POLITÉCNICA

"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Grado en Matemáticas e Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

TRABAJO DE FIN DE GRADO

Design and Implementation of an Endpoint Reputation Module

Autor: Marcos Sebastián Alarcón

Director: Juan Caballero Bayerri

MADRID, ENERO 2015

Contents

1 Introduction

1.1 Spanish

La invención de Internet y su exponencial crecimiento desde su creación ha conllevado una ingente y creciente cantidad de amenazas: desde robos masivos de datos, hasta el formar parte de una red ilegal al servicio del mejor postor. En el momento en que un dispositivo está conectado a esta gran red, la amenaza sobre ese dispositivo es real, e inminente si no se toma ninguna medida. Para tratar de prevenir estos problemas, se han propuesto e implementado multitud de medidas, como *sistemas de detección de intrusos*, *cortafuegos*, o, como explicaremos a continuación, *listas blancas* y *listas negras*.

Al hablar de listas nos referiremos al concepto de entidad (o *endpoint* en inglés) como una abstracción que abarca IPs (como 8.8.8.8), dominios (como *www.wikipedia.org*) y URLs (como *http://www.ebay.com/rpp/gift-cards*). Una lista negra o *blacklist* es un conjunto de entidades o *endpoints* de las que se sabe que son (o han sido) maliciosas. Su utilidad dentro de un sistema es la de reaccionar en el momento de conectar con estas entidades, tomando medidas como por ejemplo bloquear el acceso a ellas. Una lista blanca o *whitelist*, por el contrario, es un conjunto de entidades que se consideran no maliciosas. Una de sus más usadas utilidades es la de permitir e-mail de una serie de proveedores que se sabe que son de confianza (como *google mail*, *yahoo*, *hotmail*...). En general, un dominio benigno tiene un periodo de vida más largo que uno malicioso, pues este último en muchas ocasiones hará uso de mecanismos (como cambiar de nombre regularmente) para precisamente evadir las listas negras. Por esta razón, la lista blanca es más fiable y contiene pocos falsos positivos (hay pocas entradas en ellas que en realidad sean maliciosas). Nos centraremos a continuación en las listas negras.

Las listas negras son de gran utilidad en la prevención y deteción de amenazas: En cuanto una nueva entidad maliciosa es descubierta, es añadida a la lista, y permite a todas las partes protegidas por dicha lista (por ejemplo, *programas antivirus* o

sistemas de detección de intrusos) ser alertadas cuando un dispositivo intente conectarse a ella. Estas listas pueden estar accesibles tanto por la red, como descargables, tanto públicamente como de forma privada. Por otra parte, hay una serie de cuestiones a tener en cuenta al utilizar una lista negra. La primera es que habrá entradas que no estén actualizadas. Por ejemplo, un dominio puede haber pasado de ser malicioso a no serlo. Para ello a veces se proporciona la fecha en la que fue añadida a la lista, pero en la práctica esto no ocurre a menudo. El segundo problema es que pueden darse falsos positivos, esto son, entidades consideradas maliciosas que en el contexto en que nosotros nos movamos no sea necesario considerarlas como tal. Esto depende de los criterios que utilice cada servicio de lista negra para bloquear estas direcciones.

Principalmente por las cuestiones citadas anteriormente, en este proyecto se propone, se implementa y se evalúa un método de agregación de una serie de listas, tanto negras como blancas, tanto online como descargables, para entidades de internet. Utilizaremos información extraída de 10 diferentes listas (y servicios en general) disponibles en la red. A partir de ellas se calculará una reputación para cada entidad, que se acercará a cero en la medida en que la entidad sea considerada benigna, y a uno en la que sea considerada maliciosa.

En resumen, el proyecto de un módulo de reputatión para entidades en la red aporta:

- Una mayor flexibilidad a la hora de decidir qué medidas tomar gracias a un sistema de reputación continuo en lugar de un valor booleano.
- Reducción del número de falsos negativos, esto es, una mejor tasa de detección de amenazas gracias a la agregación de diversas listas blancas y negras así como informes de varios antivirus.

1.2 English

Since internet was invented and due to its rapid growth, the number of threats

over this big network has constantly increased: from massive data theft, to forming enormous illegal and malicious networks sold to the best bidder. Malware industry has become professional, and in the very moment a device is connected to the internet, it's exposed to all these threats, and they will take place if no measures are taken. Lots of solutions have been proposed and implemented to prevent and mitigate these threats, such as *intrusion detection systems* (IDS), *firewalls*, or as we'll explain below, *whitelists* and *blacklists*.

When talking about these lists we are referring to the concept of *endpoint*, as an abstraction wrapping IPs (like 8.8.8.8), domains (like *www.wikipedia.org*) and URLs (like *http://www.ebay.com/rpp/gift-cards*). A *blacklist* is a set of endpoints known to be (or to have been) malicious. Its function inside a system is to react in the moment of trying to connect to these endpoints and do something, like blocking the connection. A *whitelist*, on the other hand, is a set of endpoints known to be not malicious. One of the most popular uses of whitelists is to accept e-mail from a set of reliable mail providers (such as *google mail*, *yahoo*, *hotmail*...). Typically, a benign domain remains longer than a malicious one, because this last one will often try to bypass these lists implementing avoidance mechanisms (for example, changing its name with high frequency). Due to this, whitelists are known to be more reliable and contain less false positives (this is, less entries on them that are actually malicious). We'll now focus on blacklists.

Blacklists are very useful in the world of internet security for preventing and detecting threats: In the moment a new malicious endpoint is discovered, it is blacklisted, and all the systems protected under that blacklist (like antivirus software or IDSs) are alerted when a device is trying to connect to it. These lists can be accessible through the web (online) or by downloads (offline), either from a public source or from a private one. There are some issues to consider when using a blacklist. The first one is that some entries will not be updated, i.e., a domain might have been malicious in the past, but benign in the current moment. For actually measuring this, at least a last scan date is required, but this is not often provided by the black-

list services. A second issue is that in practice there will be false positives, this is, endpoints flagged as malicious when they are actually benign. This depends on the criteria used by each blacklist to block these entities.

Mostly for the issues cited above, this project proposes, implements and evaluates a method of aggregation of several blacklist and whitelist services, both online and offline, for internet endpoints. The information is extracted from 10 different services available on the cloud. We'll compute a reputation score for each endpoint, that will approximate to 0 when it's considered benign, and to 1 when is considered malicious.

To sum up, the project of a reputation module for endpoints over the internet contributes with:

- More flexibility when deciding what measures to take when facing a potentially malicious endpoint, thanks to a continuous reputation system, instead of a boolean value.
- A reduction on the number of false negatives, i.e., a better detection rate thanks to the aggregation of various services like blacklists, whitelists and reports from antivirus software.

2 Related Work

The efficiency and completeness of blacklists has not been widely studied yet. Marc Kürer et al. [26] designed a multiple blacklist parser, making an aggregation of 49 different ones. In a latter study [27], the same team analyzed the effectiveness and completeness of 15 public malware blacklists and 4 blacklists operated by antivirus vendors. The second one concludes that individual blacklists (specially public ones like the ones we'll handle) are by far incomplete, and do not cover all the threats a system is exposed at all. At the same time, the first study shows that there's a rich amount of useful information when we combine a number of these sources. Our project will follow this work in a way that makes it more integrable into a real-time

system, able to come up with an immediate result by evaluating the responses of the services queried.

On the other hand, reputation systems for web sites have also been studied. Sushant Sinha et al. [30] made a study over some of the blacklists our module incorporates, such as *Spamhaus* [21], *SORBS* [2] and *SpamCop* [1], exclusively in the field of mail spam. These lists also internally implement reputation schemes. The study concludes that these reputation mechanisms show a rate of false positives and false negatives higher than it was expected, and a low reaction time for new spammers. Chia-Mei Chen et al. [25] developed a two-stage method for milking and then sandboxing malware samples, detecting drive-by downloads aided by a web reputation system. However, this reputation is calculated based on a set of features from the domain itself (such as DNS records, zone, or WHOIS information) and applying some heuristics to them. Leyla Bilge et al. developed EXPOSURE [23], a system that employs passive DNS analysis to detect malicious domains. It uses 15 DNS features (like the Time To Live (TTL), percentage of numerical characters, number of distinct countries, and more) to separate legitimate domains from malicious (and probably auto-generated) ones. Another work, from Yuxin Meng and Lam-For Kwok [28], also utilizes the concept of IP confidence to develop a packet filter for a Network Intrusion Detection System (NIDS). As it is a packet filter, it uses packet-level information to perform statistical analysis by monitoring the connection in an adaptive manner.

We'll also take into consideration the conclusions from the work of Manos Antonakakis et al. [22] about the behaviour of randomly generated domains, frequently used by malware to bypass blacklisting and signature-based mechanisms. We also consider how malware downloaders perform their operations: how they spread and how they are identified [29] [24]. In addition, as Brett Stone-Gross et al. explain in their article about the development of FIRE (a tool to spot whole malicious networks) [31], there are some ISPs, and networks in general, that exclusively host and protect criminal activities (such as the "Russian Business Network" [17]). We'll have this in mind when querying some services, such as *Spamhaus* [21], who maintains the

"Don't Route Or Peer" (DROP) list, a collection of networks of this type.

3 System Overview

The purpose of the reputation module is to provide reputation information on different types of external endpoints, e.g., IP addresses of hosts that communicate with internal hosts. This module is aimed to be a support tool for other agents or users. As to have a general idea, the reputation module is currently being used as a support tool for the project CADENCE (Cyber Attack Detector Engineering for Commercial Exploitation) from EIT ICT Labs [12], providing information to specific agents about the endpoints the framework is monitoring.

The module provides an API that takes as input an endpoint and outputs a reputation score between zero and one that gives a measure of confidence, with zero meaning "full confidence" or "benign" and one meaning "no confidence" or "malicious". The reputation score is computed by querying 10 external reputation engines and computing an aggregate score on the combined information they return. In addition to the reputation score, the module also provides geographical information on the input endpoint. This is not used for computing the score, but for enriching the information provided, for example to the human analyst on any alarm raised.

3.1 Architecture Overview

For understanding the design of the reputation module, we introduce the concepts of *engine* and *service*.

Service is an external resource, either online or offline, that can be consulted to obtain reputation information about a given endpoint. Some (online) services may ban the IP or key of the requester if it abuses their query rate limit. Some blacklists, like the ones available from antivirus vendors (such as *Bitdefender TrafficLight* [14], *BrowserDefender* [18], or *Norton Safe Web* [7]) are private and they usually let the

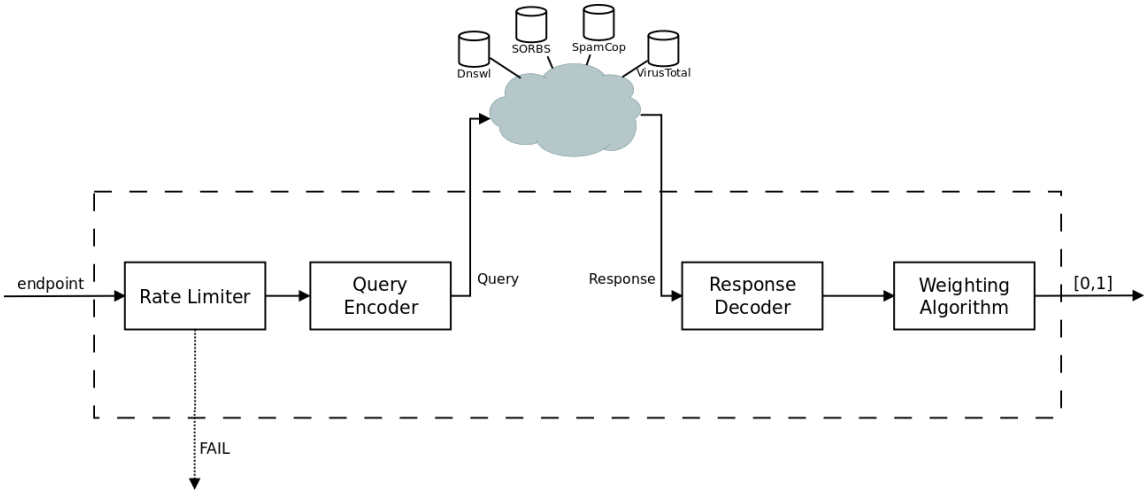


Figure 1: Overview of an online engine

user look up single endpoints through a web interface, but they don't provide an API for automated querying. Public services either provide a specific API (*VirusTotal* [15]), use plain HTML (*Google Safe Browsing* [11]) or implement some other set of mechanisms like DNS resolution *SpamHaus Zen* [21].

Engines are components of the reputation module that wrap the process of querying a service. They make transparent to the rest of the module the details of the operation. Figure 1 represents the basic configuration of an engine for querying an online service (there are slight differences between engines querying online and offline services). First, the *Rate Limiter* checks if the query limit has been reached for that session. If it hasn't, it substracts 1 to the current number of available queries and proceeds. If it has reached the limit, it fails (throwing an error) and does nothing. On the next step, the *Query Encoder* wraps the endpoint into a specific query format and sends it over the network to the address of the specific service. The encoding is service-specific. This means each engine will implement this step differently (various types of encodings will be explained in Section 4.2). When the response from the server is received, it is decoded by the *Response Decoder*. All useful information is analyzed and weighted by the *Weighting Algorithm*, also different for each engine,

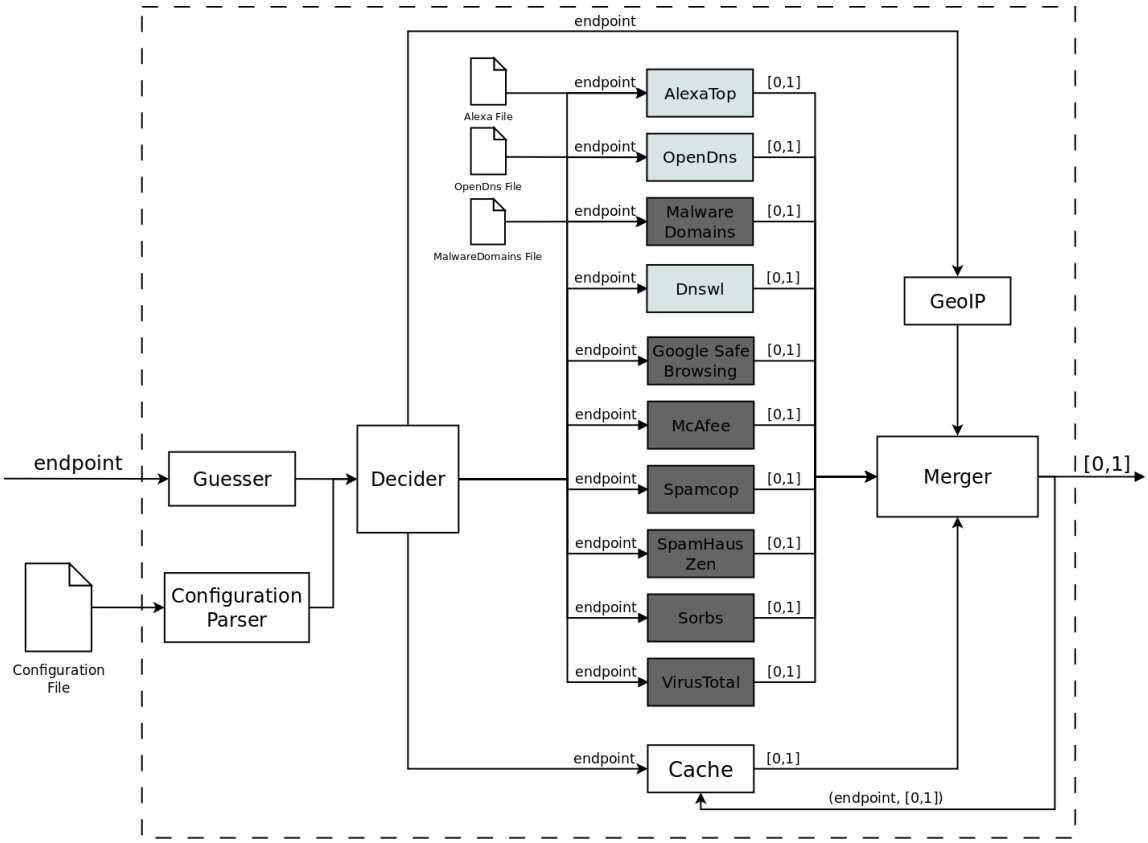


Figure 2: Overview of the module

due to the natural heterogeneity of responses from different services. Finally, an service reputation score between 0 and 1 is returned, 0 stating it's benign and 1 stating it's malicious.

Figure 2 shows the high-level structure of the reputation module. It is conformed by 3 different parts, named *Guesser*, *Decider* and *Merger*. A minor part, the *Configuration Parser* reads all the user-specific settings from an external file, such as user keys for some of the engines, and some others like the window of time in which a scan will be considered valid, for example for refreshing the cache. The *Guesser* takes as input a string representing the endpoint, and infers its type. Then both data are passed to the *Decider*, that will decide which engines to query based on the type of

endpoint received, checking first if there's recent data on the cache. It also reads from the configuration whether to give priority to the offline lists, as an optimization to consume less queries for the online services. Then the *Merger* receives a list of the engine scores from the engines queried by the Decider, and calculates an final score. It also uses the *GeoIP* component to output geographical information about the endpoint, specifically the country and city where the response comes from. It doesn't use it in the process of calculating the final score, but as additional information for the final user or agent. Finally, it writes the result on the cache and outputs the final score calculated. In Figure 2, the engines are represented as grey coloured boxes: light grey for engines querying whitelists (*AlexaTop, OpendDns* and *Dnswl*) and a darker grey for engines querying blacklists (*MalwareDomains, Google Safe Browsing, McAfee*, and so). Engines taking a file as input (*AlexaTop*, *OpenDns* and *MalwareDomains*) are querying offline services and the rest of them are querying online services.

3.2 External Services

Table 1 summarizes the 10 external services queried by the reputation module. Services are compared according to 3 main characteristics: Type of the service: blacklist (BL), whitelist (WL) and reputation (Rep), type of endpoints they accept: URL, Domain and IP, and whether it's available online and offline. Out of the 10 services, 5 are blacklists, 3 whitelists and 2 reputation services. Blacklists provide a Boolean response stating whether the queried endpoint is known to be malicious. Whitelists provide a Boolean response stating whether the queried endpoint is known to be benign. Reputation services (McAfee's TrustedSource and VirusTotal) aggregate information from multiple other detection sources, e.g. from antivirus tools. These 2 services provide enough information to compute an engine score between 0 and 1, being 0 the most reliable endpoint and 1 a malicious endpoint with total certainty.

These external services can provide information for IPs, URLs, and domains. Six of the services provide URL reputation and four only IP reputation. Some like VirusTotal can return information for different types of endpoints. For domain

Engine	BL	WL	Rep.	URL	Domain	IP	Online	Offline
Alexa [6]	✗	✓	✗	✗	✓	✗	✗	✓
Dnswl [20]	✗	✓	✗	✗	✗	✓	✓	✗
Google Safe Browsing [11]	✓	✗	✗	✓	✗	✗	✓	✗
Malware Domains [5]	✓	✗	✗	✓	✗	✗	✗	✓
McAfee TrustedSource [13]	✗	✗	✓	✓	✓	✗	✓	✗
OpenDns [8]	✗	✓	✗	✗	✓	✗	✗	✓
SORBS [2]	✓	✗	✗	✗	✗	✓	✓	✗
SpamCop [1]	✓	✗	✗	✗	✗	✓	✓	✗
Spamhaus Zen [21]	✓	✗	✗	✗	✗	✓	✓	✗
VirusTotal [15]	✗	✗	✓	✓	✓	✓	✓	✗

Table 1: External reputation services used.

reputation the module uses the services that have a *tick* (✓) on in either the *Domain* or *URL* columns, e.g., extracting the domain from the URL. For obtaining IP address reputation, it uses the services that have a *tick* (✓) in the IP column.

The last 2 columns of Table 1 indicate if the service provides an API for remote queries (*online*) or if they allow the reputation information to be downloaded (*offline*). Only three services (Alexa, OpenDns, and Malware Domains) allow downloads. From the two Whitelists, OpenDns is run by volunteers, while Alexa is a service from Amazon. Other commercial services provide an API but do not allow downloads to protect their intellectual property. Some of the external services are free. Others provide limited free access and paid subscription levels. Each online service, whether free or commercial, has some maximum query rate that the reputation module enforces through rate-limiting the queries. Paid services (like VirusTotal), and some free ones (such as Google Safe Browsing), require the use of a private key. In these two cases a free access key can be obtained by registration. Once obtained, the reputation module will read these keys from the configuration file.

The reputation module provides a score for each endpoint based on the combined information available in all services. Next we provide a brief description of each individual external service:

1. **Alexa** [6] provides a ranking of top one million most popular Web domains. It is not a whitelist per se, but we consider the top 20K domains to be benign.

2. **Dnswl** [20] is a DNS-based whitelist that provides reputation information about IP addresses of known benign mail servers. It is used by email filtering services (e.g., SpamAssasin [9]) to avoid filtering email from whitelisted senders.

3. **Google Safe Browsing** [11] provides an HTTP API for querying URLs that have been identified by Google to be malicious. For each URL, it returns an HTTP response in which the body contains "malware", "phishing", or "malware, phising". It is the service behind the malicious domain protection in the Google Chrome and Firefox browsers.

4. **Malware Domains** [5] is offered by the DNS-BH project[19]. It provides a listing of domains that are known to be used to propagate malware and spyware. This blacklist is typically used to create DNS zone files for serving fake replies to localhost for any requests to these domains.

5. **McAfee TrustedSource** [13] provides an HTTP API for querying reputation for URLs based on the McAfee (owned by Intel) internal information. It classifies URLs as "High Risk", "Medium Risk", "Minimal Risk", "Uncategorized URL", "Benign", or "Unverified". The McAfee engine assigns a different engine score to each of these risk levels.

6. **VirusTotal** [15] accumulates reputation information from many detection engines (e.g., 60 antivirus tools) and other reputation services. It provides an HTTP-JSON API that can be queried with URLs, domains, IPs, and malware files hashes. It is not designed to be used as a blacklist or whitelist directly. The reputation module outputs an Boolean engine score based on all the information returned by VirusTotal.

7. **OpenDns Top Domains** [8] contains the top 10K domain names being queried to a set of open resolvers placed all over the world. It is not a whitelist per se, since it only contains the most resolved domains by these servers. Despite this, OpenDns states that they remove any suspicious domains from the list, in this case we'll consider it as a whitelist.

8. **Spamhaus Zen** [21] is a DNS-based IP blacklist that accumulates information from 4 different Spamhaus blacklists: *XBL* for malware-related reports and 3 spam blacklists (*SBL, CLSS, PBL*).

9. **SORBS** [2] is a DNS-based blacklist. It can be used to block email from servers known to disseminate spam, phishing, and other forms of malicious email. It gives the most descriptive response from all DNS-based IP blacklists including if the IP is unknown, belongs to a hijacked network, or is a proxy server.

10. **SpamCop** [1] is a DNS-based IP blacklist that returns a Boolean response. It is a service by Cisco used for blocking spam servers. They also report the blacklisted IP addresses to the relevant ISPs.

4 Approach

The API of the module has basically one method "get_reputation(endpoint)", that takes a string as an input, representing an endpoint, and returns a reputation score. There's also a "get_rep_from_file(file)", that takes a file as input and prints through the standard output line by line the endpoint and the respective reputation score for all the endpoints in the input file (stored in the file one endpoint per each line).

4.1 Process

When first running the program, the reputation module initalizates all the engines, and calls the Configuration Parser to read all the settings from an external file that the Decider will use later. In this section, we'll describe the parts involved from calling "get_reputation" to calculating the final score. The reputation module can be called from command line (it will initalizes the module from scratch with every call) or being imported from within python. In both cases we can optionally specify the type of the endpoint that we pass as argument.

- From command line:

```
python reputation.py google.com
python reputation.py url google.com
```

In the second case we are indicating that we want "google.com" to be considered a url instead of domain, that is the type that the guesser would infer, as we will see.

- From Python:

```
>>> import reputation
>>> rp = reputation.ReputationModule()
>>> rp.get_reputation("google.com")
0
>>> rp.get_reputation("google.com","url")
0
```

Guesser. If the type of the endpoint (i.e., "ip", "url" or "domain") is not specified, the reputation module will apply regular expression matching to infer it.

1. Four numbers of length up to 3 digits separated by dots is considered an IP.
2. A string followed by a dot, another string, and a slash '/' followed by a string, is considered a URL. Any string in this context means a non-empty string.
3. The same case as identifying a URL, but if no slash '/' is found at all, or if it's at the very end of the endpoint string, means it's a domain.
4. In any other case, an error is thrown with "Unable to identify the endpoint's type".

In some cases, the same string can represent either a domain or a URL (like the case of "www.google.com" or "http://wikipedia.org"). These cases will be interpreted as domains.

Decider. This component receives the endpoint string along with its type. First, it checks if the given string is contained in the cache, within the time window defined in the settings. In a positive hit it will return that result to the merger (this is done to save online queries). If it's not found in the cache, the Decider will ask one by one to all the engines if they support that type of endpoint (by calling the public method *engine.has(type)* $\rightarrow$ *Bool*), and if so, ask for the reputation score for that given endpoint (with the method *engine.query(endpoint, type)* $\rightarrow$ *score*).

The **cache** acts as an online service, having the same interface as any other engine, with the main method *LocalDataBaseEngine.query(endpoint, type)*, and returns *null* if the endpoint is not found in it, or it is found with a timestamp older than N minutes (coded in a global variable). It is queried before any other online service, and is implemented on a local database. This database has the records of the reputation scores for endpoints already queried, along with their timestamps. The scores are stored in the cache and checked automatically by default. The database ignores what the individual scores from every engine were, to be as compact as possible. The interface to the local cache is the same as any other engine, with the method *LocalDataBaseEngine.query()*.

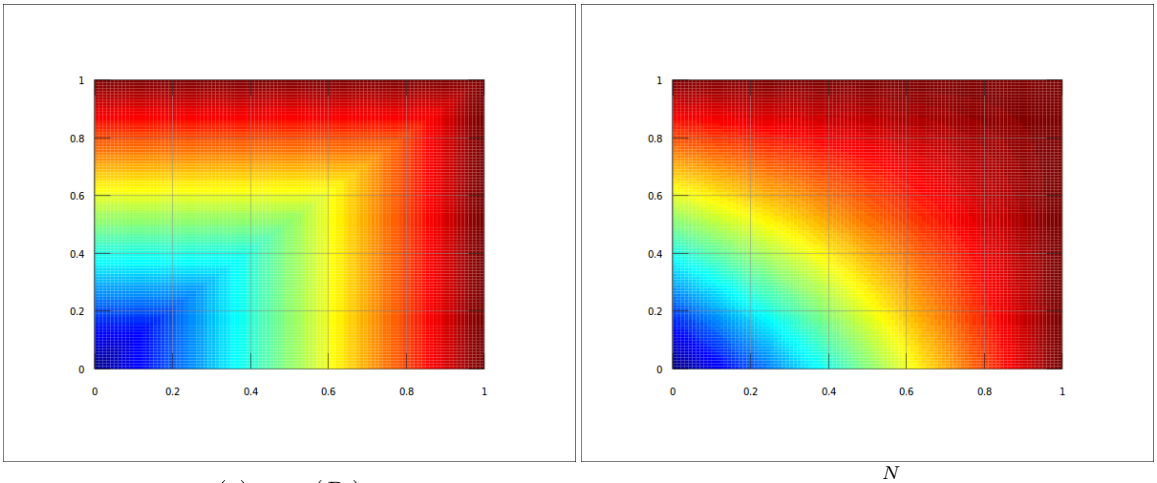
Merge. It takes the reputation scores from all engines and calculates a global reputation score S through a given function. It calculates it as follows:

$$R = 1 - \prod_{i=1}^{N} (1 - R_i)$$

Where R is the final reputation and R_i are the engine reputations from the individual engines. The function assumes $0 < R_i < 1$ for all individual scores. Calculating R this way has the following properties:

1. The higher the score for R, the more likely it is that the endpoint is malicious.

2. If any one of the engines outputs a 1, that means the endpoint is considered malicious for at least 1 service, so the final reputation R will also be 1 for sure (certainly malicious).



(a) $max(R_i)$ (b) $R = 1 - \prod_{i=1}^{N} (1 - R_i)$

Figure 3: Values of $max(R_i)$ vs. Values of R for $N = 2$

This function is close to $max(R_i)$. Figure 3 represents the reputation score R for $N = 2$, with N being the number of engines used for that endpoint. After merging the scores, the reputation module returns the final value R as a result of the previous calculation.

The **GeoIP component** is able to output the country and city where is hosted the endpoint at that given moment. It doesn't have any effect over the score, but might reveal useful information for the human analyst or end user. It operates over 2 local databases, and is accessed via its own python module "geoip2.database". The interface the reputation module offers is formed by the (private) methods "__get_country(endpoint)" and "__get_city(endpoint)", which are called by the *Merger*.

4.2 Querying Online Services

To obtain reputation information about an endpoint, the module checks first on the offline lists. If it doesn't succeed (doesn't find information on them) then it proceeds to query the described online services. This step is implemented differently on each one. We'll describe how it's done for the various engines.

DNS-based engines are used for obtaining information only about IPs. As it will be explained, they don't provide a big amount of information. The process of querying a DNS-based service is the following. We have an IP, we'll name 1.2.3.4, that we want to check. We have a service, hosted in zen.spamhaus.com (for example) that will respond to our query. With this names, the querying process is the following:

1. Put the IP in reverse order. 1.2.3.4 $\rightarrow$ 4.3.2.1.
2. Prepend the reversed ip to the URL provided by the organization for that service, with a dot between them. In this case we will prepend 4.3.2.1 to zen.spamhaus.org to obtain "4.3.2.1.zen.spamhaus.org".
3. Now we resolve the domain "4.3.2.1.zen.spamhaus.org" through DNS resolution.
4. In the case it is successfully resolved, the resulting IP for the A record will be actually encoding a response code. The meaning of this response code is the one that the owners of the service want it to be, so they'll have a (public) specification on what the response code means, and how to interpret it. In our case of zen.spamhaus.org, let's assume the domain "4.3.2.1.zen.spamhaus.org" resolves to "10.0.0.7". Then according to their documentation, we must take just the last chunk of the IP as the response code. In this case this code is "7", which means it comes from the XBL list, so it's a 3rd party exploit, such as a Trojan. In the case the endpoint is not in the database, they normally are unable to resolve it (return an "NXDOMAIN" DNS response).

The services *Dnswl, SORBS, SpamHaus,* and *SpamCop* belong to this category. The codes returned range from "blacklisted" and "not blacklisted" in the case of SpamCop, to up to 12 different classifications in the case of SORBS.

HTTP-based engines like Google Safe Browsing, McAfee, and VirusTotal. They work by sending an HTTP GET request to the URL of the service, with the parameters for the query on it, like the API key of the user (in the case of Google and VirusTotal) and the endpoint in any case. There are some extra fields, depending

on the requirements of each service. As an example, this is a query to Google Safe Browsing for the URL "http://ianfette.org/" (malware):

```
https://sb-ssl.google.com/safebrowsing/api/lookup?client=demo-app&
key=12345&
appver=1.5.2&
pver=3.1&
url=http%3A%2F%2Fianfette.org%2F
```

Google returns plain text in the body of the response. McAfee returns HTML hypertext and VirusTotal a JSON response, where fields have aggregated information from several services like antiviruses and antispyware.

4.3 Calculating Final Score

The responses from each service will be interpreted differently. In the case of blacklists and whitelists (the ones marked with a tick (✓) on the columns BL and WL in Table 1), the information extracted is simplified to a Boolean value encoding found or not found.

In the case of reputation services (the ones marked with a tick on the column Rep of 1), we consider there is enough information on the response to weight a more precise score between 0 (benign) and 1 (malicious).

- **McAfee TrustedSource** outputs the messages "High Risk", "Medium Risk", "Minimal Risk", as well as "Uncategorized URL", "Benign", or "Unverified". We interpret these levels of risk as different engine scores, being:
  - Engine reputation = 1 for "High Risk"
  - Engine reputation = 0.5 for "Medium Risk"
  - Engine reputation = 0.3 for "Unverified"
  - Engine reputation = 0 for "Minimal Risk", "Benign", and "Uncategorized URL"

These are the messages we have found so far. As there is no public documentation on the possible messages that the service can return, the reputation module throws a warning in the case it reads a new type of message that wasn't seen here, encouraging the developer to add it and assign it an engine score on the previous table.

- **VirusTotal** returns a whole JSON file as response. This file comprises multiple sections that are present only if VirusTotal has information to fill them. The weighting algorithm from the engine considers only the entries detailed below, if present:

  - *Category* is a string showing a classification of the domain. The ones we consider special cases are *Personal storage*, *Filesharing* and *Computers and software*.
  - *Detected downloaded samples* is the number of reported downloadable files that were recognized to be malicious by one or more antiviruses, with the number of positives and the scan date. An entry example of this section is:

    ```
    "detected_downloaded_samples": [{"date": "2014-03-15 22:38:53",
      "positives": 1,
      "total": 38,
      "sha256": "469d9d7329f5d8dfc1ce10368e5f837b6471
                 efb3e99bd71e8026bd454d7234ba"}]
    ```

  - *Detected urls* is the number of URLs hosted in that domain that were flagged as malicious by one or more antiviruses, with the number of positives and the scan date. An entry example of this section is:

    ```
    "detected_urls": [{"url": "http://nvidia.com/",
      "positives": 1,
      "total": 59,
      "scan_date": "2014-10-03 14:01:30"}]
    ```

This JSON structure only holds for domains. The structure when querying for a URL is completely different, and it doesn't have the sections mentioned. So this engine will handle URLs by exploring the corresponding domain. This gives information not only about the asked endpoint, but about any other resource hosted under the same domain. VirusTotal aggregates a lot of data from different antiviruses, and the response just shows this information. The objective of the VirusTotal engine is to behave as any other blacklist, returning an engine score of 0 (benign) or 1 (malicious), considering the fields mentioned before. The algorithm is implemented as follows:

1. **Create a threshold**. The idea of the threshold is to relatively measure if there are too many malicious sources inside the domain. As some categories have more downloadable files and URLs than others, we consider the *Category* field to adjust the threshold in the special cases mentioned before. If the domain is the type *Personal storage*, *Filesharing* or *Computers and software*, it's reasonable to consider that there might be some malicious sources, but we cannot flag the whole domain as malicious. The reputation will consider a threshold of 15 malicious sources for these categories, and 3 for the rest of them.
2. **Count malicious downloaded files**. If the number of malicious files hosted by the endpoint overcomes the previous threshold, it will be flagged as malicious. However, two observations are made here:
   (a) **Window of time**. VirusTotal scanned sources (both downloaded files and detected urls) have a "last scanned" date. If this date is too old, it means that no entity has requested a scan of this source in a long time, either because it hasn't showed any malicious behaviour or because it probably doesn't exist anymore. Motivated by this observation, we consider a way to implement this in the algorithm. The last scan date is compared with the current date, and old scans are ignored, as they are usually either removed or sinkholed. This window of time is set in the configuration file of the reputation module, indi-

cating the maximum number of minutes before ignoring that entry, and by default is set to 129600 minutes (90 days).

(b) **False positives** are situations in which a source is misinterpreted as malicious when in reality is not. The number of positives over these sources is sometimes very low (typically only 1 or 2 antiviruses raising an alarm). The probability that these are false positives is very high. That's why the engine also defines the minimum number of positives (3 by default) before considering the result a real threat.

After applying these filters, we compare the number of positives with the global threshold (either 3 or 15 depending on the category of the endpoint), and flag it as malicious if it corresponds.

3. **Count malicious URLs**. In the case it is not flagged with the previous step, the same process as with "malicious downloaded files" is applied with the entry "malicious urls".

5 Evaluation

For the evaluation of the reputation module, we first have to acquire a set of benign endpoints to consider it *ground truth*. This is not an easy task, since it's not trivial to know if a specific domain contains any malicious threats. Specially, when we are testing the mechanisms in which we normally rely to get the reputation for those endpoints.

In this first test case, we consider the *Alexa top 5000* file, a subset of the *Alexa top 1 million* file that the module uses, to test the online services able to query URLs. The entries of this file are not ensured to be benign on their totality, but they are a good approximation easily and publicly available. We'll the engines *McAfee TrustedSource*, *Google Safe Browsing*, and *MalwareDomains*. Since this specific test focuses on false positives, we'll run the *VirusTotal* engine only over the positives from these 3 engines. The reason is that *VirusTotal* service with a free license key

has a very low query rate limit (4 queries per minute) and it totally bottlenecks the test. The results over this first step are shown in Table 2. Each row classifies results from each engine (McAfee TS, Google SF, and MalwareDomains). Columns display the number of endpoints scoring that final reputation, along with the percentage of the total number of endpoints tested on the *Alexa top 5000*. The last row displays these numbers for the final score of the reputation module.

Reputation Score:	0	0.3	0.5	1
Endpoints by McAfee TS	4496 *(89.92%)*	407 *(8.14%)*	46 *(0.92%)*	51 *(1.02%)*
Endpoints by Google SB	4989 *(99.78%)*	-	-	11 *(0.22%)*
Endpoints by Mal.Domains	4993 *(99.86%)*	-	-	7 *(0.14%)*
Final Score	4491 *(89.82%)*	405 *(8.10%)*	45 *(0.90%)*	59 *(1.18%)*

Table 2: Reputation Scores for Alexa Top 5K Domains using 3 engines querying domains

We consider endpoints with a final reputation score of 1 to be false positives. Over these 59 endpoints, we now study the individual scores returned by the involved engines. In 55 out of 59 cases this score is either 0 or 1 for all engines, except for the endpoints "t60gfsm.com" and "findamo.com" (engine reputation of 0.5 by McAfee) and "downloadha.com" and "0427d7.se" (engine reputation of 0.3 by McAfee). Figure 4 shows the number of endpoints flagged by each engine and the intersections between these sets. As we've said earlier, it's not trivial to know if a score reflects a false positive. *McAfee TrustedSource* raises a big amount of false positives, as we can see that 37 out of the 59 endpoints are flagged as malicious exclusively by this engine, meaning "High Risk" according to McAfee's results. At the same time, *VirusTotal* and *Google Safe Browsing* engines don't flag a big number of domains as malicious, meaning their false positive rate is low. Blacklists usually take good care of having a low false positive rate, because in practice it's better to miss some malicious endpoints than to block legitimate ones. As we see in Figure 4, there are 2 domains considered malicious by McAfee, Google and VirusTotal. These domains

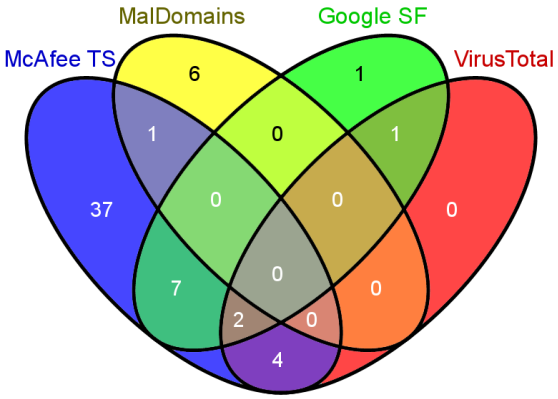


Figure 4: Number of endpoints flagged as malicious by each engine and intersections between them

are "secure-master.com" and "security-upgrade.com". We can say that these 2 domains are actually true positives, i.e., they are flagged as malicious, and are indeed malicious even though they are in *Alexa top 5000*.

In a second test case, we aimed to spot false negatives, i.e., domains flagged as benign when in fact they are malicious. For this, we would ideally need a ground truth for positive cases, this is, a set of endpoints for which we certainly know they are all malicious. This time we consider the updated file *Malicious Domains*, also part of the reputation module. This file contains updated entries of very recently detected malicious domains. We take the first 3000 endpoints of this file, corresponding to malicious domains reported from 20-12-2014 to 29-12-2014. Table 3 shows 4 reputation values, the number of endpoints from the *MalwareDomains* file scoring those values, and the percentage that they sum up over the total number of entries.

Final Reputation Score:	0	0.3	0.5	1
Number of Domains	147 *(4.90%)*	13 *(0.43%)*	22 *(0.73%)*	2818 *(93.93%)*

Table 3: Final Reputation Scores for MalwareDomains top 3000 querying the services McAfee TS and Google SB

A 4.90% of the endpoints in the file are considered false negatives. This is a reasonable number. Once again, we are not completely sure if these are true false negatives, as some of them might have been malicious when they were spotted, and not be malicious at the moment of the test. But in this case, the oldest entries were spotted two weeks before the moment of the evaluation, so we'll consider this window to be thick enough to assume they are false negatives.

6 Conclusion and Future Work

In this work we aggregate 10 lists (5 blacklists, 3 whitelists and 2 reputation lists) and calculate a joint reputation score by gathering information about a given endpoint of type IP, URL or domain. In addition, we extract geographical information for the end user. We take into consideration the rate limit for every different service, and in order to boost the number of queries we implement a cache to return recently queried data. The approach of the reputation module has the advantage of relying on multiple services, with the objective of reducing the number of false negatives, and at the same time it goes beyond considering an endpoint either malicious or benign. The reputation score conveys a scale of risk, obtaining a wider spectrum of tolerance when considering an endpoint malicious or not. To conclude, we cite the results on section 5 about the evaluation of the module, approximating at this point a false positive rate of 1.18% for the 5000 most visited domains on the internet, and a false negative rate of 4.90% for 3000 malware samples detected on the past 2 weeks.

As future work, we propose integrating Mozilla's Public Suffix List [10] as a way to extract the real domain from the endpoint. At the moment, the reputation is calculated for the Second Level Domain (SLD) assuming this is the authority re-

sponsible for all its subdomains. As an illustrative example, the reputation score for the domain *example.co.uk* will be calculated over *co.uk*, even though the *co.uk* SLD is not responsible for the domain *example.co.uk*. So in the case there is a *good.co.uk* benign domain and a *bad.co.uk* malicious one, the whole domain *co.uk* will have a final reputation close to 1, indicating that is malicious, and by doing so we are also banning *good.co.uk*.

Furthermore, a more exhaustive evaluation over the completeness and correctness of the reputation score of the concrete engines is proposed. By doing so, different algorithms for calculating the final score can be tested, and the one with the best results shall be picked. Also, the detection rate would grow by aggregating more blacklists, specially for specific threats such as banking Trojans (*Zeus* and *SpyEye*) [3], worms (*Palevo*) [3], malware like *Citadel* [4], and domains generated by Domain Generation Algorithms (DGAs) [16].

References

[1] bl.spamcop.net.

[2] dnsbl.sorbs.net.

[3] http://abuse.ch.

[4] http://botnetlegalnotice.com/citadel.

[5] http://mirror1.malwaredomains.com/files/domains.txt.

[6] http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[7] http://safeweb.norton.com.

[8] https://github.com/opendns/public-domain-lists/blob/master/opendns-top-domains.txt.

[9] http://spamassassin.apache.org/.

[10] https://publicsuffix.org.

[11] https://sb-ssl.google.com/safebrowsing/api/lookup.

[12] https://www.eitictlabs.eu/innovation-areas/privacy-security-trust-in-information-society/cadence/.

[13] https://www.trustedsource.org/en/feedback/url.

[14] http://trafficlight.bitdefender.com.

[15] http://virustotal.com.

[16] http://virustracker.info.

[17] http://www.bizeul.org/files/rbn_study.pdf.

[18] http://www.browserdefender.com.

[19] http://www.malwaredomains.com/.

[20] list.dnswl.org.

[21] zen.spamhaus.org.

[22] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou II, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, pages 491–506, 2012.

[23] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel. Exposure: A passive dns analysis service to detect and report malicious domains. *ACM Trans. Inf. Syst. Secur.*, 16(4):14:1–14:28, April 2014.

[24] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *20th USENIX*

Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings, 2011.

[25] Chia-Mei Chen, Jhe-Jhun Huang, and Ya-Hui Ou. Efficient suspicious {URL} filtering based on reputation. *Journal of Information Security and Applications*, (0):–, 2014.

[26] Marc Kührer and Thorsten Holz. An empirical analysis of malware blacklists. *Praxis der Informationsverarbeitung und Kommunikation*, 35(1):11–16, 2012.

[27] Marc Kührer, Christian Rossow, and Thorsten Holz. Paint it black: Evaluating the effectiveness of malware blacklists. In *Research in Attacks, Intrusions and Defenses - 17th International Symposium, RAID 2014, Gothenburg, Sweden, September 17-19, 2014. Proceedings*, pages 1–21, 2014.

[28] Yuxin Meng and Lam-For Kwok. Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection. *Journal of Network and Computer Applications*, 39(0):83 – 92, 2014.

[29] Christian Rossow, Christian J. Dietrich, and Herbert Bos. Large-scale analysis of malware downloaders. In *Detection of Intrusions and Malware, and Vulnerability Assessment - 9th International Conference, DIMVA 2012, Heraklion, Crete, Greece, July 26-27, 2012, Revised Selected Papers*, pages 42–61, 2012.

[30] S. Sinha, M. Bailey, and F. Jahanian. Shades of grey: On the effectiveness of reputation-based "blacklists";. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 57–64, Oct 2008.

[31] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding rogue networks. In *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pages 231–240, Dec 2009.

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Wed Jan 07 20:36:58 CET 2015
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)