Applying Soft Computing Techniques to Corporate Mobile Security Systems

Master Thesis

Paloma de las Cuevas Delgado

Department of Arquitechture and Computer Technology
School of Computer Science and Telecommunications Engineering
University of Granada

September 2014

This document has been edited with TEXIS v.1.0.

Applying Soft Computing Techniques to Corporate Mobile Security Systems

Master Thesis

Master on Computer and Network Engineering

Supervised by:
Dr. Juan Julián Merelo Guervós
Dr. Antonio Miguel Mora García

Department of Arquitechture and Computer Technology
School of Computer Science and Telecommunications Engineering
University of Granada

September 2014

Abstract

Corporate workers increasingly use their own devices for work purposes, in a trend that has come to be called the *Bring Your Own Device* (BYOD) philosophy and companies are starting to include it in their policies. For this reason, corporate security systems need to be redefined and adapted, by the corporate Information Technology (IT) department, to these emerging behaviours. This work proposes applying soft-computing techniques, in order to help the Chief Security Officer (CSO) of a company (in charge of the IT department) to improve the security policies. The actions performed be company workers under a BYOD situation will be treated as events: an action or set of actions yielding to a response. Some of those events might cause a non compliance with some corporate policies, and then it would be necessary to define a set of security rules (action, consequence). Furthermore, the processing of the extracted knowledge will allow the rules to be adapted.

The goals of this work are the following:

- Extracting information from a set of Corporate Security Polocies (CSP) (also called Information Security Policies (ISP)), and from a set of Internet connection patterns of its employees, by applying Data Mining techniques.
- Combining that information to detect interesting or suspicious patterns from the point of view of a CSO. They are relevant because they could lead to security incidents, assuming that a security incident is a pattern that is not compliant with the rules derived from security policies.
- Performing a deep study of the labelled patterns with the Weka tool, and find the best classification method and the most significant data features.

All the development of the code, research, and writing of this Thesis has been done on Github [1]. It is open and accesible at: `https://github.com/unintendedbear/TFM/`.

[1] `https://github.com`

Index

List of Figures

List of Tables

Chapter 1

Introduction

Ha, ha! Whoo hoo hoo! Ah! Geronimo!

11th Doctor. Doctor Who. *The End Of Time, part II.*

1.1. Introduction

The way in how data has been stored and accessed in the companies has completely changed over the last few years (Mozy, 2011). On the one hand, start-up companies get a great benefit either by using services of the cloud (Vaquero, 2009), or by offering their services through it (Leavitt, 2009). For bigger companies, on the other hand, the use of company-owned servers being accessed from desktop PCs (and maybe laptops) from within the company facilities has been transformed. Now, these data are distributed among a number of machines, even not all bemacselonging to the company, and working over a cloud Computing environment. Besides, being stored in the cloud or not, data are being consulted and modified through a wide amount of devices, some of them owned by the company's users. This is the so-called BYOD (*Bring Your Own Device*) (Caldwell, 2012) philosophy, which also means that people do not use their smart devices for one purpose only (personal life or business) anymore. It is becoming highly successful due to the impact that portable devices (such as smartphones and tablets) are having in the market. And because of data security and privacy are key factors for a company, it is usual to define Organisational Security Policies in order to ensure them (Blaze, 1999). Their definition is nowadays a very difficult problem, since that BYOD tendency means that several factors must be considered (Oppliger, 2011), most of them previously ignored or non considered in security systems, for instance the current mixture between personal and professional information in these devices (the user could navigate inside social networks where there could be friends and also company partners or clients).

1.1.1. Enterprise security

Now that corporate networks are becoming dynamic for being adapted to the BYOD philosophy, there is an additional risk because the devices that the employees use are not always company-owned. Thus, several solutions have arisen in order to manage the corporate security in a BYOD scenario. Some of them are focused only on smartphones, while others are thought for laptops; also some are implemented for a certain platform, but others consider multiplatform. However most of them try to be non-intrusive (regarding the users'personal data), friendly and easy to use. In Section 2.1 we present an overview of the main solutions, describing their features, advantages and flaws.

This kind of monitorization and security-aimed applications are being developed for many reasons like the aforementioned, and also because it has been demonstrated that people are the main hazard regarding the company security (Adams, 1999). With a methodology of work like the one described, users are allowed to start or continue a working session over multiple devices and locations without any significant loss of data. This new situation has a big impact from the point of view of the security (Schumacher, 2005), since company data borders have changed in the last years so now the users can access significant data from outside the enterprise, and possibly through a non absolutely secure channel.

Another way of protecting an enterprise is by means of a security policy, or in this case, a set of ISP (*Information Security Policies*), which should deal with the way of protecting a certain organization information against a security breach. Though there are standards, such as the ISO27002 or the Security Forum's Standard of Good Practice [1], and many guidelines (Cresson Wood, 2009); an ISP is built depending on the characteristics of the community/organisation that they are thought for.

Then, another issue to cope with is the elaboration of a good ISP, understandable for every user of the company, and more importantly, non-intrusive for him. A lot of researchers have studied the natural tendency of employees to comply with the ISP (Siponen, 2007; Bulgurcu, 2010; Al-Omari, 2012), reaching conclusions such as the employees compliance with the security policies increases educating/training them in information security awareness (Shaw, 2009), and decreases applying too much sanctions when a misuse or abuse occurs (Herath, 2009).

Normally, the enterprise network architecture was being adapted to cope with external attackers (Lippmann, 2005). With the incorporation of BYOD, the threat is about corporate assets being compromised due to employees'devices with vulnerabilities (Orthacker, 2012), or leaked because they are being accessed from a device connected through an unsecure (public) net-

[1] https://www.securityforum.org

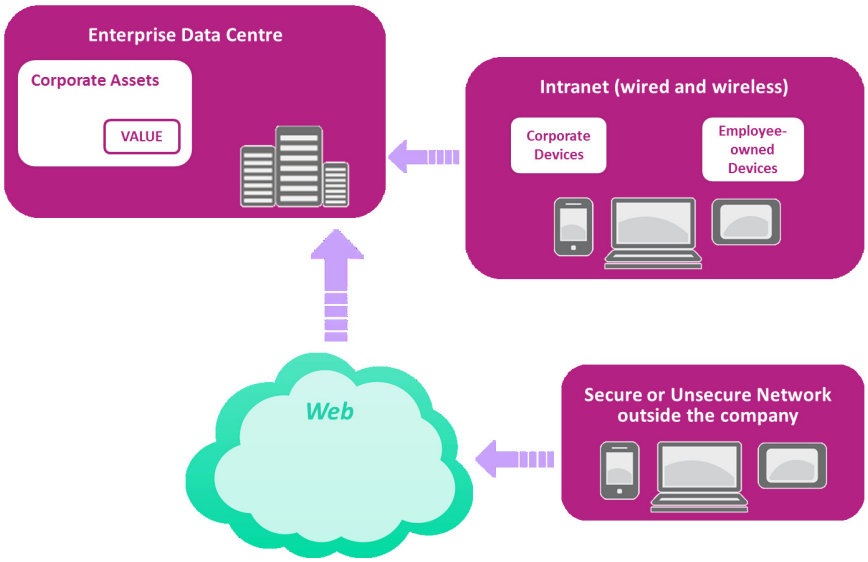


Figure 1.1: Architecture approach of an Enterprise Network assuming that the Company has adopted the BYOD philosophy.

work.

Thus now, more things should be considered than the usual ones when designing a company network architecture. In Figure 1.1 there is an approach to the system architecture of a secure environment. It includes the possibility of having employee-owned mobile (smartphones and tablets) and portable (laptops) devices, and also the opportunity that the employees have of connecting these devices either from inside or outside the company premises. Moreover, company's information assets are constantly accessed under these conditions, considering that an information asset means every *piece of information* that has a *value* (cost depending on the risk of being lost or leaked) for the company. It can be referred to files with sensitive information to certain mails, or even to company applications.

This situation leads to a need of protecting the organisation's side, but also the users'side, making non-interfering easy-to-follow ISPs, and leaving them to use their devices for personal purposes while working, without putting organisation's information assets under risk. The compliance of these requirements would compose an end-to-end security solution (protecting both enterprise and employee).

1.2. Objectives

The problem to solve is related with the application of corporate security policies in order to deal with potentially dangerous URL (*Uniform Resource*

Locator) accesses inside an enterprise. Not only there could be connections to non-confident (or non-certified) web sites (referenced by their URLs in this work), but in a company, several web pages might be also controlled for productivity or suitability reasons. Thus, an ISP normally define sets of allowed or denied pages/websites that could be eventually accessed by enterprise employees. These sets are usually included in a White (permitted) or Black (non-permitted) Lists. These lists act as a good control tool for those URL included in them as well as for the complementary, i.e. URLs that are not included in a Whitelist have automatically denial of access, for instance.

In this work we go a step beyond, trying to define a tool for automatically making an allowance or denial decision with respect to URLs that are not included in the aforementioned lists. This decision would be based in that one made for similar URL accesses (those with similar features), but considering other parameters of the request/connection instead of just the URL string, as those lists do.

Therefore, the starting point is: to have a registration of the Internet accesses made in a company, and a set of policies that say what should or should not be done.

The first objective of this work is to be able to label the whole set of entries in the access log, by means of simply applying the rules to them. To this end a dataset of URL sessions (requests and accesses) should be analysed. These data are labelled with the corresponding permission or not for that access following the aforementioned policies. The problem is then transformed into a classification one, in which every new URL request will be classified, and thus, a grant or deny action will be assigned to that pattern.

The desired outcome is to obtain as many labelled entries as possible, to proper build and train a classifier. We will have, then, labelled entries as *allowed* or *denied* by the company, and unlabelled entries.

The next objective is to obtain the best classification accuracies as possible, so a series of experiments should be performed, in order to see which classifiers work better. Also, it is important to take into account that balancing techniques may be applied if the dataset present imbalance (Japkowicz, 2002).

Finally, with a trained classifier, those patterns that were unlabelled, are expected to be now labelled (at least, most of them). Therefore, a system that adopts that classifier, would be able to adapt to new situations and new policies could be written for coping with them.

The Thesis is structured as follows. Next Chapter (Chapter 2) describes the state of the art of applications or devices related to BYOD being adopted by companies, and protection of the users'privacy. Also, it describes related work with regard to the application of Data Mining and Machine Learning techniques to security issues inside a company and, finally, introduces some

work review about the study of URLs and its use for security menaces detection. In Chapter 3, we recall again the problem this work tries to solve and describe the type of policies and rules used to label the data, as well as the datasets we have worked with. The followed methodology is described in Chapter 4, concerning the data preprocessing and the implementations done in Perl and Java languages. Chapter 5 is devoted to present the results. From a first round of experiments comparing different classification methods, to once the best of them are selected, the set of conducted experiments, the different partitions of the datasets, and result tables. Finally, the conclusions and future lines of research are presented in Chapter 6.

Chapter 2

State of the Art

We both contain the knowledge of over nine hundred years of memory and experience.

11th Doctor. Doctor Who. *The Almost People.*

The present Master Thesis tries to obtain a URL classification tool for enhancing the security in the client side, as at the end we want to get if a certain URL is secure or not, having as reference a set of rules (derived from a CSP) that allow or deny a set of known *http* requests. For this, DM (*Data Mining*) and ML (*Machine Learning*) techniques have been applied. Also, this solution tends to be included inside a BYOD environment as described at the beginning of the Section 1.2. This Chapter gives an overview in a number of solutions given to protect the user, or the company, against unsecure situations.

2.1. Tools for corporate mobile security

Now that BYOD philosophy is becoming a tendency, a number of tools appeared by now, and others are under development but expected to be released soon. They were specifically designed for CSO (*Chief Security Officer*)s and CISO (*Chief Information Security Officer*)s to secure, monitor, and act over spartphones and other personal mobile or portable devices, when they become too risky.

2.1.1. IBM Hosted Mobile Device Security Management

One of the first companies who supported the BYOD model was IBM in 2011 (IBM, 2011), as they recognized the increase of employees who brought

their personal smartphones or tablets into the workplace. To help organizations (Kao, 2011) embracing both company and employee owned mobile devices (as said, this practise is part of the BYOD model) in a security-rich environment, IBM developed a mobile device security management solution. For IBM, a mobile security strategy should focus on several key areas.

Thus, the organization should identify which business data the strategy will allow to be stored and processed on which mobile devices. This helps to determine what needs to be protected and to what degree. Then, since different mobile platforms have different native security mechanisms, the organization needs to define which mobile device platforms will be allowed in the business environment and, thus, which need to be supported in the mobile security strategy and plan. That means, to define the scope of the security plan. Also it is needed to decide the responsibility for mobile security management work, whether using the current IT (*Information Techlonogy*) security team to handle mobile devices, or outsourcing them to a managed security service provider. And no matter what the mobile environments are, a number of mobile ISPs and best-practise procedures need to be put in place and should also be identified in the company's mobile security strategic plan.

Taking into account these considerations, IBM developed a framework that specifies security domains and levels for applying various security technologies. When applied to mobile devices (not portable ones, so that laptops are not considered), the features of this framework depend on the deployment:

Identity and access Not only the use of strong passwords when accessing the devices, but also supply two way authentication and VPN (*Virtual Private Network*) access control, by supervising the authorised IP (*Internet Protocol*) addresses and adding re-authentication for accessing resources (assets) with high value.

Data protection Data security stored in the devices, related to professional life, is encrypted, also during the transmission. Moreover, when a device is lost, data can be removed remotely, and the device located or locked out (locking out a device is also made after a modifiable timeout). Finally, back ups are performed periodically once the device has been recovered.

Application security This is related to the requirement of being connected from a controlled location for authorising the download of business applications, and those applications must be certified (also the currently running on the device). Actually, IBM's tool can monitor and remove applications which are identified as untrustworthy or unsafe.

Fundamental integrity control Running both anti-malware software and personal firewall, and making the result of these tests as dependency

for VPN intranet access.

Governance and compliance Taking into consideration the mobile security as an important point in the overall risk management program of the company, and extend the periodic security audits to mobile devices too.

The considered architecture for this solution is a client-server architecture, where the controlling center is placed in the server, and the client would be installed on the mobile devices. This way, the client/device would communicate with the server as regularly as possible, to enforce policies, execute commands, and report status. With these features, and by analysing in the server the incoming data from the devices, this IBM's solution would be able to help enterprises to support ISP compliance and to recognise (consequently acting over) mobile threat landscapes.

2.1.2. Sophos Mobile Control

Sophos is a company founded in 1985 focused on IT security and data protection for businesses. Their *Mobile Device Management* main product (Sophos., 2011) is Sophos Mobile Control. It is oriented to IT administration for mobile devices, trying to offer to the users the possibility of choosing the delivery model to suit their needs, i.e., between on-premise and SaaS (*Software as a Service*).

The tool offers the possibility of managing all workers and co-workers smartphones and tablets from a single-based console. The console monitors the devices throughout their full life cycle: from the initial set up and enrolment, right through to decommissioning. Other features are similar to IBM's product, adding some new like being able to connect to an existing user directory using LDAP (*Lightweight Directory Access Control*)[1].

Additional security is provided by the incorporation of *Malware and Web protection*, so that the user does not need to install anti-malware software by him or herself. Regarding the compliance enforcement, the main goal is not to sacrifice company's security in favour of flexibility for the users, which has been demonstrated (Herath, 2009) that could result in bad (unwilling or not) users'behaviour. Thus, company's BYOD initiative should include an acceptable use policy to ensure the users are aware of any measures the company may take if a device breaches any ISPs. Sophos aims to reach this by doing three main tasks: First, by enforcing ISPs, i.e. allowing setting up user and group-based security policies separately. The security settings can also vary from one platform to another, thus looking to support all of them is necessary to set task bundles and individual actions for many different

[1]LDAP is an application protocol for accessing and maintaining distributed directory information services over an IP network.

violations. Secondly, by a risk mitigation in which the actions to perform can be set according to the severity of a breach. For minor cases, the company may want to simply inform to the user, but if sensitive data is at risk, a remote wipe may be the chosen option. As previously stated, the actions vary for each platform, but the most common platforms such as Android and iOS allow blocking email access, notifying the admin, performing a remote lock or wiping, locating a device using 3D maps, triggering a remote alarm, transferring a task bundle combining a number of actions, and Sophos's solution adds the possibility of trigger a scan. Finally, compliance check, i.e. though some of the most widely used features include allow or disallow root rights or jailbreaking, require encryption, and whitelist or blacklist apps. Sophos also allows disabling malware apps, setting maximum intervals since last *Mobile Security scan*, and allowing or disallowing suspicious apps and PUA (*Potentially Unwanted Apps*)s.

Then, the Enterprise App Store included in Sophos Mobile Control allows the company to supply the users with recommended and/or required apps directly on their devices. Both company's in-house and app store apps are suggested directly on the user's mobile device, so they can click to trigger the installation. Also, for keeping the employees working without increasing the burden for the IT department, the self-service portal built-in included in Sophos's solution has many features. Among others, it shall be mentioned that, wanting the employees to use their personal devices at work, they can register them (with a provided step-by-step process) and agree to an acceptable company-defined use policy. All profiles, including email access, would be available after registration, and the portal may be accessed from any PC with Internet access, or even from a mobile device itself. Furthermore, when a device is eventually stolen, users can choose to remotely locate, lock or wipe their devices and reset their passcode without having to contact the company help desk. From the company side, they can define which features are available in a self-service portal from the administrator console.

2.1.3. Samsung's Knox Mobile Security Suite

As part of its SAFE (Samsung for enterprise) brand, Samsung revealed at the Barcelona Mobile World Congress 2013 (Woods, 2013) the Knox application (Samsung., 2013), and was publicited at the congress again this year (News, 2014). The solution is available since December 2013. The main feature of this security package is the use of different containers, or environments, for business and personal sides. Each one even includes its own graphic configuration (wallpapers, colours, and so on), in order to be more easily recognised and distinguished by the user. There will be needed introducing a password to enter into the business side and, once *logged in* this container, no more passwords will be required for the business applications. The applications approved by the company IT department must meet Sam-

Figure 2.1: Source: `http://www.samsung.com/global/business/mobile/solution/security/samsung-knox`

sung's security standards and allow single sign-on. A Knox API (*Application Programming Interface*) will also be provided for the company to be capable of accessing over almost 205 predefined IT policies (the SAFE API grows this number to 475). Also, Knox would allow different VPNs for individual apps. Regarding the information protection methods, data files saved by applications of each environment are encrypted with AES 256-bit algorithm, in such manner that only the appropriate container can access these files. In the same way, the user won't be able to share data between the two environments, e.g. creating separate contact lists so the user cannot send a contact from one side to the other, or if the user copies data to the clipboard in the Knox container, it won't be there in the personal container. Figure 2.1 shows the device architecture with three main parts, each one in charge of deploying some of the mentioned characteristics.

Customizable Secure Boot This ensures that only verified and authorized software can run on the device. It is a primary component that forms the first line of defence against malicious attacks on devices with Samsung Knox. In addition, Samsung Knox's Secure Boot technology allows the switch of the secure boot root certificate in a secure manner after the devices are shipped.

TrustZone-based Integrity Measurement Architecture (TIMA) By a continuous integrity monitoring of the Linux kernel, it is possible to

detect that the integrity of the kernel or the boot loader is violated, and to take a policy-driven action in response. One of these policy actions disables the kernel and powers down the device.

Security Enhancements for Android This feature is the one referred to the separation of information based on confidentiality and integrity requirements. It isolates applications and data into different domains so that threats of tampering and bypassing of application security mechanisms are reduced while the amount of damage that can be caused by malicious or flawed applications is minimized.

It is important to take into account that if an enterprise decides to deploy this solution to secure its BYOD environment, it must yet work with certain MDM (*Mobile Device Management*) services, either cloud or server based. Samsung offers a list[2] of the MDM which Knox supports, each one with the number of supported policies.

2.1.4. Good's Bring Your Own Device solution

The philosophy followed by Good's solution is similar to Samsung's Knox one: to create a secure container that places an unreachable partition between personal and business data in order to protect company assets. The solutions that they offer (Technology., 2012) are similar than the previous ones. They have focused in mobile (not laptops) devices, though they support a different of OS (*Operating System*)s, and in separating personal and company data. A Good's secure Network Operations Center (NOC) is introduced for dealing with the unauthorised devices, or for providing access to secure collaboration solutions (email, PIM, calendar), intranet, and in-house or third-party mobile applications. Finally, Good offers best practise recommendations to help the company's BYOD policies such as reimbursements and stipends. There is a document available at Good's webpage (Technology., 2012) which contains several questions about ISPs and how to cope with all of them.

2.1.5. BlackBerry Balance

This security package was announced as a feature of BlackBerry 10 (Blackberry., 2012). Nevertheless, it is available with BlackBerry Enterprise Service 10, which is a device management, security and app management for BlackBerry, iOS and Android devices. It is necessary to activate BlackBerry Balance for having available some security features, all related or similar to the aforementioned. For instance, as shown in Figure 2.2, a message is displayed when the user tries to copy work data and then paste it into personal apps. Also, user attempting for actions that are not permitted in the

[2] https://www.samsungknox.com/en/knox-mdm-feature-list

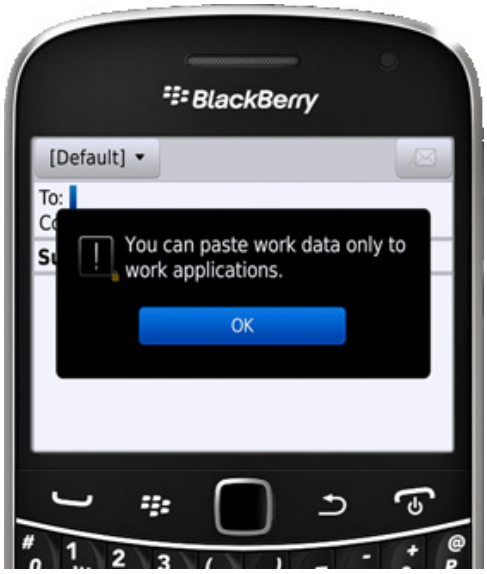


Figure 2.2: Displayed message in new Blackberry 10 when attepmting to copy sensitive company data. Source: `http://uk.blackberry.com/business/software/blackberry-balance.html`

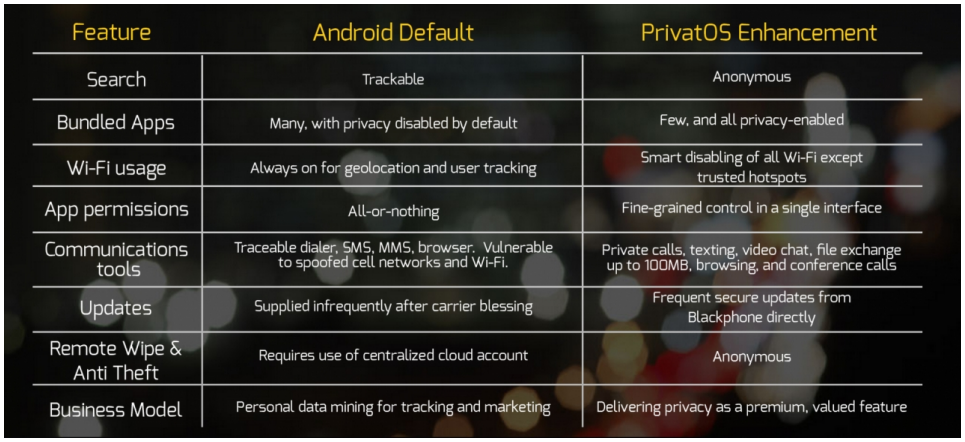
company ISP, or may cause secure work information to be in contact with personal applications, these actions won't be permitted.

On the other side, employees are able to access information and applications related to their personal lives, while staying connected to important work information when they need to perform. Finally, another known feature is also offered by Blackberry, so if the device gets lost or is stolen, or if the employee leaves the organization, there will be an option to wipe just work information which can be done remotely.

2.1.6. Blackphone

One of the most acclaimed devices that were presented in this year Mobile World Congress (Barcelona, Spain), was the Blackphone (Costa, 2014). It is a phone developed with the objective of keeping the user safe at all moments, so it is a perfect match for a BYOD ecosystem. The data stored on the device can be wiped remotely at any time, and the use of the Blackphone Security Center is really helpful for the user, as it is possible completely control every application that is installed.

Blackphone's Operative System, called PrivateOS, is a version of Android customised to offer the maximum security, and it has the hability, for example, of ignoring any Wi-Fi spot except for those that are trusted by the user. In Figure 2.3 there is a list of the main advantages and security and



Feature	Android Default	PrivatOS Enhancement
Search	Trackable	Anonymous
Bundled Apps	Many, with privacy disabled by default	Few, and all privacy-enabled
Wi-Fi usage	Always on for geolocation and user tracking	Smart disabling of all Wi-Fi except trusted hotspots
App permissions	All-or-nothing	Fine-grained control in a single interface
Communications tools	Traceable dialer, SMS, MMS, browser. Vulnerable to spoofed cell networks and Wi-Fi.	Private calls, texting, video chat, file exchange up to 100MB, browsing, and conference calls
Updates	Supplied infrequently after carrier blessing	Frequent secure updates from Blackphone directly
Remote Wipe & Anti Theft	Requires use of centralized cloud account	Anonymous
Business Model	Personal data mining for tracking and marketing	Delivering privacy as a premium, valued feature

Figure 2.3: The features are compared with the ones in Android from the point of view os security and user privacy. Source: `https://www.blackphone.ch`

privacy enhancements that PrivatOS has with respect to a standard Android OS.

2.1.7. Citrix

XenMobile with Worx App SDK, made by Citrix Systems, provides BYOD security services to companies using fine-grained policies to prevent users from performing unallowed actions (like using the mobile phone's camera, GPS or microphone) (Citrix, 2013). These policies can be turned on or off using its own GUI. Citrix is framework-enabled, and it is aware of some (or all) apps installed on the device. All the apps that are Worx-enable are capable of interacting, and thus offering the user a better experience.

It is also important to point out that Citrix differenciates between apps used by the user privately and those used for business, locating both on a secure mobile container that is encrypted, and can be locked remotely for safety reasons. Another feature of Citrix is that it uses dedicated micro VPN to connect to Citrix-protected backend services.

2.1.8. WSO2 Enterprise Mobility Manager

WSO2 Enterprise Mobility Manager (WSO2 EMM) (WSO2) is an open source platform that also works with the BYOD program. Some of this WSO2 EMM key features are:

Mobile Device Management that is used to manage both user and corporate owned devices, providing support for Android and iOS at the

moment. It should be noted that this tool allowes the tracking of every enrolled device, as well as obtaining reports and analytics of their use.

Mobile Application Management With regard to the software, this platform is able to allow or deny the use of applications on enrolled devices based on the role of the user or policies, thus restricting the use of some apps to certain users.

Enterprise App Store This store provides users with both enterprise and public applications approved by the company.

Mobile Data Security WSO2 also allows the user's data to be encrypted via password.

2.1.9. Azzurri Icon Mobile Device Management

ICON Mobilise is a Managed Service that enables organisations to secure, manage and maximise their mobile devices and application. Azzurri (Communications) offers ICON (Intelligent Cloud Optimised Network) as a cloud mobile management tool for Blackberry, Windows Phone, Android and iOS.

Although Blackphone's main feature was his ultra-secure customised OS, ICON centrally deployes, administers and secures all mobile devices regardless of type or OS.

2.1.10. Oesis Framework

OESIS Framework (Opswat., 2014) is a cross platform, open development framework that enables software engineers and technology vendors to develop products that detect, classify and manage thousands of third-party software applications. The extensive functionality of this robust framework gives solutions the ability to perform detailed endpoint assessment and management on Windows, Mac, Linux, and mobile devices.

2.2. Extracting knowledge from Data

Due to the nature of the data (URL accesses performed by humans), the used set of data is highly unbalanced (Chawla, 2005). In order to deal with this problem there exist several methods in the literature, but all of them are mainly grouped in three techniques (Japkowicz, 2002):

- *Undersampling the over-sized classes*: i.e. reduce the considered number of patterns for the classes with the majority.
- *Oversampling the small classes*: i.e. introduce additional (normally synthetic) patterns in the classes with the minority.

- *Modifying the cost associated to misclassifying the positive and the negative class* to compensate for the imbalance ratio of the two classes. For example, if the imbalance ratio is 1:10 in favour of the negative class, the penalty of misclassifying a positive example should be 10 times greater.

The first option has been applied in some works, following a random undersampling approach (Guo et al., 2008), but it has the problem of the loss of valuable information.

The second has been so far the most widely used, following different approaches, such as SMOTE (Synthetic Minority Oversampling Technique) (Chawla et al., 2002), a method proposed by Chawla et al. for creating 'artificial'samples for the minority class, in order to balance the amount of them with respect. However this technique is based in numerical computations, which consider different distance measures, in order to generate useful patterns (i.e. realistic or similar to the existing ones).

The third option implies using a method in which a cost can be associated to the classifier accuracy at every step. This was done for instance by Alfaro-Cid et al. in (Alfaro-Cid et al., 2007), where they used a GP (*Genetic Programming*) approach in which the fitness function was modified in order to consider a penalty when the classifier makes a false negative (an element from the minority class was classified as belonging to the majority class). However almost all the approaches deal with numerical (real, integer) data.

One interesting point about URL classification is that the study of the distance between URLs may be based in the distance between two strings, but Blanco et al. (Blanco et al., 2011) argues that the lexical distance between two URLs is not enough to classify them. In addition, the heuristic study of URLs for security purposes in the user side is not a novel practice. Also, the use of Blacklists (in this work, the *denied* URLs) and Whitelists (*allowed* URLs) are very extended practices. For instance, phishing is a problem of security that Sheng et al. and Khonki et al. (Khonji et al., 2011) tried to solve. The first work uses Blacklists as reference to avoid phishing attacks made by e-mail; the second one aims for an heuristic analysis of the URLs domain names and its ranks, in a way that a phished URL can be detected.

Also, doing some web searching we have found that a lot of companies stands for the use of one between Blacklist and Whitelist (Townsend, 2011). While whitelisting is the more restrictive solution and therefore the more secure, we think that the best solution is to use both, and for this reason the set of rules that we used covers a succession of either allowed and denied web sites.

What refers to the used techniques, DM, as well as ML, has been used since long ago in many scientific fields, and given that research in computer security was growing since the eighties (Anderson, 1980), it was in the nineties when these techniques began to be applied to security issues (Clifton y

Marks, 1996).

On the one hand, DM helped to develop new solutions to computer forensics (de Vel et al., 2001), being the researchers able to extract information from large files with events gathered from infected computers. Another important advance took place after the 9/11 events, when *clustering techniques* and *social network analysis* started to be performed in order to detect pontential crime networks (Chen et al., 2003). On the other hand, and more focused on the user side like our approach, there exist some user-centric solutions to problems like user authentication in a personal device, who Greenstadt and Beal (Greenstadt y Beal, 2008) proposed to address using collected user biometrics along with machine learning techniques.

Then, when a ISP is going to be applied, P.G. Kelley et al. (Kelley et al., 2008) found important to include the user in the machine learning process for refining the policy model. They called it *user-controllable policy learning*. Another approach to the refinement of user's privacy policies has been described by Danezis in (Danezis, 2009), for he uses ML techniques over the user's settings in a social network, being capable of restricting permissions to other people depending on their interaction with the user.

In the same line, Lim et al. propose a system (Lim et al., 2008b,a) that evolves a set of computer security policies by means of GP, taking again into account the user's feedback. Furthermore, Suarez-Tangil et al. (Suarez-Tangil et al., 2009) take the same approach as Lim et al., but also bringing event correlation in. These two latter author's works are interesting for ours, though they are not focused on company ISPs - for instance, our case with the allowed or denied http requests -.

2.3. URL filtering

As in this work we are dedicating an important part to URL, its different parts and the way they influence in the classification process, this section is devoted to review some works related with the study of URLs. In particular, we found interesting those works that try to identify malicious sites (like the ones that want to perform a pishing attack). It is also interesting if they study the URL lexical features, like the one performed during this Master Thesis, and because we consider this kind of study better than to download and proccess the page (the thing that in fact is trying to be avoided).

Hence, Kan and Thi (Kan, 2005) focus their work in lexical features in order to classify as dangerous, URLs that were not previously in Blacklists servers. They gather features like the URI components, length, ortographic data, or segments by entropy reduction. Their results are close to 95 % of accuracy.

On the other hand, this work not only focuses in lexical features of the requested URL, but also in other data that appears in the log files. And not

exactly log data but Zhang et al. (Zhang, 2007), with CANTINA, detect pishing URLs by studying lexical features, content related features, and a WHOIS query (obtaining the date when the domain were registered, which if it is too new, it can be less trustful). They also obtain a 95 % of accuracy.

The most important work we have found related to this were of J. Ma et al (Ma, 2011), whose aim is to detect malicious URLs, mainly related with pishing attacks throug e-mailing, but without processing the content or other private data of the user. They extract information from the lexical features (62 % of the total of the gathered features), and also from the host that has the URL. It is important to point out that they perform the study over 100 days, and that they work with a quantity of almost 2 million of features. In addition, they implement an online classifier (instead of a batch one), and obtain a 99 % of accuracy.

Chapter 3

Data description and preprocessing

Substantial data was received, master, yes, however I am unable to assimilate it.

K9. Doctor Who. *Full Circle.*

This Chapter is devoted to the first part, a very important one, in every Data Mining process: data preprocessing. In the following sections, the data we have worked with will be deeply described, as well as the way it has been preprocessed. At the end, data will be ready to be analysed with a methodology that will be specified in the next Chapter.

3.1. Introduction

The data (log files as well as security policies and rules) we have been working with belong to a real Spanish company that volunteered to donate them for their academic study. This way, we have started from a series of policies written in Drools language (as described in Section 3.2.1), and a log of connections made by company employees.

In order to perform an analysis over the obtained logs such as more frequent events, dangerous or suspicious IPs (according to the connections they made), or more triggered rules, some steps have been followed. This analysis could provide the company CSO with mechanisms to visualise interesting facts about the data. Given that, the following steps have been included in our work:

Classification The aim is to use database information (logs) to look for patterns that had been allowed or denied, in order to build a classifier.

When an incoming connection (or session) has no rule to apply on, the classifier should tell the similarity with previous (and already labelled) patterns.

Clustering The patterns could be grouped considering some similarity criteria, in order to deal with them as a set. This could be used for providing data visuallization mechanisms. In order to make it easier to interpret the data interaction and the distribution in clusters with respect to the different properties/features of the patterns.

Feature Classification It consists of extracting the most important features from the data. This could be done by means of one of the previous techniques, and could be useful if we want to discard non-key features. This is useful in order to reduce the database weight, or for improving the classification running time and even the performance of the system.

3.2. Security Policies vs. Rules

Goguen et al. (Goguen, 1982) defined the concept of *security policy* as the definition of certain requirements needed for a system to be secure. However, in this work we have decided to separate between those, which we consider written in formal language (as any official document in a company), and the *rules*, which are derived from them, and are the specification of the policies in a programming language. It can be said, also, that 'a policy is composed by a set of rules'. In this section we detail how to obtain a set of rules in a format that we can then apply to a set of log data (Section 3.3).

3.2.1. Extracting rules from Policies

In this work we have considered Drools (Team, 2013c) as the tool to create, and therefore, manage rules in a business environment. This so called Business Rule Management System (BRMS) has been developed by the JBoss community under an Apache License and it is written in Java. Though this platform consist of many components, here we focus on Drools Expert and the DRL (*Drools Rule Language*) (Team, 2013b)). Then, the defined rules for a certain company are inside of a file with a `.drl` extension, the file that needs to be parsed to obtain the final set of rules. In Figure 3.1, (a), there is the typical rule syntax in DRL. Two main things should be obtained from the parsing method: both left and right sides of the rule, taking into account that the left side is where the company specifies the conditions required to apply the action indicated in the right side. Also, for describing the conditions, Squid syntax is used (see Section 3.3.1), having thus the following structure: `squid:Squid(`*`conditions`*`)`. Finally, from the right side

of the rule, the *ALLOW* or *DENY* label to apply on the data that matches with the conditions, will be extracted.

The Perl parser that we have implemented applies two regular expressions, one for each side of the rule, and returns a hash with all the rules with the conditions and actions defined. The 'before and after'performing the parsing over the `.drl` file is in Figure 3.1 (b and c). They show how the rule has been *translated* into Perl and Java languages. Figure 3.1 (b) shows the rule as a Perl hash of hashes; the main key is the number of the rule (for instance, 'rule15'meaning 'the 15th rule in the set of rules'), and the value is another hash that includes the fields of the rule:

field is a string that contains the data type (content type, main content type, url, bytes, etc).

relation is a string with four possible values: $<$, $>$, and $==$ for numbers, and *matches* for comparing strings.

value is a string the value of the data type in *field* that we are looking for.

action as shown in the Figure, a string indicating if the rule is supossed to deny or allow the pattern that meets the conditions.

On the other hand, Figure 3.1 (c) shows the way to represent the same rule but as a Java Object. The attributes of its constructor are: a `String action`, the same as in the Perl hash, and a Java List of another Java Object that we have define, `Condition`. Thus, Figure 3.2 depicts the constructor of this Condition Object. This Object has three strings: `dataType`, `relationship`, and `value`. They have the same meaning as in the Perl hash.

3.3. Company Log

The analysed data come from an `access.log` of the Squid proxy application (Team, 2013a), in an actual Spanish company. This open source tool works as a proxy, but with the advantage of storing a cache of recent transactions so future requests may be answered without asking the origin server again (Wessels, 2004). Every pattern, namely a URL session has ten variables associated, which we describe in Table 3.1, indicating if the variable is numeric or nominal/categorical.

The dependent variable or class is a label which inherently assigns an decision (and so the following action) to every request. This can be: *ALLOW* if the access is permitted according to the ISP, or can be *DENY*, if the connection is not permitted. These patterns are labelled using an 'engine'based in a set of security rules, that specify the decision to make. This process is described in Chapter 4.

Variable name	Description	Type	Rank/Number of Values (if categorical)
`http_reply_code`	Status of the server response	Categorical	20 values
`http_method`	Desired action to be performed	Categorical	6 values
`duration_milliseconds`	Session duration	Numerical	integer in [0,357170]
`content_type`	Media type of the entity-body sent to the recipient	Categorical	11 values (main content), 85 values (whole content)
`server_or_cache_address`	IP address	Categorical	2343 values
`time`	connection hour (in the day)	Date	00:00:00 to 23:59:59
`squid_hierarchy`	It indicates how the next-hop cache was selected	Categorical	3 values
`bytes`	Number of transferred bytes during the session	Numerical	integer in [0,85135242]
`client_address`	IP address	Categorical	105 values
`URL`	Core domain of the URL, not taking into account the TLD	Categorical	976 values

Table 3.1: Independent Variables corresponding to a URL session (a connection to a URL for some time). The URLs are parsed as detailed in Section 3.3.1.

```
rule "name"
     attributes
     when
          /* Left Side of
          the Rule */
     then
          /* Right Side of
          the Rule */
end
```

(a) Drools Rule

```
%rules = (
     rule =>{
          field =>xxx
          relation =>xxx
          value =>xxx
          action =>[allow,
          deny]
     },
);
```

(b) Rule as a Perl hash

```
public class Rule{
     private List<Condition> conditions;
     private String action;
     public Rule (List<Condition> conditions, String action){
          this.conditions = conditions;
          this.action = action;
     }
}
```

(c) Object Rule in Java

Figure 3.1: (a) Structure of a rule in Drools Expert. (b) Resulting rule, after the parsing, in a global hash of rules in Perl. (c) Java constructor of the object type Rule.

During the time of research for this Master Thesis, we have had access to a set containing data that were gathered along a period of two hours, from 8.30 to 10.30 am (30 minutes after the work started), monitoring the activity of all the employees in a medium-size Spanish company (80-100 people), obtaining 100000 patterns. We consider this dataset quite complete because it contains a very diverse amount of connection patterns, going from personal (traditionally addressed at the first hour of work) to professional issues (the rest of the day). The file was in CSV format.

3.3.1. Extracting data from a Log File

Usually, the instances of a log file have a number of fields, in order to have a registration of the client who asks for a resource, the time of the day when the request is made, and so on. In this case, we have worked with an *access.log* file, converted into a CSV format file so it could be parsed and transformed in another hash of data. All ten fields of the Squid log yield a hash like the ones depicted in Figure 3.3 and Figure 3.4.

Once the two hashes of data were created, they were compared in such

```
1  public Condition(String dataType, String relationship, String value) {
2
3        this.dataType = dataType;
4        this.relationship = relationship;
5        this.value = value;
6
7     }
```

Figure 3.2: Constructor method of the object type Condition in Java.

a way that for each rule in the hash of rules, it was determined how many entries in the data log hash are covered by the rule, and so they were applied the label that appears as 'actionín the rule.

One of the problems was to extract from a whole URL the part that was more interesting for our purposes. It is important to point out that in a log with thousands of entries, an enormous variety of URLs can be found, since some can belong to advertisements, images, videos, or even some others does not have a domain name but are given directly by an IP address. For this reason, we have taken into account that for a domain name, many subdomains (separated by dots) could be considered, and their hierarchy grows from the right towards the left. The highest level of the domain name space is the TLD (*Top Level Domain*) at the right-most part of the domain name, divided itself in country code TLDs and generic TLDs. Then, a domain and a number of subdomains follow the TLD (again, from right to left). In this way, the URLs in the used log are such as *http://subdomain...subdomain.domain.TLD/ other_subdirectories*. However, for the ARFF[1] file to be created, only the domain (without the subdomains and the TLD) should be considered, because there are too many different URLs to take into consideration. Hence, applying another regular expression, the data parser implemented in Perl obtains all the core domains of the URLs, which makes 976 domains in total.

3.3.2. Legal aspects

Taking into account that the study performed along this Master Thesis has the objective of being finally implemented inside a real system (see Appendix A), we must identify the issues related to legal aspects. Then, the set of rules to apply to a specific unlabelled connection intent, are selected depending on some sensitive data related to the user, such as:

- User's work station IP. This could be useful for limiting the rules to apply, since they can be grouped according to it. The identification of an individual user could be considered by the system if the aim is to adapt the set of rules to an especific person (it would be an important

[1] Format of Weka files.

“http_reply_code”;“http_method”;“duration_milliseconds”;“content_type”;
“server_or_cache_address”;“time”;“squid_hierarchy”;“bytes”;
“url”;“client_address”

200;“GET”;1114;“application/octet-stream”;
“x.x.x.x”;“08:30:08”;“DEFAULT_PARENT”;106961;
“http://www.one.example.com”;“x.x.x.x”

(a) Squid Log Entry in CSV

```
%logdata = (
   entry =>{
      http_reply_code =>xxx
      http_method =>xxx
      duration_miliseconds =>xxx
      content_type =>xxx
      server_or_cache_address =>xxx
      time =>xxx
      squid_hierarchy =>xxx
      bytes =>xxx
      url =>xxx
      client_address =>xxx
   },
);
```

(b) Hash Log Entry in Perl

Figure 3.3: (a) Structure of a log entry (fields and an entry example) of Squid. (b) Perl hash with an example entry. The actual hash used for this work has a total of 100000 entries, with more than a half labelled as *ALLOW* or *DENY* after the comparing process.

feature if we want to build a user-centric system, for example). In this case an anonymisation step should be done (encryption), maybe during the data gathering step, to avoid a leak of information in the event of hacking.

- If we can identify the user's IP, his or her behaviour too. This means that for a certain user, it can be known the amount of time spent in leisure, thus, not being productive. However, this kind of information is not that important from the security point of view, so any conclusion about that should be erased for the sake of the user's privacy.

```
1  public LogEntry (int http_reply_code, String http_method, int
       duration_milliseconds, String content_type_MCT, String
       content_type, String server_or_cache_address, String time, String
       squid_hierarchy, int bytes, String url, String url_core, String
       url_TLD, String client_address) {
2        this.http_reply_code = http_reply_code;
3        this.http_method = http_method;
4        this.duration_milliseconds = duration_milliseconds;
5        this.content_type_MCT = content_type_MCT;
6        this.content_type = content_type;
7        this.server_or_cache_address = server_or_cache_address;
8        this.time = time;
9        this.squid_hierarchy = squid_hierarchy;
10       this.bytes = bytes;
11       this.url = url;
12       this.url_core = url_core;
13       this.url_TLD = url_TLD;
14       this.client_address = client_address;
15   }
```

Figure 3.4: Constructor method of the object type Log Entry in Java.

Chapter 4

Methodology

DOCTOR: Oh, that's never going to work, is it? CLARA: What's wrong? Do you think it's not done yet?

11th Doctor and Clara. Doctor Who. *The Time of the Doctor.*

This Chapter describes the two implementations that followed this research work. First, a fast analysis of both security policies and log data files required a language whith tools for an easy implementation of a parser for these kind of files. Section 4.1 describes why Perl was used for this task and how the first system (as a set of Perl scripts) was implemented. Then, for processing the data that has been analysed, and trying as many classification methods as possible, to see which ones work better with our data, we worked with Weka. Followed methodology with Weka is reported in section 4.2. Finally, after a first round of experiments was done, before continue doing more experiments, and due to the high quantity of different files that needed to be created (separation of the original file in training and test files, as explained in Section 4.4), a second implementation, not with Perl, was proposed. That implementation was required to be compatible with the use of Weka, in order to make things faster and automated. As Weka is developed in Java, the second part of the implementation was made in it, and the details are in Section 4.3.

4.1. First implementation, Perl

Perl was chosen for the initial development of the environment in which the data would be prepocessed and experiments would be developed. As a scripting language, it allows a faster application development than other languages such C or Java (although we have used Java in this work too, and

in Section 4.3 it is explained why), and faster performance too. In (O'Reilly, 2007), Tim O'Reilly, founder and CEO (*Chief Executive Officer*) of O'Reilly Media Inc [1], and Ben Smith deeply describe the advantages of using Perl for every application, and also they present 11 actual cases in where the use of Perl enhanced their situations.

But the most important advantage of Perl is that is the best language for Information Processing. Thus, by means of its regular expressions, patterns of any type (depending on what the programmer is looking for) can be recognised, processed, stored, or studied. And even though Java uses regular expressions too, Perl scripting nature makes easier to use them. Furthermore, URL study and HTML (body of free form text) processing is the ideal application for this Perl feature.

4.1.1. Implementation

We already presented in Sections 3.2.1 and 3.3.1 the structure of the data to be processed and the desired output. As said before, Perl has the possibility of a quick parsing of files (bigger or small size) by using regular expressions. The next subsections describe how the parsing methods for Drool Rules and Squid Log Files were implemented. Then, it is explained how we ended with a Weka File (ARFF format) after a labelling process, and this way we accomplish the first objective of this work (see Section 1.2).

4.1.1.1. Rules

As seen in Figure 3.1 (a), we are interested in both left and right side of the rule. In Table 4.1 is depicted an example of the format of those sides of a normal rule, extracted from the policy "Every intent of visualisation of a MP4 video streaming will be denied from the system". More precisely, this example rule has four conditions, which have to occur in a log entry to finally apply a *DENY* over it. At the same time, a condition has three fields: data type, relationship, and value.

To this end, we have applied the regular expressions defined in Figure 4.1 and obtained a hash with the format shown in Figure 3.1 (b).

/* Left Side of the Rule */
squid:Squid(dif_MCT == "video", bytes >1000000, content_type matches "*.application.*, url matches "*.p2p.*)
/* Right Side of the Rule */
PolicyDecisionPoint.deny();

Table 4.1: Specification of a Rule in Drools, using also Squid syntax.

[1] `http://www.oreilly.com/`

```
1 #Regular expressions to detect the left side of the rule: the
      conditions.
2 /\s*(.*)(==)"(.+)"/
3 /\s*(.+)([>|<|=])(\d+)/
4 /\s*(.+) (.+) "?\*\.(.+)\.\*"?/
5 #Regular expression to detect the action to apply to a log entry that
      meets the conditions.
6 /^\D+\.(.+)\(\)\;/
```

Figure 4.1: Perl regular expressions for obtaining information from the Rules.

4.1.1.2. Log Data

The first file to be processed was already mentioned in Section 3.3, and it is a file with CSV extension, with the hundred thousand entries belonging to two work hours of the company. There is an example of one of the log entries in Figure 3.3 (a), and some points have to be taken into account:

1. The first row of the CSV file contains the fields that identify each one of the values in the subsequent rows.
2. The values are separated by a semicolon.
3. Some values are in between quotation marks, and others are not.
4. The format of the data in the values is important when constructing a regular expression (for instance, the time).
5. We are interested in extracting information about some values before storing them. First, "content_type" (MIME types)[2] is usually formed by the MCT (*Main Content Type*) (application, video, etc) and a subtype, and the parsing method should extract both (when present).
6. Also, the lexical features of the URL should be studied (see Section 2.3). In this first research, we have worked with the core domains, due to many many rules in the supplied policies were focused in avoiding employees to enter dangerous sites.

Hence, the created regular expressions for the creation of a hash with the data (shown in Figure 3.3 (b)) are described in Figure 4.2.

4.1.1.3. Labelling and creating the ARFF file

To finally fullfill the first objective, and now that we have two hashes with information, a method should be created in order to see if any log entry meets enough conditions to be labelled. It may happen, though, that for a

[2] `http://www.w3.org/TR/html4/types.html#h-6.7`

```
1  #Cleaning quotation marks of the values which have them
2  /"(.+)"/
3  #Detecting and storing separately the Main Content Type and the
       subtype (in case there is a subtype)
4  /^(\w+-*\w+)[\/?]\w+/
5  #Detecting time format
6  /^\d{1,2}\:\d{2}\:\d{2}/
7  #Regular expressions related to URLs, they are already prepared for
       storing other lexical properties for its study, such as Top Level
       Domains, subdomains, and paths.
8  /^(ht|f)tps?:\/\/([\.\-\w]*)\.([\-\w]+)\.(\w+)\/[\/*\w*]*/
9  /^(ht|f)tps?:(\/\/)([\-\w]+)\.(\w+)\/[\/*\w*]*/
10 /^NONE\:\/\// #This has to be included in order to avoid carrying
       errors during the process.
```

Figure 4.2: Perl regular expressions for obtaining information from the Log File.

certain connection, more than one rule could be applied. If those rules have the same action to take (all allow the action, or all deny it), there is no conflict, but it is the other way around. When a conflict appears between an allow and a deny label, we always apply the deny label over the allow, for security reasons.

Therefore, we make a verification for each entry, to see what rules can be apply to it and, at the end, we add a label as another field of the data hash. If there is no rule to apply to the connection in the log entry, the label field contains *no_ label*.

Finally, data is needed to be in a format that Weka can manage, and hence we translate the obtained hash of labelled data to ARFF format, they way it is described in Figure 4.3. Ir worths a comment that in Perl is very easy to obtain the rank of values for a categorical attribute, by following this process:

1. Create a hash for each categorical attribute.

2. Go over the values of the attribute, adding them to the hash as keys.

3. If a value is repeated, the value of the pair "key-value" is incremented by 1.

4. We obtain the rank of values simply accessing to the array of keys of the hash.

Then, we fill the rest of the ARFF file by adding the same entries as were in the CSV, but taking care of the new features (including them in the same order as they are initialised in the attributes specification) and their format.

```
my $header=<<EOC;
\@RELATION logsUrl

EOC
  $header .= "\@ATTRIBUTE http_reply_code { ".join(",", @respuestas ).
    " }\n\@ATTRIBUTE http_method { ".join(",", @metodos).
      " }\n\@ATTRIBUTE duration_milliseconds REAL".
   "\n\@ATTRIBUTE content_type_MCT { ".join(",", @MCTs ).
   " }\n\@ATTRIBUTE content_type { ".join(",", @ctype ).
   " }\n\@ATTRIBUTE server_or_cache_address { ".join(",", @serveradd )
       .
   " }\n\@ATTRIBUTE time DATE \"HH:mm:ss\"".
   "\n\@ATTRIBUTE squid_hierarchy { ".join(",", @squidh ).
   " }\n\@ATTRIBUTE bytes REAL".
   "\n\@ATTRIBUTE url { ".join(",", @coredomains ).
   " }\n\@ATTRIBUTE client_address { ".join(",", @clientadd ).
   " }\n\@ATTRIBUTE label { ".join(",", @etiquetas ).
   " }\n\n\@DATA\n";

my $salida = "$header\n";
my $count = 0;
for my $r (@rows ) {
   $salida .= join(", ", @$r )."\n";
   $count++;
}
```

Figure 4.3: Perl code for transforming a hash of data into ARFF format, in order to be read by Weka.

4.2. Weka

As said in Section 3.3, the data used for this work is not only numerical or nominal, thus, only classification algorithms that support both types of data have been considered. Weka (of Waikato, 1993) is a collection of State-of-the-Art machine learning algorithms and data preprocessing tools that are key for data mining processes (Frank y Witten, 2011). With such a great number of possible algorithms to work with, we have conducted a preselection phase trying to choose those which would yield better results in the experiments. More specifically, we have focused on rule-based and decision-tree-based algorithms.

In this way, a decision-tree algorithm is a group of conditions organised in a top-down recursive manner in a way that a class is assigned following a path of conditions, from the root of the tree to one of its leaves. Generally speaking, the possible classes to choose are mutually exclusive. Furthermore, these algorithms are also called "divide-and-conquer" algorithms. On the other hand, there are the "separate-and-conquer" algorithms, which work creating rules one at a time, then the instances covered by the created rule are removed and the next rule is generated from the remaining instances.

A reference to each Weka classifier can be found at (Frank y Witten,

2011). Below are described the top five techniques, obtained from the best results of the experiments done in this stage, along with more specific bibliography. Naïve Bayes method (Domingos y Pazzani, 1997) has been included as a baseline, normally used in text categorization problems. According to the results, the five selected classifiers are much better than this method. Deeper results are shown in Chapter 5.

Naïve Bayes It is the classification technique that we have used as a reference for either its simplicity and ease to understand. Its basis relies on the Bayes Theorem and the possibility of represent the relationship between two random variables as a Bayesian network (Rish, 2001). Then, by assigning values to the variables probabilities, the probabilities of the occurrences between them can be obtained. Thus, assuming that a set of attributes are independent one from another, and using the Bayes Theorem, patterns can be classified without the need of trees or rule creation, just by calculating probabilities.

J48 This classifier generates a pruned or unpruned C4.5 decision tree. Described for the first time in 1993 by (Quinlan, 1993), this machine learning method builds a decision tree selecting, for each node, the best attribute for splitting and create the next nodes. An attribute is selected as 'the best'by evaluating the difference in entropy (information gain) resulting from choosing that attribute for splitting the data. In this way, the tree continues to grow till there are not attributes anymore for further splitting, meaning that the resulting nodes are instances of single classes.

Random Forest This manner of building a decision tree can be seen as a randomization of the previous C4.5 process. It was stated by (Breiman, 2001) and consist of, instead of choosing 'the bestáttribute, the algorithm randomly chooses one between a group of attributes from the top ones. The size of this group is customizable in Weka.

REP Tree Is another kind of decision tree, it means Reduced Error Pruning Tree. Originally stated by (Quinlan, 1987), this method builds a decision tree using information gain, like C4.5, and then prunes it using reduced-error pruning. That means that the training dataset is divided in two parts: one devoted to make the tree grow and another for pruning. For every subtree (not a class/leaf) in the tree, it is replaced by the best possible leaf in the pruning three and then it is tested with the test dataset if the made prune has improved the results. A deep analysis about this technique and its variants can be found in (Elomaa y Kaariainen, 2001).

NNge Nearest-Neighbor machine learning method of generating rules using non-nested generalised exemplars, i.e., the so called 'hyperrectangles'for

being multidimensional rectangular regions of attribute space (Martin, 1995). The NNge algorithm builds a ruleset from the creation of this hyperrectangles. They are non-nested (overlapping is not permitted), which means that the algorithm checks, when a proposed new hyperrectangle created from a new generalisation, if it has conflicts with any region of the attribute space. This is done in order to avoid that an example is covered by more than one rule (two or more).

PART It comes from 'partial'decision trees, for it builds its rule set from them (Frank y Witten, 1998). The way of generating a partial decision tree is a combination of the two aforementioned strategies "divide-and-conquer" and "separate-and-conquer", gaining then flexibility and speed. When a tree begins to grow, the node with lowest information gain is the chosen one for starting to expand. When a subtree is complete (it has reached its leaves), its substitution by a single leaf is considered. At the end the algorithm obtains a partial decision tree instead of a fully explored one, because the leafs with largest coverage become rules and some subtrees are thus discarded.

These methods are then deeply tested on the datasets that result from applying balancing methods over the initial one, and by makin a partition in training and test files. These processes are described in Section 4.4.

4.3. Second implementation, Java

Weka had been implemented in Java, and for that reason we started a second implementation with this language. As said in the previous Section 4.1, the developing time with Java is higher, but in general the performance is better (see results in Section 4.3.2), and we also have the possibility of developing a method for lauching the experiments and not doing them one by one. But if Perl had the advantage of being the best for preprocessing with the regular expressions, Java has the feature of directly process Weka output and, this way, obtaining reports and rankings with the best techniques.

4.3.1. Implementation

The part of data preprocessing through regular expressions is not worth to be detailed, because they are the same as the ones in Perl (see Section 4.1.1), with little or no difference, and implemented with `Pattern` and `Matcher` classes. One difference with Perl is that here is not necessary to have both rules and log data hashes to compare, but here we have created three objects: `Condition.java`, `Rule.java`, and `LogEntry.java`, so the data is stored as an `ArrayList` of the correspondant objects. These objects has been described in Chapter 3.

Now, with regard to the direct management of Weka in Java, an *ExperimentRunner.java* has been implemented and is called like is shown in Figure 4.4. There, it can be seen the syntax for calling the classifiers. Then, starting from the location where the ARFF file is, and the desired type of classifier, we create an Object of the type `Experiment` (from the weka.jar JAR) and start to customise it:

```
public static Double experimenter(String[] ARFF_File_input,
                                  String classifier_type)
                                  throws Exception {

System.out.println("Setting up...");
Experiment exp = new Experiment();
exp.setPropertyArray(new Classifier[0]);
exp.setUsePropertyIterator(true);

SplitEvaluator se  = new ClassifierSplitEvaluator();
Classifier sec = ((ClassifierSplitEvaluator) se).getClassifier();
```

In the code above it is also specified that we are performing classification and not regression. After that, we specify that (for now) the classifier will be trained and tested by a crossvalidation process with 10 folds. This means that the dataset is divided in 10 equal parts and that the classifier will be trained an tested 10 times: each time it is trained with 9 of the 10 parts and the 10th is for testing. After the 10 generations, the results are calculated by averaging.

```
CrossValidationResultProducer cvrp = new CrossValidationResultProducer();
cvrp.setNumFolds(10);
cvrp.setSplitEvaluator(se);

PropertyNode[] propertyPath = new PropertyNode[2];
try {
 propertyPath[0] = new PropertyNode(se,
  new PropertyDescriptor("splitEvaluator",
    CrossValidationResultProducer.class),
    CrossValidationResultProducer.class);
 propertyPath[1] = new PropertyNode( sec,
    new PropertyDescriptor("classifier", se.getClass()), se.getClass());
} catch (IntrospectionException e) {
    e.printStackTrace();
}
exp.setResultProducer(cvrp);
exp.setPropertyPath(propertyPath);
```

The next step is to set the times that the experiment will be repeated.

```
exp.setRunLower(1);
exp.setRunUpper(1);
```

And then it is time for setting up the classifier. First, the Weka method `Utils.splitOptions(classifier_type)` takes the argument with which we called our ExperimentRunner, and given its syntax (see Figure 4.4), parses the options. With those options we build an Object of the type `Classifier`.

```
String[] options = Utils.splitOptions(classifier_type);
String classname = options[0];
options[0]  = "";
Classifier c = (Classifier) Utils.forName(Classifier.class,
                                          classname,
                                          options);
exp.setPropertyArray(new Classifier[]{c});
```

What is left is to tell the Experimenter where to find the data set whose data we want to use for training and test the classifier. We set the data set with `exp.setDatasets(model)` and the configuration is almost finished.

```
DefaultListModel model = new DefaultListModel();
File file = new File(ARFF_File_input[1]);
model.addElement(file);
exp.setDatasets(model);
```

All the results are displayed in a file with ARFF extension, like the one which contains the dataset to process. This way, here we set up the name of the output file (the same as the input one but with the suffix *_experimenter_results*), and the location, the same as the input file.

```
InstancesResultListener irl = new InstancesResultListener();
String ARFF_File_name = ARFF_File_input[0].substring(0,
                                          ARFF_File_input[0].length()-5);
File ARFF_File_output = new File("/home/paloma/workspace/KRS_Prototype
                    /ARFF/"+ARFF_File_name+"_experimenter_results.arff");
irl.setOutputFile(ARFF_File_output);
exp.setResultListener(irl);
```

Now, the configuration of the experiment is complete and we finish the method by initialising and running the experiment. But at the end, in Figure 4.5 it can be seen how the results of the experiments are managed and stored in the set ARFF output file.

```
1  /* Definition of the Weka experiments. */
2  String[] experiments = new String[11];
3  experiments[0] = "NaiveBayes";
4  experiments[1] = "DecisionTable -X 1 -S \"weka.attributeSelection.
       BestFirst -D 1 -N 5\"";
5  experiments[2] = "JRip -F 3 -N 2.0 -O 2 -S 1";
6  experiments[3] = "OneR -B 6";
7  experiments[4] = "PART -M 2 -C 0.25 -Q 1";
8  experiments[5] = "ZeroR";
9  experiments[6] = "DecisionStump";
10 experiments[7] = "J48 -C 0.25 -M 2";
11 experiments[8] = "RandomForest -I 10 -K 0 -S 1 -num-slots 1";
12 experiments[9] = "RandomTree -K 0 -M 1.0 -V 0.001 -S 1";
13 experiments[10] = "REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0";
14
15 /* Array percentages for storing the accuracy obtained after each
       experiment. */
16 Double[] percentages = new Double[experiments.length];
17
18 /* Loop for launching the experiments. */
19 int i;
20 for ( i = 0; i < experiments.length; i++) {
21    System.out.println("Launching "+experiments[i]+"...");
22    Double temp_percentage = ExperimentRunner.experimenter(
          ARFF_File_name, experiments[i]);
23    if (!temp_percentage.isNaN()) {
24       percentages[i] = temp_percentage;
25       System.out.println(percentages[i]);
26    }
27 }
```

Figure 4.4: Java code for setting up the experiments with Weka and obtain an array of percentages of successfulness with each classifier.

```
System.out.println("Initializing...");
exp.initialize();
System.out.println("Running...");
exp.runExperiment();
System.out.println("Finishing...");
exp.postProcess();
```

Our method returns a *double* variable with the percentage of correct predictions by the classifier or, in other words, its accuracy. Once we have the whole array of percentages, it is easy to choose which ones will be tested with the rest of the files, that have been obtained as detailed in Section 4.4.

4.3.2. Perl performance vs. Java performance

There are two implementations, in both Perl and Java, and each one has its advantages and disadvantages. In this subsection, there is a study of the time spent for each language in executing the tasks of parsing the data (Rules, first, and then the Log File), and label the data and storing

```
1  PairedTTester tester = new PairedCorrectedTTester();
2  Instances result = new Instances(new BufferedReader(new FileReader(irl
       .getOutputFile())));
3  tester.setInstances(result);
4  tester.setSortColumn(-1);
5  tester.setRunColumn(result.attribute("Key_Run").index());
6  tester.setFoldColumn(result.attribute("Key_Fold").index());
7  tester.setDatasetKeyColumns(
8  new Range(
9     ""
10    + (result.attribute("Key_Dataset").index() + 1)));
11 tester.setResultsetKeyColumns(
12 new Range(
13    ""
14    + (result.attribute("Key_Scheme").index() + 1)
15    + ","
16    + (result.attribute("Key_Scheme_options").index() + 1)
17    + ","
18    + (result.attribute("Key_Scheme_version_ID").index() + 1)));
19 tester.setResultMatrix(new ResultMatrixPlainText());
20 tester.setDisplayedResultsets(null);
21 tester.setSignificanceLevel(0.05);
22 tester.setShowStdDevs(true);
23
24 /* Fill result matrix (but discarding the output) */
25 tester.multiResultsetFull(0, result.attribute("Percent_correct").index
       ());
26 ResultMatrix matrix = tester.getResultMatrix();
27 return matrix.getMean(0, 0);
```

Figure 4.5: Java code for obtaining the output in an ARFF file as well as returning a double with the accuracy percentage.

the results in Weka format. The results can be found in Table 4.2. They are completely expected because the strength of Perl is that of extracting data, and because of that it spends 15 seconds less than Java performing this task. On the other hand, Java is much more efficient in labelling the data and storing the data once labelled. Actually, that difference is of 24 seconds with respect to Perl.

It is expected to have a quicker performing by developing in Java, but instead of chosing only one, we have just demonstrated that the best performance can be obtained by using both languages, for different tasks.

	Average time (sec)	
Action performed	Perl	Java
Parsing Rules and Data	2.156	16.663
Labelling and Creating ARFF format File	74.355	50.816

Table 4.2: Comparison of the performance time while preprocessing the information between Perl and Java.

4.4. File partitioning

At this point, we have explained how we reached the first objective we set on the Section 1.2 and for accomplish the second, there is a series of different files the should be created, in order to test the classifiers and see how can we obtain the best results. The initial Log File, after being labelled, presents a total of 57502 labelled patterns, which 38972 patterns are allowed and 18530 are denied. We observed that more than half of the initial amount of patterns are labelled, and that the ratio is 2:1 in allows to denies. Hence, after a first set of experiments we obtained the results of Table 5.1. From these results, we took the top five classifiers and then prove them with the rest of the files.

The 2:1 ratio means that the data is unbalanced, and that balancing techniques may affect in the results when applied. Also, as it may be recalled, the data tested with the classifiers was used both for training and test by crossvalidation. For the results to be more accurate, the next step is to build separate training and test files to assure that the data used for training is not used again for testing, and vice versa.

The way they have been generated is pretty trivial, but what matters is that each time that some file is created randomly, me made 3 generations of files and the same experiments with every one of them. Then, if an 80 % training file was created by taking the log entries generating a random number, we repeated the process two times more and at the end we had three sets of training and test files, and three group of five experiments for each one. After that, the means and standard deviations were calculated. The statistics of

the number of patterns and how the labels are distributed can be found at Tables 4.6, 4.7, and 4.9. These are tables that contain the number of samples (patterns, or log entries) that were obteined after each separation on training and test files. So these tables show that the implemented method would effectively split them. As it is shown, percentages are obtained, although not exact, almost equal on random generations, and equal on consecutive.

This being said, the work has been distributed as follows:

Step 1 Figure 4.10 shows the first step: experiments were made with files of unbalanced and balanced data (with both undersampling and oversampling balancing techniques), and then again but with those files separated in training and test files, both randomly and consecutively taken. Two different separations were made: 80 % for training, %20 for testing, and 90 % for tranining, 10 % for testing. The reason for generating these files also keeping the sequence of the initial data and not only randomly, is because the requested URLs, or the time of the requesting, can be related.

Step 2 After having analised the log of connecting patterns, we studied the field *squid_hierarchy* and saw that had two possible values: `DIRECT` or `DEFAULT_PARENT`. The Squid FAQ reference (Team, 2014), and the Squid wiki (Wiki, 2014) explain that, as a proxy, the connections are made, firstly to the Squid proxy, and then, if appropriate, the request continues to another server. These connections are registered in Squid the same way, with the same fields, with the exception of the client and server IP addresses. From the point of view of classification, if one of these two entries happens to be in the training file, and the other in the testing file, it would mean that the second would be well classified because of all the attribute values that both have in common. However, this means also that the good percentages that we obtained may not be real, but faked. That is why the second step is about removing entries that we called "repeated" (in the explained sense). After the removal, a new set of files was created.

Step 3 Another 'issue'related to classification accuracies, was taking care of the repeated core domains. Like with the connections, if a Log Entry with a certain core domain, goes to a training file, and another one (even if the rest of the fields are different) goes to the test file, it may affect the results. As a result, a whole new set of files were created and studied.

Step 4 As it will be described in Section 5.2 of next Chapter, we saw that the rules created by the classifiers were too focused on the URL core domain feature. Thus, we wanted to add another set of experiments, represented as *step 4* in Figure 4.10, because we did the experiments

again with the original file, but including as a feature only the TLD of the URL, and not the core domain. This was the last set of files created for this research work.

On steps 2 and 3, the separation 60 % for tranining, 40 % for testing was added when possible, for studying the generalization of the classifiers. In the next chapter we will present the results of the experiments, and in Chapter 6, the derived conclussions.

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	57502	38972	18530	100	67.78	32.22
Training 80 %	45837	31026	14811	79.71	67.69	32.31
Test 20 %	11665	7946	3719	20.29	68.12	31.88
Training 90 %	51764	35102	16662	90.02	67.81	32.19
Test 10 %	5738	3870	1868	9.98	67.45	32.55

(a) Training and Test files, random

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	57502	38972	18530	100	67.78	32.22
Training 80 %	46002	31178	14824	80.00	67.78	32.22
Test 20 %	11500	7794	3706	20.00	67.77	32.23
Training 90 %	51752	35075	16677	90.00	67.78	32.22
Test 10 %	5750	3897	1853	10.00	67.77	32.23

(b) Training and Test files, consecutive

Figure 4.6: Statistics of the files created from the initial file of unbalanced data. It can be seen that after the partition, the ratio between allows and denies remains the same. (a) Files generated by randomly taking patterns from the original file. (b) Files generated by consecutively taking patterns from the original file.

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	38081	19569	18512	100	51.39	48.61
Training 80 %	30486	15664	14822	80.06	51.38	48.62
Test 20 %	7594	3905	3689	19.94	51.42	48.58
Training 90 %	34184	17536	16648	89.77	51.30	48.70
Test 10 %	3896	2033	1863	10.23	52.18	47.82

(a) Training and Test files, random

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	38081	19569	18512	100	51.38	48.61
Training 80 %	30465	15655	14810	80.00	51.39	48.61
Test 20 %	7615	3914	3701	20.00	51.40	48.61
Training 90 %	34273	17612	16661	90.00	51.39	48.61
Test 10 %	3807	1957	1850	10.00	51.41	48.59

(b) Training and Test files, consecutive

Figure 4.7: Statistics of the files created after the **undersampling** technique was applied. It can be seen that after the partition, the ratio between allows and denies remains the same, and they are balanced now. (a) Files generated by randomly taking patterns from the original file. (b) Files generated by consecutively taking patterns from the original file.

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	76032	38972	37060	100	51.26	48.74
Training 80 %	60748	31104	29644	79.90	51.20	48.80
Test 20 %	15283	7867	7416	20.10	51.48	48.52
Training 90 %	68477	35047	33430	90.06	51.18	48.82
Test 10 %	7554	3924	3630	9.94	51.95	48.05

(a) Training and Test files, random

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	76032	38972	37060	100	51.26	48.74
Training 80 %	60826	31178	29648	80.00	51.26	48.74
Test 20 %	15205	7793	7412	20.00	51.25	48.75
Training 90 %	68429	35075	33354	90.00	51.26	48.74
Test 10 %	7602	3896	3706	10.00	51.25	48.75

(b) Training and Test files, consecutive

Figure 4.8: Statistics of the files created after the **oversampling** technique was applied. It can be seen that after the partition, the ratio between allows and denies remains the same, and they are balanced now. (a) Files generated by randomly taking patterns from the original file. (b) Files generated by consecutively taking patterns from the original file.

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	38619	26318	12301	100	68.15	31.85
Training 80 %	30080	20261	9819	77.89	67.36	32.64
Test 20 %	8539	6057	2482	22.11	70.93	29.07
Training 90 %	33917	22981	10936	87.82	67.76	32.24
Test 10 %	4702	3337	1365	12.18	70.97	29.03
Training 60 %	21771	14519	7252	56.37	66.69	33.31
Test 40 %	16848	11799	5049	43.63	70.03	29.97

(a) Training and Test files, random, 1st generation

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	38619	26318	12301	100	68.15	31.85
Training 80 %	30034	20394	9640	77.77	67.90	32.10
Test 20 %	8585	5924	2661	22.23	69.00	31.00
Training 90 %	33839	22873	10966	87.62	67.59	32.41
Test 10 %	4780	3345	1335	12.38	69.98	27.93
Training 60 %	22447	15213	7234	58.12	67.77	32.23
Test 40 %	16172	11105	5067	41.88	68.67	31.33

(b) Training and Test files, random, 2nd generation

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	38619	26318	12301	100	68.15	31.85
Training 80 %	30096	20198	9898	77.93	67.11	32.89
Test 20 %	8523	6120	2403	22.07	71.81	28.19
Training 90 %	33854	22793	11061	87.66	67.33	32.67
Test 10 %	4765	3525	1240	12.34	73.98	26.02
Training 60 %	22390	15026	7364	57.98	67.11	32.89
Test 40 %	16229	11292	4937	42.02	69.58	30.42

(c) Training and Test files, random, 3rd generation

File	Total	Allow	Deny	% of Original	%Allows	%Denies
Original	38619	26318	12301	100	68.15	31.85
Training 80 %	30895	21054	9841	80.00	68.15	31.85
Test 20 %	7723	5263	2460	20.00	68.15	31.85
Training 90 %	34757	23686	11071	90.00	68.15	31.85
Test 10 %	3861	2631	1230	10.00	68.14	31.86
Training 60 %	23172	15791	7381	60.00	68.15	31.85
Test 40 %	15446	10526	4920	40.00	68.15	31.85

(d) Training and Test files, consecutive.

Figure 4.9: Statistics of the files created after the duplicated connections were removed. It can be seen that after the partitions, the ratio between allows and denies remains the same, unbalanced. (a), (b), and (c) Three generations of files generated by randomly taking patterns from the original file. (d) Files generated by consecutively taking patterns from the original file.

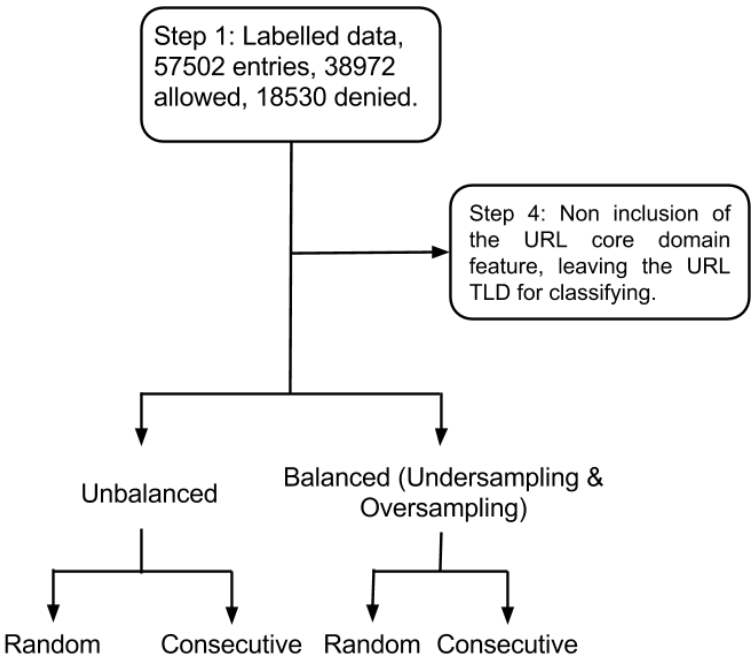


Figure 4.10: This Figure shows that experiments were made with files of unbalanced and balanced data (with both undersampling and oversampling balancing techniques), and again but with those files parted in training and test files, both randomly and consecutively taken.

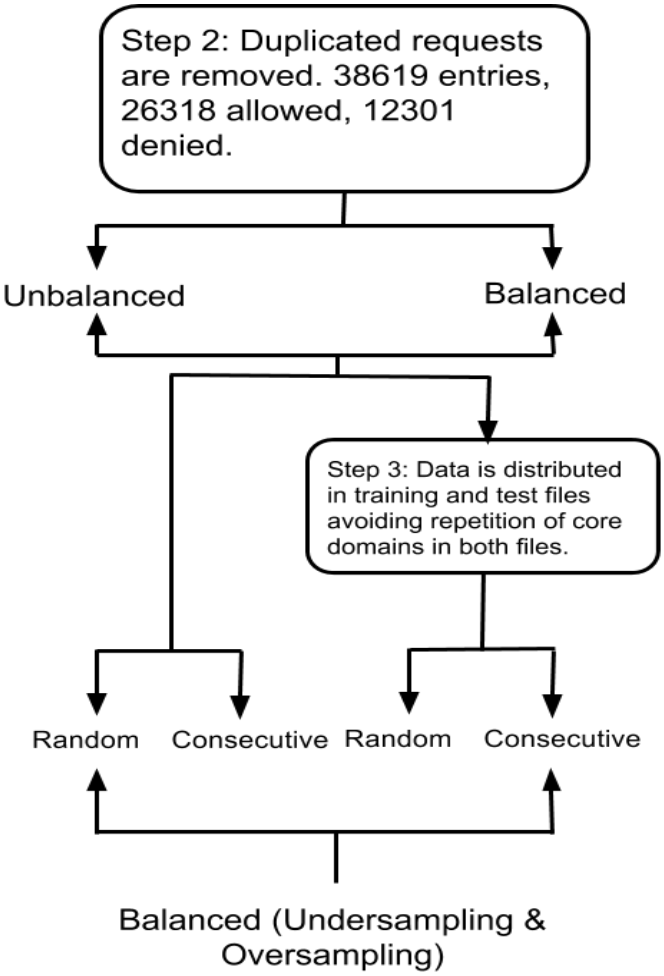


Figure 4.11: Two main files were created. First, requests between the client and the server that passed through the proxy (duplicated, from the point of view of classification training), were reduced. Then, when parting the main file into training and test files, it was made avoiding repetition of core domains in both files.

Chapter 5

Results

You know the Doctor. You understand him. You will predict his actions.

DALEK. Doctor Who. *The Parting of the Ways.*

This Chapter reports all the experiments thas have been conducted, from the first stage using the initial (and unbalanced) log file, to the last in where we have used separated files for training and testing (see Section 4.4). At the beggining, and given that we were working with rules, the initial log file (with the entries that have been labelled as allowed or denied) was tested following a *cross-validation* partition, with all the possible rule and tree classifiers given in Weka. With this initial ranking, we took the five classifiers that obtained the best results and continue testing with all the rest of the partitions. Also, for each section, best results are commented, and therefore conclusions will be given in the next Chapter.

5.1. Experiment results

As explained in Section 4.2, an initial group of experiments was conducted and te results can be consulted at Table 5.1. The top five was formed by the classifiers: J48, Random Forest, REP Tree, NNge, and PART, plus Naïve Bayes as a reference. These five classifiers were then training and tested with both training and test files, as explained in Section 4.4. First results can be found in Table 5.2 [1]. Results in that Table were obtained after a balancing process applied over the initial file, as explained in Section 4.4. Then, when performing *undersampling*, half of the total of the 'allowed'patterns were randomly removed, and when *oversampling* was applied, each of the the

[1] All result tables are publicly accessible through the following link: `https://drive.google.com/folderview?id=0B0Rvxfh8NwQRMi1RYU9uT3FmQzg&usp=sharing`

'denied'patterns were repeated till the total of 'denied'patterns was the same as 'allowed'.

	Undersampling	Oversampling
Naïve Bayes	91.02 ± 0.10	91.77
Conjunctive Rule	60.00 ± 0.14	60.02
Decision Table	94.31 ± 0.20	90.29
DTNB	94.92 ± 0.14	95.65
JRip	90.03 ± 0.07	92.47
NNge	**96.47** ± 0.02	**98.76**
One R	93.53 ± 0.08	93.70
PART	**96.35** ± 0.09	**97.54**
Ridor	86.60 ± 0.55	89.87
Zero R	51.22 ± 0.16	51.26
AD Tree	77.65 ± 0.08	77.68
Decision Stump	60.00 ± 0.14	60.02
J48	**96.99** ± 0.04	**98.00**
LAD Tree	78.93 ± 1.71	79.97
Random Forest	**96.94** ± 0.07	**98.84**
Random Tree	95.59 ± 0.40	98.35
REP Tree	**96.74** ± 0.04	**97.67**

Table 5.1: Results of all tested classification methods on balanced data. The best ones are marked in boldface.

	80 % Training - 20 % Test		90 % Training - 10 % Test	
	Random (mean)	Sequential	Random (mean)	Sequential
Naïve Bayes	91.60 ± 1.25	85.53	92.89 ± 0.12	83.84
J48	97.56 ± 0.20	88.48	**97.70 ± 0.15**	82.28
Random Forest	**97.68 ± 0.20**	**89.77**	97.63 ± 0.13	82.59
REP Tree	97.47 ± 0.11	88.34	97.57 ± 0.01	83.20
NNge	97.23 ± 0.10	84.41	97.38 ± 0.36	80.34
PART	97.06 ± 0.19	89.11	97.40 ± 0.16	**84.17**

Table 5.2: Percentage of correctly classified patterns for unbalanced data. The best ones are marked in boldface.

As it can be seen, all five methods achieved a high performance classifying in the right way the test dataset. The given numbers are the mean taken from three genetations of undersampled files from the original, due to the *random* nature of the process. Also, these results are not like this by chance, as shown by a low standard deviation.

Next step described in Section 4.4 was, then, to separate the original file into training and test files. The results are depicted in Table 5.2. Obtained results for the first division, 80 % of the original dataset taken for training, and the 20 % for testing the classifiers, are quite good (*Random Forest* reached a 97.68 % of accuracy with a standard deviation of 0.2). As was expected,

the results from the 90 %-10 % division were slightly better, and due to this a more aggressive division was executed in the following experiments so the methods can be really proved with much less training data.

What matters to the results of the experiments made with the sequential data, they are worse than the obtained from the random data, but they are still good ($> 85\,\%$). This is due to the occurrence of new patterns from a certain time (maybe there are some requests that are made just at one specific time in a day, or in settled days), and then there is no sufficient similarity between the training data and the classifying of the test data set may fail. The loss of 5 to 6 points in the results of the 90 %-10 % division is the first unexpected or unlogical result of the experiments, but they also reinforce the previous theory.

The technique that lightly stands out over the others is *Random Forest*, being the best in almost every case, even in the experiments with the most complex sequential divisions. Also, *J48* reached high accuracies with a lesser standard deviation. In general, all the obtained results present robustness as can be seen in the standard deviations.

For its part, results obtained from unbalanced data are shown in Table 5.1. As explained in Section 4.4, the corresponding to the random partitions come from the mean of three blocks of experiments, and so are specified the standard deviations. The Table illustrates two segments of results, obtained from the undersampled data and from the oversampled data. For each one, the 90 %-10 % and 80 %-20 % divisions were also made. We distinguish the results depending on:

Applying Undersampling In comparison with those results from Table 5.2, these go down one point (in the case of randomly made divisions) to six points (sequential divisions). The reason why this happens is that when randomly removing ALLOW patterns, we are really losing information, i. e. key patterns that could be decisive in a good classification of a certain set of test patterns.

Applying Oversampling Here we have duplicated the DENY patterns so their number could be up to that of the ALLOW patterns. However, it does not work as well as in other approaches which uses numerical computations for creating the new patterns to include in the minority class. Consequently, the results have been decreased.

It is noticeable in both cases that taking the data in a sequential way, instead of randomly, lower the results. It is clear that due to the fact that performing undersampling some patterns are lost while in the case of oversampling they all remain, *undersampling results* are better. Then, in this case the algorithm with best performance is *J48*, though *Random Forest* follows its results very closely in random datasets processing, and *REP Tree*,

	80 % Training - 20 % Test			
	Undersampling		Oversampling	
	Rand (mean)	Sequential	Rand (mean)	Sequential
Naïve Bayes	91.30 ± 0.20	84.94	91.74 ± 0.13	85.43
J48	**97.05 ± 0.25**	84.29	**97.40 ± 0.03**	85.66
Random Forest	96.61 ± 0.17	**88.59**	97.16 ± 0.19	**89.03**
REP Tree	96.52 ± 0.13	85.54	97.13 ± 0.25	85.41
NNge	96.56 ± 0.42	85.28	96.90 ± 0.28	83.46
PART	96.19 ± 0.14	85.16	96.82 ± 0.09	84.50

(a) 80 % for training, %20 for testing

	90 % Training - 10 % Test			
	Undersampling		Oversampling	
	Rand (mean)	Sequential	Rand (mean)	Sequential
Naïve Bayes	91.18 ± 0.16	**82.35**	**91.77** ± 0.28	**81.81**
J48	96.85 ± 0.35	76.44	97.37 ± 0.06	74.24
Random Forest	**96.99 ± 0.13**	79.98	97.25 ± 0.33	81.33
REP Tree	96.55 ± 0.10	77.65	97.14 ± 0.09	76.81
NNge	96.33 ± 0.05	81.93	96.91 ± 0.06	78.73
PART	96.09 ± 0.10	79.70	96.68 ± 0.11	78.16

(b) 90 % for tranining, 10 % for testing

Figure 5.1: Percentage of correctly classified patterns for balanced data (undersampling and oversampling). (a) 80 % for training, %20 for testing. (b) 90 % for tranining, 10 % for testing. Best results are marked in boldface.

which is better than the rest when working with sequential data. Nevertheless, generally speaking and given the aforementioned reasons, performing data balancing methods yields worse results.

5.1.1. Removal of duplicated requests

After performing the step 2 that is depicted in Figure 4.11, a whole new ranking of classification performance was created, in order to see if the new situation has some influence on the results. However, Table 5.3 shows that the best results are from the same classifiers, with slightly difference when performing oversampling, where *Random Tree* was better than *REP Tree* but less than one point.

The process is again the same, and the results are displayed in Tables 5.2 (a) and (b). We can see that the results are slightly worse than the ones obtained in step 1, but they are still good, and definitely better than Naïve Bayes, our result reference. It is not a surprise that, again, results for files with the patterns taken consecutively lower significantly. As previously, this is because the possible loss of information. Best results are obtained by both *Random Forest* and *REP Tree* classifiers, with a 96 % of accuracy.

	Unbalanced	Undersampling	Oversampling
Naïve Bayes	92.30	90.77 ± 0.06	91.77
Conjunctive Rule	73.31	59.53 ± 0.15	60.02
Decision Table	95.21	93.73 ± 0.18	90.29
DTNB	95.55	94.75 ± 0.07	95.65
JRip	92.95	89.64 ± 0.32	92.47
NNge	**97.18**	**96.33 ± 0.14**	**98.76**
One R	94.86	93.53 ± 0.04	93.70
PART	**97.41**	**96.32 ± 0.10**	**97.54**
Ridor	89.21	86.19 ± 0.79	89.87
Zero R	68.14	51.68 ± 0.17	51.26
AD Tree	85.23	78.01 ± 0.10	77.68
Decision Stump	73.31	59.53 ± 0.15	60.02
J48	**97.37**	**96.65 ± 0.04**	**98.00**
LAD Tree	86.87	80.24 ± 0.04	79.97
Random Forest	**97.57**	**96.75 ± 0.05**	**98.84**
Random Tree	96.11	95.45 ± 0.66	**98.35**
REP Tree	**97.32**	**96.38 ± 0.07**	97.67

Table 5.3: Results of all tested classification methods on unbalanced and balanced data, after the repeated requests have been removed. The best ones are marked in boldface.

5.1.2. Enhancing the creation of training and test files

The last step taken in this research was to try to better separate the patterns from the initial file. Thus, we tried to avoid possible classification errors by assuring that entries with the same core domain end in the same file (either for training o testing) (see Figure 4.11).

It must be pointed out that, for the moment, there are no generations of training and test files with the patterns taken consecutively. The reason for this can be seen also in Table 5.3: taking into account this way of separating the data, when a certain log entry happens to be placed in the training, or testing, file, all the others which share the same URL core domain, will follow to the same file, and therefore is impossible to reach the desired file distribution.

Furthermore, taking a look at the results in Table 5.4, they are by far the worst we obtained. It seems that a lot of information is lost by performing this step. These Tables show a comparison between the tested classifiers during this process. It can be seen that the best classifiers are *J48* and *REP Tree* (between 70.69 % and 74.72 % of accuracy).

5.1.3. Filtering the features of the URL

Last step explained in Section 4.4, was the repetition of the experiments done at the beggining, but studying what would it happen if we trained the

	80 % Training - 20 % Test		90 % Training - 10 % Test	
	Random (mean)	Sequential	Random (mean)	Sequential
Naïve Bayes	93.01 ± 0.32	82.61	93.09 ± 0.91	83.04
Random Forest	**96.97 ± 0.47**	**91.03**	**96.79 ± 0.97**	80.60
J48	96.90 ± 0.26	87.78	96.50 ± 1.00	84.49
NNge	96.21 ± 0.28	81.17	96.11 ± 1.13	81.92
REP Tree	**96.97 ± 0.40**	87.75	96.62 ± 0.87	**85.57**
PART	96.84 ± 0.18	86.68	96.55 ± 0.87	83.61

(a) 80 % for training, %20 for testing, and 90 % for tranining, 10 % for testing

	60 % Training - 40 % Test	
	Random (mean)	Sequential
Naïve Bayes	92.76 ± 1.34	77.74
Random Forest	**96.45 ± 0.67**	87.50
J48	96.44 ± 0.19	87.42
NNge	95.98 ± 0.34	84.73
REP Tree	96.41 ± 0.27	**88.91**
PART	96.05 ± 0.45	81.29

(b) 60 % for tranining, 40 % for testing

Figure 5.2: Percentage of correctly classified patterns for unbalanced data, after after the removal of entries that coul lead to missclassification.. (a) 80 % for training, %20 for testing, and 90 % for tranining, 10 % for testing. (b) 60 % for tranining, 40 % for testing. Best results are marked in boldface.

classifiers including the TLD feature of the URL, but excluding the core domaind used until now. Tables 5.5, 5.6, and 5.7 show the results of this whole new set of experiments. We focused directly in testing the known-top five of classifiers, plus Naïve Bayes, with partitions for training and testing, including the 60 % for training and 40 % for testing, to see the ability for generalisation of the classifiers. Also, as we had obtained the best results for the partitions made randomly, we excluded sequential partition in these experiments, in order to focus in the best results.

This way, Table 5.6 shows that *Random Forest* is again the best classifier (95.42 % of accuracy in the separation 80 % for training, %20 for testing, even though *NNge* was the best but just in 0.05 points). The importat point derived from these results is that the accuracy percentages do not lower after not taking into account the core domain feature of the URL, so maybe the classification is not that dependant on it.

Percentage training & test	80 % - 20 %	90 % - 10 %	60 % - 40 %
Naïve Bayes	48.67 ± 11.70	44.99 ± 18.59	44.58 ± 10.73
Random Forest	67.48 ± 9.61	74.21 ± 6.01	70.29 ± 5.06
J48	**71.19 ± 10.71**	**74.72 ± 6.08**	**70.69 ± 6.51**
NNge	58.19 ± 4.18	65.82 ± 5.70	53.77 ± 5.59
REP Tree	**71.19 ± 10.71**	**74.72 ± 6.08**	**70.69 ± 6.51**
PART	68.37 ± 13.36	67.52 ± 9.60	64.23 ± 5.82

Table 5.4: Percentage of correctly classified patterns for unbalanced data after assuring that same values for a feature of the URL go all to training or testing. There are three different sets of files: 80 % for training, %20 for testing; 90 % for tranining, 10 % for testing; and 60 % for tranining, 40 % for testing. Best results are marked in boldface.

Percentage training & test	80 % - 20 %	90 % - 10 %	60 % - 40 %
Naïve Bayes	81.56 ± 2.68	79.68 ± 0.67	79.85 ± 0.77
Random Forest	95.42 ± 0.32	**95.69 ± 0.27**	**95.16 ± 0.22**
J48	95.00 ± 0.27	94.89 ± 0.15	93.97 ± 0.15
NNge	**95.47 ± 0.09**	94.50 ± 0.42	93.97 ± 0.22
REP Tree	94.20 ± 0.30	93.95 ± 0.16	93.12 ± 0.26
PART	94.77 ± 0.12	95.02 ± 0.08	94.13 ± 0.11

Table 5.5: Percentage of correctly classified patterns for unbalanced data the core domain feature was removed for classification, and TLD was added. There are three different sets of files: 80 % for training, %20 for testing; 90 % for tranining, 10 % for testing; and 60 % for tranining, 40 % for testing. Best results are marked in boldface.

Percentage training & test	80 % - 20 %	90 % - 10 %	60 % - 40 %
Naïve Bayes	79.70 ± 0.25	79.96 ± 0.72	79.43 ± 1.72
Random Forest	**94.01 ± 0.28**	**95.46 ± 1.73**	**93.32 ± 0.11**
J48	92.93 ± 0.37	93.23 ± 0.14	92.15 ± 0.39
NNge	93.07 ± 0.23	93.28 ± 0.26	92.51 ± 0.14
REP Tree	91.80 ± 0.24	91.90 ± 0.35	90.94 ± 0.16
PART	93.03 ± 0.09	94.46 ± 1.69	92.28 ± 0.49

Table 5.6: Percentage of correctly classified patterns for balanced data (with undersampling technique) after the core domain feature was removed for classification, and TLD was added. There are three different sets of files: 80 % for training, %20 for testing; 90 % for tranining, 10 % for testing; and 60 % for tranining, 40 % for testing. Best results are marked in boldface.

Percentage training & test	80 % - 20 %	90 % - 10 %	60 % - 40 %
Naïve Bayes	79.57 ± 0.98	79.87 ± 0.20	79.49 ± 0.88
Random Forest	**97.09 ± 0.06**	**97.90 ± 0.22**	**96.24 ± 0.14**
J48	95.77 ± 0.42	96.25 ± 0.17	94.92 ± 0.12
NNge	96.82 ± 0.27	97.66 ± 0.25	95.78 ± 0.16
REP Tree	94.63 ± 0.27	95.23 ± 0.34	93.80 ± 0.09
PART	95.73 ± 0.30	96.12 ± 0.05	94.86 ± 0.16

Table 5.7: Percentage of correctly classified patterns for balanced data (with oversampling technique) after the step 4. There are three different sets of files: 80 % for training, %20 for testing; 90 % for tranining, 10 % for testing; and 60 % for tranining, 40 % for testing. Best results are marked in boldface.

5.2. Rules obtained, a study case

With regard to the obtained rules/trees, we want to remark that the majority are based on the URL in order to discriminate between the two classes, however we also found several ones which consider variables/features different of this to make the decision. For instance:

```
IF server_or_cache_address = "90.84.53.17"
THEN DENY

IF server_or_cache_address = "173.194.78.103"
THEN ALLOW

IF content_type =
 "application/vnd.google.safebrowsing-update"
THEN DENY

IF server_or_cache_address = "173.194.78.94"
AND content_type_MCT = "text"
AND content_type = "text/html"
AND http_reply_code = "200"
AND bytes > 772
THEN ALLOW

IF server_or_cache_address = "173.194.34.225"
AND http_method = "GET"
AND duration_milliseconds > 52
THEN ALLOW

IF server_or_cache_address = "90.84.53.49"
```

```
AND time <= 33758000
THEN ALLOW
```

These rules are the most interesting for our purposes, since they are somehow independent of the URL to which the client requests to access. Thus, it would be potentially possible to allow or deny the access to unknown URLs just taking into account some parameters of the session.

Of course, some of these features depend on the session itself, i.e. they will be computed after the session is over, but the idea in that case would be 'to refine'somehow the existing set of URLs in the White List. Thus, when a client requests access to a Whitelisted URL, this will be allow, but after the session is over, and depending on the obtained values, and on one of these classifiers, the URL could be labelled as DENIED for further requests. This could be a useful decision-aid tool for the CSO in a company, for instance. In case that the features considered in the rule can be known in advance, such as `http_method`, or `server_or_cache_address`, for example, the decision could be made in real-time, and thus, a granted URL (Whitelisted) could be DENIED or the other way round.

File	Total	Allow	Deny	% of Original
Original	38619	26318	12301	100
Training 80 %	30466	20169	10297	78.79
Test 20 %	8153	6149	2004	21.11
Training 90 %	29964	20454	9510	77.59
Test 10 %	8655	5864	2791	22.41
Training 60 %	19272	13705	5567	49.90
Test 40 %	19347	12613	6734	50.10

(a) Training and Test files, random, 1st generation

File	Total	Allow	Deny	% of Original
Original	38619	26318	12301	100
Training 80 %	28368	18206	10162	73.46
Test 20 %	10251	8112	2139	26.54
Training 90 %	31463	20776	10687	81.47
Test 10 %	7156	5542	1614	18.53
Training 60 %	21945	14814	7131	56.82
Test 40 %	16674	11504	5170	43.18

(b) Training and Test files, random, 2nd generation

File	Total	Allow	Deny	% of Original
Original	38619	26318	12301	100
Training 80 %	31068	21862	9206	80.45
Test 20 %	7551	4456	3095	19.55
Training 90 %	32034	21119	10915	82.95
Test 10 %	6585	5199	1386	17.05
Training 60 %	23337	14416	8921	60.43
Test 40 %	15282	11902	3380	39.57

(c) Training and Test files, random, 3rd generation

Figure 5.3: Statistics of the files created after the duplicated connections were removed. It can be seen that after the partitions, the ratio between allows and denies remains the same, unbalanced. (a), (b), and (c) Three generations of files generated by randomly taking patterns from the original file. (d) Files generated by consecutively taking patterns from the original file.

Chapter 6

Conclusions and future work

Whatever happens today, will change future events, create its own timeline, its own reality. The future pivots around you, here, now. So do good, for humanity, and for Earth.

11th Doctor to Amy. Doctor Who. *Cold Blood.*

Finally, this last Chapter is devoted to draw the conclusions over this research work, and names the papers that followed it. Also, as we want to continue researching in this topic, new objectives of future work are set out.

6.1. Discussion

During the elaboration of this Master Thesis, a set of classification methods have been applied in order to perform a decision process inside a company, according to some predefined corporate security policies. This decision is focused on allowing or denying URL access requests, but just considering previous decisions on similar requests, not by having specific rules in a White/Black List defined including those URLs. Thus, the proposed method could allow or deny an access to a URL, based in additional features rather than just the specific URL string or its lexical properties. This would be very useful since new URLs could be automatically 'Whitelisted', or 'Blacklisted', just depending on some of the connection parameters, such as the `content_type` of the response or the `IP` of the client which makes the request.

To this aim, we have started from a big dataset (100000 patterns) about employees'URL sessions information (Section 3.3), and considering a set of URL access permissions (Section 3.2), we have composed a labelled dataset (more than 57000 labelled patterns, see Table 4.6). Over that set of data, we have tested several classification methods, after some data balancing

techniques have been applied. Then, the top five have been deeply proved over several training and test divisions (Section 4.4), and with two methods: using sequential patterns (consecutive URL accesses), and taking them in a randomly way.

We have performed experiments following a series of steps (see Figures 4.10 and 4.11), and studying how the results changed and why.

The results shown at the beginning experiments, that classification accuracies are between 95 % and 97 %, even when using the unbalanced datasets (Section 5.1). However, they have been diminished because of the possible loss of data that comes from performing an undersampling (removing patterns) method; or taking the training and the data sets in a sequential way from the main log file, due to the fact that certain URL requests can be made only at a certain time.

Then, taking into account the way that the connections client/server passed through a proxy, we performed a removal of patterns that could be considered 'duplicated'(see Section 5.1.1). And we still hade good accuracies (more than 96 %). Then, another step was taken forward and we tried to avoid repetition of the same core domain feature of the URLs in the same training/test file. This means a high loss of information and so it is seen in the results. The last experiments shown accuracies not higher than 74 %.

Finally, a last set of experiments was conducted, in which we trained the classifiers including the TLD feature of the URL, but excluding the core domaind. Accuracies were again quite high, showing resultd between 94 % and 95 %, with low standard deviations.

Hence, after all the experiments done in different situations, we can conclude that the approach has been successful and it would be a useful implementation on a tool for an actual enterprise.

Furthermore, we have found that for the data sets taken consecutively, the methods always classify worse the DENY labels, as they label them as ALLOW patterns. This is worth further study because it is the worst situation. It would be preferable to have a false positive in a DENY pattern, rather than a false negative and permit a request that is forbidden in the ISP.

6.2. Scientific exploitation

As a result of the work that has been done, three papers have been accepted (two of them also already presented, and one is to be presented in October 2014) in three conferences.

The paper titled "*MUSES: A corporate user-centric system which applies computational intelligence methods*", by A.M. Mora, P. De las Cuevas, and J.J. Merelo, was accepted in a special track session of the **ACM SAC** conference, celebrated at Gyeongju, Korea, in March 2014. This special session

was called **TRECK**, from *Trust, Reputation, Evidence and other Collaboration Know-how*. The paper was presented at the conference by A.M. Mora.

Abstract This work presents the description of the architecture of a novel enterprise security system, still in development, which can prevent and deal with the security flaws derived from the users in a company. Thus, the Multiplatform Usable Endpoint Security system (MUSES) considers diverse factors such as the distribution of information, the type of accesses, the context where the users are, the category of users or the mix between personal and private data, among others. This system includes an event correlator and a risk and trust analysis engine to perform the decision process. MUSES follows a set of defined security rules, according to the enterprise security policies, but it is able to self-adapt the decisions and even create new security rules depending on the user behaviour, the specific device, and the situation or context. To this aim MUSES applies machine learning and computational intelligence techniques which can also be used to predict potential unsafe or dangerous user's behaviour.

The paper titled "*Enforcing Corporate Security Policies via Computational Intelligence Techniques*", by A.M. Mora, P. De las Cuevas, J.J. Merelo, S. Zamarripa, and Anna I. Esparcia-Alcázar, was accepted in a worshop at the **GECCO** conference, celebrated at Vancouver, Canada, in July 2014. This workshop was called **SecDef** - Workshop on *Genetic and Evolutionary Computation in Defense, Security and Risk Management*. The paper was presented at the conference by J.J. Merelo and Anna I. Esparcia-Alcázar.

Abstract This paper presents an approach, based in a project in development, which combines Data Mining, Machine Learning and Computational Intelligence techniques, in order to create a user-centric and adaptable corporate security system. Thus, the system, named MUSES, will be able to analyse the user's behaviour (modelled as events) when interacting with the company's server, accessing to corporate assets, for instance. As a result of this analysis, and after the application of the aforementioned techniques, the Corporate Security Policies, and specifically, the Corporate Security Rules will be adapted to deal with new anomalous situations, or to better manage user's behaviour. The work reviews the current state of the art in security issues resolution by means of these kind of methods. Then it describes the MUSES features in this respect and compares them with the existing approaches.

The third paper derived from this research is titled "*Going a Step Beyond the Black and White Lists for URL Accesses in the Enterprise by means of Categorical Classifiers*", by A.M. Mora, P. De las Cuevas, and J.J. Merelo. It

was accepted at the **ECTA** conference, which is going to be held at Rome, Italy, in October 2014.

Abstract Corporate systems can be secured using an enormous quantity of methods, and the implementation of Black or White lists is among them. With these lists it is possible to restrict (or to allow) the users the execution of applications or the access to certain URLs, among others. This paper is focused on the latter option. It describes the whole processing of a set of data composed by URL sessions performed by the employees of a company; from the preprocessing stage, including labelling and data balancing processes, to the application of several classification algorithms. The aim is to define a method for automatically make a decision of allowing or denying future URL requests, considering a set of corporate security policies. Thus, this work goes a step beyond the usual black and white lists, since they can only control those URLs that are specifically included in them, but not by making decisions based in similarity (through classification techniques), or even in other variables of the session, as it is proposed here. The results show a set of classification methods which get very good classification percentages (95-97 %), and which infer some useful rules based in additional features (rather that just the URL string) related to the user's access. This led us to consider that this kind of tool would be very useful tool for an enterprise.

6.3. Future Work

Future lines of work include conducting a deeper set of experiments trying to test the generalisation power of the method, maybe considering bigger data divisions, bigger data sets (from a whole day or working day), or adding some kind of 'noise'to the dataset. Also, as the last experiments were succesful, more experiments will be conducted includind other features of the URL. As found by reviewing the State of the Art (see Section 2.3), the more features are extracted from the connection requests, the better results are obtained (Ma, 2011). Then, we will also extract additional information from the URL string, than could be transformed into additional features that could be more discriminative than the current set. Moreover, a data process involving summarizing data about sessions (such as number of requests per client, or average time connection) will be also considered.

Thus, one of the future steps to follow will be to perform experiments with other two datasets which we have recently had access to:

- A log file provided in JSON format, which contains log entries from 12 days, and 5 million patterns. As can be found in CPAN (*Comprehensive Perl Archive Network*) (Hannyaharamitu, 2005), there is a Perl module

to directly work with this format, so is a great opportunity to continue testing.

- Another log file, in CSV format as the one processed during this research work, that is a subset of the previous data set and contains 1 million entries. This one colud be processed before the one in JSON format, given that we already have the implementation to preprocess CSV log files.

Furthermore, considering the good classification results obtained, another next step could be the application of these methods in the real system from which data was gathered, counting with the opinion of expert CSOs, in order to know the real value of the proposal. The study of other classification methods could be another research branch, along with the implementation of a Genetic Programming approach, which could deal with the imbalance problem using a modification of the cost associated with misclassifying, could be done (as the authors did in (Alfaro-Cid et al., 2007)).

Appendix A

MUSES

A.1. Introduction

This work is embedded inside a system named MUSES [1], from Multiplatform Usable Endpoint Security, which is a device-independent end-to end user-centric tool that is currently being developed, based in a set of security rules defined as specifications of the CSP (*Company Security Policies*), but with the capability of 'learning'from the user's past behaviour and adapt, even inferring new ones, the set of rules in order to deal with potential future security incidents due to the user's actions. Then it will react, in a non-intrusive way, to the potentially dangerous sequence of actions (events) that he or she is conducting at any time.

To this end MUSES analyses the users'behaviour by means of DM techniques and ML methods, extracting a set of patterns which will be later processed by means of CI (*Computational Intelligence*) algorithms.

MUSES includes some important modules in the loop, such as an Event Processor, which performs an event correlation task, matching occurred events with rules to deal with them and extracting threads that a Real-Time Risk and Trust Analysis Engine (RT2AE) will process (along with other trust data and profiles), to select the best subset of rules to apply.

A.2. Multiplatform Usable Endpoint Security System

The MUSES system overview is presented in Figure A.1. As it can be seen, in this system, the user interacts with the devices, own or corporate, through the MUSES graphical interface inside his or her own context (situation, connection, status).

[1] MUSES European project (FP7-318508)

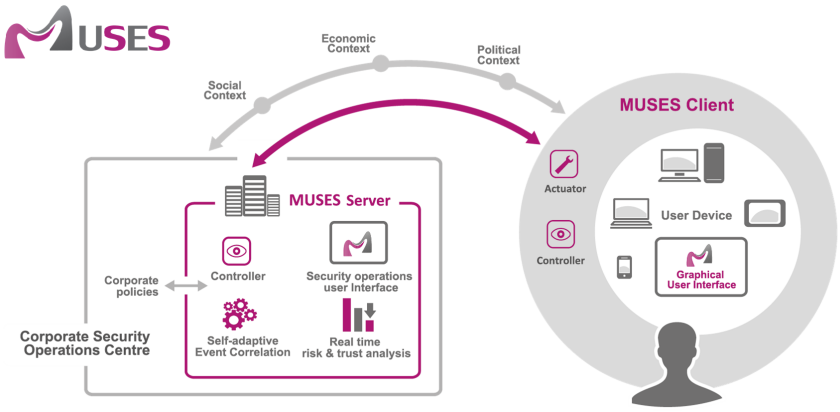


Figure A.1: MUSES system overview. (Design art by S2 Grupo. http://www.s2grupo.es/)

As a summary, this application includes two modules: a *controller* and an *actuator*. The first one monitors the environment (context) and the user's behaviour, translating his/her actions into a sequence of events. These events, along with the patterns defining the user's conduct, are processed by the system in real-time by means of a Risk and Trust Analysis Engine (RT2AE) and an Event Correlation module. Then, a decision is taken in the corporate security operations centre (SOC) side, considering the RT2AE output and the set of security rules adapted to that specific user and context. The corresponding feedback is communicated to the user through the *actuator*, which is also in charge of triggering the recommended actions to stop the user's or application's doings, in case it is required.

The designed MUSES architecture is shown in Figure A.2. It is a *client/-server* approach in which the *client* program will be installed in every user's mobile or portable device, independently of the platform (operating system and type of device). The *server* side would be installed in the corporate SOC. Both sides are connected through a secure channel (using HTTPS) over Internet. In that figure just the high-level components in each part are shown, along with the information flows labelled as 'Info XX'.

The rationale for this approach is based on two main reasons:

1. there is need for a high computational power in order to perform the event correlation and self-adaptation processes, so a quite powerful machine should be used (server);

2. there are two clearly separated parts in the system, namely the users (client) and the enterprise (server).

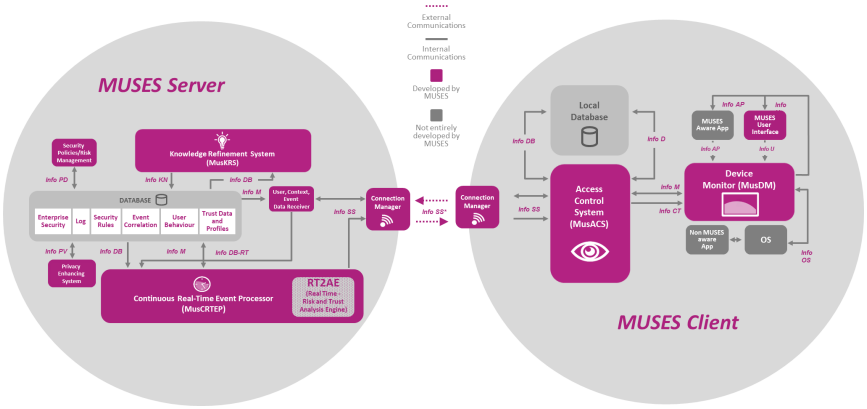


Figure A.2: Proposed architecture. (Design art by S2 Grupo. http://www.s2grupo.es/)

The system considers two working modes for every device: online and offline. In the *online* mode the device can connect with the MUSES server, so it can request the server to make a decision. On the other hand, in *offline* mode the server cannot be reached by the device (there is not an available connection between them), so all the decisions should be made locally. Anyway, in this mode, the gathered information by the sensors in the device will be stored for later submission to the server side (when a connection is available), in order to be processed in the knowledge refinement process.

A.2.1. Client/Device architecture

There are three main components in this side:

- *Local Database*: it is a local security-based storage, which includes the set of security rules to be applied locally (Device Policies), user authentication data, and a cache of gathered events and information. The latter will be useful when the device is in offline mode, so these data are stored to be later submitted to the server side.

- *Device Monitor (MusDM)*: module which gathers the events and information produced by the user. It also acts following the decisions made by the system, in order to allow or deny the controlled application (or the user) for doing something. It is composed by a monitoring and an actuator submodules.

- *Access Control System (MusACS)*: module in charge of making the decisions considering the gathered data. These decisions can be made locally (if possible), or can be requested to the server (if there is no rule which matches with the occurred events).

The rest of the components are: the *MUSES User Interface*, the application through which the user interacts with the system; and the *Connection Manager* which controls the communications between client and server sides. In addition, there are two types of applications considered in this system: on the one hand the *MUSES Aware App*, which is an application adapted to MUSES, so the system can directly interact with it (monitoring and acting). This application must be implemented using the MUSES API [2]. On the other hand the *Non MUSES Aware App* is that which MUSES cannot directly interact with. Usually it will be accessed through the OS.

A.2.2. Server architecture

As in the client, there are three main components:

- *System Database*: it stores all the information that the system will manage, including authentication data, enterprise security policies, assets'values, user-related information (trust, context), events data, and, of course, security rules to apply (regarding the security policies).

- *Continuous Real-Time Event Processor (MusCRTEP)*: this component is the core of the MUSES system with respect to the decision making process. It includes an *Event Processor*, the module in charge of performing the event correlation process, gathering the set of occurred events and doing a rule-based threat identification. The output of this module is taken by the *RT2AE*, which also considering information such as trust data and profiles, assets'values, user reputation, or opportunity, performs a risk and trust analysis task (Seigneur et al., 2013), and extracts the set of potential rules to consider in the analysed situation. Then, these rules are transformed into decisions (or Device Policies) and submitted to those devices to which they apply.

- *Knowledge Refinement System (MusKRS)*: this module is in charge of analysing the information stored in the system database, identifying relevant data, such as important patterns, key features, or security incidents. These are later processed for tuning up the existing set of rules or for inferring new ones.

There are some other components, namely: the *Security Policies/Risk Management* tool, that lets the company CSO to define and manage Security Policies and Rules in a friendly way. It also lets the management of risk-related information, useful for the RT2AE process, such as the assets'values. The *Privacy Enhancing System* which is a module aimed to fit

[2] For every application desired to be MUSES Aware, it should be implemented using this API.

with the legal compliance of the system regarding the user's data anonymisation. The *User, Context, Event Data Receiver* is devoted to receive data from the device side (events, user-related, etc) and to distribute them among the components (storing in the database, and requesting the Event Processor to start the correlation task). Finally, there is another *Connection Manager* which controls the communication with the device side.

As previously stated, one of the main features of the presented system will be the self-adaptation (to the user and context) of the set of security rules. To this end, the MusKRS's task will be run asynchronously to the system working. This process is composed by two steps: first, a Data Mining procedure (See Chapter 3) will be performed, considering the whole amount of historic information mainly regarding user's behaviour and context. Some methods such as Clustering (grouping data), Pattern Recognition (usual situations identification), or Feature Selection (main variables/values in the data) will be applied. The second step consists in a refinement and inference process, performed by means of ML and CI techniques. Regarding the latter, some methods will be used, such as Evolutionary or Genetic Algorithms for parameters/values optimisation; or Genetic Programming for the modification of security rules.

Thus the set of rules will be adapted to every user in the system, updating some of the existing and creating new ones (always respecting the corporate security policies). Moreover, some predictive models will be also obtained applying other soft computing techniques, so the user's potentially dangerous behaviour will be anticipated.

A.3. MUSES Advantages Against other Solutions.

MUSES is mainly a free, open-source, platform independent solution, so these set of features make it a very good option in a first comparison against the proprietary, close and system-specific tools presented in Section 2.1. Moreover most of the existent tools take into account only smartphones and tablets, but MUSES also covers laptops and company PCs too. Moreover the companies which desires work with those systems need specific operative system and server (such as Windows Server, for instance). Being even more strict in some cases, as in Samsung Knox (see Section 2.1.3), in which companies must work with a specific (though they can choose from a list) MDM service for being able of deploy Knox in their environments.

With the Blackphone, that was presented in Section 2.1.6, the comparison may have no sense, because to build a whole device is not the scope of MUSES, but to add the enhancements that this device has to a normal device. This way, the company is not forcing the employees to buy a new device to be compliant with its policies.

Another comparison concerns that the existing products are mostly policy-

based, but MUSES makes its decisions not only considering policies, but also based on the terminals/users context (location, connected networks and so on), to really understand the real danger of a specific action. Also related to this, a very big plus of the MUSES system is its new feature of self-adaptivity. Thus, MUSES is able to adapt to changes, either regarding corporate security policies, newly discovered vulnerabilities or threats, different environments of use or user profiles.

Moreover, MUSES has the feature of having two connection modes, depending on the client being able to connect with the server or not, which favours a real-time decision making process.

Though these are clearly advantages of MUSES over the aforementioned products, they also establish a progress beyond the state of the art. Other issues that MUSES aims to progress on are related to risk and trust data analysis, human-computer interaction (or HCI), device monitoring, and legal compliance. First, as mentioned previously, it will implement a self-adaptive event correlation, including a novel hybrid technique of rule refinement and rule adjustment extracting relevant information from processing huge amounts of data. Then, the project defines a new approach to risk management taking into account threats (with their costs) as well as the innovative concept of opportunity, i.e. the following beneficial outcome from a situation on which, for instance, a user is able to work while waiting at the airport if risk is low enough. About HCI and usability of mobile devices, this will set up a significant advance in the state of the art because of the novelty of the BYOD philosophy, and trying to look for individual differences among the users in susceptibility to persuasive strategies for secure behaviour. Regarding device monitoring, MUSES will also take into account the so-called context observation, by which private or professional scenarios will be detected, or predicted, based on advanced machine learning techniques. Finally, the project is concerned about legal compliance in regards of Information Security Policies, so that it will contribute to the proposal for the EU Data Protection: legal binding force and legal certainty of company policies, and end-user responsibility.

Bibliography

Books. People never really stop loving books.

10th Doctor. Doctor Who. *Silence in the Library.*

Adams, S. A., A. Users are not the enemy. *Communications of the ACM Magazine.*, vol. 42(12), páginas 40–46, 1999.

Al-Omari, E.-G. O. D. A.-W. J., A. Security policy compliance: User acceptance perspective. En *45th Hawaii International Conference on System Science (HICSS)* (editado por J. Shavlik), páginas 3317–3326. IEE, 2012. ISBN 978-1-4577-1925-7. ISSN 1530-1605.

Alfaro-Cid, E., Sharman, K. y Esparcia-Alcázar, A. A genetic programming approach for bankruptcy prediction using a highly unbalanced database. En *Applications of Evolutionary Computing* (editado por M. Giacobini), vol. 4448 de *Lecture Notes in Computer Science*, páginas 169–178. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-71804-8.

Anderson, A. J. P. Computer security threat monitoring and surveillance. Informe técnico, James P. Anderson Co., Fort Washington, PA, 1980.

Blackberry. Blackberry balance. 2012. Disponible en `http://www1.good.com/secure-mobility-solution/bring-your-own-device.html` (último acceso, September, 2014).

Blanco, L., Dalvi, N. y Machanavajjhala, A. Highly efficient algorithms for structural clustering of large websites. En *WWW '11 Proceedings of the 20th international conference on World wide web.*, páginas 437–446. ACM, 2011. ISBN 978-1-4503-0632-4.

Blaze, F. J.-I. J. K. A., M. The role of trust management in distributed systems security. En *Secure Internet Programming* (editado por J. C. Vitek, J.), vol. 1603 de *Lecture Notes in Computer Science*, páginas 185–210. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-66130-6.

BREIMAN, L. Random forests. *Machine Learning*, vol. 45(1), páginas 5–32, 2001.

BULGURCU, C. H.-B. I., B. Information security policy compliance: an empirical study of rationality-based beliefs and information security awareness. *MIS Quarterly.*, vol. 34(3), páginas 523–548, 2010.

CALDWELL, Z. S.-G. K., C. Byod (bring your own device). *Competition Forum*, vol. 10(2), páginas 117–121, 2012.

CHAWLA, N. Data mining for imbalanced datasets: An overview. En *Data Mining and Knowledge Discovery Handbook* (editado por R. L. Maimon, O.), páginas 853–867. Springer US, 2005. ISBN 978-0-387-24435-8.

CHAWLA, N. V., BOWYER, K. W., HALL, L. O. y KEGELMEYER, W. P. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, vol. 16(1), páginas 321–357, 2002. Disponible en `http://dl.acm.org/citation.cfm?id=1622407.1622416`.

CHEN, H., CHUNG, W., QIN, Y., CHAU, M., XU, J. J., WANG, G., ZHENG, R. y ATABAKHSH, H. Crime data mining: An overview and case studies. En *Proceedings of the 3rd National Conference for Digital Government Research (dg.o 2003)*, vol. 130, páginas 1–5. Digital Government Society of North America, 2003.

CITRIX. Mobile application management with xenmobile and the worx app sdk. 2013. Disponible en `https://www.citrix.com/content/dam/citrix/en_us/documents/products-solutions/mobile-containers-with-citrix-mdx.pdf` (último acceso, September, 2014).

CLIFTON, C. y MARKS, D. Security and privacy implications of data mining. En *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, páginas 15–19. 1996.

COMMUNICATIONS, A. Icon (intelligent cloud optimised network). ???? Disponible en `http://www.azzurricommunications.com/~/media/Resources/Brochures/Media/icon_mobilise_brochure_online.ashx` (último acceso, September, 2014).

COSTA, D. Blackphone unveils super-secure smartphone at mwc. Disponible en `http://www.pcmag.com/article2/0,2817,2453964,00.asp`.

CRESSON WOOD, L.-D., C. *Information Security Policies Made Easy Version 11*. Information Shield, Inc., 2009.

DANEZIS, G. Inferring privacy policies for social networking services. En *Proceedings of the 2Nd ACM Workshop on Security and Artificial Intelligence*, AISec '09, páginas 5–10. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-781-3.

DOMINGOS, P. y PAZZANI, M. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, vol. 29, páginas 103–137, 1997.

ELOMAA, T. y KAARIAINEN, M. An analysis of reduced error pruning. *Artificial Intelligence Research*, vol. 15(-), páginas 163–187, 2001.

FRANK, E. y WITTEN, I. H. Generating accurate rule sets without global optimization. En *Fifteenth International Conference on Machine Learning* (editado por J. Shavlik), páginas 144–151. Morgan Kaufmann, 1998.

FRANK, E. y WITTEN, I. H. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, third edición, 2011. ISBN 978-0-12-374856-0.

GOGUEN, M.-J., J. A. Security policies and security models. En *Proceedings of the EEE Symposium on Security and Privacy, 1982*, páginas 11–20. IEEE, 1982. ISBN 0-8186-0410-7.

GREENSTADT, R. y BEAL, J. Cognitive security for personal devices. En *Proceedings of the 1st ACM Workshop on Workshop on AISec*, AISec '08, páginas 27–30. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-291-7.

GUO, X., YIN, Y., DONG, C., YANG, G. y ZHOU, G. On the class imbalance problem. En *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, vol. 4, páginas 192–201. 2008.

HANNYAHARAMITU, M. Json, a perl module. 2005. Disponible en `http://search.cpan.org/~makamaka/JSON-2.90/lib/JSON.pm` (último acceso, September, 2014).

HERATH, R. H., T. Protection motivation and deterrence: a framework for security policy compliance in organisations. *European Journal of Information Systems*, vol. 18(2), páginas 106–125, 2009. ISSN 0960-085X.

IBM. Hosted mobile device security management. 2011. Disponible en `http://www-935.ibm.com/services/uk/en/it-services/managed-security-services-cloud-computing-hosted-mobile-device-security-management.html` (último acceso, September, 2014).

JAPKOWICZ, S. S., N. The class imbalance problem: A systematic study. *Intell. Data Anal.*, vol. 6(5), páginas 429–449, 2002. Disponible en `http://dl.acm.org/citation.cfm?id=1293951.1293954`.

KAN, T.-H. O. M., M-Y. Fast webpage classification using url features. En *Proceedings of the 14th ACM international conference on Information and knowledge management.*, páginas 325–333. ACM New York, NY, USA, 2005. ISBN 1-59593-140-6.

KAO, I.-L. Securing mobile devices in the business environment. 2011.

KELLEY, P. G., HANKES DRIELSMA, P., SADEH, N. y CRANOR, L. F. User-controllable learning of security and privacy policies. En *Proceedings of the 1st ACM Workshop on Workshop on AISec*, AISec '08, páginas 11–18. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-291-7.

KHONJI, M., JONES, A. y IRAQI, Y. A novel phishing classification based on website features. En *GCC Conference and Exhibition (GCC)*, páginas 221–224. IEE, 2011. ISBN 978-1-61284-118-2.

LEAVITT, N. Is cloud computing really ready for prime time? *IEEE Computer*, vol. 42(1), páginas 15–20, 2009. ISSN 0018-9162.

LIM, Y. T., CHENG, P. C., CLARK, J. y ROHATGI, P. Policy evolution with genetic programming: A comparison of three approaches. En *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, páginas 1792–1800. 2008a.

LIM, Y. T., CHENG, P. C., ROHATGI, P. y CLARK, J. A. Mls security policy evolution with genetic programming. En *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, páginas 1571–1578. ACM, New York, NY, USA, 2008b. ISBN 978-1-60558-130-9.

LIPPMANN, I. K.-S. C. P. K.-K. K. A. M. C. R., R.P. Evaluating and strengthening enterprise network security using attack graphs. 2005.

MA, S. L. K. S. S. V.-G. M., J. Learning to detect malicious urls. *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2(3), 2011.

MARTIN, B. *Instance-Based learning: Nearest Neighbor With Generalization*. Proyecto Fin de Carrera, University of Waikato, Hamilton, New Zealand, 1995.

MOZY. The past, present, and future of data storage. Disponible en `http://mozy.com/infographics/the-past-present-and-future-of-data-storage/`.

NEWS, S. K. New knox solutions being announced at mwc. Disponible en `https://www.samsungknox.com/es/blog/new-knox-solutions-being-announced-mwc`.

OPPLIGER, R. Security and privacy in an online world. *IEEE Computer*, vol. 44(9), páginas 21–22, 2011. ISSN 0018-9162.

OPSWAT. Oesis framework. 2014. Disponible en `http://www.opswat.com/products/oesis-framework` (último acceso, September, 2014).

O'REILLY, S. B., T. The importance of perl. Informe técnico, 2007.

ORTHACKER, T. P. K. S. L. G.-G. M. M. A. L. J. P. O., C. Android security permissions - can we trust them? En *Security and Privacy in Mobile Information and Communication Systems* (editado por F. K. S. A. U. L.-A. R. G. L. F. L. Prasad, R.), vol. 94 de *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.*, páginas 40–51. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30243-5.

QUINLAN, J. R. Simplifying decision trees. *Man-Machine Studies*, vol. 27(3), páginas 221–234, 1987.

QUINLAN, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

RISH, I. An empirical study of the naive bayes classifier. En *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, páginas 41–46. 2001.

SAMSUNG. Samsung knox. 2013. Disponible en `https://www.samsungknox.com` (último acceso, September, 2014).

SCHUMACHER, F.-B. E. H. D. B.-F. S. P., M. *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & sons, 2005.

SEIGNEUR, J.-M., KÖLNDORFER, P., BUSCH, M. y HOCHLEITNER, C. A survey of trust and risk metrics for a byod mobile working world. En *Third International Conference on Social Eco-Informatics*. 2013.

SHAW, C.-C. H. A. H. H.-J., R.S. The impact of information richness on information security awareness training effectiveness. *Computers & Education.*, vol. 52(1), páginas 92–100, 2009.

SIPONEN, P.-S. M. A., M. Employees adherence to information security policies: An empirical study. En *New Approaches for Security, Privacy and Trust in Complex Environments* (editado por E.-M. L. L. E. J.-v. S. R. Venter, H.), vol. 232 de *IFIP International Federation for Information Processing*, páginas 133–144. Springer US, 2007. ISBN 978-0-387-72366-2.

SOPHOS. Sophos mobile control. 2011. Disponible en `http://www.sophos.com/en-us/products/mobile-control.aspx` (último acceso, September, 2014).

SUAREZ-TANGIL, G., PALOMAR, E., FUENTES, J., BLASCO, J. y RIBAGORDA, A. Automatic rule generation based on genetic programming for event correlation. En *Computational Intelligence in Security for Information Systems* (editado por Á. Herrero, P. Gastaldo, R. Zunino y E. Corchado), vol. 63 de *Advances in Intelligent and Soft Computing*, páginas 127–134. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04090-0.

TEAM, S. Squid website. Disponible en `http://www.squid-cache.org/`.

TEAM, S. Squid faq - squid log files. 2014. Disponible en `http://docstore.mik.ua/squid/FAQ-6.html` (último acceso, September, 2014).

TEAM, T. J. D. Drools documentation. version 6.0.1.final. Disponible en `http://docs.jboss.org/drools/release/6.0.1.Final/drools-docs/html/index.html`.

TEAM, T. J. D. Drools website. Disponible en `http://www.jboss.org/drools.html`.

TECHNOLOGY., G. Good's technology byod solution. 2012. Disponible en `http://es.blackberry.com/business/software/blackberry-balance.html` (último acceso, September, 2014).

TOWNSEND, K. Whitelisting vs blacklisting. Disponible en `http://kevtownsend.wordpress.com/2011/08/24/whitelisting-vs-blacklisting/`.

VAQUERO, R.-M. L. C. J. L.-M., LM. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, vol. 39(1), páginas 50–55, 2009. ISSN 0146-4833.

DE VEL, O., ANDERSON, A., CORNEY, M. y MOHAY, G. Mining e-mail content for author identification forensics. *SIGMOD Record*, vol. 30(4), páginas 55–64, 2001.

OF WAIKATO, U. Weka. 1993. Disponible en `http://www.cs.waikato.ac.nz/~ml/weka/` (último acceso, September, 2014).

WESSELS, D. *Squid: The Definitive Guide*. O'Reilly Media, Inc., 1st edición, 2004. ISBN 978-0-596-00162-9.

WIKI, S. Squid hierarchy. 2014. Disponible en `http://wiki.squid-cache.org/Features/CacheHierarchy` (último acceso, September, 2014).

WOODS, B. Mwc 2013: Samsung's knox system takes byod fight to blackberry. Disponible en `http://www.zdnet.com/mwc-2013-samsungs-knox-system-takes-byod-fight-to-blackberry-7000011770/`.

WSO2. Wso2 enterprise mobility manager. ???? Disponible en `http://wso2.com/products/enterprise-mobility-manager/` (último acceso, September, 2014).

ZHANG, H.-J. I. C. L. F., Y. Cantina: a content-based approach to detecting phishing web sites. En *Proceedings of the 16th international conference on World Wide Web.*, páginas 639–648. WWW '07, 2007. ISBN 978-1-59593-654-7.

List of Acronyms

API ..... *Application Programming Interface*

BYOD ..... *Bring Your Own Device*

CEO ..... *Chief Executive Officer*

CI ..... *Computational Intelligence*

CISO ..... *Chief Information Security Officer*

CPAN ..... *Comprehensive Perl Archive Network*

CSO ..... *Chief Security Officer*

CSP ..... *Company Security Policies*

DM ..... *Data Mining*

DRL ..... *Drools Rule Language*

GP ..... *Genetic Programming*

IP ..... *Internet Protocol*

ISP ..... *Information Security Policies*

IT ..... *Information Techlonogy*

LDAP ..... *Lightweight Directory Access Control*

MCT ..... *Main Content Type*

MDM ..... *Mobile Device Management*

ML ..... *Machine Learning*

OS ..... *Operating System*

PUA ..... *Potentially Unwanted Apps*

SaaS ..... *Software as a Service*

TLD ..... *Top Level Domain*

URL ..... *Uniform Resource Locator*

VPN ..... *Virtual Private Network*