to C&C servers, malicious web servers for malware distribution or to detect DNS spoofing.

Process work flow

The whole process of NetFlow analysis consists of four main phases:

NetFlow traffic sniffing

This phase consists of the sniffing of the NetFlows in the monitored target infrastructure, either by hardware (a router) or by software (e.g. Softflowd¹).

Traffic capture and parsing

This phase consists on using the capture interface to feed the NetFlow information to the internal DB, so the sensors can use the data to monitor for botnets.

Behavioural analysis

This phase analyses NetFlow data. Botnets detected in this phase normally compromise a vulnerable router or switch device (usually not properly configured), giving the C&C server the control over the network to recruit all the hosts in the corresponding subnet to perform malicious activities.

P2P analysis

This phase analyses NetFlow data over a period of time for the identification of clusters of hosts with unusual high rates of inter-connections that simulate the behaviour of regular peer-to-peer networks but are actually an active botnet in disguise.

The sensors, belonging to both modules, are being constantly tested in different environments, thus, the functionalities of some of the components might change or be entirely discarded and replaced by different ones.

6.1.3 Experiments

The module analyses the pre-processed DNS traffic data captured on the DNS servers.

At first, the sensors worked on raw traffic captures in pcap format. This was highly ineffective and, because of the huge amount of data, caused performance issues for the prototypes. To improve the performance, an ad-hoc component was introduced to pre-process and extract in advance the data necessary for the analysis. It solved the performance issues produced by the volume of data coming from the DNS servers and allowed the storage of pre-analysis data.

The tests were carried out on traffic captured during two consecutive months: April 2015 and May 2015. The main focus was on detecting DNS

¹ Softflowd is a flow-based network traffic analyser, capable of Cisco NetFlow™ data export. Softflowd semi-statefully tracks traffic flows recorded by listening on a network interface or by reading a packet capture file. These flows may be reported via NetFlow™ to a collecting host or summarised within Softflowd itself.

6.1. DNS-BASED DETECTION OF MALICIOUS CLOUD OUTBOUND TRAFFIC

amplification attempts and Fast Flux domains, although additional brute force attacks and port scanning attempts that originated from other cloud services and targeted our infrastructure were also detected.

Table 6.1: DNS amplification attempts - April 2015

Attempts	Failed	Country	Victims ASN	URL
678 452	678 422	FR	Societe Francaise du Radiotelephone S.A	isc.org
209 725	205 553	NL	LeaseWeb B.V.	saveroads.ru
134 174	134 095	NL	TransIP B.V.	saveroads.ru
132 718	132 559	NL	TransIP B.V.	saveroads.ru
70 302	66 167	NL	Duocast B.V.	saveroads.ru
36 486	36 477	NL	TransIP B.V.	saveroads.ru
14 931	14 931	PL	Netia SA	isc.org
13 643	13 641			isc.org
11 643	11 643	FR	Free SAS	isc.org
10 529	10 460	IT	Seflow S.N.C. Di Marco Brame' and C.	saveroads.ru
94 14	94 13	FR	Bouygues Telecom S.A.	isc.org
83 20	83 20	FR	Free SAS	isc.org
81 08	81 02	US	SSASN2 - SECURED SERVERS LLC	saveroads.ru
79 67	79 66	FR	Free SAS	isc.org
79 23	79 22	FR	MANCHE TELECOM	isc.org
69 06	69 06	PT	TVCABO Autonomous System	isc.org
66 91	66 91	FR	Orange S.A.	isc.org
62 99	62 99	FR	Free SAS	isc.org
57 06	57 06	RO	Voxility S.R.L.	isc.org

6.1.3.1 DNS Amplification Report - April 2015

Table 6.1 shows attempts of utilizing our DNS servers to participate in an amplification attack. The table highlights the most prominent attempts.

By far, the most utilized domain was `isc.org` followed by `saveroads.ru`. An educated guess would be that so called “script kiddies” have been using the command found on a blog that appears very high in the results of a Google search for “dns amplifications attack”. Although, the modus operandi of the attackers seems a little bit random, by analyzing the traffic, a trend does stand out: high peaks of attacks are performed in non-working hours, and especially on vacation days, when the servers are not monitored.

¹<https://blog.cloudflare.com/deep-inside-a-dns-amplification-ddos-attack/>

Figure 6.3a shows a notorious chain of attacks during Easter holidays (2 and 3 of April), with over 100k attempts per hour at peak time.

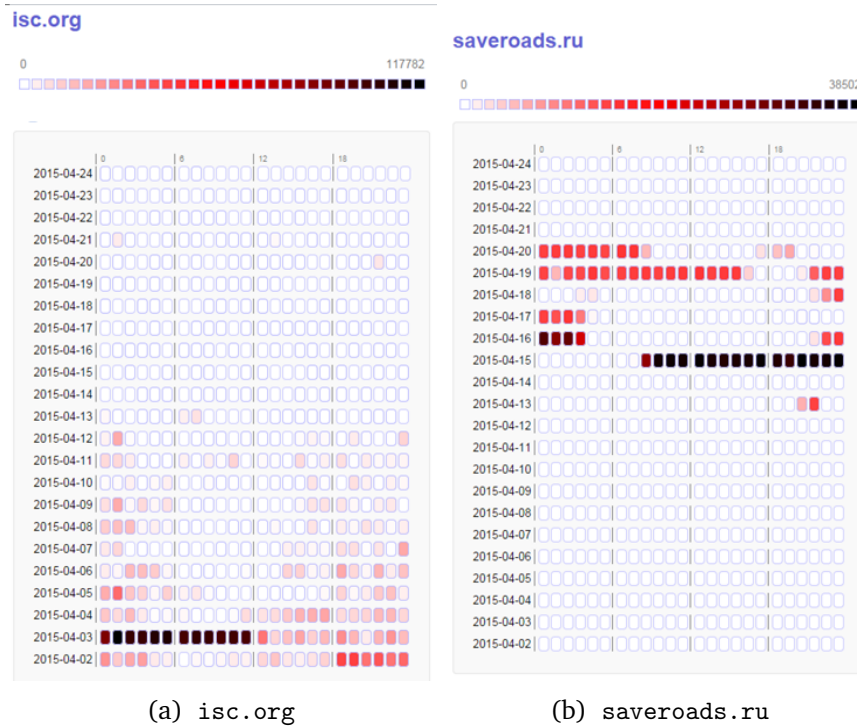


Figure 6.3: DNS amplification attempts - April 2015.

With `saveroads.ru` (cf. Fig. 6.3b), the pattern is not so obvious, but the attacks are focused and persistent when they occur.

6.1.3.2 DNS Amplification Report - May 2015

As seen in Table 6.2, May has been a rather calm month for DNS Amplification attacks compared to April, having only a tenth of the attack attempts, and `saveroads.ru` leading the URLs used this time.

The attackers using `isc.org` (cf. Fig. 6.4a) have been showing a clear pattern focusing the attacks on holidays, but for this month, only a few and scarce attacks were made towards the last days, mostly on a weekend.

The attackers using `saveroads.eu` (cf. Fig. 6.4b) have shown a very strong and focused activity this time. They chose to focus the attack on Sunday 3rd, lasting for about 16 hours, with a complete silence during the rest of the month.

6.1. DNS-BASED DETECTION OF MALICIOUS CLOUD OUTBOUND TRAFFIC

Table 6.2: DNS amplification attempts - May 2015

Attempts	Country	Victims ASN	URL
61 451	NL	Cyso Hosting B.V., Alkmaar, The Netherlands	saveroads.ru
27064	FR	Free SAS	isc.org
12959	US	Google Inc.	isc.org
10791	FR	Free SAS	isc.org
10081	DE	Kabel Deutschland Vertrieb und Service GmbH	isc.org
8013	FR	Societe Francaise du Radiotelephone S.A	isc.org
7717	NL	WorldStream	isc.org
4219	FR	Orange S.A.	isc.org
3617	FR	Free SAS	isc.org

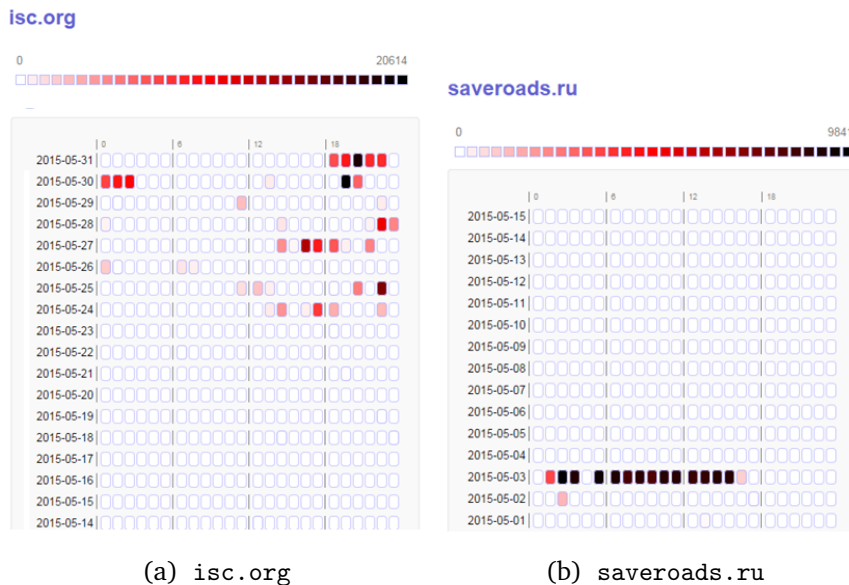


Figure 6.4: DNS amplification attempts - May 2015.

6.1.3.3 Brute force attacks - May 2015

This kind of attack has been occurring every single day, tens to thousands of times a day. The problem is that the attackers hide behind servers that provide static IPs. This is not bad by itself, but the users exploit the fact that they get a free IP they can exploit for port scanning or brute force attacks on servers, and then just leave that IP and change to another one.

Table 6.3: DNS amplification attempts - May 2015.

Source attack domain	Attempts
"amazonaws"	76210
"netbreeze"	65418
"mtc"	29263
"melbourne"	24820
"163data"	22537
"optonline"	18407
"secureserver"	16204
"spqnet"	12222
"telkom"	9025
"Other"	38840

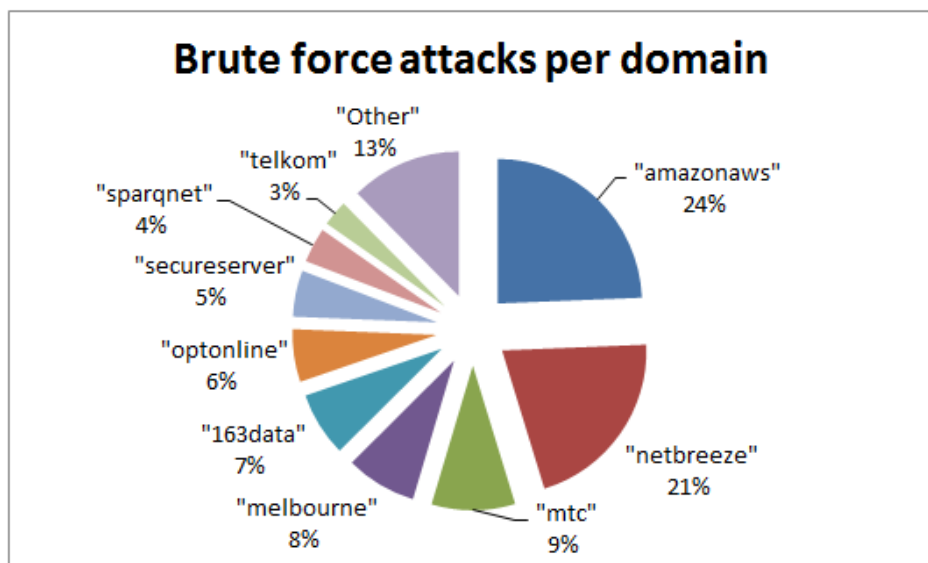


Figure 6.5: Brute Force attacks per domain - May 2015.

Table 6.3 and the pie chart in Fig. 6.5 show the amount and rate of attacks per server, the Amazon Cloud computing services being the biggest source, followed closely by Netbreeze (not the Swiss company bought by Microsoft, but the Russian domain netbreeze.ru).

6.1.4 Experience gained

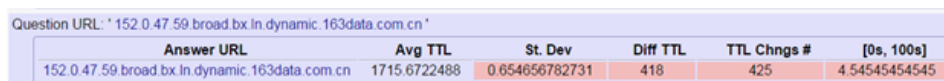
With the kind of traffic we monitor, the most reliable results come from the Time Based and TTL Based sensors, with the first one finding short lived domains and irregular patterns in the queries, and the second one finding anomalies or malicious ranges in the TTL values. Among the biggest offenders regarding those two sensors, we have found the following services:

- 163data.cn
- secureserver.net
- rev.poneytelecom.eu
- adsl-pool.jlccptt.net.cn
- ertlecom.ru

6.1.4.1 Time Based Sensor

The Time Based analysis can determine the short lived domains by checking if there are sudden peaks of traffic in a short amount of time. A domain is either popular or is not, but it is rare to have a domain with hundreds of thousands of queries within the same hour, and with the contiguous hour blocks having tens of requests at most.

6.1.4.2 TTL Based Sensor



Question URL: '152 0.47 59 broad.bx.in.dynamic.163data.com.cn'					
Answer URL	Avg TTL	St. Dev	Diff TTL	TTL Chngs #	[0s, 100s]
152 0.47 59 broad.bx.in.dynamic.163data.com.cn	1715.6722488	0.654656782731	418	425	4.54545454545

Figure 6.6: TTL analysis example.

The TTL Based sensor is also reliable for finding fast flux domains, due to the erratic fluctuation of the TTL values as opposed to benign domains that have a rather stable set of values. In the screenshot shown in Fig. 6.6, we can see the standard deviation with a high value (with zero being completely uniform, and 1 being chaotic). The Diff TTL column shows the number of unique TTL values this domain has had, for example: If the TTL values are [100, 230, 22], this column should show “3”. The next column shows the number of changes the TTL values had had with respect to the immediate former one, for example: if the values are [100, 40, 40, 200] the column should display “2”. The last column shows the percentage of TTL values that fall into the malicious range (0 seconds to 100 seconds), here showing 4.54%.

6.1.4.3 Scalability

The module is highly scalable and can work under heavy load, although the results are not obtained in real time. The tests were carried out in a corporate network and proven to be successful. However the module has to be configured to work in a specific environment as the thresholds may vary depending on the network structure and traffic volume.

6.1.4.4 Value added

The results obtained during tests are 100% accountable and the module proved to reliable when configured properly.

6.2 Text Mining Approach for Estimating the Vulnerability Score

In this research, we developed a method that can automatically estimate CVSS base-metrics, the score of vulnerability impact, by analyzing threat information written in natural language.

Most modern systems rely on software and software bugs often increase the risk that remote attackers can gain unauthorized access to such systems. Therefore, it is important to find suitable methods for managing vulnerabilities in order to protect society from these attacks. However, the number of vulnerabilities increases along with the amount of software, since bugs pervade every level of modern software [39]. Therefore, it is important to share information in order to form a knowledge base that can facilitate management of these vulnerabilities.

The National Vulnerability Database (NVD) is a popular knowledge base. It is composed of the Common Vulnerability and Exposures (CVE) dictionary of vulnerabilities [11] and the Common Vulnerability Scoring System (CVSS) [29], which estimates the impacts of those vulnerabilities.

However, the announcement of impacts can be delayed by a day or two following receipt of reports by the Computer Security Division of the National Institute of Standards and Technology (NIST), the organization that manages the NVD. This delay can potentially increase the risks, since it can delay the awareness of serious vulnerabilities.

Herein, we present a way to rapidly estimate the impacts predicted by the CVSS and we develop an automated method for estimating these impacts. We then use techniques for processing natural language to analyze the CVE descriptions, classify the documents and estimate CVSS base-metrics. We performed a preliminary experiment to determine a suitable estimation method; we compared several machine learning algorithms: the

naive Bayes classifier, latent Dirichlet allocation (LDA) [10], latent semantic indexing (LSI) [13], and supervised LDA (SLDA) [9]. As the training dataset, we considered approximately 60,000 vulnerabilities reported during the period January 2002 to December 2013 and we used 1,300 definitions reported during the period January 2014 to May 2014.

We also propose a new learning algorithm that introduces an annual parameter. Within the algorithm, the training dataset is separated by year, and a model is generated for each year. For each model, the algorithm assigns a weight that reflects the annual effects of the CVE documents. Our results indicate that this often improves the results.

6.2.1 Previous study

Earlier analyses of CVE documents (CVEs) focused on extracting topics from security vulnerability information. Neuhaus and Zimmerman [32] used CVEs published during the period 1999 to 2009 to analyze the trend of cyberthreats. They used the LDA [10] to classify the CVEs into 40 categories, such as cross-site scripting, SQL injection, and buffer overflows. Their results showed that by eliminating these vulnerabilities and making PHP more secure, the majority of the CVEs were fixed.

To the best of our knowledge, no past study attempted to analyze the CVEs in order to assess the possible impact of a given vulnerability, and thus the CVSS score was not calculated.

6.2.2 Estimating CVSS base-metrics from CVE documents

We decided to use the NVD [30] provided by the NIST. The NVD consists of the CVE-IDs and descriptions provided by MITRE ² and CVSS base-metrics calculated by the NIST team.

When extracting the CVE descriptions, we used the Porter stemming algorithm [35] to remove the inflectional endings from words. The list of the stopwords used in the algorithm was available online ³.

We considered various methods to determine a suitable algorithm for the analysis. An earlier analysis of CVEs [32] showed that LDA [10] is a feasible method. LDA is a probabilistic model for extracting topics from a corpus of documents, and it uses dimensional reduction to determine the co-occurrence of textual patterns within the documents. The naive Bayes classifier (NBC), a common method for classifying texts, is also feasible. We attempted to use it with the multivariate Bernoulli model. For the dimension reduction, we also considered the LSI [13]. It should be noted that the NBC, LSI, and LDA each use unsupervised learning. We also considered using SLDA [9] to deal with the labeled documents. Since the NVD can

²The MITRE Cooperation: <http://www.mitre.org>

³stemming: <https://pypi.python.org/pypi/stemming>

Table 6.4: Preliminary analysis of the performance of our method for estimating CVSS base-metrics

	Category	#	NBC	LSI	LDA	SLDA
AV	LOCAL	115	0.607	0.086	-	0.621
	ADJACENT NETWORK	41	-	0.067	-	-
	NETWORK	1158	0.962	0.609	0.981	0.962
AC	HIGH	50	-	0.084	0.069	-
	MEDIUM	624	0.650	0.456	0.644	0.676
	LOW	640	0.715	0.124	0.009	0.718
AU	MULTIPLE INSTANCES	5	-	-	-	-
	SINGLE INSTANCE	239	0.371	-	-	0.451
	NONE	1070	0.920	0.868	0.893	0.925
A	NONE	499	0.814	0.204	0.116	0.823
	PARTIAL	459	0.659	0.485	0.492	0.623
	COMPLETE	356	0.667	0.005	0.033	0.693
C	NONE	480	0.755	0.041	0.555	0.792
	PARTIAL	565	0.742	0.573	0.266	0.708
	COMPLETE	269	0.570	0.007	0.065	0.624
I	NONE	420	0.697	0.048	0.035	0.751
	PARTIAL	639	0.761	0.630	0.617	0.741
	COMPLETE	255	0.576	-	0.025	0.622

be regarded as a set of labeled documents, we used SLDA to classify the vulnerability information along with the topics.

We then performed a preliminary evaluation with the aim of using text mining to estimate the impact score. For this evaluation, we used 10-fold cross-validation and determined the average performance. We used the f_1 measure, which is defined as follows:

$$f_1measure = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}. \quad (6.1)$$

The results are summarized in Table 6.4, where the first column denotes the factors of CVSS base-metrics, as follows:

- Access Vector (AV) denotes the place where the vulnerability is accessed. This information is categorized as *LOCAL*, *ADJACENT NETWORK*, or *NETWORK*.
- Access Complexity (AC) denotes the difficulty of the conditions required to exploit the vulnerability. This is categorized as *HIGH*, *MIDDLE*, or *LOW*.

- Authentication (AU) denotes the number of times that the attacker must authenticate in order to exploit the vulnerability. This is categorized as *MULTIPLE INSTANCES*, *SINGLE INSTANCE*, or *NONE*.
- Availability (A) denotes the impact on the availability of a system when the system is attacked. This is categorized as *NONE*, *PARTIAL*, or *COMPLETE*.
- Confidentiality (C) denotes the impact on the confidentiality of the data in a system when the system is attacked. This is categorized as *NONE*, *PARTIAL*, or *COMPLETE*.
- Integrity (I) denotes the impact on the integrity of a system when the system is attacked. This is categorized as *NONE*, *PARTIAL*, or *COMPLETE*.

The second column gives the category of each factor, and the third shows the number of the test datasets that were found to belong to each category. The remaining columns denote the f_1 measures for the NBC, LSI, LDA, and SLDA, respectively. The symbol – means that the f_1 measure could not be calculated because both the recall and the precision were zero. The training data were a set of CVEs published during the period 2012 to 2013 (CVE-2012 to CVE-2013), and the test dataset was a set of CVEs published during the period January to May 2014 (CVE-2014).

In the preliminary evaluation, we observed that the SLDA usually performed better than the other algorithms. We note that the dimension reduction hindered both the LSI and the LDA. We tried to classify the CVEs as High, Medium, or Low without regard to genre, but we note that the topics estimated from an unsupervised model may correspond to genres, if that is the dominant structure in the corpus [9].

6.2.3 Annual weight assignment

In this section, we briefly summarize a history of threat information in order to consider the annual effect of the CVEs.

According to a report by the SANS institute [41], the serious cyberthreats reported in the first decade of the 2000s targeted operating systems and their default installed services. For example, CVE-2001-0500 described the vulnerability in a Web service for Windows that allows remote attackers to penetrate, i.e., the CodeRed worms. CVE-2002-0649 describes the Slammer worm, which exploits Microsoft's SQL services; CVE-2003-0352 describes the Blaster worm, which exploits Microsoft's RPC services; and CVE-2003-0533 describes the Sasser worm, which exploits the Active Directory service; all of these targets belong to the family of Microsoft Windows operating systems. In addition, CVE-2002-0392 describes the Scalper worm, which

Table 6.5: Performance of the basic SLDA and the annual weight-assignment algorithms

	Category	#	SLDA	(linear)	(sigmoid)
AV	LOCAL	115	0.649	0.663	0.637
	ADJACENT NETWORK	41	-	-	-
	NETWORK	1158	0.964	0.960	0.960
AC	HIGH	50	-	-	-
	MEDIUM	624	0.609	0.636	0.669
	LOW	640	0.731	0.769	0.765
AU	MULTIPLE INSTANCES	5	-	-	-
	SINGLE INSTANCE	239	0.622	0.303	0.506
	NONE	1070	0.940	0.914	0.931
A	NONE	499	0.783	0.860	0.855
	PARTIAL	459	0.597	0.726	0.707
	COMPLETE	356	0.632	0.694	0.695
C	NONE	480	0.781	0.794	0.798
	PARTIAL	565	0.701	0.761	0.742
	COMPLETE	269	0.562	0.653	0.634
I	NONE	420	0.708	0.753	0.750
	PARTIAL	639	0.741	0.780	0.770
	COMPLETE	255	0.571	0.634	0.622

exploits an Apache web service; CVE-2001-0011 describes the Lion worm, which exploits a BIND DNS service; and CVE-2002-1337 describes a vulnerability in the Sendmail email services. The modus operandi for all of these attacks was a buffer overflow, in which the data stored in the stack area of the computer are overwritten. These data are not only read/write-able, but also executable. If these attacks are successful, the attacker can run an arbitrary program on the host computer. In order to mitigate the risks of a buffer overflow, source code validation tools [31, 19], compilers [45, 18], and CPU supports, such as eXecute Disable/No eXecute, have been developed.

Instead of operating systems and their default installed services, web browsers and applications became the primary target during the middle of the first decade of the 21st century. In 2007, the most critical vulnerability in a client computer was ActiveX, the browser extension of Microsoft's Internet Explorer. According to a report by AV-test [5], attackers tended to target Java, Adobe Flash, and PDF, all of which were available as browser extensions. The modus operandi for these attacks was heap spraying, rather than buffer overflows. In addition to client computers, host computers also suffered attacks, primarily SQL injection, script injection (cross-site scripting), and OS command injection.

We therefore considered that the annual effect of the CVEs is a feasible parameter for providing better estimates, and designed an algorithm to assign weights that reflect the annual effect.

The training data consisted of CVE-2002, \dots , CVE-2013, and the test data were CVE-2014. We calculated twelve sets of the parameters (ϕ_d, η_d) , and we used either a linear or a sigmoid function to assign weights for each parameter. The predicted value $Y_{predicted}$ can be calculated as

$$Y_{predicted} = \operatorname{argmax}(evaluated_value(Y_d)), \quad (6.2)$$

where $evaluated_value(Y_d)$, where document $d \in \text{CVE-2014}$, can be calculated as

$$evaluated_value(Y_d) = \sum_{k=2002}^{2013} \eta_{d,k} \cdot \phi_{d,k} \cdot \omega_k. \quad (6.3)$$

We calculated the annual parameter ω_d for a linear function (Equation 6.4) and for a sigmoid function (Equation 6.5):

$$\omega_k = \frac{(k - 2001)}{12} \quad (6.4)$$

$$\omega_k = \frac{1}{1 + e^{\text{gain}(-1+2(k-2001)/k)}} \quad (6.5)$$

where $k \in (2002, \dots, 2013)$. The results are summarized in Table 6.5, where the first column lists CVSS base-metrics, the second column lists the category, the third lists the number of test datasets classified into that category, the fourth denotes the f_1 measures for the SLDA, and the fifth and sixth columns list the performance obtained using the annual weight-assignment algorithm with linear and sigmoid function, respectively.

In many cases, our algorithms performed better than did the basic SLDA. Our algorithms gave better estimates of the Availability, Confidentiality, and Integrity. In the cases of Access Complexity and Access Vector, we observed that, in some categories, the SLDA performed better than our algorithms, and it did so in all categories in the case of Authentication. However, of the 18 categories, in 13 cases, our algorithm with the linear function performed better than the SLDA, and in 12 cases, this was true of our algorithm with the sigmoid function. Hence, we believe that our algorithms can improve the estimation performance.

6.3 Correlation between Phishing Identification and Eye Movement

It is often said that the eyes are the windows to the soul. If that is true, then it may also be inferred that looking at web users' eye movements could potentially reflect what they are actually thinking when they view websites.

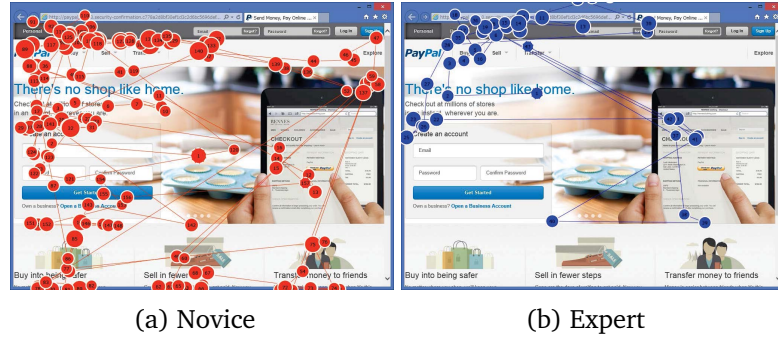


Figure 6.7: Eye-tracking in a phishing website

In this research, we conducted a set of experiments to analyze whether user intention in relation to assessing the credibility of a website can be extracted from eye movements.

According to the theory of mind [36], eye movements are different between users with a motivation to find any particular objects and users without a motivation. In the context of phishing identification, expert users may gain information from intentionally looking at the browser's address bar, and evaluating this information based on their knowledge. By contrast, novice users would look at the address bar with no particular motivation, simply because they are unable to evaluate this piece of information due to their lack of knowledge. Instead, novice users may intend to gain information from the website's contents even if the contents may not give any meaningful indications with regards to phishing identification.

It implies that analysis of eye movement may lead to estimating whether a user is likely to fall victim to phishing. Therefore, we assessed this hypothesis with a participant based experiment in which 23 participants had their eye movements monitored while taking a test where they needed to determine which websites are phish sites among twenty samples and provide their decision's criteria. Based on our experiment, it might be reasonable to consider that the analysis of eye movement is feasible for estimating the users' intention and their decision.

6.3.1 Previous studies

This section explains the background of this research. Section 6.3.1.1 shows the eye movement analysis carried out in *NECOMA* project, and section 6.3.1.2 describes the correlation between the users' implicit intention and eye movement.

6.3. CORRELATION BETWEEN PHISHING IDENTIFICATION AND EYE MOVEMENT

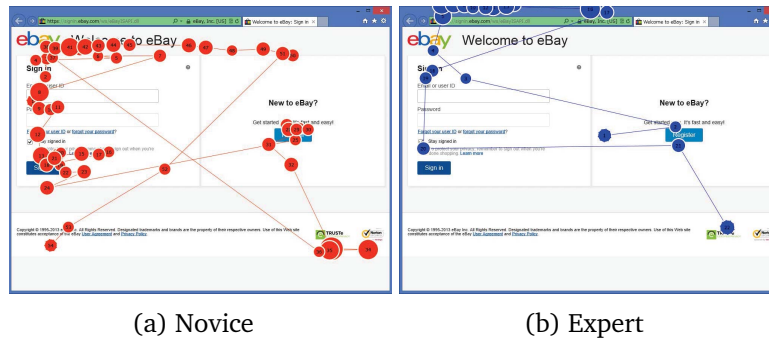


Figure 6.8: Eye-tracking in a legitimate website

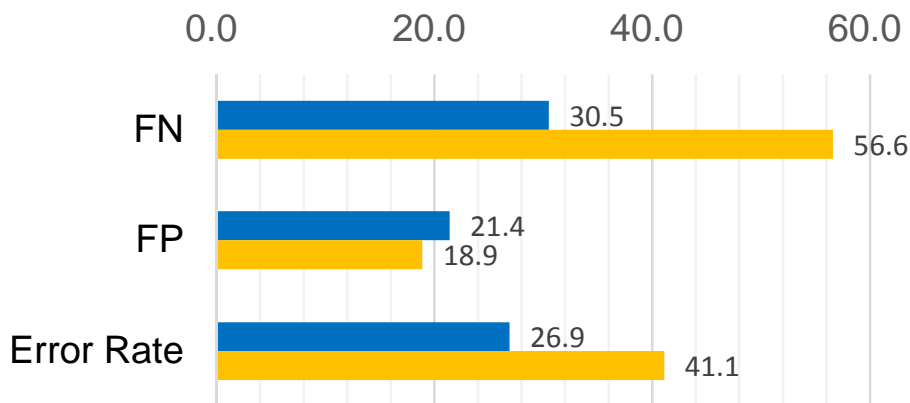


Figure 6.9: The average false positive, false negative and error rate for users that looked (blue) and did not look (orange) at the address bar

6.3.1.1 Preliminary analysis of eye movement

We recruited 23 participants to observe their eye movement. The volunteers were mainly males in their twenties. With their consent, their eye movements were recorded by our prepared eye-tracking device, Tobii TX300⁴. A separate calibration procedure for each participant was required.

Fig. 6.7 and 6.8 show typical eye-movement records on both phishing and legitimate website, for a novice and an expert respectively. Circles 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 during the experiment, as shown in Fig. 6.7a. 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. 6.8a. In contrast, an expert tends to evaluate the site's URL and/or the browser's SSL

⁴Tobii Technologies: <http://www.tobii.com>

indicator rather than the contents of the web page to judge the credibility of the sites, as shown in Fig. 6.7b and 6.8b.

We then analyzed the detection accuracy of participants who looked at the address bar and those who did not look. The results are shown in Fig. 6.9, where the blue bars denote the average rates for the participants that looked at the address bar of the browser, and the orange bars depict the average rates for the participants that did not look at the bar.

Out of the 331 times the bar was gazed, 89 (26.9%) misjudgments were observed. In the case of phishing websites, the participants looked at the bar 200 times in total, and 61 were misclassified (30.5% false negatives), i.e., a phishing website was labelled as legitimate. In the case of legitimate websites, participants looked at the bar 131 times in total, and 28 (21.4%) false positives occurred, i.e., a legitimate website was labelled 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 considered that our assumption, i.e., *checking the browser's address bar is beneficial to end users in making them aware of phishing*, is reasonable.

6.3.1.2 Correlation between users' implicit intention and eye movement

Prior studies [34, 27] exhibit the correlation between eye fixation and intention. The intention refers to an idea or plan of what a person is going to do. The theory of mind states that a person has a natural way to predict, represent and interpret intention expressed explicitly or implicitly [36]. A person expresses explicit intentions using different sequences of actions. For example, during an interaction, a person tends to express intention explicitly through speech, gesture, and facial expression. By contrast, implicit human intentions are subtle, vague and otherwise often difficult to interpret. Since the explicit expression alone may not be enough to understand the intention of a person, it is critical to also understand the implicit intention.

According to [34, 27], the implicit intentions can be identified through the following biomedical signals during a visual stimulus.

- *Navigational intention* refers to an idea or plan of a person to find any object in a visual input without a particular motivation.
- *Informational intention* refers to an idea or plan of a person to find a particular object of interest or to behave with a motivation.

6.3. CORRELATION BETWEEN PHISHING IDENTIFICATION AND EYE MOVEMENT

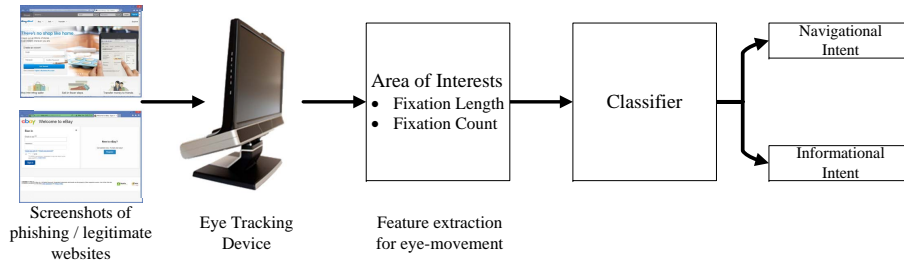


Figure 6.10: Block diagram of the experiment

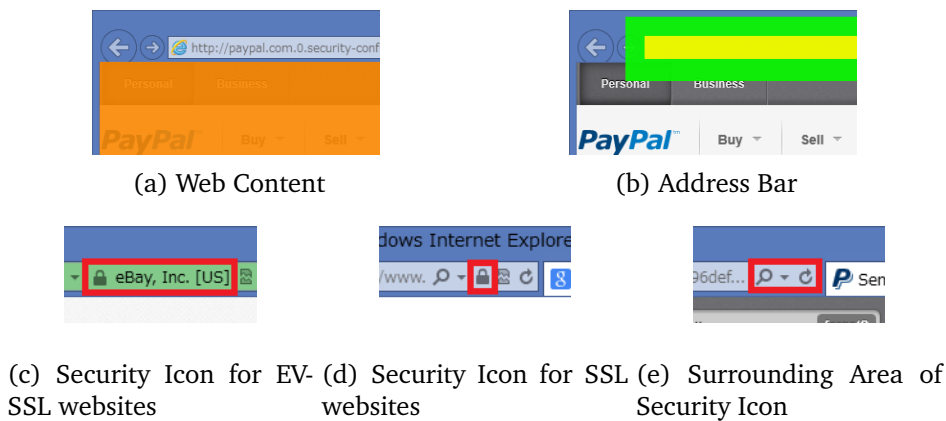


Figure 6.11: Definition of AoI

The authors have built classifiers for identifying intentions based on Support Vector Machine (SVM), and observed that there were positive correlations between the intentions and eye movement patterns.

6.3.2 Correlation between phishing identification and eye movement

Figure 6.10 shows the block diagram for recognizing the participants' intention. In our experiments, we employed a Tobii TX300 eye tracking system to analyze the eye movement data. With the users' consent, we measure their eye movements after we calibrated the eye tracking device for each participant. The participants were also shown several options to indicate their decision's criteria: "Content of Web page," "URL of the site," "Security Information of Browser," and "Other Reason." The participants were requested to mark all options that applied (multiple answers allowed), and described in details their reason when selecting the "Other Reason" option.

Table 6.6: Participants' recognition performance by eye movement analysis

Type of Intent		AER	AUC
Content of Web page	entire time period	32.4%	0.741
	initial ten seconds period	32.2%	0.759
URL of the site	entire time period	28.0%	0.741
	initial ten seconds period	27.8%	0.759
(removed noise)	entire time period	21.3%	0.890
	initial ten seconds period	19.7%	0.917
Security Information from browser (AoI of the address bar)	entire time period	14.5%	0.855
	initial ten seconds period	14.3%	0.855
(AoI of the padlock icon)	entire time period	13.5%	0.841
	initial ten seconds period	13.7%	0.809

6.3.3 Extraction of implicit intention

We hereafter examine the feasibility of extracting implicit intention from observing the user's eye movements. Based on the number and duration of fixations in each Area of Interest (AoI) of a given input stimulus image, we construct classifiers with SVM to differentiate the participant's implicit intention into navigational and informational intentions.

At first, we evaluate the following hypothesis: *the analysis of the eye movement can extract a user's intentions while watching web pages*. Since novices tend to assess credibility by the "Content of Web page," their eye movements would be different from the eye movements of experts. The feature vectors include the number and duration of fixations towards the web content AoI (as shown in Figure 6.11a). The objective variable is a binomial value that denotes whether the participant checked the "Content of Web page" option or not in our questionnaire. The average error rate (AER) was 32.4% and the area under the curve (AUC) was 0.741 as shown in Table 6.6. Additionally, we assumed that some participants would try to find some trustworthiness information as soon as they have begun browsing the websites. From this perspective, we also extracted the fixation count and duration within the first ten seconds. In this case, the AER was 32.2% and the AUC was 0.759. Hereafter, "initial ten seconds period" means the analysis of eye movement within the first ten seconds, and "entire time period" means the analysis of the entire time while making decision.

We wished for the participants to check the browser's address bar intentionally since the browser's attention on address bar gives trustworthy information such as the URL and security related information. The feature vectors are composed of the duration and number of fixations towards the address bar AoI (as shown in Figure 6.11b), and the objective variable is a binomial value that denotes whether the participant checked the "URL of

the site” option or not in our questionnaire. The AER was 28.0% and the AUC was 0.741 in the case of the entire time period. In the experiment, we found that several participants labeled “URL of the site” as their decision making criteria without actually gazing at the address bar. Even when we redefined the AoI in order to add the surrounding margins, as shown by the green rectangle in Figure 6.11b, their eye fixations towards the AoI still could not be accounted for. If we remove such falsely motivated decisions, the AER would be 21.3% and the AUC would be 0.890. Additionally, the margined AoI did not improve the performance: in the case of the entire time period, the AER was 22.1% and the AUC was 0.842.

We also assumed that some participants would choose to look at the address bar to find a security indicator. The feature vectors are the number and duration of fixations for that particular AoI, and the objective variable is a binomial value that denotes whether the participant checked the “Security information of browser” option or not in our questionnaire. The possible AoIs are the address bar and the padlock icon. The AER was 14.5% and the AUC was 0.855.

We defined the AoIs of the padlock icon, as shown in Figure 6.11c, 6.11d, and 6.11e, for an EV-SSL certificate where the AoI is around the name of the entity as well as the padlock icon, for an SSL certification, it is a rectangle around the padlock icon, and in the case of non-SSL websites, the AoI was a surrounding area for icons displayed in the address bar, respectively. For this last case, we assumed that some participants would check the nonexistence of the SSL certificates. In total, the AER was 13.5% and the AUC was 0.841.

We found that some participants tend not to check the “Security Information of Browser” option, even when the website displayed an SSL padlock icon. The predictor therefore indicates that all participants did not intentionally look at this AoI. We concluded that the AoIs were not as useful to construct a good predictor, however, the AER was 7.6% and the AUC was 0.785. In the case of the websites that displayed an EV-SSL padlock icon, the AER was 33.3% and the AUC was 0.711. When the websites had no certificate, the AER was 10.5% and the AUC was 0.775.

6.3.4 Estimation of participant’s likelihood to be victim

In this experiment, the hypothesis is that *the analysis of the eye movement can estimate whether or not a user is going to fall victim to phishing*. The feature vectors in this scenario are the number and duration of fixations towards the three types of AoIs (web content, address bar, and security icons). The objective variable is a binomial value that denotes whether the participant judged correctly or not.

The results are shown in Table 6.7. By using the combination of all types of AoIs, we observed that the AER was 20.7% and the AUC was 0.873, in the

Table 6.7: Estimation of participants who were going to be victims of phishing

Area of Interest		AER	AUC
Web Content	entire time period	24.8%	0.799
	initial ten seconds period	25.1%	0.818
Address Bar	entire time period	24.1%	0.820
	initial ten seconds period	25.7%	0.815
Security Icons	entire time period	24.4%	0.782
	initial ten seconds period	24.8%	0.761
All types of AoIs	entire time period	20.7%	0.873
	initial ten seconds period	21.1%	0.853

case of the entire time period. The lowest error rate was observed at 8.7% in Websites 10 and 15, and followed by Websites 1, 4, 8, and 9 with 13.0%. Since Website 10 is displayed in English, and since a significant number of the participants were non-native English speakers, we therefore assume that the participants had attempted to assess the website based on the address bar rather than the content.

Additionally, we performed a 10-fold cross validation with tuning parameters by grid search. The results showed that the AER was 29.3% in the case of the entire time period, and 30.8% in the case of the initial ten seconds period.

6.4 Anti-Phishing Visual Analysis to Mitigate Users' Excessive Trust in SSL/TLS

HTTPS websites are often considered safe by the users, due to the use of the SSL/TLS protocol. As a consequence phishing web pages delivered via this protocol benefit from that higher level of trust as well.

In this research work, we assessed the relevance of heuristics such as the certificate information, the SSL/TLS protocol version and cipher-suite chosen by the servers, in the identification of phishing websites. We concluded that they were not discriminant enough, due to the close profiles of phishing and legitimate sites. Moreover, considering phishing pages hosted on cloud service platform or hacked domains, we identified that the users could easily be fooled by the certificate presented, since it would belong to the rightful owner of the website.

Hence, we further examined HTTPS phishing websites hosted on hacked domains, in order to propose a detection method based on their visual identities. Indeed, the presence of a parasitic page on a domain is a disruption to the overall visual coherence of the original site. By designing an intelligent perception system responsible for extracting and comparing these divergent

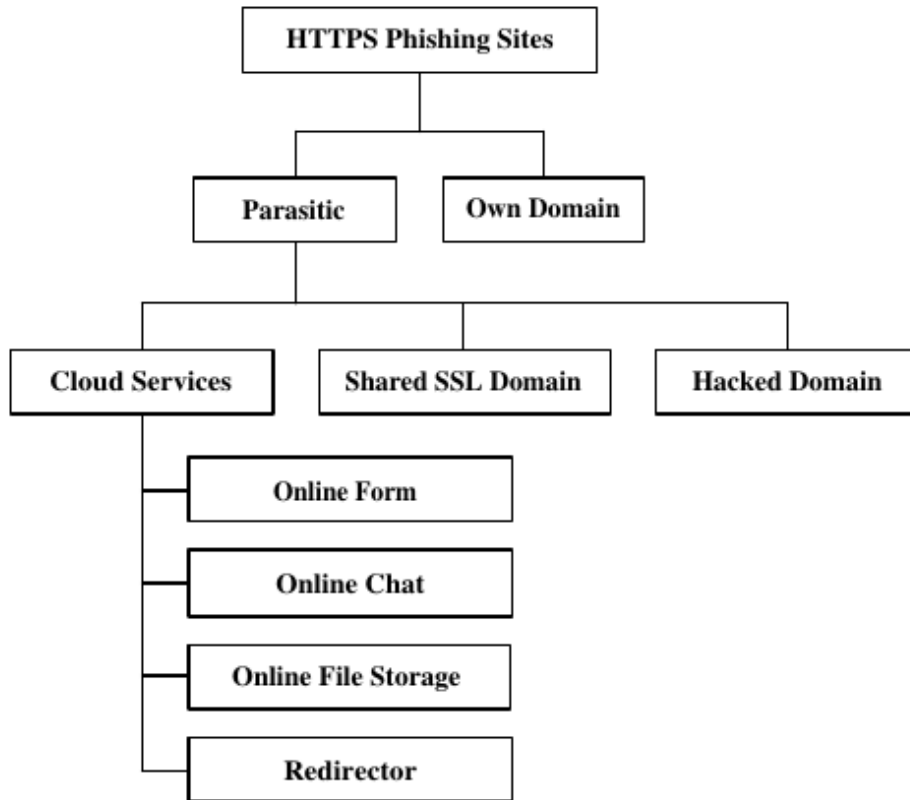


Figure 6.12: Taxonomy of HTTPS phishing sites.

renderings, we were able to spot phishing pages with an accuracy of 87% to 92%.

6.4.1 Previous studies

Two previous surveys, carried out in *NECOMA* project, prompted us to validate the discriminating effect of SSL/TLS handshake information to the detection of HTTPS phishing pages.

6.4.1.1 A Taxonomy of HTTPS Phishing Webpages

A taxonomy of HTTPS phishing webpages can be based on their hosting method. In this work, HTTPS phishing websites have been classified in two main categories: websites hosted on their own domains and websites taking advantage of third-parties (cloud services, hacked domains, shared SSL domains). These two categories were further refined as illustrated in Fig. 6.12.

Phishing websites hosted on their own domain are identified using the following indicators:

- The reported phishing URL is a domain with no subfolder
- The domain name looks clearly incriminating (similarity with legitimate ones, presence of security-related keywords, well-crafted sub-domains)
- No other uses for the domain were found in search engines or archive.org

In that case, the SSL/TLS certificate presented is provided by the attacker, incurring further financial cost.

Parasitic sites are described as being hosted on a domain that does not belong to the phisher: a shared SSL domain, a cloud services platform or a hacked domain. That domain is most of the time shared with other legitimate users: all of them will present the certificate of the owner of the platform. Such method is usually less costly to the attackers, except for the one that requires pre-emptive compromise of an existing domain (*hacked domain*).

6.4.1.2 Security Assessment of SSL/TLS Servers

The second work on which we based our present research deals with the assessment of SSL/TLS servers [38]. In this work, reported in Deliverable D2.1, two scores were given to a particular server according to specific criteria:

- the first group of criteria is relative to the information contained in the certificate, *i.e.* the presence of suspicious entries in the fields (see Table 6.8).
- the second group is relative to the handshake information of an SSL/TLS server, specifically the protocol and cipher-suite chosen by the server. These parameters were classified as being secure, risky or insecure, according to known flaws and vulnerabilities.

6.4.2 HTTPS Phishing Webpages: Building upon a False Sense of Security

In order to highlight the fact that SSL/TLS features are not sufficiently discriminant in the fight against phishing, we first analyzed phishing samples based on our previously reported work on assessing the security of SSL/TLS servers, to see where these phishing webpages stand. Indeed, we assume that phishing webpages may be equipped with SSL/TLS implementations in order to fend off suspicions from users.

6.4. ANTI-PHISHING VISUAL ANALYSIS TO MITIGATE USERS' EXCESSIVE TRUST IN SSL/TLS

Table 6.8: Suspicious values in the certificate.

C (Country)	O (Organization)	OU (Organizational Unit)	S (State/province)	CN (Common Name)
XY, NON-STRING-VALUE, single/double quotation	SomeState, Some-province, SomeOrganization, MyCompany, self-signed, 127.0.0.1, any compromised CA or cheap reseller CA	single/double quotation, Single dot, SomeState, Some-province, SomeOrganizationUnit, Division, section, self-signed, 127.0.0.1, any compromised CA or cheap reseller CA	SomeState, Someprovince, SomeState, Select one, Default, default	localhost.local-domain, 127.0.0.1

6.4.2.1 Datasets

The datasets used in this work are as follows:

- 1,213 legitimate certificates retrieved by connecting to Alexa Top 3,000 websites [1]: Legit_cert
- 1,170 phishing certificates retrieved by connecting to websites identified in Phishtank database [33]: Phish_cert
- SSL/TLS protocol and cipher-suite information of 103 online phishing websites: Phish_SSL
- SSL/TLS protocol and cipher-suite information of 102 online legitimate websites randomly selected among Alexa Top 3000 websites: Legit_SSL

The last two datasets have been obtained using the backward compatible Firefox configuration to contact the servers [2].

6.4.2.2 Analysis

We categorized websites according to the taxonomy previously described. This classification has its importance, since we assume that malicious websites are less likely to present properly filled certificates. Due to the fact that a certificate is shared, the information presented is not manipulated by malicious parties and comes directly from the administrators of the platform,

not from the cybercriminals; thus, it cannot be used in order to gain knowledge regarding the status of the website, even if it is qualified as insecure by the certificate module.

When connecting to a website, we extracted informations in the meta tag and searched for keywords indicating the use of a cloud service, such as : online, form, shared, storage, cloud, etc. On the other hand, we matched the Common Name (CN) in the retrieved certificate against a known list of shared certificates CNs, and identified the use of wildcard certificates. These steps allowed us to single out websites presenting shared certificates, the others being considered as presenting their own certificates.

The research on the assessment of SSL/TLS servers [38] identified as malicious certificates with suspicious values (SomeCity, SomeState, etc.) or those that lacked one of the following fields:

- Common Name (CN)
- Organizational Unit (OU)
- Organization (O)
- Country (C)
- State (S)

However, an analysis of Legit_cert dataset showed that even secure certificates do not always include these 5 fields at once. 90% of the dataset is constituted of certificates presenting a subset of 2 or 3 of these fields. While Common Name is always provided, along with Organization or Organizational Unit, Country and State fields are most of the time left blank. We adapted the script in order to include these use cases, identifying as insecure a certificate presenting suspicious values or less than two of the aforementioned fields. The results of this reviewed implementation are presented in Table 6.9.

Besides, we only slightly modified the script provided in [38] in order to include the new cipher-suites we encountered during the collection of Legit_SSL and Phish_SSL. We relied on known flaws and vulnerabilities associated with a particular protocol version or cipher-suite to rate the servers. The results obtained after the analysis of Phish_SSL and Legit_SSL are presented in Table 6.10 and Table 6.11.

6.4.2.3 Results

From the results in Table 6.9 obtained via an analysis of Phish_SSL, only 1.90% of phishing websites have been identified as being insecure after evaluating their certificates. Among these insecure certificates, 72.72% are own

6.4. ANTI-PHISHING VISUAL ANALYSIS TO MITIGATE USERS' EXCESSIVE TRUST IN SSL/TLS

Table 6.9: Repartition of HTTPS phishing certificates according to their types (Total of 1,170 certificates).

Secure certificates %		Insecure certificates %	
98.1		1.9	
Own certificates %	Shared certificates %	Own certificates %	Shared certificates %
63.88	36.11	72.72	27.28

Table 6.10: Repartition of servers hosting HTTPS phishing websites based on the chosen protocol and cipher-suite (Total of 103 servers).

Secure servers %		Risky servers %		Insecure servers %	
83.5 %		12.62 %		3.88 %	
Own certificates %	Shared certificates %	Own certificates %	Shared certificates %	Own certificates %	Shared certificates %
51.16	48.84	76.92	23.08	100	0

Table 6.11: Repartition of servers hosting legitimate websites based on the chosen protocol and cipher-suite (Total of 102 servers).

Secure servers %	Risky servers %	Insecure servers %
91.17	0	8.83

certificates. In order to be more accurate with the profile of legitimate websites we loosened the rules allowing us to label the certificates, however this fact is an impediment to our ability to detect insecure certificates too. Indeed, phishing and legitimate websites tend to share similar characteristics (presenting only the CN, OU and O fields) and hackers make an effort to present trustworthy values.

As shown in Table 6.10 and Table 6.11, more insecure servers can be encountered among the legitimate ones than among those that host phishing websites; however this last population presents more risky servers. Further investigations showed us that no insecure certificates have been presented by insecure or risky servers. Overall, only approximately 16% of servers (risky + insecure) hosting phishing websites have been identified as risky or insecure, against 9% for servers hosting legitimate websites.

6.4.3 Detection of Hacked Domains Hosting Phishing Webpages

The main advice given to a user in order to identify a phishing page is to never trust the look and feel of the content he is presented, but rather rely on different indicators. However, what if that perception could actually be

engineered in a different way to help the detection in the case of hacked domains?

Starting from the assumption that every website has its own visual identity allowing a user to distinguish it from another one, the fact of introducing a page impersonating a different website is more likely to disrupt that uniqueness. We were actually able to observe this characteristic among hacked domains present in the Phishtank database. If a user took the time to explore the website, instead of directly trusting the form presented on the page he landed, he could realize that incoherence. This way, by analysing the look and feel he receives not only from the first page presented, but from the global domain, he might be able to correctly identify a deceptive content.

Since it is not realistically possible to expect users to execute that analysis themselves, we aimed to introduce a system based on intelligent perception to automatically perform that behaviour. Thereby, being able to extract and compare the visual identity of the global website and the currently browsed URL gives a way to deem a website as being hacked if too dissimilar results are obtained.

This approach distinguishes itself from the research performed by Zhang and al. [48], as well as Fu and al. [21] which both used visual analysis. We propose to compare a page to others hosted within the same website in order to get their level of similarity, and this way spot anomalies that could allow us to detect phishing pages mimicking an unknown target.

6.4.3.1 Perceptual Image Hashing

Hashing algorithms are used to convert files into a fixed-length string, representing the fingerprint associated with a particular input file. They refer in general to cryptographic hash algorithms, that allow to associate two files with a slight variation in the content, to hashes that are extremely distinct. However, in our approach, it is essential to carry the information relative to the similarity of the input files in the hash obtained. It is performed by the use of perceptual hashing, a different category of hashing algorithms applied on pictures. Perceptual hashing allows to maintain the correlation that exists between the input files by generating similar hashes for similar pictures. It is then possible to assess that level of similarity by the use of a comparison algorithm such as the Hamming distance on the generated hashes. In our system, we opted for the use of dHash [28], a perceptual hashing algorithm that realizes the following steps:

- reduction of the size of the picture
- conversion of the image to a gray scale picture
- computation of the difference between two adjacent pixels

- assignment of the bits based on whether the left pixel is brighter than the right one

Its output is then a hexadecimal string representing the hash of the image.

6.4.3.2 AJNA: Anti-Phishing JS-based Visual Analysis

In order to experiment with this idea, a Firefox plugin has been implemented. Its purpose is to send the currently browsed URL to an application running on a NodeJS server, responsible for running the visual analysis during the loading of the page.

Upon reception of the URL, the server-side application retrieves all the links present on the home page of that domain and randomly selects two of them: URL1 and URL2. These URLs, along with the currently browsed one are then passed to a headless browser (Phantom JS) that takes a snapshot of the three rendered pages. The window is calibrated to get only the top of the pages, where the menu bar is more likely to be localized; the text content of paragraphs and links is hidden before rendering.

For each of the snapshots, a hash is obtained by the use of dHash, the perceptual hashing algorithm we introduced in the previous section.

The computed hashes are compared using the Hamming distance in order to get their level of similarity:

- the Hamming distance between the hash of URL1 and the one of URL2 gives a reference on how dissimilar we can expect two pages on the same website to be: H_{ref}
- the Hamming distance between the hash of URL2 and the hash of the currently browsed URL gives us the divergence between a page of the website and the page we want to assess: H_{test}

The heuristic exploited is the difference between these two hamming distances: $(H_{test} - H_{ref})$. The graph on Fig. 6.13 shows the dispersion of this heuristic, according to the status of the website: phishing or legitimate. Fig. 6.14 illustrates an implementation of our mechanism.

Along with the parameter $(H_{test} - H_{ref})$, we introduced legacy features used in previous researches, but also new ones discovered during the analysis of our phishing dataset.

The first legacy features used is the **presence of forms asking for password or credit card information on the browsed page** (feature F1). We automatically identify a page without forms as being safe, and make sure to check not only the HTML source code, but the JavaScript source code as well.

The second legacy feature is a **suspicious URL** (feature F2). We based our categorization on the presence of an IP address, a suspicious port or

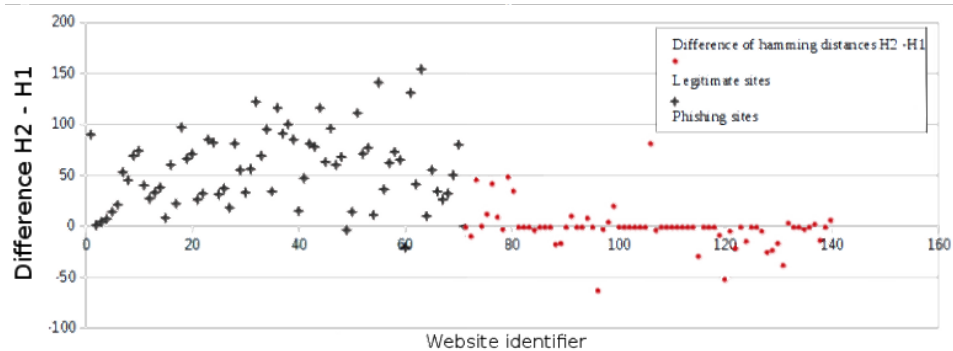


Figure 6.13: Representation of the heuristic ($H_{test} - H_{ref}$).

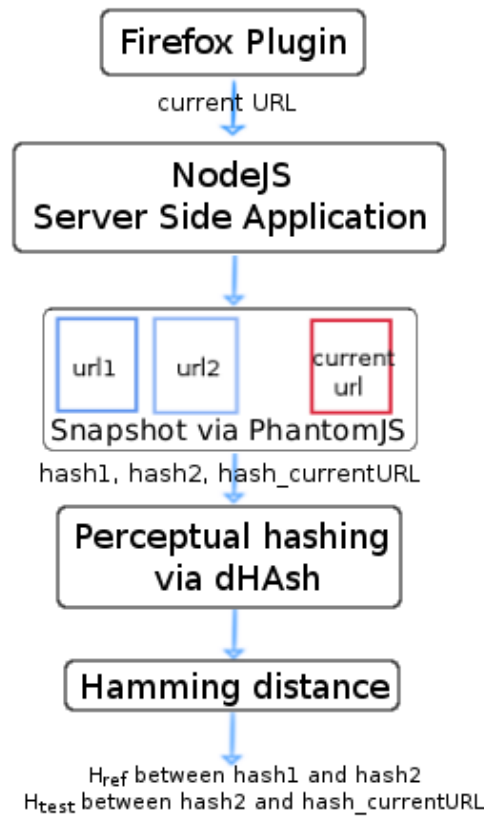


Figure 6.14: Implementation details.

character, the number of dots in the domain name, a suspicious redirect or path, as already used in past works. However, we also performed a survey of phishing URLs active between August 2014 and June 2015 that allowed us to obtain five classes of keywords recurrent in this population.

6.4.3.3 Experiments

We collected the previous heuristics on a set of 140 websites:

- 70 websites gathered via Phishtank database, they represent hacked domains hosting phishing pages
- 70 websites being legitimate pages belonging to the Alexa Top 3000

A fluctuating acquisition time has been observed for each website: from 1 to 10 seconds for 70% of the sample, and up to 20 seconds for the rest. Further investigation allowed to pinpoint the rendering of the web page by the headless browser as being the bottleneck of the system. Indeed, the more extensive the text content is, the more processing time is needed, as it is necessary to discard that content to avoid the inclusion of more divergence in the snapshots.

In order to understand the impact of each feature in the detection, we separated them into three sets:

- the first set, containing F1 and F2, the selected features used in previous works
- in the second set, we introduced our first heuristic F3 in addition to F1 and F2
- in the third set, we used the four heuristics F1, F2, F3 and F4

Facing a classification problem, we fed our dataset to the Support Vector Machine (SVM) algorithm, a simple machine learning classifier. SVM allows to find a decision boundary which separates the space in two regions. The hyperplan found is as far from all the samples as possible.

Our goal here was to train the classifier to discriminate between hacked domains and legitimate websites.

The 10-Fold cross validation method was used in order to evaluate the performance of the classifier: the dataset was randomly split into 10 sets, each one containing a training set of 126 websites and a testing set of 14 websites.

We ran this 10-Fold cross validation on each of the 3 sets of features previously introduced, and plotted the Receiver Operating Characteristic (ROC) curve for each of the experiments. The average of the metrics for each experiment can be observed in Table 6.12.

From the results in Table 6.12, we can notice the evolution of the metrics, according to the chosen set. The more features we have in the set, the better are the results. F3 and F4, the parameters introduced in this study, allow to have a more accurate classification. Indeed, the best metrics are obtained for the third set, which combines F1, F2, F3 and F4, where the phishing detection accuracy reaches 90%. That is also observable in Fig.

Table 6.12: Average of the metrics observed during 10-Fold validation, across the 3 set of features.

	TPR	TNR	FNR_score	FPR	Precision	Recall	Error	F1_score
Set 1	0.47	0.46	0.53	0.54	0.51	0.47	0.54	0.49
Set 2	0.54	0.58	0.46	0.42	0.71	0.54	0.44	0.62
Set 3	0.89	0.90	0.11	0.10	0.90	0.89	0.11	0.90

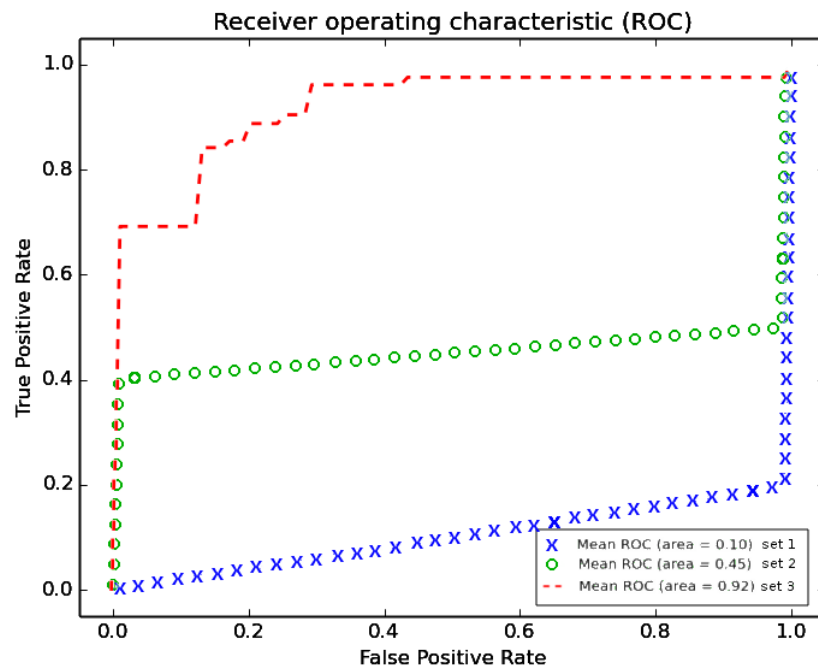


Figure 6.15: Representation of the Mean ROC curve for the three sets of features.

6.15: we start from a mean Area Under the Curve (AUC) of 0.4 with the first set of features, to reach a mean AUC of 0.6 with the second set. The best performances are obtained with the third set of features where the mean AUC is equal to 0.92.

From this experiment, we can conclude that the new heuristics we introduced allow to discriminate between legitimate sites and hacked domains without having to depend on a database of known targets.

These results seem promising, even if they should be further confirmed with a larger set of examples.

Summary

This deliverable reports the final architecture of the threat analysis platform developed by the NECOMA consortium. While matching the initial project goals, the final design of the platform also reflects some of the unexpected challenges faced during the implementation and practical uses of the platform, such as data sharing issues. The implemented prototype is based on the Hadoop framework which facilitates analysis of large datasets. The prototype provides a set of unified interfaces for accessing heterogeneous information, suitable both for batch and real-time processing.

The number and variety of implemented analysis modules permit to monitor numerous incidents and attack patterns. In particular, the deployment of analysis modules across different layers and source types reveals numerous aspects of attacks which are crucial for a comprehensive analysis of Internet threats.

The significant number of collected datasets and the abundant number of detected events, however, is a potential obstacle to pinpoint relevant incidents and focus only on useful data. To address this issue, dataset rating is considered as an essential approach to improve the usability of the platform. Furthermore, the set of threat metrics presented in this deliverable permits to estimate the importance of detected events, in terms of attack intensity, sophistication level and report confidence. Therefore, users are able to assess the threat posed by detected events, thus, prioritize responses to important attacks.

Bibliography

- [1] <http://aws.amazon.com/alexatopsites/>. [consulted 2015/09/13, Online].
- [2] https://wiki.mozilla.org/Security/Server_Side_TLS. [consulted 2015/09/13, Online].
- [3] A. Freier, P. Karlton, P. Kocher. The Secure Sockets Layer (SSL) Protocol Version 3.0, August 2011. RFC6101.
- [4] D. Anstee, A. Cockburn, and G. Sockrider. Arbor Special Report: Worldwide Infrastructure Security Report. Available at: <http://pages.arbornetworks.com/rs/arbor/images/WISR2014.pdf>, 2014.
- [5] AVTEST. Adobe & Java Make Windows Insecure. Available at: <http://www.av-test.org/en/news/news-single-view/adobe-java-make-windows-insecure>, December 2013.
- [6] X. Bao and H. Hong. NSFOCUS DDoS Threat Report 2013. Available at: <http://www.nsfocus.com/SecurityReport/NSFOCUS%20DDoS%20Threat%20Report%202013.pdf>, 2013.
- [7] P. Bieganski, J. Riedl, J. Cartis, and E. Retzel. Generalized suffix trees for biological sequence data: applications and implementation. In *27th Int. Conf. on System Sciences*, volume 5, 1994.
- [8] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis. In *NDSS*, page 17, 2011.
- [9] D. M. Blei and J. D. McAuliffe. Supervised topic models. In *Proceedings of the 21st Annual Conference on Neural Information Processing System*, pages 121–128, December 2007.
- [10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(5):993–1022, 2003.
- [11] M. Corporation. Common Vulnerability and Exposure. Available at: <https://cve.mitre.org>.
- [12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC5280.
- [13] S. Deerwester, S. T. Dumais, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of The American Society for Information Science*, 41(6):391–407, 1990.

BIBLIOGRAPHY

- [14] T. Duong and J. Rizzo. Flickr's API Signature Forgery Vulnerability. Technical report, September 2009. Available at: http://netifera.com/research/flickr_api_signature_forgery.pdf.
- [15] T. Duong and J. Rizzo. Here Come The ☹ Ninjas. May 2011. Unpublished manuscript.
- [16] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Cryptology*, 2(3):3–72, July 1990.
- [17] B. Eli. New types of cryptanalytic attacks using related keys. *Cryptology*, pages 229–246, 1994.
- [18] H. Etoh and K. Yoda. propolice: Improved Stack-smashing Attack Detection. *IPSSJ Journal*, 43(12):4034–4041, 2002. (in Japanese).
- [19] D. Evans and D. Larochelle. Improving Security Using Extensible Lightweight Static Analysis. *IEEE Software*, 19(1):42–51, 2002.
- [20] J. François, S. Wang, R. State, and T. Engel. BotTrack: Tracking Botnets Using NetFlow and PageRank. In *NETWORKING 2011*, pages 1–14. Springer, 2011.
- [21] A. Y. Fu, W. Liu, and D. Xiaotie. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *Dependable and Secure Computing, IEEE Transactions on* 3.4, pages 301–311, 2006.
- [22] H. O. Alanazi, B. B. Zaidan, A. A. Zaidan, H. A. Jalab, M. Shabbir and Y. Al-Nabhani. New Comparative Study Between DES, 3DES and AES within Nine Factors. *Computing*, 2(3):152–157, March 2010.
- [23] N. Hachem, Y. Ben Mustapha, G. G. Granadillo, and H. Debar. Botnets: lifecycle and taxonomy. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–8. IEEE, 2011.
- [24] D. Holmes. The DDoS Threat Spectrum. Available at: <https://f5.com/Portals/1/Cache/Pdfs/2421/the-ddos-threat-spectrum-.pdf>, 2012.
- [25] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *NDSS*, 2008.
- [26] C.-H. Hsu, C.-Y. Huang, and K.-T. Chen. Fast-flux bot detection in real time. In *Recent Advances in Intrusion Detection*, pages 464–483. Springer, 2010.
- [27] Y.-M. Jang, R. Mallipeddi, S. Lee, H.-W. Kwak, and M. Lee. Human intention recognition based on eyeball movement pattern and pupil size variation. *Neurocomputing*, 128:421–432, 2014.
- [28] N. Krawetz. Kind of like that. <http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html>. [consulted 2015/09/13, Online].
- [29] P. Mell, K. Scarfone, and S. Romanosky. A Complete Guide to the Common Vulnerability Scoring System Version 2.0. Available at: <https://www.first.org/cvss/cvss-guide>, June 2007.
- [30] National Institute of Standards and Technology. National Vulnerability Database. Available at: <https://nvd.nist.gov>.
- [31] N. Nethercote and J. Seward. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, pages 89–100, June 2007.
- [32] S. Neuhaus and T. Zimmermann. Security Trend Analysis with CVE Topic Models. In *Proceedings of the 21st International Symposium on Software Reliability Engineering (ISSRE)*, pages 111–120, November 2010.
- [33] OpenDNS. PhishTank - Join the fight against phishing. Available at: <http://www.phishtank.com>.

- [34] U. Park, R. Mallipeddi, and M. Lee. Human Implicit Intent Discrimination Using EEG and Eye Movement. In *Proceedings of the 21st International Conference on Neural Information Processing*, Nov. 2014.
- [35] M. F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, 1980.
- [36] D. Premack and G. Woodruffa. Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1:515–526, 1978.
- [37] Prolexic. Akamai’s State of the Internet Security - Global DDoS Attack Report Q2 2014. Available at: <https://www.stateoftheinternet.com/downloads/pdfs/2014-state-of-the-internet-web-security-global-ddos-attack-report-2014-q2.pdf>, 2014.
- [38] S. Pukkawanna, Y. Kadobayashi, G. Blanc, J. Garcia-Alfaro, and H. Debar. Classification of SSL Servers based on their SSL Handshake for Automated Security Assessment. In *Proceedings of the 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 2014.
- [39] D. A. Ramos and D. Engler. Under-Constrained Symbolic Execution: Correctness Checking for Real Code. In *Proceedings of the 24th USENIX Security Symposium*, pages 49–63, August 2015.
- [40] S. Moriai, A. Kata, and M. Kanda. Addition of Camellia Cipher Suites to Transport Layer Security (TLS), July 2005. RFC4132.
- [41] SANS Institute. SANS Information Security Training. Available at: <http://www.sans.org>.
- [42] R. Sharifnya and M. Abadi. A novel reputation system to detect DGA-based botnets. In *Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on*, pages 417–423. IEEE, 2013.
- [43] H. Tazaki, K. Okada, Y. Sekiya, and Y. Kadobayashi. MATATABI: Multi-layer Threat Analysis Platform with Hadoop. In *3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, page 8, Sept. 2014.
- [44] P. Vixie. Extension mechanisms for dns (edns0). 1999.
- [45] P. Wagle and C. Cowan. Stackguard: Simple stack smash protection for gcc. In *Proceedings of the GCC Developers Summit*, pages 243–255, May 2003.
- [46] A. R. Wilcox. Indices of qualitative variation and political measurement. *The Western Political Quarterly*, pages 325–343, 1973.
- [47] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting algorithmically generated domain-flux attacks with DNS traffic analysis. *Networking, IEEE/ACM Transactions on*, 20(5):1663–1677, 2012.
- [48] W. Zhang, G. Zhou, and X. Tian. Detecting Phishing Web Pages Based on Image Perceptual Hashing Technology. *International Journal of Advancements in Computing Technology* 4.2, 2012.