Pattern Mining
A Key To Improve Network Usage Understanding

Nuno Miguel Vaz Afonso

Dissertação para a Obtenção do Grau de Mestre em Engenharia Informática e Computadores

Júri
Presidente: Prof. Joaquim Armando Pires Jorge
Orientador: Prof. Cláudia Martins Antunes
Co-Orientador: Prof. Carlos Nuno da Cruz Ribeiro
Vogal: Prof. Fernando Henrique Corte-Real Mira da Silva

November 2011
No great discovery was ever made without a bold guess.

- Isaac Newton
Resumo

As tecnologias de informação têm vindo a adquirir um papel preponderante no que respeita ao suporte do negócio das organizações, sendo hoje em dia uma ferramenta indispensável a um número crescente de áreas de aplicação. Com a criticidade dos sistemas, acrescem as preocupações com a segurança da informação e das infraestruturas, pois a sua exposição ou adulteração poderão comportar pesados danos para a organização.

As tecnologias e ferramentas existentes actualmente na área da segurança encontram-se ainda maioritariamente focadas na segurança de cada um dos sistemas individualmente, e não na globalidade da infraestrutura. Acresce ainda que as políticas de segurança utilizadas têm vindo a ser construídas ao longo do tempo com base na experiência dos peritos na área e seguem uma abordagem predominantemente preventiva, ou seja, procuram reconhecer à partida comportamentos de utilização nocivos e impedir que estes cheguem a ocorrer na infraestrutura.

Não obstante a sua correcção e predominância, este modelo torna difícil a gestão e verificação das políticas de segurança definidas pelos peritos, para a infraestrutura em causa.

Como exemplo ilustrativo, no contexto dos sistemas de detecção de intrusões (em inglês, intrusion detection systems - IDS), existem dois grupos principais de sistemas.

Os sistemas locais, ou *host-based IDS*, focam-se nas características da máquina, através de *checksums* aos ficheiros e registo dos tempos de acessos, efectuando periodicamente a verificação da sua integridade.

Os sistemas de rede, ou *network IDS*, focam-se na informação que recebe vinda da rede, através das características dos pacotes de rede e dos tipos de comunicações que ocorrem. Este tipo de sistemas é habitualmente instalado no ponto de acesso que separa a rede local do exterior. Contudo, um sistema desta natureza não consegue ter em consideração as comunicações existentes entre os diferentes pontos no seio da rede.

Este trabalho propõe a aplicação de métodos de data mining nos pontos de chegada, em particular, descoberta de padrões sequenciais aplicada aos ficheiros de log dos serviços disponibilizados pela infra-estrutura, com o objectivo de analisar a forma como a infra-estrutura é habitualmente usada pelos clientes, e testar a adequação das políticas de utilização à sua aplicação real.

É esperada uma melhoria dos resultados na detecção de padrões sequenciais de utilização quando esta detecção considera a infra-estrutura de forma abrangente, e não apenas cada um dos pontos individualmente.
Information technologies have been acquiring an important role with regard to business support organizations, and is today an indispensable tool for a growing number of application areas. With the criticality of the systems, plus concerns about information security and infrastructure, as their exposure or tampering could include heavy damage to the organization.

Technologies and tools that currently exist in the area of security are still largely focused on the security of each system individually, not in the overall infrastructure. Furthermore, the security policies used have been built over time based on the experience of experts in the field and follow a predominantly preventive approach, that is, seek to recognize at the outset to use harmful behaviors and prevent them from reaching the place the infrastructure.

Regardless of its accuracy and dominance, this model makes it difficult to manage and verify security policies defined by the experts to infrastructure concerned.

As illustrative example, in the context of intrusion detection systems (IDS), there are two main groups of systems.

Local systems, or host-based IDS, focus on the characteristics of the machine via the checksums of files and registry access time, by periodic checking of its integrity.

Network systems, or network IDS, focus on the information it receives from the network, through the characteristics of the network packets and communication delays that occur. This type of system is usually installed in the access point d which separates the LAN from outside. However, such a system fails to take into account the communications between the different points within the network.

This work proposes the application of data mining methods at the point of arrival, in particular, discovery of sequential patterns applied to the log files of the services provided by the infrastructure, in order to analyze how the infrastructure is often used by customers, and test the adequacy of the use of its actual application.

It is expected an improvement in results in the detection of sequential patterns of use when detecting Consider this infrastructure in a comprehensive manner and not just each point individually.
Abstract

In corporative network environment there are certain sets of events that are usually individually analyzed. Whenever events are correlated this way of "looking through" processes jeopardizes the ability of establishing a coordination amongst them. Nowadays security systems are, in its essence, focused on the security of each one of the infrastructure component. It is possible to reinforce complementarily the security systems adding knowledge about the infrastructures usage, collected by the application of sequential pattern mining techniques in application log files.

Keywords:
Sequential pattern mining, event correlation, security, log files.
# Contents

Resumo iii  
Resume iv  
Abstract v  
List of Figures viii  
List of Abbreviations x

1 Introduction 1

2 State of the Art 2  
2.1 Network Infrastructure 3  
2.2 Data Mining 6  
2.2.1 Intrusion Detection 6  
2.2.2 Anomaly Detection 7  
2.2.3 Failure Detection 12  
2.2.4 Anomaly Detection 14  
2.3 Log Tools 20  
2.3.1 Log Analyzer Tools 20  
2.3.2 Log Indexer Tools 22  
2.4 Intrusion Detection Systems 23  
2.5 Open Issues 24  
2.6 Goal 25

3 Network Usage Exploration 26  
3.1 Methodology 26  
3.2 Business Understanding: Target Infrastructure Model 28  
3.2.1 Data Understanding 30  
3.2.2 Notes about the data types 31  
3.3 Target Infrastructure Model 32  
3.3.1 Gateway 33  
3.3.2 Name Server 34  
3.3.3 FTP 36  
3.3.4 Web Server 38  
3.3.5 Mail Server 41  
3.4 Data Preparation 41  
3.4.1 Parsing Sources 41
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.2 Basket Construction</td>
<td>42</td>
</tr>
<tr>
<td>3.4.2.1 Baskets from Log Messages</td>
<td>43</td>
</tr>
<tr>
<td>3.4.2.2 Labeling Clients Address</td>
<td>47</td>
</tr>
<tr>
<td>3.4.2.3 Address Origins Classification</td>
<td>48</td>
</tr>
<tr>
<td>3.4.2.4 Labels Classification Criteria</td>
<td>49</td>
</tr>
<tr>
<td>3.4.2.5 Time Line Sorting</td>
<td>50</td>
</tr>
<tr>
<td>3.4.3 Reduction Process</td>
<td>52</td>
</tr>
<tr>
<td>3.5 Mining Phase</td>
<td>54</td>
</tr>
<tr>
<td>4 Case Study</td>
<td>55</td>
</tr>
<tr>
<td>4.1 Network Configuration</td>
<td>55</td>
</tr>
<tr>
<td>4.2 Components and Services</td>
<td>56</td>
</tr>
<tr>
<td>4.2.1 Servers</td>
<td>56</td>
</tr>
<tr>
<td>4.2.2 Services</td>
<td>56</td>
</tr>
<tr>
<td>4.2.3 Applied criteria</td>
<td>57</td>
</tr>
<tr>
<td>4.3 Efficiency Analysis</td>
<td>57</td>
</tr>
<tr>
<td>4.3.1 Evaluation Before Reduction</td>
<td>58</td>
</tr>
<tr>
<td>4.3.2 Data Reduction</td>
<td>59</td>
</tr>
<tr>
<td>4.3.3 Evaluation After Reduction</td>
<td>60</td>
</tr>
<tr>
<td>4.3.4 Comparative Analysis</td>
<td>61</td>
</tr>
<tr>
<td>4.3.5 Scalability Analysis</td>
<td>62</td>
</tr>
<tr>
<td>4.3.6 Qualitative Analysis</td>
<td>65</td>
</tr>
<tr>
<td>5 Conclusion</td>
<td>67</td>
</tr>
<tr>
<td>Appendices</td>
<td>70</td>
</tr>
<tr>
<td>.1 Case Study Taxonomies</td>
<td>70</td>
</tr>
<tr>
<td>.2 Detailed Scalability</td>
<td>75</td>
</tr>
<tr>
<td>Bibliography</td>
<td>83</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Generic organizational structure</td>
<td>2</td>
</tr>
<tr>
<td>2.2</td>
<td>Topology of a basic network infrastructure</td>
<td>3</td>
</tr>
<tr>
<td>2.3</td>
<td>Topology of a basic Microsoft infrastructure</td>
<td>4</td>
</tr>
<tr>
<td>2.4</td>
<td>A typical web interaction</td>
<td>5</td>
</tr>
<tr>
<td>3.1</td>
<td>CRISP-DM Adaptation</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Data Sources</td>
<td>29</td>
</tr>
<tr>
<td>3.3</td>
<td>Wide Range Partition Example</td>
<td>31</td>
</tr>
<tr>
<td>3.4</td>
<td>Infrastructure Components</td>
<td>32</td>
</tr>
<tr>
<td>3.5</td>
<td>GateWay Taxonomy</td>
<td>34</td>
</tr>
<tr>
<td>3.6</td>
<td>Name Server Taxonomy</td>
<td>36</td>
</tr>
<tr>
<td>3.7</td>
<td>FTP Taxonomy</td>
<td>38</td>
</tr>
<tr>
<td>3.8</td>
<td>Web Taxonomy</td>
<td>40</td>
</tr>
<tr>
<td>3.9</td>
<td>Web SSL Taxonomy</td>
<td>41</td>
</tr>
<tr>
<td>3.10</td>
<td>Preprocessing schema</td>
<td>41</td>
</tr>
<tr>
<td>3.11</td>
<td>Gateway log sample</td>
<td>43</td>
</tr>
<tr>
<td>3.12</td>
<td>Name Server queries log sample</td>
<td>45</td>
</tr>
<tr>
<td>3.13</td>
<td>Name Server reverse queries log sample</td>
<td>45</td>
</tr>
<tr>
<td>3.14</td>
<td>Name Server edns disabled log sample</td>
<td>46</td>
</tr>
<tr>
<td>3.15</td>
<td>Name Server lame servers log sample</td>
<td>46</td>
</tr>
<tr>
<td>3.16</td>
<td>Name Server update security log sample</td>
<td>46</td>
</tr>
<tr>
<td>3.17</td>
<td>Time Line Sorting example</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Case Study Network Topology</td>
<td>56</td>
</tr>
<tr>
<td>4.2</td>
<td>Data Frequency Histogram Before Reduction</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Number of Sequential Patterns Before Reduction</td>
<td>58</td>
</tr>
<tr>
<td>4.4</td>
<td>Time Before Reduction</td>
<td>59</td>
</tr>
<tr>
<td>4.5</td>
<td>Name server association rules for data reduction</td>
<td>59</td>
</tr>
<tr>
<td>4.6</td>
<td>Gateway association rules for data reduction</td>
<td>60</td>
</tr>
<tr>
<td>4.7</td>
<td>Data Frequency Histogram After Reduction</td>
<td>60</td>
</tr>
<tr>
<td>4.8</td>
<td>Number of Patterns After Reduction</td>
<td>61</td>
</tr>
<tr>
<td>4.9</td>
<td>Time After Reduction</td>
<td>61</td>
</tr>
<tr>
<td>4.10</td>
<td>Reduction Impact in the Number of Patterns</td>
<td>62</td>
</tr>
<tr>
<td>4.11</td>
<td>Reduction Impact in Time</td>
<td>62</td>
</tr>
<tr>
<td>4.12</td>
<td>Nr. of Sequences Over Time</td>
<td>63</td>
</tr>
<tr>
<td>4.13</td>
<td>Nr. of Baskets Over Time</td>
<td>63</td>
</tr>
<tr>
<td>4.14</td>
<td>Parsing Duration Over Time</td>
<td>64</td>
</tr>
<tr>
<td>4.15</td>
<td>Nr. of Discovered Sequential Patterns</td>
<td>64</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.16</td>
<td>Time of Sequential Patterns Discovery</td>
<td>64</td>
</tr>
<tr>
<td>4.17</td>
<td>Sequences Length Proportion</td>
<td>65</td>
</tr>
<tr>
<td>1</td>
<td>Case Study GateWay Taxonomy</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>Case Study Name Server Taxonomy</td>
<td>73</td>
</tr>
<tr>
<td>3</td>
<td>Case Study FTP Server Taxonomy</td>
<td>74</td>
</tr>
<tr>
<td>4</td>
<td>Nr. of Processed Baskets Over Time</td>
<td>76</td>
</tr>
<tr>
<td>5</td>
<td>Time of Blocks Integration in a Time Line</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>Time of Baskets Extraction</td>
<td>77</td>
</tr>
<tr>
<td>7</td>
<td>Nr. of Processed Baskets Over Time in the Gateway</td>
<td>77</td>
</tr>
<tr>
<td>8</td>
<td>Baskets Extraction Time in the Gateway</td>
<td>78</td>
</tr>
<tr>
<td>9</td>
<td>Nr. of Processed Baskets Over Time in the Primary Name Server</td>
<td>78</td>
</tr>
<tr>
<td>10</td>
<td>Baskets Extraction Time in the Primary Name Server</td>
<td>79</td>
</tr>
<tr>
<td>11</td>
<td>Nr. of Processed Baskets Over Time in the Secondary Name Server</td>
<td>79</td>
</tr>
<tr>
<td>12</td>
<td>Baskets Extraction Time in the Secondary Name Server</td>
<td>80</td>
</tr>
<tr>
<td>13</td>
<td>Nr. of Processed Baskets Over Time on FTP Service in the FTP Server</td>
<td>80</td>
</tr>
<tr>
<td>14</td>
<td>Baskets Extraction Time on FTP Service in the FTP Server</td>
<td>81</td>
</tr>
<tr>
<td>15</td>
<td>Nr. of Processed Baskets Over Time on HTTP Service in the FTP Server</td>
<td>81</td>
</tr>
<tr>
<td>16</td>
<td>Baskets Extraction Time on HTTP Service in the FTP Server</td>
<td>82</td>
</tr>
</tbody>
</table>
List of Abbreviations

API - Application Programming Interface
CRISP-DM - CRoss-Industry Standard Process for Data Mining
DC - Domain Controller
DNS - Domain Name System
DoS - Denial of Service
HMM - Hidden Markov Model
IDS - Intrusion Detection System
IP - Internet Protocol
KDD - Knowledge Discovery in Database
QoS - Quality of Services
SMTP - Simple Mail Transfer Protocol
SVM - Support Vector Machine
URL - Universal Resource Locator
1. Introduction

The security of an organizational computer network is usually considered the sum of the security of its components. The experience in this area has shown that the network security, or in a wide sense, the general infrastructure, should be regarded as a whole, and not only as the sum of its parts. This derives from the fact that to ensure the security of an infrastructure, it is not enough that each one of its components is secure... It is also necessary that the interaction between components is secure too.

Today’s classical computer network security tools are single computer oriented. Because of this, it is difficult to have a wide overview about what is happening in all infrastructure, and not only in each single component. To improve the overview about the infrastructure activity, this work applies sequential pattern mining methods to discover information and identify typical behaviors on network usage. It is believed that the identification of those network usage patterns will help system administrators to have a more accurate perception of the infra-structure usage, and in consequence, will help them to adjust their configuration criteria and policies to better fit the usage necessities.

This work studies how sequential pattern mining techniques can contribute improve network usage understand, analyzing the network infrastructure as a whole.

This thesis proposes a methodology of sequential usage patterns discovery in a network infrastructure. It is also proposes a way to systematize the domain knowledge of the infrastructure services about to be mined. To discuss the proposed method, it is presented the analysis of a case study.

The rest of this document is organized as follows. Chapter 2 introduces some basic concepts about organizational structure, network infrastructures, services and some security concerns, data mining and the goal of this work. Chapter 2.2 presents related work divided in clustering and classification. It also presents related products for log processing and analyzing, and security applications. Chapter 3 presents the methodology proposed for sequential pattern discovery in log files of an infrastructure services, and the schedule plan for its implementation. Chapter 4 presents the case study. Chapter 5 concludes this work.
2. State of the Art

Since the early human evolution security concerns are present in almost all working subjects. Security goal is to preserve the characteristics and the access control of a subject. That means that security tries to avoid that the subject is accessible to those who do not have access authorization, and also avoid that the characteristics of the subject are adulterated.

Those aspects are particularly relevant when the subjects are the information systems. Nowadays, information systems are a vital piece of organizations living and playing a decisive role in its success. An information system can be defined technically as a set of interrelated components that collect (or retrieve), process, store and distribute information to support decision making and control in an organization [28].

Information systems can be computer based platforms and other platforms like paper or black board. This work will only consider computer based platforms. The set of computers (and other equipments) that support information systems are called infrastructure.

A typical organizational structure is divided in four basic layers [28], as shown in Fig. 2.1. From bottom to top, there are the hardware layer, system software layer, application software layer and the users layer.

![Figure 2.1: Generic organizational structure](image)

At the bottom is the hardware layer, composed by computers, switches, routers and firewalls, which is defined by machinery communicating with other machinery. The second bottom layer is the system software layer, composed by the operating system and other applications (usually services) that support the network usage. Domain name systems (DNS), web and file servers, mail servers and authentication servers are examples of these kind of applications. Organizations usually have their own specific applications, the application software, which contains, for example, an intranet portal or a content management system (CMS) to support the business processes. Those applications have some kind of business knowledge and each organization has its own specific applications. This layer lies above system software applications. Above the application software layer is the human resources layer (Users), where people interact with application software and the infrastructure.

For each one of these layers, as presented in Fig. 2.1, there are specific security concerns that should be considered.

The hardware layer security concerns about questions like:

- Is it easy to change our router configurations and add a new route somewhere else?
- Is our firewall allowing traffic to forbidden computers?
2.1 Network Infrastructure

Technology is usually a poor approach to solve human nature or social problems. Security in Users layer is managed mainly by social and juridical rules, so computers by itself, in general, can only be a complement to these rules. Application software layer is usually specific to the organization, so security is also specific to these applications and the organization too. Hardware layer and system software layer are closely related, and have some general topologies that are common in organizations infrastructures world wide. This work will mainly focus in these two lower layers of the organizational structure to research how sequential pattern mining techniques can help the security management improvement.

Hardware layer and system software layer are the network infrastructure which supports the organization application software and business processes.

A typical infrastructure, Fig. 2.2, is composed by a gateway which commonly has firewall capabilities, a name server which ensures the translations from names to computer addresses, a mail server which provide the e-mail function to the organization, and a web server or other file servers that provide an easy way to share files.

![Figure 2.2: Topology of a basic network infrastructure](image)

A particular case of the previous topology is a Microsoft Windows Domain [18], Fig. 2.3, which consists in a domain controller, or a forest of domain controllers. Domain Controllers
(DC) [4] mainly provide domain policies control and authentication. It is also possible to provide name resolution and share files with domain controllers, for example. In a non Microsoft Windows infrastructure, authentication can be provided by a lightweight directory access server (LDAP) [23].

![Figure 2.3: Topology of a basic Microsoft infrastructure]

A gateway connects networks that are using different network protocols. A gateway is responsible for transferring traffic from one network to another by transforming the format of that traffic into a form compatible with the protocol or transport method used by each network [14]. For example, from the Internet to a local network. It is a common practice to have a gateway with firewall and traffic shaper capabilities.

A firewall is a network device used to filter traffic and is typically deployed between a private network and a link to the Internet, but it can be deployed between departments within an organization. Firewalls filter traffic based on a defined set of rules, also called filters or access control lists. They are basically a set of instructions that are used to distinguish authorized traffic from unauthorized and/or malicious traffic. Only authorized traffic is allowed to cross the security barrier provided by the firewall.

Firewalls are useful for blocking or filtering traffic. They are most effective against unrequested traffic and attempts to connect from outside the private network and for blocking known malicious data, messages, or packets based on content, application, protocol, port, or source address. They are capable of hiding the structure and addressing scheme of a private network from the public. Most firewalls offer extensive logging, auditing, and monitoring capabilities, as well as alarms and basic intrusion detection system (IDS) functions. Firewalls are unable to block viruses or malicious code transmitted through otherwise authorized communication channels, prevent unauthorized but accidental or intended disclosure of information by users, prevent attacks by malicious users already behind the firewall, or protect data after it passes out of, or into the private network [14].

A name server is a computer running the service responsible for the translation between computer names and network addresses using the domain name system (DNS). This translation provide an abstraction between the organization service names known by clients and the infrastructure topology, so organizations can make changes in its infra-structure without changes in clients access behavior. Because clients usually know computer names rather than computer addresses.

A mail server is a computer running the service responsible to distribute the clients e-mails messages, called mail transfer agent (MTA) [17]. Usually this service is composed by a simple
mail transfer protocol (SMTP) [42] service that ensures the reception of e-mail messages sent by other mail transfer agents and the local delivery to organization clients\(^1\). This service also makes the reverse. That means, mail transfer agent provides e-mail delivery from outside to inside the organization, from inside to outside and between internal organization clients.

A web server is a computer running the service responsible to share information and files using the hipper text transfer protocol [45]. This information is usually read only for clients. A file server is a computer running the service responsible for sharing files, but differs from the web server in the way the files are shared. Usually it uses file transfer protocol [41] to transfer files to client computers, or uses a network file system [49][11] to share files so client computers can access files with almost the same semantic as a local file system.

These services, are based on a client-server architecture [62], and a client can be any program starting a request, waiting the response from the server. For example, a web browser like Microsoft Internet Explorer [20] or Mozilla Firefox [21] act as clients and Apache HTTP [53][45] server acts as server. Only as a complement, there is a slightly different client-server concept not based on request-response as in Distributed Systems [16] but on remote procedure call [16]. For example, Microsoft Exchange Server Protocol [8] uses this model. Remote procedure call client-server model will not be discussed in this work.

There are typical behaviors in the interaction between clients and the services of the infrastructure. For example, usually before a web page request or an e-mail delivery, a domain name resolution is requested, as shown in Fig. 2.4. Since this is a typical behavior, there is nothing forbidding a client that previously aware of the mail server IP address, to directly contact the mail server without contacting name server first.

These kind of interactions can be defined as series of sequential actions over time. That means actions have causal relationship since a client cannot receive a response before sending the request. Typical behaviors are formed by interactions that vary in its content but preserve almost the same structure or pattern. For example, different clients can access the same mail server using the same steps sequence, preserving the interaction structure shown in Fig. 2.4. To explore sequential patterns, data mining techniques will be used.

\(^1\) The mail service depends critically on domain name service. Mail transfer agent usually have the domain name when the e-mail message is to be delivered, instead the network address. This is because a domain name is easier for humans than simply numbers.
2.2 Data Mining

Data mining [48] is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, XML repository, etc. Also data mining is known as one of the core processes of Knowledge Discovery in Database (KDD) [73].

Generally speaking, there are two classes of data mining techniques, descriptive and predictive. Descriptive mining is to summarize or characterize general properties of data in data repository, while predictive mining is to perform inference on current data, to make predictions based on the historical data. There are various types of data mining techniques such as association rules, classifications and clustering. Based on those techniques web mining and sequential pattern mining are also well researched [73].

Association rule mining, is a well researched technique of data mining [34], that aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories [73]. For example, for each web server log entry there is a name server log entry from the same client in 87% of the occurrences, Fig. 2.4, than it is possible to establish an association rule that says “web server requests have a name server request for 87% of the time”. The percentage value shown in this example is usually called support, and is used to provide suplementary information about the association rule relevance.

Sequential patter mining, besides the frequent items analysis, it is also considered the order in the items occurrence in data base. In previous example, sequential pattern mining establishes the pattern “when a web page is requested, name server is contacted before the web server in 87% of the occurrences”. It is common to tag frequent items with a temporal identification to have order properties in the item sets defining a pattern, but if an order can be defined by a different criteria, sequential pattern mining methods can be applied even without termporal tags. Sequential pattern mining can be used in DNA sequence analysis.

2.2.1 Intrusion Detection

Mining Intrusion Detection Alarms for Actionable Knowledge [26] tries to address this problem to improve the efficiency handling such amount of intrusion detection system information. High volume of intrusion detection systems information generated is a real problem that often affects the decision of deploying or not a network intrusion detection system. This is why so much experience and sensibility is required to adequately configure and monitor a intrusion detection system.

This work attempts to mitigate those aspects and facilitate the management task. Authors proposes learning approach using episode rules and clustering techniques based on historical false positives triggered. This method does not simply eliminate those false positive patterns, because it is risky to discard information that can encapsulate some kind of unusual treat.

Episode rules lays on the discovery of association rules principle, trying to predict the occurrence of certain alarms based on the occurrence of other alarms. Authors does not explicitly reveals the method used to discover association rules, but they refer the use of support and confidence parameters used, the same used in Apriori [72] algorithm.

Authors also define parallel and serial episodes in a certain time line. Serial episodes are
2.2. DATA MINING

ordered, since parallel episodes do not. The definition of episode are important to identify groups of related alarms and to cluster them, because, as authors describe, some alarms entail other redundant alarms so it is possible to fuse them into a single but semantically richer meta-alarm. A natural evolution of this concept is the conceptual clustering introduced by the authors. Conceptual clustering not only group objects which the category are alike. It also tries to group up events in a conceptual generalization, for example, it does not tries only to group up events with the same source IP address, instead it tries to group up events in which the source IP addresses belongs to the same network segment, like a local network.

Authors advertise that one aspect could be dangerous, called over-generalization. Author advertise to this aspect, because there are certain situations that can compromise the process. As an example, one can define a cluster of events with the source IP address belonging to 192.168.0.0/24 as the cluster of events originated by the local network computers. But if the same principle is applied to events that does not belong to 192.168.0.0/24 as external events, it could be an over-generalization because it encapsulates several network sources and there is a loss of information. This kind of conceptual clustering needs domain knowledge, otherwise it is possible to introduce errors in the process and make assumptions that are not correct.

2.2.2 Anomaly Detection

Anomaly-based Data Mining for Intrusions (ADMIT) [52] presents a method to catch terminal\(^2\) users behavior based on their command sequences, and build a usage profile. Every time a specific user logs in, the user commands are analyzed in search of fraud or abnormal utilization, trying to differentiate a legitimate user from a disguised one.


K-means algorithm is a straightforward and widely used clustering algorithm. Given a set of objects or records, the goal of clustering or segmentation is to divide these objects into groups or clusters such that objects within a group tend to be more similar to one another as compared to objects belonging to different groups. In other words, clustering algorithms place similar points in the same cluster while placing dissimilar points in different clusters and is often referred to as unsupervised learning. Because there is no need for labeled data, unsupervised algorithms are suitable for many applications where labeled data is difficult to obtain. Unsupervised tasks such as clustering are also often used to explore and characterize the dataset before running a supervised learning task. Since clustering makes no use of class labels, some notion of similarity must be defined based on the attributes of the objects. The definition of similarity and the method in which points are clustered differ based on the clustering algorithm being applied.

The greedy-descent nature of k-means on a nonconvex cost implies that the convergence is only to a local optimum, and indeed the algorithm is typically quite sensitive to the initial centroid locations. In other words, initializing the set of cluster representatives differently can lead to very different clusters, even on the same dataset. A poor initialization can lead to very poor clusters.

To measure the similarity or distance between sequences, ADMIT uses the Longest Common Sequence (LCS) [66], but this technique has some inherent difficulties.

First difficulty is sometimes the great variety of commands typed by a certain user. It can

\(^2\) Unix like terminal
only be done if the user has a regular set of tasks, with a regular set of system commands. Otherwise it will be almost impossible to define such profile. Indeed, it is possible to identify commands that a regular user is not supposed to use, like network interface configuration command. Of course, if the system is correctly configured, it will not be more than a try. And could be reported, once again, as a try.

A second difficulty is the learning period. Either this learning is obtained in an isolated environment and then taken in production, or this learning must be followed by an human resource. If not, how can the system know if it is learning a correct behavior or a fraud behavior posing as a legitimate user?

There is one last observation. Depending on the detail level of such monitoring, it is always possible to encapsulate non profile commands in a script\textsuperscript{3} or batch file. In this kind of situation, the only thing this method can see is something like the text editor execution. To overcome this kind of command encapsulation, it is necessary some command fork\textsuperscript{4} audit or other system call support in the system. This support, usually adds overhead to the system.

A final consideration about some authors assumptions can be made. Authors argue that sometime user passwords used are weak, so a usage profile definition would identify a masqueraded user logged in the system. Recent systems usually have some kind of complexity requirements for user passwords\textsuperscript{5}. For example, a password must have at least two numerical characters, and must have twelve characters length minimum.

In ADMIT, authors only consider the users command sequences, without the arguments, i.e., if a user types \textquoteleft vi tile.txt\textquoteright, ADMIT will only consider \textquoteleft vi <1>\textquoteright which means the \textquoteleft vi\textquoteright command with one argument \textsuperscript{[29]}. To achieve this, the first step is to filter the list of user commands and exchange the arguments contents by the number of arguments. To ADMIT, each user session is a list of commands and the corresponding arguments number. The next step of the authors process is to build a set of clusters using a K-Means based algorithm. The number of clusters (K) and the number of iterations (t) are adjusted dynamically with the help of some extra procedures providing merge clusters and split clusters to manage groups of clusters.

For example, if two clusters seems to be close to each other, according to some distance criteria, the algorithm will merge the two clusters. In complement, centroids are not calculated every time a sequence is added because of the inherent overhead, and depending of the number of elements in a cluster, it would add a minimal change. Instead, it only recalculates the centroids after a reasonable number of sequences added. Concluding, a profile is defined by a set of clusters, and user sequence is analyzed and evaluates the distance to the centroids compared to a threshold value. If the distance does not reach the threshold, this sequence is added to the closest cluster. If the sequence does not meet any cluster, it is a suspicious sequence.

Traditional approaches to system management have been largely based on domain experts through a knowledge acquisition process that translates domain knowledge into operating rules and policies. This has been well known and experienced as a cumbersome, labor intensive, and error prone process. In addition, this process is difficult to keep up with the rapidly changing environments.

---
\textsuperscript{3}A Unix like shell script or similar, for example, a Perl script or Python script.
\textsuperscript{4}A fork or exec system call as in Unix Network Programming Vol 2 \textsuperscript{[59]}
\textsuperscript{5}As CrackLib project \textsuperscript{[19]} provides.
An Integrated Framework on Mining Logs Files for Computing System Management [63] presents a framework for mining system log files for automatic management. It applies text mining techniques to categorize messages in log files into common situations, improve categorization accuracy by considering the temporal characteristics of log messages, develop temporal mining techniques to discover the relationships between different events, and utilize visualization tools to evaluate and validate the interesting temporal patterns for system management.

Authors argue that when problems occur, traditional approaches for trouble shooting rely on the knowledge and experience of domain experts to figure out ways to discover the rules or look for the problem solutions laboriously. It has been estimated that, in medium and large companies, anywhere from 30% to 70% of their information technology resources are used in dealing with problems.

Authors considers unrealistic and inefficient to depend on domain experts to manually deal with complex problems in ever-changing computing systems.

Modern computing systems are instrumented to generate huge amounts of system log data. The data in the log files describe the status of each component and record system operational changes, such as the starting and stopping of services, detection of network applications, software configuration modifications, and software execution errors.

Analyzing log files, as an attractive approach for automatic system management and monitoring, has been enjoying a growing amount of attention. However, several new aspects of the system log data have been less emphasized in existing analysis methods from data mining and machine learning community and pose several challenges calling for more research.

The aspects include disparate formats and relatively short text messages in data reporting, asynchronism in data collection, and temporal characteristics in data representation. Authors identify three major characteristics.

First, the heterogeneous nature of the system makes the data more complex and complicated. As one knows, a typical computing system contains different devices (e.g., routers, processors, and adapters with different software components (e.g., operating systems, middleware, and user applications), possibly from different providers (e.g., Cisco, IBM, and Microsoft). These various components have multiple ways to report events, conditions, errors and alerts. The heterogeneity and inconsistency of log formats make it difficult to automate problem determination. For example, there are many different ways for the components to report the start up process. Some might log the component has started, while others might say that the component has changed the state from starting to running. This makes it difficult to perform automated analysis of the historical event data across multiple components when problems occur as one need to know all the messages that reflect the same status, for all the components involved in the solution. To enable automated analysis of the historical event data across multiple components, we need to categorize the text messages with disparate formats into common situations.

Second, text messages in the log files are relatively short with a large vocabulary size. Hence, care must be taken when applying traditional document processing techniques.

Third, each text message usually contains a time stamp. The temporal characteristics provide additional context information of the messages and can be used to facilitate data analysis.

In this work, authors describe their research efforts to address the above challenges in mining
system logs. In particular, it is proposed to establish an integrated framework for computing system management by acquiring the needed knowledge automatically from a large amount of historical log data, possibly from different types of information sources such as system errors, resource performance metrics, and trouble ticket text records. Specifically, authors try to apply text mining techniques to automatically categorize the text messages into a set of common categories, incorporate temporal information to improve categorization performance, develop temporal mining techniques to discover the relationships between different events, and utilize visualization tools to evaluate and validate the interesting temporal patterns for system management.

Authors consider that the disparate logging mechanisms impede problem investigation because of no standard approach for classifying them. In order to create consistency across similar fields and improves the ability to correlate across multiple logs, it is necessary to transform the messages in the log files into a common base, for example, by defining a set of common categories.

First, authors manually determine a set of categories as the basis for transformation. The set of categories is based on the Common Base Event (CBE) format established by IBM initiative. CBE provides a finite set of canonical situations after analyzing thousands of log entries across multiple IBM and non-IBM products. Authors use CBE as an internal presentation where the problem and the proposed approach are generic and can be extended for other data formats as well. Specifically, the set of categories includes start, stop, dependency, create, connection, report, request, configuration, and other.

Given the set of common categories, messages reported can be categorized by different components into the prescribed categories. This can be viewed as a text categorization problem where the goal is to assign predefined category labels to unlabeled documents based on the likelihood inferred from the training set of labeled documents.

In this work, authors use naive Bayes [72] in classification approach as it is among the most successful known algorithms for learning in text categorization. System log files usually have a large size of vocabulary and most of the system log messages contain a free format 1024-byte ASCII description of the event.

Basically naive Bayes assumes that the text data is generated by a parametric model, and uses training data to calculate Bayes-optimal estimates of the model parameters. Then, it classifies test data using the generative model by calculating the posterior probability that a class would have generated the test data in question. The most probable class is then assigned to the test data.

Authors consider that the classification performance achieved by the naive Bayes classifier is not satisfactory by the authors tests, if a sequence of log messages are considered, the accuracy of categorization for each message can be improved as the structure relationships among the messages can be used for analysis. For example, the components usually first start up a process, then stop the process at a later time. That is, the stop message usually occurs after the start message.

Hidden Markov Model (HMM) is another approach to incorporate the temporal information for message categorization. The temporal relations among messages are naturally captured in HMM [30]. The Hidden Markov Model is a finite set of states, each of which is associated
with a (generally multidimensional) probability distribution. Transitions among the states are governed by a set of probabilities called transition probabilities [46]. In a particular state an outcome or observation can be generated, according to the associated probability distribution. The model describes a probabilistic generative process whereby a sequence of symbols is produced by starting in some state, transitioning to a new state, emitting a symbol selected by that state, transitioning again, emitting another symbol and so on until a designated final state is reached.

In authors experiment, the category labels are regarded as states. The emitting observation symbols are the log messages corresponding to their state labels. HMM explicitly considers the state transition sequence. It can also be used to compare all state paths from the start state to the destination state, and then choose the best state sequence that emits a certain observation sequence. Authors consider that it works well in labeling continuous log messages.

When one log message has been assigned several competitive state labels by text classification, HMM selects a certain label by traversing the best state sequence. The state transition probability is calculated from the training log data sets. The probability of emitting messages can be estimated as the ratio of the occurrence probabilities of log messages to the occurrence of their states in the training data. Viterbi algorithm [66] is used to find the most possible state sequence that emits the given log messages, that is, finding the most possible state labels to assign to these log messages.

Once the messages in the log file are transformed into common categories, it is then possible to analyze the historical event data across multiple components to discover interesting patterns embedded in the data. Each message in log files usually has a time stamp and the process will try to find the mining temporal patterns. Temporal patterns of interest appear naturally in the system management applications. Specifically, a computer system problem may trigger a series of symptom events or activities. Such a sequence of symptom events provides a natural signature for identifying the root cause. Authors state that a problem manifests itself as a sequence of events propagating from origin and low layer to high software layer through the dependency tree. Thus knowing the temporal patterns can help to pinpoint the root cause and take proper actions.

Authors argue that previous work of temporal mining focuses on frequent item sets with a predefined time window. It fails to address two important aspects often required by applications.

First, the fixed time-window scheme can not explore precise temporal information within a window, and misses the opportunity to mine temporal relationship longer than the window size. In authors practice on system management applications, the temporal relationships discovered have time distances ranging from one second to one day.

Second, as well-known for transaction data, frequent pattern framework misses significant, but infrequent patterns. In most system management applications, frequent patterns are normal operations and service disruptions are usually infrequent but significant patterns.

To address the above two problems, authors develop algorithms for discovering temporal patterns without predefined time windows.

The problem of discovering temporal patterns is divided into two sub-tasks. One, using cheap statistics for dependence testing and candidates removal. Two, identifying the tempo-
reral relationships between dependent event types. The dependence problem is formulated as
the problem of comparing two probability distributions and is solved using a technique reminis-
cent of the distance methods used in the spatial point process, while the latter problem
is solved using an approach based on Chi-Squared tests. The statistical properties provide
meaningful characterizations for the patterns and are usually robust against noise.

Let $T_a$ and $T_b$ be two point processes for event $a$ and $b$ respectively. One would like to
check whether $T_b$ is dependent on $T_a$. Authors use distance methods instead of quadrat
methods (e.g., time window segment). This approach is motivated by the methods of find-
ing spatial associations between spatial point processes developed in statistical community.
The underlying idea is: if the occurrence of $b$ is predictable by the occurrence of $a$, then the
conditional distribution which models the waiting time of event type $b$ given event type $a$’s
presence would be different from the unconditional one.

2.2.3 Failure Detection

Classification of Software Behaviors for Failure Detection: A discriminative Pattern Mining
Approach [12] trays to classify software behavior based on past history or runs. This method
build profiles based on the observations of software system calls, and try to detect a faulty
behaviors if their occurred.

Once again, software execution can have a Large variability in its system calls, which could
originate a huge volume of different cases to analyze. Authors of this work have choose a
relatively regular behavior application (MySQL server) which minimizes this aspect. Authors
also advocate that the presented method can help software development process in terms of
detection of new bugs or errors based on earlier versions of developed software.

Authors proposed classification method includes three major steps: Iterative pattern min-
ing, feature selection and model learning.

First step, iterative pattern mining, concerns about mining system trace sequences for
Closed Iterative Pattern. A pattern is a Closed Iterative Pattern if it is composed by a
set of unique events, and there is no other sequence with the same events that contains that
pattern. Events are system calls within the program system traces. Authors use this definition
to reduce the number of generated iterative patterns, and to avoid mining single events with
extremely high support but with little meaning called noise, since those noisy events are
usually originated from external utility libs, for example, $toString()$ or $hashCode()$. These
patterns are mined in a depth-first fashion using a recursive algorithm trying to combine
events and grow patterns. Every time a Closed Iterative Pattern condition is satisfied, this
pattern is added to classification pattern set.

Second step, feature selection, tries to assign a discriminative measure using Fisher score
[44] approach. This method scores iterative patterns based on its support and on its proximity
with similar patterns, so similar patterns have similar scores and different patterns have
different scores. This way, is suitable manner to differentiate instances from different classes.

The last step, model learning, trains a classifier with discriminative values calculated in
the second step. This classifier has an additional difficulty which is the disparity between the
portion of anomalous behaviors and normal behaviors. Authors refers that anomalous behav-
iors are less than 10%. Such disparity dificults the classification because normal behaviors
will dominate the results and anomalous behaviorsould be poorly represented. To avoid
this, authors try to identify those anomalous patterns and duplicate them several times until
the portions are balanced between anomalous and normal behaviors.

Learning, Indexing, and Diagnosing Network Faults [67] introduce a class of indexable fault signatures that encode temporal evolution of events generated by a network fault as well as topological relationships among the nodes where these events occur. Modern communication networks generate massive volume of operational event data, for example alarm and alert, which can be used by a network management system (NMS) to diagnose potential faults.

In its simplest form, a network can be modeled as a graph where the vertices of the graph represent network entities and the edges represent pairwise interactions between network entities.

Authors propose a class of models suitable for learning, indexing, and diagnosing a wide range of network phenomena while focusing on faults in the communication networks to exemplify their techniques. Authors argue that large communication networks have hundreds of thousands of network entities that are typically managed by a centralized network management system (NMS), which collects monitoring data from network entities to diagnose network faults. When a fault occurs in a network entity, it tends to influence the neighboring entities. Consequently, faults often result in a large burst of messages being sent to the NMS from the affected entities. Each message contains a time stamp, an identifier of the affected device, and a type that signifies an event at the affected device. The goal of NMS is to correlate the events occurring in the whole network, and identify the root-cause fault event(s), suppress dependent events, and discard routine operational events.

Events triggered by a fault are typically generated by a small, constant-size subset of nodes that are topologically related to the faulty node.

Authors propose the Meta (Monitoring network Events with Topology Assistance) framework that utilize topologically-aware event patterns to perform network fault diagnosis.

Typically the operational events caused by network faults occur in bursts. Interval filter, as authors define, segments the event sequence generated by each node into a series of event bursts.

An appropriate value can be obtained by analyzing the distribution of inter-arrival time of events in a historical dataset. Since events in a burst occur within a very small time window, authors arbitrarily define the time stamp of an event burst as the time stamp of its first event.

Given a set of event bursts generated by a node, support filter treats each burst as a set and ignores the temporal ordering of events. Authors use support filter to denote both the event burst and its corresponding event set.

Authors also define the periodicity filter responsible to further reject event sets that occur periodically since periodic event sets typically correspond to regular network operations, for example heartbeat messages.

Modeling event bursts takes the events at a node, after the filtering of the first phase as input, and utilizes Markov chains to produces a set of chains as the summarization of event bursts at a node. Authors substantiate that Markovian properties have been verified to be common in network operational events. However, they do admit that it could not be the optimal approach. Authors emphasize that the framework is flexible enough to support many other models that can be used to summarize events at a node.
Authors argue that a feature that significantly distinguishes Meta from existing solutions lies in its incorporation of network topology information in learning and matching faults. They consider the following set of relationships, \{selfing, neighboring, containing/contained, down/up-streaming, tunneling\},

Intuitively, authors construct the fault signature based on the following two fundamental observations. First, typically when a fault occurs at a root-cause node, symptom events may be triggered in affected nodes that are topologically related to. Second, the triggered event burst at an affected node differs depending on the topological relationship between the two nodes. For example, in an Internet Protocol (IP) network if node v is a direct neighbor of node u, the failure of u may lead to the event burst of Interface Down, Neighbor Down at v, while if u is on a tunnel with v as one end, the failure of u may cause the event burst of Failed Connection Attempt, Open Tunnel Failure at v.

Therefore, to learn fault signatures from historical data, authors make the following assumptions: each event burst, observed at a node v has been classified into a Markov chain cv, and represented as a pair \((v, cv)\). The number of fault types is known and all faults are reflected in the historical data. The time-window size is set as the maximum delay between observing the first and the last event bursts triggered by a single fault.

2.2.4 Anomaly Detection

*Learning Rules for Anomaly Detection of Hostile Network Traffic (LERAD)* [13] presents an algorithm that learns rules for finding rare events in time-series data. This algorithm tries to find anomalies based on network packets and TCP sessions information to detect novel intrusion attempts.

Authors present an Apriori based algorithm with some variations. The first one is the way for association rules generation randomly instead of the frequency of the of the items. An efficiency improvement claimed by the authors, directly related with the fact that unlike Apriori algorithm, this method does not generates rules with one item only because its frequency is above the threshold. Authors also advocate that this method characterizes the network behavior in a better way, citing the *Wide-Area Traffic: The Failure of Poisson Modeling* [38] paper which discusses the inappropriateness of the Poisson [36] distribution in network modeling, because of the majority of network requests occur in bursts instead of time repeating requests. Which means, in general, a network requests behavior does not follow a Poisson distribution. This paper presents several protocol type analysis and show that Telnet and FTP protocols are well represented by Poisson distributions, but SMTP, HTTP, FTP-data and NNTP protocols do not.

LERAD execution is composed by two phases. First, a candidate rule set is generated from a random sample of training data. Then the rules are trained by collecting the set of allowed values for each antecedent. After this training, the rules are validated on a portion of training data to remove the lower rated rules because authors consider that those lower rated rules are not representative of the test data. The anomaly rating value for a certain rule is given by \(tn/r\), where \(t\) is the time since the last anomaly occurred, \(n\) is the number of training instances satisfying the antecedent or support and \(r\) is the number of allowed values. The idea is to identify rare events. Those which have not occurred for a long time with a large \(t\) value and where the average rate of “anomalies” in training is low, with a small \(r/n\) value. If
the total anomaly score summed over all violated rules exceeds a threshold, then an alarm is generated. Since the rule generation is made from a training data set, and by the authors, an attack-free training set, this can raise some questions.

Weighting versus Pruning in Rule Validation for Detecting Network and Host Anomalies [61] proposes a method to improve IDS rules management by the attribution of a weight to each rule according to quality criteria, instead of using a boolean approach. IDS rules management techniques often discards low rated rules, but it bears the risk of eliminating relevant information in some circumstances.

Intrusion detection has two general approaches. Signature detection, where patterns signaling well known attacks are searched, and anomaly detection, where deviations from normal behavior are flagged. Signature detection works reliably for known attacks, but has a limitation of missing new attacks. Through anomaly detection can detect novel attacks, it has the drawback of not being able to discern intent, it can only signal that some event is unusual, but not necessarily hostile, thus generating false alarms.

Authors cite LERAD work as an efficient method to learn a succinct set of comprehensible rules and detect attacks unknown to the algorithm, but instead of discarding non validated rules, authors present an improving method through weighting rules. Authors present three weighting schemes. Winnow specialist based weighting, equal reward apportioning and weight of evidence.

Winnow is an incremental weight updating algorithm for voting experts, which correspond to rules in this work. Initially all rule weights are assigned a value 1, signifying equality of belief across the rule set. For any instance, a rule must either hold good or be inapplicable, in which case it abstains from voting. Any rule violation in the verification set corresponds to a false alarm and reduces trust in culprit rule. If a rule formed during training is not useful, it is likely to be violated many times. Such rules are penalized by multiplicative decay of their weight. On the other hand, if a rule is conformed by a dta instance in the validation set when other rules where violated, it stresses upon validity of the rule and increases trust. Since the rule formed during training is expected to hold true in validation as well, it is increased its weight by a small fraction. The intent is to levy a heavy penalty by decreasing the weight by a factor $\alpha$ when the rule is violated, but increase the weights by factor $\beta$ for a conforming rule. The strategy to update weights is formally defined by $\omega \times \alpha$ if the evaluating rule is violated, and $\omega(1 + \beta)$ if the rule is conformed.

For example, assuming $\alpha = 0.5$ and initial weight 1, the weight is equal to 0.5 the first time the rule is violated. It is reduced to 0.25 upon second violation and so on. On the other hand, weight is updated as 1.5, 2.25, 3.375 for the first three conformances with $\beta = 0.5$, respectively. It can be noted that pruning is a special case of this weighting strategy, with $\alpha = \beta = 0$.

Equal reward apportioning is a variant of the winnow specialist based approach. One can observe that in winnow specialist approach weights for correct rules are incremented by a constant factor $\beta$. This results in varied weights increments across conforming rules. For example, given $\alpha = \beta = 0.5$, current weights 1.0 and 0.5 of two conforming rules are updated as 1.5 and 0.75 respectively. The winnow specialist based scheme thus favors rules with already higher weights by increasing their weights even more, resulting in potential imbalance. Moreover, the amount of weight increase is independent of whether a hight or low support
rule was violated.

The equal reward apportioning scheme adopts an impartial approach towards all conforming rules, irrespective of their current weights. This weights scheme aggregates the total weight reduction due to violation of rules, and rewards the conforming rules by equally distributing the consolidated weight mass amongst them. The amount of weight increase for conforming rules is thus dependent on the amount of weight decreased for violated rules. Following the example above, if a violated rule has weight 0.6, weights for conformed two rules are incremented by the same amount (0.15), resulting in weights 1.15 and 0.65 respectively. On the other hand, if a higher trust rule is violated with weight 1.0, it provides greater boost to the conforming rules by incrementing their weights by 0.25 each.

Weight of evidence is defined as the measure of evidence provided by an observation in favor of a target attribute value as opposed to other values for the same target attribute. This measure is based on information theory and has been applied in classification tasks based on event association. It is used a scheme of associations of weights with the rules in the rule set. The weights is computed for each rule based on its evidence. Contrary to the previous two incremental weighting techniques, this involves batch weighting where evidence is consolidated from the validation set as a whole. Moreover, weight of evidence can be positive, negative or zero. A positive value reflects high trust in the rule whereas a negative or zero value implies otherwise. Only rules with positive weights are kept and the remaining may be eliminated. One can also scale the values by a linear shift of the axis such that all weights are positive. Now the positive weight of evidence of the high support rules have high positive weights, whereas the low (negative or zero weight of evidence) trust rules have low positive weights.

*Learning Nonstationary Models of Normal Traffic for Detection Novel Attacks* [33] tries to combine information retrieved from signature based network intrusion detection systems (NIDS) with pattern mining to improve security by detecting non documented network attacks or anomalies. Signature based systems like classical Intrusion Detection Systems (IDS) or Anti-Virus are good to detect well known attacks, but are often weak in new or unknown attacks.

Authors tries to model a normal network traffic from a training set. In a learning process based on a training set applied to a such variable environment like a computer network, some difficulties should be expected. It could be difficult to reproduce the real environment. And if it could be well defined, it is because this environment is already known, learning, could not be a major necessity.

This work relies on non stationary probabilities models to model the computer network events. A stationary model is based on the average rate of the events over time. Contrarily, a non stationary model is based on the time since the last event as occurred. Based on Wide-Area Traffic: The Failure of Poisson Modeling [38] authors claims that stationary probabilistic models, based on the Poisson distribution, does not correctly models the network behaviors. Instead, authors argue that network behaviors reveals fractal behaviors, in which no sample, no matter how short or long, can predict the rate of events for any other sample, that is, network behaviors can be described by a nonstationary model. Events tend to occur in bursts separated by long gaps in all time scales, from milliseconds to months.
2.2. DATA MINING

In this work authors define the probability of an event \( x \) as \( 1/t \), where \( t \) is the time in seconds since last event \( x \). In anomaly detection systems the focus are events that have the lowest probability, otherwise those events become normal events. Authors simplify this point by assigning anomaly scores to events that have never occurred in the training phase. This model lays its classification model in a dimension \( n \) sample, and for each event \( x \) it computes the amount of times that event as occurred. For each occurrence it computes \( tn/r \), where \( r \) is the number of outcomes, and sums this anomaly score to a variable. If this sum exceeds a threshold value, an alarm is triggered.

Events are selected using two criteria: Packet Header Anomaly Detection (PHAD) and Application Layer Anomaly Detection (ALAD). PHAD monitors the fields of the network packet headers, from the data layer, network layer and transport layer \[62\]. The value of each field is an integer, so it is impractical to represent every observed value. To overcome this difficulty, authors define a set of ranges, clusters, to represent those values. If the cluster number reaches a limit value \( C \), then the closest two ranges are merged.

ALAD assigns a score value to the incoming TCP connections, instead of incoming network packets. ALAD uses a set of combinations of source and destination of IP addresses, and also port number. In complement, it uses the knowledge of application protocol to analyze the correction of some keywords used, as well as standard application ports.

Like PHAD, ALAD uses the same criteria, \( n/r \) to score abnormal events. It should be observed that encrypted communications could not be easily analyzed, unless it uses some special method applications, for example, Microsoft ISA server\(^6\) \[55\]. Otherwise, it will not be easy to intercept and decrypt the encrypted connections.

*Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URL’s* \[27\] describes a method based support vector machines to automate URL classification, using statistical methods to discover the tell-tale lexical and host-based properties of malicious Web site URL’s.

Support vector machines is an instance classification method. Usually each instance has a numerical mapping and is translated to a \( n \) dimensional point. This method assumes that the classes are linearly separable, and there is a training set composed by several point defining the separation line or hyperplane between the classes, called support vector\[71\].

Internet users have become targets of a wide range of criminal enterprises such as spam-advertised commerce like counterfeit watches or pharmaceuticals, financial fraud like phishing or 419-type scams, as well as malware propagation. Although the precise commercial motivations behind these schemes may differ, the common thread among them is the requirement that unsuspecting users visit their sites. These visits can be driven by email, Web search results or links from other Web pages, but all require the user to take some action, such as clicking, that specifies the desired URL.

Clearly, if one could inform users beforehand that a particular URL was dangerous to visit, much of this problem could be alleviated. To this end, the security community has responded by developing blacklisting services, appliances and search engines that provide

\(^6\)Microsoft ISA server implements a *legal version* of a method called *man in the middle attack* \[14\]. This method intercepts the communication between the client and the server, and act as *store and forward* mechanism. The client talks only with ISA server that inspects all traffic before forwarding the traffic to server, and the same happens from the server to client traffic.
precisely this feedback. These blacklists are in turn constructed by a range of techniques including manual reporting, honeypots\textsuperscript{7}, and Web crawlers combined with site analysis heuristics. Inevitably, many malicious sites are not blacklisted, either because they are too new, were never evaluated, or were evaluated incorrectly, for example, due to cloaking. To address this problem, some client-side systems analyze the content or behavior of a Web site as it is visited. But, in addition to run-time overhead, these approaches can expose the user to the very browser-based vulnerabilities that should be avoided.

This paper focuses on a complementary part of the design space, lightweight URL classification that is, classifying the reputation of a Web site entirely based on the inbound URL. Authors' motivation is to provide inherently better coverage than blacklisting based approaches, correctly predicting the status of new sites, while avoiding the client-side overhead and risk of approaches that analyze Web content on demand.

Authors explore the use of statistical methods from machine learning for classifying site reputation based on the relationship between URL’s and the lexical and host-based features that characterize them.

Authors advocate that the system selects new, non-obvious features that are highly predictive and yield substantial performance, supported by the evaluation of across 20,000 to 30,000 URL’s drawn from different sources, and argue that it can obtain an overall prediction accuracy of 95 to 99%, detecting a large fraction of malicious websites while maintaining a very low false positive rate.

The classification of the sites is based only on the relationship between URL’s and the lexical and host-based features that characterize them, and it is not considered two other kinds of potentially useful sources of information for features: the URLs page content, and the context of the URL, for example, the page or email in which the URL is embedded. Although this information has the potential to improve classification accuracy, it is excluded for four main reasons.

First, avoiding downloading page content is strictly safer for users.

Second, classifying a URL with a trained model is a lightweight operation compared to first downloading the page and then using its contents for classification.

Third, focusing on URL features makes the classifier applicable to any context in which URL’s are found like Web pages, email or chats, rather than dependent on a particular application setting.

Finally, reliably obtaining the malicious version of a page for both training and testing can become a difficult practical issue. Malicious sites have demonstrated the ability to cloak the content of their Web pages, i.e., serving different content to different clients. For example, a malicious server may send benign versions of a page to honeypot IP addresses that belong to security practitioners, but send malicious versions to other clients. Authors state that

\textsuperscript{7}Honey pots are individual computers or entire networks created to serve as a snare for intruders. They look and act like legitimate networks, but they are 100 percent fake. Honey pots tempt intruders by containing unpatched and unprotected security vulnerabilities as well as by hosting attractive and tantalizing but faux data. They are designed to grab an intruders attention and direct them into the restricted playground while keeping them away from the legitimate network and confidential resources. Legitimate users never enter the honey pot; there is no real data or useful resources in the honey pot system. Thus, when honey pot access is detected, it is most likely an unauthorized intruder. Honey pots are deployed to keep an intruder logged on and performing their malicious activities long enough for the automated IDS to detect the intrusion and gather as much information about the intruder as possible. The longer the honey pot retains the attention of the intruder, the more time an administrator has to investigate the attack and potentially identify the person perpetrating the intrusion.
classifying on lexical features of the URL and features about the host are sufficient for highly accurate prediction.

The justification for using lexical features is that URL's to malicious sites tend to look different in the eyes of the users who see them. Hence, including lexical features allows to methodically capture this property for classification purposes, and infer patterns, if possible, in malicious URL's that would otherwise miss through ad-hoc inspection. This work distinguishes the two parts of a URL. The hostname and the path. As an example, with the URL www.geocities.com/usr/index.html, the hostname portion is www.geocities.com and the path portion is usr/index.html.

Lexical features are the textual properties of the URL itself, not the content of the page it references. These properties include the length of the hostname, the length of the entire URL, as well as the number of dots in the URL. Additionally, it creates a binary feature for each token in the hostname, delimited by “.”, and in the path URL, strings delimited by “/”, “?”, “:”, “;”, “=”, “,”, “.”, and “.”. The order of the tokens is not preserved, and it is made a distinction between tokens belonging to the hostname, the path, the top-level domain (TLD) and primary domain name.

The reason presented by the authors for using host-based features is that malicious Web sites may be hosted in less reputable hosting centers, on machines that are not conventional web hosts, or through disreputable registrars. To an approximate degree, host-based features can describe where malicious sites are hosted, who own them, and how they are managed. The following are properties of the hosts that are identified by the hostname part of the URL. Authors argue that some of these features overlap with lexical properties of the URL.

- IP address properties: Is the IP address in a blacklist? Are the IP’s of the A, MX or NS records in the same autonomous systems (ASes) or prefixes as one another? To what ASes or prefixes do they belong?
- WHOIS properties: What is the date of registration, update, and expiration? Who is the registrar? Who is the registrant? Is the WHOIS entry locked?
- Domain name properties: What is the time-to-live (TTL) value for the DNS records associated with the hostname? Additionally, the following domain name properties are used in SpamAssassin Botnet plugin for detecting links to malicious sites in emails: Does the hostname contain client or server keywords? Is the IP address in the hostname? Is there a PTR record for the host? Does the PTR record in turn resolve one of the hosts IP addresses?
- Geographic properties: In which continent, country or city does the IP address belong? What is the speed of the uplink connection (broadband, dial-up, etc)?

Authors use these features to encode individual URLs as very high dimensional feature vectors. Only a subset of the generated features may correlate with malicious Web sites, and it is not know in advance which features are relevant. More generally, when there are more features than labeled examples, it enters the regime in which statistical models are most prone to overfitting.

This work uses three different classification methods. Naive Bayes, support vector machines (SVM) and Logistic Regression.
Naive Bayes, commonly used in spam mail filters, assumes that for a given label, the individual features of URL's are distributed independently of the values of other features. The classification is made by computing a conditional probability of an URL being malicious given its features set. Author argue that naive Bayes classifier is most easily trained by computing the conditional probabilities from their maximum likelihood estimates. For real value features, authors model the probability by a Gaussian distribution whose mean and standard deviation are computed over the respective component from component vector in training set. For binary value features, authors estimates the probability as a fraction of the feature vector in the training set.

The model parameters in the Naive Bayes classifier are estimated to maximize the joint log-likelihood of URL features and labels, as opposed to the accuracy of classification. Optimizing the latter typically leads to more accurate classifiers, notwithstanding the increased risk of overfitting.

SVM's are trained to maximize the margin of correct classification, and the resulting decision boundaries are robust to slight perturbations of the feature vectors, thereby providing a hedge against overfitting.

The decision rule in SVMs is expressed in terms of a kernel function that computes the similarity between two feature vectors and non-negative coefficients that indicate which training examples lie close to the decision boundary. SVMs classify new examples by computing their (signed) distance to the decision boundary. The sign of this distance indicates the side of the decision boundary on which the example lies. SVMs are trained by first specifying a kernel function and then computing the coefficients that maximize the margin of correct classification on the training set.

Logistic Regression is a parametric model for binary classification where examples are classified based on their distance from a hyperplane decision boundary. The decision rule is expressed in terms of the sigmoid function which converts these distance into probabilities that feature vectors have positive or negative labels.

With the evolution of the information technologies and with its adoption from organizational environments, the amount of available information has grown, as well as the necessity of extracting benefits from this information to increase the knowledge. Different tools try to address different domains, and sometimes different tools consider different concepts of the same definition. For example, the Simple Event Correlator and Splunk 4 presents different concepts for event correlation, as it is presented next.

2.3 Log Tools

2.3.1 Log Analyzer Tools

*Simple Event Correlator (SEC)* [10] is an open source and platform independent tool, written in Perl [51], that tries to match predefined patterns rules in a set of input streams like a pipe or a file.

It works in a “IF-THEN” basis and if a set of events match a predefined rule than it triggers an action like running a command or sending an e-mail. Rules are Perl regular expressions and subroutines that specifies the events pretended to be caught, stored in a configuration file.
2.3. LOG TOOLS

Fulfilling with an example, rules have the form “if I find an event A, and within 30 seconds I find the event B then I execute the command C”, where events A and B are define with Perl regular expressions. SEC can be complemented with Log Pre Processor (LPP) [43], a tool for preprocessing log files, trying to match lines with rules, like regular expressions, convert those lines according to given templates and write them to the destination.

LPP actually can act as a filter preprocessing log files and weed out irrelevant lines, reducing the overhead of the subsequent analysis. As an attentive can reveal, the correlation concept used in SEC is different for the classical correlation used in data mining based, for example, in Apriori based algorithms [72], which uses the probabilistic definition [36]. In data mining, two events are correlated if one can express the occurrence of an event based on the probability of another particular event. In SEC, the correlation is already defined in the configuration files, so while data mining can reveal unknown correlation patterns, SEC is unable to.

**Swatch** [60] is designed to monitor system activity. In order for Swatch to be useful, it requires a configuration file which contains pattern(s) that looks for and action(s) to perform when each pattern is found.

**Tenable network security** [64] has a suite of security products including event correlation and anomaly detection components. As commercial product, the information provided is marketing oriented and does not explain how the event correlation is made, or the technology employed.

**LogSurfer and LogSurfer+** [32] Logsurfer is a program for monitoring system logs in real-time, and reporting on the occurrence of events.

It is similar to the Swatch program on which it is based, but offers a number of advanced features which swatch does not support. Logsurfer is capable of grouping related log entries together - for instance, when a system boots it usually creates a high number of log messages. In this case, Logsurfer can be setup to group boot-time messages together and forward them in a single Email message to the system administrator under the subject line “Host xxx has just booted”.

Swatch just could not do this properly. Logsurfer is written in C, what makes it efficient, an important factor when sites generate a high amount of log traffic. Authors advocate that have used Logsurfer at a site where a logging server was recording more than 500,000 events per day, and Logsurfer had no trouble keeping up with that load. Swatch, on the other hand, is based on Perl and runs into trouble even when dealing with a much smaller rate of log traffic.

**Logsurfer+** is a branched version of the standard Logsurfer package from DFN-CERT, and it has been modified to add a few features to improve what can be done with it.

**Microsoft** is an undeniable reality in today’s information systems. In some aspects Microsoft takes a different approach from a wide range of operating systems to try to achieve its goals. Because of this, Microsoft systems appear often in separate sections. To present some motivation for this differentiation, it can be said that Microsoft have a very poor pipes & filters pattern employment, once Microsoft puts the majority of its effort in point & click interfaces, and it can also be said that as enterprise in opposition to open-source platforms, Microsoft’s main drivers are economical so Microsoft only have incentive to solve a certain problem if it will be profitable, otherwise the solution will be delayed or simply ignored.
As Microsoft uses a different model, Microsoft Windows logs are not stored in a text files that can be directly processed using text manipulation tools. Instead they are stored in a kind of system database, so to query Microsoft Windows log system, it is necessary to communicate with this system database.

Microsoft Log Parser 2.2 [1] is a tool provided by Microsoft that provides both connection to log system database and a text-based output.

It can be seen as an attempt to eliminate the gap between Microsoft Windows and other systems or tools. Log Parser 2.2 claims to be a very versatile tool which is able to process a wide variety of input format, such log files, XML and CSV files or Microsoft Windows Event Log or Microsoft Windows Registry system. It outputs the results in text-based format, syslog or even SQL. In resume, it can be said that Microsoft Log Parser 2.2 pretends to be an integration tool that aggregates external tools and the native system.

To complement the log analysis task, there are some well-known tools.

SmarterStats [57] is a web log analytics tool with several capabilities including statistical analysis and some data mining principles. SmarterStats is oriented to web site commercial improvement according to hit rate calculation and statistical evaluation.

It claims the ability to find hidden customer behavior patterns in navigation data using data mining techniques, in a question and answer format. For example, “What are the referrers of this file?” or “What sites were people on before they hit this file?”.

As a commercial tool, its objective is to be user-friendly and targeted to its clients, but in an academic sense, a major consideration can be made. Since there is a set of fixed questions whose the answer comes from the data, the main pattern types and analysis are already defined, so it could be a little bit restrictive in some circumstances because it is possible to have the necessity to formulate different questions. Say, the documentation questions are path find questions in users navigation. Questions are “what happens before/after” questions type. Different question types could be raised like “it is possible group up a certain set of pages in a different section?”, which tries to find similarities in the navigation process.

2.3.2 Log Indexer Tools

Splunk 4 [58] is a log aggregator and analyzer capable to index large amounts of information and provides a large set of search types and reports. Splunk 4 is not a security exclusive application. This work will focus in data mining aspects related to security improvement.

Considering event correlation features, Splunk 4 uses user defined rules like SEC but is more user interactive and less “real time” automated. That means that like SEC, Splunk 4 does not provide a purist data mining event correlation. Instead it provides a couple of functionalities which allows the user to research what is relevant to them.

Splunk 4 is not automated with predefined rules like SEC, because it uses a more interactive way, presenting the results to the user, and with the results, the user will decide what will be searched next. Following this reasoning, Splunk 4 uses two mainly functionalities, event correlation through time event correlation with search commands and.

Event correlation through time generally uses domain knowledge like regular use of a certain resource during a certain period of time, and tries to extract information by analyzing a specific period of time in contrast with the domain known information. For example, the work periods are usually known, like what days and what hours, Splunk4 provides methods to know the resources usage out of work periods, and possibly discover uncommon usage
2.4. **INTRUSION DETECTION SYSTEMS**

patterns. It can bring, in some cases, possible security concerns.

Event correlation with search commands offers a way to apply consecutive filters, reducing the amount of information by restricting the number of results with even greater detail level, as well as its relevance. For example, to find unsuccessfully SSH password attempts, first it would be searched all SSH password attempts, and the result would be searched with ssh failed attempts. This method is interactive because the user can adjust the search criteria in each step based on the results and what it is wanted to be researched.

*Sisyphus* [56] is a log data-mining toolkit that tries to provide a way to find important information in log files by fast index-based searches when one knows what to find, but in a complementary way to *splunk* 4, it tries to help to find anomalies when one do not know what to look for.

It ranks log files and highlights words based on *information theory* [65] trying to answer “what is the strangest log file, and why?”. It also provides both file and word statistics in tabular and plot format.

*Sisyphus* makes use of an external tool called *LogHound* [31]. *LogHound* is a tool from the authors of *Mining Sequential Patterns: Generalizations and Performance Improvements* [70] that was designed for finding frequent patterns from event log data sets with the help of a breadth-first frequent item set mining algorithm. *LogHound* can be employed for mining frequent line patterns from raw event logs, but also for mining frequent event type patterns from preprocessed event logs.

*Sisyphus* has been developed specifically for use with supercomputer *syslogs*, based on the premise that similar computers correctly executing similar workload should produce similar logs, but it claims to be general enough to be used on other log types using some custom made configurations.

**2.4 Intrusion Detection Systems**

*Advanced Intrusion Detection Environment (AIDE)* [3] is a host-based intrusion detection system (HIDS) pretended to be a free replacement for *Tripwire* [68].

*AIDE* is a file integrity checker that constructs a database with a predefined list of files to be verified. The *AIDE* database stores several file attributes like file size and permissions, *inode* number, user and group information, modification time *mtime*, number of links and other information. *AIDE* also creates a cryptographic checksum or hash of each file using a digest algorithm like SHA-1 or RMD160 [50].

The deployment process lays in the principle that a system administrator will create an *AIDE* database on a new system before it is brought onto the network. This first *AIDE* database is a snapshot of the system in its normal state and the yardstick by which all subsequent updates and changes will be measured. The database should contain information about key system binaries, libraries, header files and all files that are expected to remain the same over time. The database should not contain information about files which change frequently like log files, mail spools, *proc* file systems, user’s home directories, or temporary directories. Otherwise these files will always trigger unnecessary alerts.

Files can be later checked against the previously build database because while it is easy to manipulate file dates and sizes, it is much more difficult to manipulate a single cryptographic checksum like MD5, and exponentially more difficult to manipulate each of the entire array of
checksums. By rerunning AIDE after a break-in, a system administrator can quickly identify changes to key files and have a fairly high degree of confidence as to the accuracy of these findings.

**Snort** [22] is an open source network intrusion prevention and detection system (IDS/IPS) that tries to combine the benefits of signature, protocol and anomaly-based inspection.

**Snort** can be configured to run in three distinct modes: sniffer mode, network intrusion detection system (NIDS) mode and inline mode.

In sniffer mode, Snort simply reads the packets of the network and displays them in a continuous stream on the console.

In network intrusion detection system Snort analyzes network traffic for matches against a user-defined rule set and performs several actions based upon that.

Finally, in inline mode Snort obtains packets from *iptables* instead from *libpcap* and causes *iptables* to decide if it drops or passes packets based on Snort evaluation. Packet inspection is usually used in live traffic and protocol analysis or in forensic investigation.

Network intrusion detection is an even more complex activity because it not only limits itself to inspect packet contents, it tries to detect strange behaviors like, for example, two dozen connection requests from the same IP address at the same second. It is a port scan typical behavior.

**Snort** integrations with complementary tools like the *iptables* packet filter (or firewall tool) is a way to extend the firewall system to have intrusion prevention mechanisms. To achieve this, a Snort instance is running in daemon mode and *iptables* calls the Snort API to analyze the event, dropping or accepting the packet according with the Snort decision.

These tools help system administrators to preserve important system files integrity by alerting if some file attributes changed or some strange network behavior is detected.

Other intrusion prevention system solutions are available. Cisco [9] or Radware [47] appliances are product examples. These products prevention capabilities are usually based on threshold values of specific network traffic, and protocol inspection.

When threshold values are reached, preventive action is triggered and, for example, the traffic is blocked for a certain amount of time. This is a common behavior used to avoid service degradation in the infrastructure.

Protocol inspection is its essence an application firewall, for example, Apache ModSecurity [35], that monitors all the traffic of a specific application in real time, searching the protocol exchanged information that matches the rules, allowing only authorized traffic.

**2.5 Open Issues**

There are some research works that try to improve the accuracy and configuration of some security tools, or simply infer knowledge based on terminal users interaction, but it was not found relevant work trying to relate interactions in network services of an infrastructure.

Relevant log analyzer tools have different goals amongst them, from the indexation of huge amounts of log data to the identification of predefined event sequences in log files, but it was not found any tool that has the discovery of sequential patterns in clients interaction in a network infrastructure.
Data mining have great potential of data analysis to help to discover non trivial usage patterns. This work researches the application of sequential pattern mining techniques to discover typical behaviors in network infrastructure.

2.6 Goal

The security of an infrastructure is based on security policies. These policies define what restrictions should be applied in the infrastructure to allow users to work, but prevent the infrastructure from getting harmed. But absolute security is virtually impossible to reach. Instead it is tried a trade off between the cost to secure a value, and the necessary effort to break the security.

Nowadays network security applications can do a great job preventing a reasonable amount of threats, but even today, these security applications continue focused in each system individually, what makes the task of security policies validation an hard task. Data mining, particularly sequential pattern mining, provide method to analyze infrastructure data from each system, and try to extract patterns that represent network usage behaviors. With these patterns, a security expert can evaluate how security policies are adjusted with the real infrastructure usage.
3. Network Usage Exploration

Nowadays, network security approaches have a preventive perspective and tend to focus in the security of each single system. Prevention is essential because it aims to intercept and act over previously known harmful behaviors, but it is also important to be able to validate the prevention policies and, when possible, to detect undesired behaviors in the network usage.

This chapter presents a method to analyze server log files regarding the infrastructure as a whole, employing data mining techniques and trying to discover usage patterns.

The goal of this work is to identify the common usage behavior in the infrastructure aiming to prevent undesirable behaviors, and to improve the infrastructures efficiency, in the future.

In order to reach this goal, sequential pattern mining is used to discover frequent usage patterns. One of the main problems of pattern mining is the combinatorial explosion of patterns. To minimize this problem, the logging data is reduced using domain knowledge.

This chapter is organized as follows: Sec. 3.1 describes the methodology followed by this work, Sec. 3.2 describes the data sources criteria that can be used, Sec. 3.3 describes the infrastructure issues, Sec. 3.4 describes how the selected data should be composed and Sec. 3.5 describes how composed data can be mined.

3.1 Methodology

CRoss-Industry Standard Process for Data Mining (CRISP-DM) was conceived in 1996 by a consortium formed by NCR Systems Engineering Copenhagen (USA and Denmark), DaimlerChrysler AG (Germany), SPSS Inc. (USA), aiming to systematize some of the data mining processes previously developed by these consortium entities individually.[39]

CRISP-DM [54] organizes the data mining process into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. Fig 3.1 presents an adaptation of the CRISP-DM model focused on this work subject.

Business understanding, is one of the most important phases of any data mining project. This goals phase focuses on understanding the project from a business perspective, converting this knowledge into a data mining problem definition, and then developing a preliminary plan designed to achieve identified goals. In order to understand which data and how it should later be analyzed, it is vital for data mining practitioners to fully understand the business for which they are finding a solution. The business understanding phase involves several key steps, including determining business objectives, assessing the situation and determining the data mining goals.

Regarding to a network infrastructure, the business objectives are the availability and quality of the services provided. A basic network infrastructure with name service, mail service, web service. Other organization specific services can exist, like authentication or file sharing services.

Data understanding phase starts with an initial data collection, to identify data quality
3.1. METHODOLOGY

Figure 3.1: CRISP-DM Adaptation

problems, to discover initial insights into the data, or to detect interesting subsets to form hypotheses about hidden information. The data understanding phase involves four steps, including the collection of initial data, the description of data, the exploration of data, and the verification of data quality.

Log files are the main source of information containing output messages of the network services. These messages can be error messages or simply warnings, usually starting with a time stamp identifying when the event occurred. Error messages can be found with the explicit error word, or with a specific application code number. For example, Apache web server [53] identifies its messages with specific http [45] numbers. According to each service, entries have the address of the client that originates the event message. This property helps to correlate log entries amongst different services.

Data preparation phase covers all activities to construct the final data set or the data that will be fed into the modeling tools from the initial raw data. Tasks include table, record, and attribute selection, as well as transformation and cleaning of data for modeling tools. The five steps in data preparation are the selection of data, the cleansing of data, the construction of data, the integration of data, and the formatting of data.

The selection of data consists in the identification of the relevant log files for data mining process. Log files are usually semantically organized in each system and have a file for each subject. Subjects can be, for example, authentications, periodic tasks, kernel messages or services provided by the server.

The cleansing of data consists in the elimination of noisy data entries from log files to avoid the presence of frequent item sets. These entries can be originated by monitoring, automatic maintenance tasks and other entries that does not add relevant data to the process. For example, specific hardware related messages.

The formatting of data consists in the selection of the relevant parts of log event messages, applying the necessary manipulations so data can be processed. For example, manipulating the date format to start by year, then month and day, which is easier to sort, instead of month, day and year. Log files are, in general, text files so log files can be easily processed by text tools, but in data mining processes it is often convenient to convert those values from
The Integration consists in collecting and put together the relevant data to be processed. In this work, data mining process considers several log files from different servers. These log entries are merged according to time stamps in each entry to form a set time ordered events. The resulting set is composed by entries from different servers in the same time line.

**Modeling** phase consists in the selection of convenient discovery techniques and parameters calibration values.

This work uses sequential pattern mining by the application of the PrefixSpan [25] algorithm or other sequential pattern mining algorithms. Constraint Relaxations can be used to analyze patterns that differ less than number of items.

Before proceeding to **deployment** phase, it is important to more thoroughly evaluate the model and review the models construction to be certain that it properly achieves the business objectives. This is critical to determine if some important business issue has not been sufficiently considered. The key steps here are the evaluation of results, the process review, and the determination of next steps.

Sequential pattern mining, by its nature, can only be evaluated after being applied. In the scope of this thesis, it is expected to have sets of frequent sequences of events (item sets) as result. This result is highly dependent of the analyzed data.

Considering that different item sets sequences have different frequencies, the **threshold** parameter controls the limit value for a sequence to be considered frequent.

**Evaluation** will be made comparing the infrastructure access patterns using local access information, and using no local access information. It is expected that the classification using local access information improves the classification accuracy.

The **development** of end-user tool for deployment in a production environment is beyond the scope of this thesis. The deployment phase would consist in producing an end user tool, tuned for this problem and producing results in a format useful for those who need it.

### 3.2 Business Understanding: Target Infrastructure Model

To begin a mining process, the starting point is the definition of the sources of data based on domain understanding. The aim of this work is to discover sequential patterns in clients interaction with a service network infrastructure. The mining process can also accomplish any data source that is considered relevant.

Fig. 3.2 shows the coverage of this work. The client interacts with the public services available in the infrastructure, but each service has other data that can be considered, for example, the local system kernel messages. It also can be considered data from some local service running in the server, for example, local mail delivering system\(^1\).

The infrastructure can be composed by POSIX\(^2\) compliant systems, Linux for example,

---

\(^1\)In certain systems, it is possible to have local mail delivering messages to system administrators for particular events types.

\(^2\)POSIX compliant systems are systems that use the POSIX\(^{[40]}\) standard. Linux or OpenBSD are examples of POSIX compliant systems.
3.2. BUSINESS UNDERSTANDING: TARGET INFRASTRUCTURE MODEL

Microsoft Windows systems or an hybrid combination of different systems.

POSIX systems commonly have some configured log facilities by default. Facilities are sections with semantic organized logging information. These facilities configuration defaults are defined to achieve the system correct functions acting like filters to put the right information in its correct place. Other external systems installed, for example, a remote login server (SSH), can also take advantage of the default facilities or define new ones if needed. Messages usually have attached the severity level, for example, notification or error.

Microsoft systems follow a similar philosophy, but properly conceived to Microsoft technology. Which means that the log information in Microsoft systems holds the same basic principles, but have some implementation differences. For example, Microsoft have a system log section for the systems own events and an application log sections to store information generated by external installed applications. Microsoft also define severity levels for logging messages, for example, information or error.

The mining process does not have to be necessarily a passive process. This means that it does not have the constrain of collecting only some existent data. It can be added data deliberately built to the process. For example, it can be logged the average load of the server over time, memory consumption, network traffic speed or other data considered to be a useful contribution to the knowledge discovery.

In a POSIX compliant infrastructure the information can be obtained by reading the log files, which are in simple text format for the majority of the cases, to be easily human readable. In exceptional situations the log information can be stored in binary format, and have a proper tool to convert the logged data to a human readable format.

In a Microsoft infrastructure the information can be obtained through the Microsoft application programming interface (API) connecting to Microsoft Windows event logging system.

In an hybrid infrastructure, these two alternatives must be combined, which means that it must exist an application to read Microsoft Windows events from the event viewer and another application to read the necessary log files in the POSIX compliant system. It is also necessary to normalize and integrate the information gained from the two data sources.
3.2.1 Data Understanding

To apply the mining process, it is necessary to identify the data sources and, ultimately choose the files to be mined.

POSIX systems usually have a proper directory for log files. There are services that create a subdirectory in the log directory to store its own files bringing additional organization to the log system.

Considering a POSIX system gateway, with the defaults system tools, the log information will be stored in defaults system log files. For example, the kernel log files. It can have additional complexity once the kernel log have all other messages generated by the kernel. To overcome this aspect, it is necessary to build filters to extract only the relevant events. If the gateway is an appliance, it will have its log system, for example, as a particular service different from the system kernel itself.

A Linux name service, for example, BIND service, when it is possible follows a different approach constructing a separated directory tree for all service, including log files, and makes the high level directory of the created tree the root of the system file system, only to the service. A service that uses this technique is called a chrooted\[6\] service. Some other services, like Apache web service, can use this technique. In such cases, the log files are not stored in the systems log directory, but are stored in a complete separated directory.

FTP services in a Linux system, for example, pure-ftpd server, usually store its log files in a separated directory in the logs directory. The log files have all generated information by the service.

If the Linux system that hosts the FTP service provides files using other services, via rsyn or HTTP for example, these other services usually have a separate directory for each service itself.

A Linux web service, for example, Apache HTTP server, usually have logs separated in two categories. The access category and the error category, containing the access entries and the generated errors, respectively. If the server is configured to support secure socket layer (SSL), then there will exist two new separated files logging accesses and errors originated by connections established over SSL.

In Microsoft Windows servers it is necessary to identify which are the relevant events to be mined. Microsoft systems have an optional integrated services, Internet Information Services (IIS), containing a web service, a mail service and a FTP service. Microsoft name servers are usually configured in Microsoft Windows domains. These Microsoft components naturally integrate its log information with the system logs.

Indeed, it is possible to have these services by installing other components than Microsoft components. In such situations it is common to have the service logging to text files in a defined directory, similarly to a Linux service.

\[6\]It is beyond the scope of this work to discuss the security properties of the chroot concept, but more information can be found in [6].
3.2. BUSINESS UNDERSTANDING: TARGET INFRASTRUCTURE MODEL

3.2.2 Notes about the data types

Sequential pattern mining was not designed to deal with continuous data types. Because of this, not all data types are good to be used directly in the process. For example, a “client logged in message” or a “file download successfully” can be easily tagged and used in the mining process. For example, the “file download successfully” message can be converted into a tag named “FDS” representing the information contained in the text message. But in the opposite, if it is considered the number of bytes transferred by the network interface, which are a continuous data type, so each recorded value is tagged to the process. This creates a problem, once these recorded values are not good candidates to be frequent items. Only long idle periods will make certain values frequent. So, in similar circumstances, it should be considered instead, the speed values of the transferred information.

Considering the network interface speed in detriment of transferred bytes by the network interface resolves the problem of dealing with a continuous data type, but leads to a different problem. The problem of dealing with a wide range data type.

Recording the network speed over a time line, can contain lots of different values, which will originate a large set of different labels in the process, with low frequency for each label. This fact will bring overhead to the process and will produce poor results because, after a large number of generated candidates is processed there is a low probability of these low frequency candidates are selected to be part in the next iteration. For example, considering a 100 Mbps network interface, the speed value can be any number between 0 and 100, and floating point values are not excluded.

One way to overcome this problem is to consider a set of partitions of the range. For example, if in a certain moment of the time the network speed is 78.6 Mbps and in the next moment the value is 79.2 Mbps, it makes sense to gather these values in a range with an expressive label. This way, a possible classification can be “min” for values between 0 and 19.99, “low” for values between 20 and 39.99, “medium” for values between 40 and 59.99, “high” for values between 60 and 79.99, and “maximum” for values between 80 and 100 as shown in Fig. 3.3.

![Figure 3.3: Wide Range Partition Example](image)

Regarding to Fig. 3.3, the five groups defined to gather a wide range of values seams acceptable for a first look, and certainly it would comply with most applications necessities in a similar area. Of course, the number of groups will be highly dependent of the specific application in which the classification will be applied. If the number of groups is a large number, for example, if the range interval is divided in groups of two integers, say from 0 to 1.99, from 2 to 3.99 and so on, it will generate a large set of labels and consequently each label will have a low support, and could be dropped in the mining phase.

On the other hand, if the number of groups is to small, for example, from 0 to 49.99 and then, from 50 to 100, it is applied a kind of binary classification. In that case, each group will appear with high support because there are only two groups, but the information added is poor because there are only two groups that can contain a high number of values, condensing relevant information.
The number of groups should achieve a balance between the number of combinations that it could generate and the information that the groups contain.

### 3.3 Target Infrastructure Model

Target infrastructure model systematizes the relevant information about the organization used by this process. This section starts with presentation of a generic service infrastructure, as represented in Fig. `fig:infrastructure-components`. Then, it is presented an information representation for each component.

A simple organizational infrastructure commonly has three basic types of components. First, a gateway that connects the network segments. Second, the servers network providing the services to be available to the public. Third, the clients. Clients can be any computer interacting with the services, from within the organization, or from the Internet.

![Figure 3.4: Infrastructure Components](image)

1. **Gateway.** The gateway is a very important component in the network, once it usually have, not only routing capabilities, but have also firewall and traffic shaping functions.

2. **Name Server.** As a critical component, usually this service is replicated in two (or more) servers, the primary name server and the secondary name server. This replication aims to mitigate unavailability of service by an hardware problem or misconfiguration of the service.

3. **Web Server.** The web server usually contains important services for the organization like the public web page or a web service available to the Internet to the clients. It can also contains other important functionalities to the organization itself, like an “intranet” system or simply a discussion forum.

4. **FTP Server.** The FTP server usually contains different types of public files provided by the organization. These public files can be specific software, like a device driver or a client software. Depending on the organization, it can also be a community software sharing the organization resources to cooperate with the project.

5. **Mail Server.** The mail server is usually important to the organization, once it supports e-mail communications between the organization and other entities.
3.3. TARGET INFRASTRUCTURE MODEL

A common infrastructure can also have other important services to the organization. For example, authentication servers that will control the infrastructure accesses from clients.

Next it is presented a description of the relevant services running in the infrastructure servers, and a taxonomic representation of services information is also proposed. The taxonomy diagrams follows a hierarchical structure where each level consists in a refinement of the upper level. The logical "OR" symbol represents an exclusive OR which means that only one value of the bellow level can appear at a time. The remaining branches can be read as logical AND branches, which means that these fields are mandatory to appear in the log messages.

Taxonomies are a possible way to model and represent existent domain knowledge. Through taxonomies it is possible to make decisions about the information that will be mined, once each down lower level is a refinement of its higher level, it can be decided which is the desired detail level of a certain branch of the information tree.

If it is used too much detailed data, i. e., detailing the source address by defining the clients IP address instead of defining groups for local, national and international, would be more precise, but would also generate a higher amount of data as mining process result, rising the problem of treating the resultant information to be in a form that a human can easily evaluate and make decisions if needed. At the most detailed level patterns cannot be found because everything is unique.

If it is used few detailed data, the mining process is lighter because there is few data to process, but the results will be poor and almost useless because of the aggressive aggregation criteria. Results will reveal information that is somehow already known.

Mining requires generalization and it is necessary to achieve a compromise to get satisfactory results.

3.3.1 Gateway

The gateway does not constitute a service\(^4\) by itself once it does not provide relevant data to clients. However, the gateway, as one of the most important components in the network, records important data about its interaction on the network due its functions.

Fig. 3.5 represents a taxonomy for a Juniper\(^{[37]}\) gateway appliance. The information logged is composed by the time stamp field which records the event occurring time, the host name of the appliance, the session close data (flow session close), the service name, the applied policy name, and the connection (or NAT) information.

The time stamp consists of time (hour, minute and second), day and month when the event has occurred.

The host name consists in the network name of the (appliance) gateway.

The flow session close data consists of close reason, protocol, elapsed time of the connection and the data statistics information. The close reason can be a TCP FIN, TCP RESET, ICMP Response Received or unset. The protocol can be TCP, UDP or ICMP. The elapsed time contains the connection time, in seconds. The data contains a set of fields containing statistical information of the connection composed by packets from client, bytes from client, packets from server and bytes from server. Service name consists of a set of possible known service names as http of ftp, for example.

\(^4\)In this context, it is not being considered the concept of service in its pure sense of an application running, usually in background, but as an application available to the public.
CHAPTER 3. NETWORK USAGE EXPLORATION

Policy name consists of rule name, source interface and destination interface\(^5\). Juniper appliance usually has several interfaces and rules can be defined in a way to cover any space in the Cartesian product formed by the routes from source interfaces to destination interfaces. This way it is possible to record the traffic from a subset of interfaces as source and a subset of interfaces as destination. It is also possible to define rules to record traffic only in one direction but not in the opposite direction. For example, it possible to define traffic from interface 1 to interface 2 and 3, and in the opposite way only from the interface 2 to interface 1, leaving the interface free 3. Rule name represents the name of the filter that allows the session creation\(^6\). Source and destination interfaces represent the subnet connected to the interface.

NAT information gathers the source and destination address information (source and destination IP address and port). This field can also contain network address translation, if it exists. If no network address translation exists, the source and destination NAT rule will always have the same field name “none”\(^6\). In the mining process, fields with a constant value that appear in all occurrences have maximum support and at the same time does not add any relevant information, because the value is already known. In addition, it is also important the elimination of constant values to improve the mining algorithms performance, once the elimination avoids the necessity of generating useless candidate sets.

\[\text{Figure 3.5: GateWay Taxonomy}\]

### 3.3.2 Name Server

The name server usually has the \textit{BIND} service, or the \textit{djbdns} service.

---

\(^5\) The policy name field contains knowledge information about the network structure once network interfaces are used to separate the infrastructure in different segments, usually applying appropriated security and usage policies. It is common that the interface names reflect what the interface is connecting. For example, the users laboratory network can be named “labs” or the public services network name can be “pub”.

\(^6\) Rule name also reflects knowledge about the domain application. Different rules state for different logging criteria, and the names of the rules usually have a proper meaning to the applied domain.
3.3. TARGET INFRASTRUCTURE MODEL

Fig. 3.6 represents a taxonomy for a bind service. The information logged is composed by a time stamp field which records the event occurring time, the host name\(^7\), the service name and the query type.

The time stamp consists of time (hour, minute and second), day and month when the event has occurred.

The host name consists in the network name of the server in the network.

Queries are composed by client data, network domain view, recursive desired flag, resource record type and destination address do be resolved. The client data is composed by the client IP address and port. The view field states the client request origin. Internal-in means that the request arrives from within of the name server, and external-in means the opposite. Recursion desired field states if the client asks the server to perform the recursive queries it self or not. This field also contains other flags that define the name server behavior. Resource record defines the type of the action performed by the name server. For example, an “A” record represents a request for resolving a certain domain name to the corresponding IP address, while a “PTR” record represents the opposite, i. e., what is the domain name for a certain IP address. The destination states the destination address that client is asking to be resolved.

The EDNS disabled usually represents internal service information about the readjustment of the name server in communication with a certain server of an external domain. For example, adjusting the packet size to avoid some kind of packet loss.

The lame server field represents internal service data about name servers in other external domains that are discovered to be incorrect or fake. The composing fields state the errors that can be identified by the name server when it tries to communicate with a server found to be incorrect. It also contains the query address which is trying to be resolved.

The update security update states internal service information about clients trying to update its register in the name server. For security reasons, usually these kind of operations is forbidden to clients, but the clients update tentative it recorded. This field contains the client source IP and port, the view field (usually with internal-in value) and the network (which is the local network) and the denied message.

Note: In the mining process, the source IP address and the destination fields are gathered in three basic groups, local, national and international. This aggregation is necessary in order to avoid data scattering once the frequency of the items is one of the starting point values to the data processing. Grouping the addresses it will be increased the frequency of the items, and at the same time it is reduced the number of the items. This reduction is very important because it will generate and process item combinations. If the number of items is smaller, the combinatorial result will be also smaller and easy to treat.

It is also relevant to note that when a lame server record is found, it is usually originated by a client request. This means that when the name server tries to resolve some address and finds a lame server, most of the time the address to be resolved starts from a client request. It could make sense to make a connection between the client that originates the request, and the message informing that the name server for that domain is incorrect. It could be done, but it would have a preprocessing cost. One to handle these situations is to have a cache for the clients requests and when a lame server is found later (usually few seconds later) the respective record is associated with the clients original request through the service process

\(^7\)In a name service, due its availability requirements, it is common to have two or more servers replicating the service in case of a server fail, others can ensure the service availability, or simply to distribute the load. In the mining process, the existence of more than one host name will imply a split of the data even when the service is the same. To avoid this data split, is recommended to merge the records defining only one host name.
CHAPTER 3. NETWORK USAGE EXPLORATION

Figure 3.6: Name Server Taxonomy

identification (pid) and then eliminating the entry found. The cost of this operation is the storage in memory of all other client requests, and for each lame server entry search the clients request to found the corresponding entry that originates the lame server message. It is costly in terms of memory and processing overhead.

One way to avoid this necessity is to assume that when a lame server record is found, the record origin is the name server itself. There is a information lost once the relation between the client that has originated the request and the lame server is not established, but it is a mining process decision.

3.3.3 FTP

FTP server usually has a FTP service like the pure-ftpd or vsftpd to serve public domain files. It is also common to have these public domain files available via HTTP using a web server, for example, the Apache web server. In the organizations where collaborative open source projects exists, it is also common to have a rsync service to synchronize file directory trees among other project mirror servers.

Fig. 3.7 represents a taxonomy for a pure-ftpd, Apache and rsync services, divided in a branch for each service.

FTP server records are divided in three major groups, one group for each service. The FTP group contains the pure-ftpd service information, the Logger group contains the Apache web service information and the rsync service.

The FTP service is composed by the time stamp, the host name, the service name, the event type and the client identification. The time stamp consists of time (hour, minute and second), day and month when the event has occurred. The host name consists in the network name of the server in the network. The service name consists in the service running. The event type can be one of the three types, information, error and notice. Information field states information about the beginning and the end of clients data, for example, a new client login or logout. The error field states internal data about some client actions attempt that
3.3. TARGET INFRASTRUCTURE MODEL

generates an error message, for example, trying to access to an inexistent file. The notice field states statistical data about the available resources accessed by clients, for example, which file was downloaded with the respective size and connection speed.

Note: In the notice field, it is visible the presence of domain knowledge in the downloaded field. The values present in this field reflect the present directory tree in a public FTP mirror of some well known open source projects like Eclipse IDE platform or Ubuntu linux distribution.

In the mining process, if the entire directory is processed considering the file full path, it will originate a large data set with low support in each considering item. This fact will force the need to lower the minimum support of the items, which will increase the combinatorial explosion in candidate sequences generation. To avoid this problem, an adjustment can be made in order to reduce the number of disperse files path and increase the support of the considering items. Instead of considering each particular file path, it should be considered a portion of the file path, for example, the higher level directory name. This way to aggregate the data follows a natural grouping criteria that also respects the concentration of the data regarding to the most accessed resources by clients. For example, if the Ubuntu linux resources have more client accesses that the Eclipse IDE platform, even with the loss of information concerning the particular accessed file, if only the higher directory name is recorded, it will preserve the original access proportions.

A technical aspect should also be considered. In certain situations, like an open source project under constant development, it is common to have file changes in the directory tree due the presence of new releases or simply bug fixes some times a day. For example, it is common to have rsync processes running every four hours period. Depending on the time period applied in the mining process, it can be a relevant aspect, because the larger time period is considered, the higher is the amount of files changed. This file changing will add an artificial data partition once it is possible a file to be replaced by a newer version which implies the elimination of the older one from the directory tree and the addition of the new one, probably with different file name. Considering the higher directory name, this problem is eliminated but the access properties are preserved. If the particular resources are important to the end user of the mining process, it can be used a subdirectory name referring the resource needed to know.

The Apache service is composed by the time stamp, the host name, the client address, the request, the HTTP error code and the resource size accessed by the client.

The time stamp consists of time (hour, minute and second), day and month when the event has occurred. The host name consists in the network name of the server in the network. The source IP is the client’s address.

The request consists in three fields. The type field represents the protocol request type. The resource field is the file accessed by the client. The HTTP version is an identification tag of the protocol version.

The error code does not mean that an error has occurred, but instead is the HTTP code

---

8In public FTP servers, it is common to have the same directory tree served via FTP protocol as well as HTTP, in order to allow clients to access files, even when a client does not have an easy FTP client to use. Considering these aspects it can be assumed that the directory tree served via HTTP is similar to the directory tree served via FTP. In the mining process, the directory tree in HTTP can be processed in a similar way to the directory tree in FTP.
value returned to client when a resource is requested.

The response size is the number of bytes sent to client in the message response.

The rsync service is composed by the time stamp, the host name, the service name and the event identification.

The time stamp consists of time (hour, minute and second), day and month when the event has occurred. The host name consists in the network name of the server in the network. The service name consists in the rsync service running.

The event identification is composed by the service process identification, the client address and the event type.

The event type consists in one of the following messages: Connect, rsync, send/receive, build index and error. The connect message records connections from clients identified by the client’s IP address. The rsync message records rsync requests on a certain directory from external mirror servers (or clients). The send/receive message records statistical information about the rsync operation. The build file list is an internal information message that states a index information was built. The error messages is an internal information that records detected errors in the clients interaction, like syntax errors, for example.

3.3.4 Web Server

The web server usually has the Apache service, most of the time supported by other types of software, for example, an organizational intranet system or a CMS. It can also be a discussion forum like the Vbulletin or a web mail interface like the Horde.

In order to reduce the dimensions of the web server information diagram, this diagram is divided in two parts, one part concerns to the web server activity without SSL support, and the other part concerns to the activity with SSL support. Fig. 3.8 represents a taxonomy for an Apache web service serving its contents without SSL and Fig. 3.8 represents a taxonomy
3.3. TARGET INFRASTRUCTURE MODEL

for an *Apache* web service serving its contents with SSL.

Web server records are divided in two major groups, the access group that records the access activity of the clients requests, and the error group that records accesses that does not conclude successful and other occasional server error. When an *Apache* service is configured to also serve its contents via a SSL connection, these two groups appear also in the SSL records. That means that a separation is made, one log file records accesses without SSL and a separated log file records accesses with SSL. Same applies to error events.

The access log group records accesses without SSL support and is composed by the time stamp, the client source address, the response error code\(^9\), the resource requested by the client and the response size.

The time stamp consists of time (hour, minute and second), day and month when the event has occurred. The source address consists in the source client address. The error code consists in the HTTP returning message type. The request consists in three fields, the request type, the resource\(^10\) requested by the client and the HTTP protocol version used in the communication with the client. The response size states the size of the response sent by the server to the client.

The error log group records error events\(^11\) and can be divided in two major groups. Invalid URI messages and internal server errors.

Invalid URI messages consists of time (hour, minute and second), day, weekday and month when the event has occurred. The source IP is the client location. The error message consists of the internal descriptive message generated by the clients incorrect action. The HTTP version states which HTTP version is used to communicate with the client.

Note: The resources field should be treated in the same way as in the FTP, considering only the first part of the directory path in order to avoid a high level of dispersion in data with low support which implies less combinations in sequence items.

In some cases it is convenient to ensure that the contents are served via SSL. This can be achieved by configuring the web server to redirect every request to HTTPS instead of HTTP. In such situations the HTTP log file only will state the 301 error code\(^13\) messages. If the HTTPS is optional in the organization, then the HTTP and HTTPS logs should present a similar spectrum of messages.

Resources provided via SSL are divided in SSL access log and SSL error log.

The SSL access log group records accesses with SSL support and is composed by the time stamp, the client source address, the response error code, the resource requested by the client and the response size.

The time stamp consists of time (hour, minute and second), day and month when the event has occurred. The source address consists in the source client address. The error code\(^9\) Nevertheless the word “error”, it does not mean that an error, in the wide sense, have in fact occurred. The “error” term derives from the *Apache* service nomenclature that classifies the results in categories according to HTTP specification. In the Fig. 3.8, the error code 301 states for “Moved Permanently” returned to the client.

\(^{10}\)The resources provided via HTTP can be treated in a similar way as the resources provided via FTP.

\(^{11}\)The error log file, contrarily to “error code”, records errors generated by client requests, for example, clients requests for nonexistent files, or application error.

\(^{12}\)URI is acronym to Uniform Resource Identifier.

\(^{13}\)According to HTTP specification[45].
CHAPTER 3. NETWORK USAGE EXPLORATION

Figure 3.8: Web Taxonomy

consists in the HTTP returning message type\textsuperscript{14}. The request consists in three fields, the request type, the resource requested by the client and the HTTP protocol version used in the communication with the client. The response size states the size of the response\textsuperscript{15} sent by the server to the client.

The SSL error log group records error events and can be divided in two major groups. Invalid URI messages and internal server errors.

Invalid URI messages consists of time (hour, minute and second), day, weekday and month when the event has occurred. The source IP is the client location. The error message consists of the internal descriptive message generated by the clients incorrect action. The HTTP version states which HTTP version is used to communicate with the client\textsuperscript{16}.

Observation: Some organizations concerns about the web communications security and define policies that enforces HTTP to be served via SSL. One way to enforce such policy is to configure the web server with a permanent redirect from HTTP to secure HTTP via an SSL connection for all available resources. This redirect works as follows: when a client connects to the server without SSL the server responds with a 301 error code message to client informing that the resource was permanently moved to the same location but via SSL. The client receives the message and keeping its intention to access the desired resource starts a new connection, this time via SSL, following the returned address from the server.

The mining process can take advantage of this domain knowledge and, for example, avoid to process access log file in conjunction with the access log file via SSL.

Applying such criteria using domain knowledge can greatly improve the mining process in several aspects. First of all it will not be tried to present information that is already known as result. Besides this aspect, processing access log file (without SSL) would return frequent 301 error code events, which would be one more frequent element to be combined with the other frequent events found increasing significantly the number of combinations to be processed. Combining this processing with access log file via SSL would return an high correlation between 200 error code of an event and an immediately earlier 301 for the same event, suggesting this way that there exists a frequent usage pattern consisting in clients contacting secured

\textsuperscript{14} Error code with value 200 means “OK”.
\textsuperscript{15} This field does not state the size of the resource required by the client.
\textsuperscript{16} It is possible to be found HTTP 1.0 requests but these are no frequent neither relevant from the mining process.
HTTP resources after contacting HTTP (without SSL), adding non relevant information.

### 3.4. Data Preparation

Regarding the whole process, it can be divided in four major steps. First it is necessary to execute a preliminary parse to apply an initial filter and extract desired filters that will help posterior steps. Second follows an integration according the temporal organization. Third is the extraction of the sequences of baskets (or itemsets). Fourth is the application of sequential pattern mining algorithms. Fig. 3.10 illustrates the schema of the process.

![Preprocessing schema](image)

**Figure 3.10: Preprocessing schema**

#### 3.4.1 Parsing Sources

The preliminary parse has two major goals.

The first goal is to make a preliminary extraction of important fields used in the following process steps. These steps are the introduction of a format containing the time stamp of
the event standardized to all log source files, the source address of the client, and the construction of a preliminary basket (described in Sec. 3.4.2). This preliminary step is important once it will turn the integration step easier.

The second goal is the application of some defined criteria, for example, the application of some discretization rules labeling the clients source address as “local”, “national” of “international” as in Sec. 3.2.2

The number of groups can vary according to the application and the goals, for example, the “local” label can be divided in a label with the subnet, “labs”, “pub”…

This step is also important to make data cleaning, eliminating log entries that are not relevant for the process. For example, if logs contain service restarting information, and if it is considered to be irrelevant according to the mining goals, the preliminary parse will drop these entries avoiding that information to propagate in the remaining process.

3.4.2 Basket Construction

Basket construction consists in grouping the important components of log file entries. Once a log entry is a set of elements that occur at the same instant, it is considered a basket (or an itemset). Sequences are build by grouping sequences of baskets over time, identified by the clients source address.

The discovery of sequential patterns in log files can take advantage of some intrinsic properties. Sec. 3.3 presents a representation of some services. These taxonomic representations highlights the existence of different message types defined by the branches.

For example, considering the gateway taxonomy (Fig. 3.5) it can be identified only one type of messages, once the “event” node is not a conjunction of different branches. Semantically it is also correct because of the data that a common gate way is supposed to record, that is, the source and the destination addresses of a connection, and information about the service and protocol used in the connection.

Gateways can have more capabilities than the example turning the log messages more complex. In such situations there will exist a corresponding number of brunches, each brunch corresponding to each different message type.

A name service naturally has several message types, once its information is more complex than a regular gateway. Following the example in Sec. 3.3, it is possible to identify different query types a name service (Fig. 3.6).

A technical note can be made. While the log files have a proper syntax, it could be used a regular parser to process log files to extract the necessary label values, but once the log files does not have a syntax as complex as a normal programming language, a simple method like the application of regular expressions[15] can be used instead. The only thing that is necessary is the implementation of a tokenized mechanism that returns the required tokens. For example, log files syntax does not have recursion. A regular programming language usually has.
3.4. DATA PREPARATION

3.4.2.1 Baskets from Log Messages

1. Gateway.

Fig. 3.11 is an example of a log message in a Juniper gateway appliance. Its structure is described in Sec. 3.3.1. The basket construction for this log messages type is made considering the relevant fields of the log message and attach the field values with a label making pairs of (label, value), applying the defined criteria regarding to the detail level the mining process should comply.

Mar 10 00:00:01 gw RT_FLOW[] RT_FLOW_SESSION_CLOSE: session closed unset: aaa.bbb.ccc.ddd/ii->xxx.yyy.zzz.www/jj junos-dns-udp
aaa.bbb.ccc.ddd/ii->xxx.yyy.zzz.www/jj None None 17 rule-name source-interface destination-interface 25995 1(73) 1(327) 0

Figure 3.11: Gateway log sample

Relevant field are, the time stamp which will allow to sort the events in a time line in the integration phase, the message type which is a “session close” message, the reason why the session was closed (if known), the source IP address and port, the destination IP address and port, the rule name, the source interface and the destination interface.

A possible representation can be: “1299715201 aaa.bbb.ccc.ddd (gw_reason=unset, gw_src_ip=international, gw_src_port=ii, alpha_dst_ip=local, gw_dst_port=jj, gw_service_type=junos-dns-udp, gw_protocol=udp, gw_rule_name=rule-name, gw_src_if=source-interface, gw_dst_if=destination-interface)”.

The first number is the unix time\(^\text{17}\) representation of the time stamp. It is possible to use another time representation, but in order to sort events overtime, integer values provide a natural order.

Each label has a prefix “gw” concerning to the host name. Adding such prefix makes the label unique to a certain server, and this way it is possible to distinguish “src_ip” label in the gateway from the “src_ip” of the name server, for example. The addition of the host name in each label is only a way to turn the labels unique. It is possible to use other ways to hit the same goal, that means, to ensure the label uniqueness. If the labels are not unique, there is a loss of information in the process.

This parse is also a good place to apply some translation criteria such the definition of the local, national and international groups for the addresses. There are other refinements that can be made. For example, it can be defined some other values for the address labels, adding suitable information appropriated to the application domain, like a value for each relevant server. Adding new values, the address labels will have, not only the local, national or international values, but also, NAMESERVER, MAILSERVER, WEBSERVER...

Nevertheless, it is important to notice that increasing the number of values in a label will increase the number of alphabet elements and consequently the number of combinations. In a regular infrastructure, the number of important servers is often limited, so a refinement of this nature will not cause a combinatorial explosion. If the domain

\(^{17}\)Unix time or POSIX time considers the number of seconds since epoch, is the number of seconds since midnight of January 1 of 1970.
application demands a greater number of values, it should be found a trade off between
the number of possible generated combinations and the knowledge gained with the add-
dition of the new values.

Not all fields of a log entry must be included in the basket. For example the NAT\textsuperscript{18} fields
with the value “None” are not included. It is because in a network without NAT these
fields always have “None” as value. Including such labels containing always the same
value would increase the number of combinations to process in the algorithms compu-
tation, increasing processing overhead without an addition of relevant information as a
result.

This issue can be evaluated semantically by knowing that the network configuration
does not have NAT rules defined. Considering that, any NAT information does not
bring interest to the process and can be ignored.

It is also possible to evaluate this subject statistically by analyzing a representative
sample of the data and observing that if some labels referring to NAT rules do exist, it
will be observed that these labels will have always the same value with a 100\% support.
A label with a maximum support will always be frequent in the mining process and will
always generate candidates without useful information.

Considering these aspects, it is important to not include labels that will not add useful
information, and also avoid to add unnecessary complexity to the process.

The remaining fields (bytes and packets sent or received) can be discarded using Sec.
3.2.2 and can be processed according to it.

2. Name Server.

A typical name service, like \textit{Bind}, has five message types, as can be seen in Fig. 3.6.
These are resolve queries, reverse resolve queries, edns disabled\textsuperscript{19} notifications , lame
servers notifications and update security messages.

Resolve queries occur when a computer needs to know the IP address of another com-
puter identified by a name (DNS name), than the name server will be asked to translate
the identification name for its corresponding IP address.

Fig. 3.12 presents an example of a resolve query log entry.

Relevant fields are, the time stamp, the entry type, the source address, the destination
address, the resource record type (“IN A” means a resolve request), the recursive desired

\textsuperscript{18} Acronym for network address translation.

\textsuperscript{19} edns-disabled Log queries that have been forced to use plain DNS due to timeouts. This is often due to the
remote servers not being RFC 1034 compliant (not always returning FORMERR or similar to EDNS queries
and other extensions to the DNS when they are not understood). In other words, this is targeted at servers
that fail to respond to DNS queries that they dont understand.

Note: the log message can also be due to packet loss. Before reporting servers for non-RFC 1034 compliance
they should be re-tested to determine the nature of the non-compliance. This testing should prevent or reduce
the number of false-positive reports.

Note: eventually named will have to stop treating such timeouts as due to RFC 1034 non compliance and
start treating it as plain packet loss. Falsely classifying packet loss as due to RFC 1034 non compliance impacts
on DNSSEC validation which requires EDNS for the DNSSEC records to be returned.[7]
3.4. DATA PREPARATION

Mar 10 00:00:00 ns named[3005] 10-Mar-2011 00:00:00.193 queries: info: client aaa.bbb.ccc.ddd#ii: view internal-in: query: www.example.com IN A + (mmm.nnn.ppp.qqq)

Figure 3.12: Name Server queries log sample

flags and the view. The log entry ends with the name server IP address. Once this information is known in the first place, there is no need of inclusion in the process because it is redundant. One final note, the number that follows the service name (“named”) is the process identifier number, that is a wide range data type, and can be treated as discussed in Sec. 3.2.2.

A possible representation can be: “1299715200 aaa.bbb.ccc.ddd (ns info=query, ns src ip=MAILSERVER, view=internal-in, ns dst addr=international, ns resource record=A, ns recursive desired=+)”.

As described in the gateway baskets construction, here the same principles should be applied. The labels should be unique, and its values should be coherent with the construction of the other services. In a network environment it is important to keep the consistence of chosen values to the address labels between mined servers.

This is crucial to achieve the relations establishment in clients interactions. For example, to be able to relate the destination address in the gateway to the corresponding DNS name resolve query for a certain server, like the FTP server. This way it is being added dimension to the process avoiding a possible excessive flattening of the data, if only one value exists to characterize the local infrastructure.

In complement to the DNS resolve queries there are the reverse queries. If the resolve queries aim to know which is the corresponding IP address of a certain DNS name (for example, “http://www.example.com”), the resolve queries aim to know the DNS names, if they exists, corresponding to certain IP address.

Fig. 3.13 shows an example of a reverse resolve log query.

Mar 10 00:00:00 ns named[3005] 10-Mar-2011 00:00:00.192 queries: info: client aaa.bbb.ccc.ddd#ii: view internal-in: query: www.zzz.yyy.xxx.in-addr.arpa IN PTR + (mmm.nnn.ppp.qqq)

Figure 3.13: Name Server reverse queries log sample

Relevant fields are, the time stamp, the entry type, the source address, the destination address, the resource record type (“IN PTR” means a resolve request), the recursive desired flags and the view.

A possible representation can be: “1299715200 aaa.bbb.ccc.ddd (ns info=query, ns src ip=MAILSERVER, view=internal-in, ns dst addr=international, ns resource record=PTR, ns recursive desired=+)”.

EDNS disabled log messages occur when the name server have been forced to use plain DNS due to a timeout. Fig. 3.14 shows an example of a edns disabled log message.

20 The “view” field records where the request come from, that means, if the client belongs to the same domain as the name server or not.
CHAPTER 3. NETWORK USAGE EXPLORATION

Figure 3.14: Name Server edns disabled log sample

Relevant fields are, the time stamp, the entry type, the destination address, and the resource record.

A possible representation can be:

```
"1299715208 aaa.bbb.ccc.ddd (ns_info=edns_reducing_pkt_size, ns_dst_addrs=international, ns_resource_record=AAAA)"
```

Lame server messages can occur in three different forms, but essentially they share the same principle, and are caused by misconfigurations in remote servers, discovered by BIND service when trying to query these servers during resolution[7]. Fig. 3.15 shows an example of lame servers log messages.

Figure 3.15: Name Server lame servers log sample

Relevant fields are, the time stamp, the entry type, the source address, the destination address, the resource record, if it is an error message, and the lame server address.

A possible representation for each message can be:

```
"1299715208 aaa.bbb.ccc.ddd (ns_error=servfail, ns_src_ip=NAMESERVER, ns_dst_addrs=international, ns_resource_record=A, ns_lame_srv_ip=xxx.yyy.zzz.www)"
```

for the first message,

```
"1299715209 aaa.bbb.ccc.ddd (ns_error=connection_refused, ns_src_ip=NAMESERVER, ns_dst_addrs=international, ns_resource_record=A, ns_lame_srv_ip=xxx.yyy.zzz.www)"
```

for the second message, and

```
"1299715222 aaa.bbb.ccc.ddd (ns_info=resolution, ns_src_ip=NAMESERVER, ns_dst_addrs=international, ns_lame_srv_ip=xxx.yyy.zzz.www)"
```

for the third message, respectively.

Update Security messages occur when a client tries to send a DNS update to the name server. For security reasons the access to the dynamic update facility should be strictly limited. Fig. 3.16 shows an example of an update security log message.

Figure 3.16: Name Server update security log sample

A possible representation can be:

```
"1299715235 aaa.bbb.ccc.ddd (ns_info=update_security, ns_src_ip=local)"
```
3.4. DATA PREPARATION

According to BIND 9 official documentation, in earlier versions of BIND, the only way to do this was based on the IP address of the host requesting the update, by listing an IP address or network prefix in the allow-update zone option. This method is insecure since the source address of the update UDP packet is easily forged. Also note that if the IP addresses allowed by the allow-update option include the address of a slave server which performs forwarding of dynamic updates, the master can be trivially attacked by sending the update to the slave, which will forward it to the master with its own source IP address causing the master to approve it without question.

For these reasons, BIND 9 documentation strongly recommend that updates be cryptographically authenticated by means of transaction signatures (TSIG). That is, the allow-update option should list only TSIG key names, not IP addresses or network prefixes. Alternatively, the new update-policy option can be used.

Some sites choose to keep all dynamically-updated DNS data in a subdomain and delegate that subdomain to a separate zone. This way, the top-level zone containing critical data such as the IP addresses of public web and mail servers need not allow dynamic update at all[7].

Other service logs can be parsed in same way that gateway logs and name server does. In resume, after defining the log file information structure (similar to a taxonomy), apply some necessary filters to discard noisy log entries, and labeling relevant information according defined principles. Than, constructed baskets can be outputted in a way that become easy to sort in time. Sorting in a time line is presented in Sec. 3.4.2.5.

3.4.2.2 Labeling Clients Address

Typically gateways logs identify source and destination addresses by its network IP address, whether the address is a local address or not. Local IP address are domain knowledge and can be classified without doubt, that means that in such process it is possible to identify where the IP address belongs too.

But in the Internet, there exist a subtle problem. It is possible to know what IP address ranges are assigned to a country, because there are responsible entities to certain globe regions IP address assignment, and also institutional entities like universities are known. IP address assignment changes are a process slow enough to be considered a problem, so this work does not consider the fact of an IP address range being delegated from a country to another a serious problem.

The problem exists when considering private enterprises, because when a private corporation becomes bigger it tends to have delegations in several countries bringing some difficulties and uncertainties to its classification regarding to its real country. A consequent problem is the delegation problem that can exist when an entity sells an IP range to another entity, or simply abandons an IP address, for example, if it extinguishes a certain entity delegation.

This problem can be seen as a world intrinsic problem, and not a problem of the data mining applications.
3.4.2.3 Address Origins Classification

As referred in Sec. 3.4.2.2, a typical gateway identifies its addresses by its network IP addresses, but a name server deals with other address formats beyond the IP addresses only. It deals with DNS names, so it is common to have to deal with a mixture of both.

Considering the problem of source IP address classification, in a name server is not limited to IP addresses. It also comprises the DNS names once countries have its own DNS suffix, for example, “.pt is delegated to Portugal”, but there are other name suffixes that are not delegated to a specific exclusive entity. For example, different countries can acquire a “.com” address.

Once again, there are some addressees that are difficult to classify, because of the words behavior, not by data mining process itself.

At this point, to find a reasonable compromise to solve such situation, some possibilities can be addressed.

According to the defined criteria to the process, labeling of IP addresses and DNS names can have different detail levels. In this work there are three suggested labels, “local”\(^{21}\), “national” and “international”, but for some reason it could be important to detail these labels, or some. For example, “local” label can be divided in the number of subnets in the infrastructure or even include important servers. By the same reason, the “international” label can be divided by continents or by countries inside some continents, for example.

In the case of the “local” label, there is no problem with it because it is part of domain knowledge once the server names are well known and addressed. Same can be said about subnetworks in the infrastructure.

The problem rises in the “international” label, as mentioned above. The problem can not be completely solved because of its nature, but can be minimized. There are entities that makes the effort\(^{22}\) to have lists of IP addresses known to belong to each country. It is reasonable to consider that, if an IP address is not in the country list, it is an international IP address. Of course, it is possible that an IP address does not exist in the list, but in fact belongs to the country.

In some way, it is reasonable to consider other DNS suffixes than “.pt” as “international”. For example, “www.example.com” is not a “.pt” suffix so it is considered “international”.

There is a subtle aspect that can improve the labeling in some situations. Considering that a local Internet service provider (ISP) is known to belong to the countries IP address ranges, if DNS names are translated to the corresponding IP address, a national company that have bought a “.com” address, is correctly classified as “national”, even with a DNS name that could be classified as “international” in first place.

But this translation can carry a serious problem, depending on the implementation. If this

\(^{21}\) The “local” label is conveniently divided in the case of the address is a service to be mined, like the name server, or the FTP server.

\(^{22}\) Effort because some ranges change over the world, so it is necessary to regularly update the IP address lists. It is not a process that once concluded, remains unchanged over time. It is necessary to periodically update the delegation lists.
process is implemented in an infrastructure and run every night, trying to discover the frequent sequential patterns from the previous day, it would parse several log entries and would address an amount of DNS reverse queries to the name server. These queries would be logged by the name server and parsed in the next day. If no action is taken, it would introduce an unnecessary exponential load to the name server and the infrastructure.

To overcome this possible implementation problem, there are at last two possible solutions. One solution can be the application of a filter in the preliminary parsing phase eliminating all DNS queries that are originated by the computer running the sequential pattern mining process. This way, it is only added entries corresponding to the previous day, avoiding an exponential growth from the process daily feedback. Other solution can be the implementation of slave name server that only serves the process, eliminating the interference in the infrastructures name servers.

The gateway can also have corresponding log entries to these reverse queries. These entries should also be filtered to avoid the process to mines itself introducing undesired errors.

There is one last implementation aspect in address classification. Clients request to name server the resolution of a certain address, for example, “what is the IP address of www.example.com”, and in some situations the name server concludes that the server supposed to be the authoritative name server for that domain in reality does not behavior as the name server would, classifying is as a “lame server”. The name server logs these events, the clients resolve request, and posteriorly the “lame server” detection.

The problem is the presence of two related messages, having a direct correspondence between the client that requests the address that originates the “lame server” message and the “lame server” detection in the name server. To properly solve this problem it is necessary to keep in memory the log entries and for each “lame server” detection, find the correspondent client request. Depending the infrastructure amount of traffic, this task can be very resource consuming whether in terms of memory or in terms of time.

A possible trade off in the implementation process is to define the name server as the origin of the “lame server” messages. This way it is avoided to keep almost the log file in memory and also, it is avoided to search the “lame server” origin in all previous read messages.

### 3.4.2.4 Labels Classification Criteria

Source addresses are important mainly in two aspects.

Source addresses are important in the application domain because it can provide a good source of knowledge, for example, discovering that a substantial amount of clients requests to a specific service of the organizations web site, like “products” or “orders” are originated from spanish spoken countries, it can originate the creation of a spanish section in the organizations web site to improve its communication with spanish spoken users.

Source addresses are also important in the security evaluation because if an incident is detected, the sources are one of the first issues to be analyzed. Isolated incidents, by its nature, does not constitute a frequent item and consequently have no expression using this method. However, there are frequent events generated by automated tasks that are worthy of attention. For example, certain types of network scans with services version guess trying to identify any exploitable vulnerability.

These two aspects motivates the discussion about the reasonable detail level to the domain application, and what are the decisions consequences.
It is desirable to have the highest detail level that it is possible to achieve. A higher detail level can lead to an huge number of output entries. That means that if the decision is to classify source addresses by country, it is possible to have a large number of sources, which will originate a large output. A possible way to reduce such amount of possible sources is to classify source addresses by continent, making groups of countries. If it is necessary continue to reduce the number of different source addresses, it can be considered only the difference between national and international sources, for example.

It is important to note that this proposed division follows a semantic hierarchy of locations, but there is no conceptual limitations forbidding different classifications, for example, classifying source addresses by continent and adding relevance to specific countries that reveals to be important to the process, like spanish spoken countries. This way the classification can consist in the group of the five continents plus the group of the spanish spoken countries. This number of source addresses is bigger than only the continents but less smaller than the group of all countries. Same applies if the classification is the country sources and a certain organization subnetwork.

The detail level of the labels is affected by the data itself and should not be regarded as a matter of definition only. If the detail level is too high specific, for example, if the sources are labeled by its country, depending on the data, it is possible to have an insufficient frequent items, leading to very poor results.

In Sequential patterns discovery in network clients interaction, it is important to consider the time period that the process will be applied.

If the time period is too short, for example, half of one hour, it is possible to have a non representative sample of the clients activity in the network, depending on the amount of network traffic.

On the other hand, if the time period is too long, for example, three months it is possible to have an overlapping of time periods, losing information. Depending on the network available services, it is possible in most cases to identify specific time periods with particular network usage. For example, the nocturnal period can have less activity than the daily period. Analogously, holidays periods can reveal less activity than school term.

There are organizations with online services available to the world, like Amazon, with an huge amount of network requests. In such cases, time periods should be adjusted to return useful results.

### 3.4.2.5 Time Line Sorting

Time line sorting is an operation consisting in the integration of the log files to be mined in one single file containing the events sorted according to temporal principles. That means that it is made a temporal merge of recorded events, simplifying the process of basket sequences built.

This is not an indispensable step, once it is possible to implement the log files integration while the basket sequences are built, but integrating files in a earlier step, it significantly simplifies the basket sequences building step.

Time line sorting respects some necessary conditions. First it respects the inherent order of the events in each log file. Second it respects the order of log files processing. For example, if log files order is “log file 1”, “log file 2”, . . . , “log file N”, then for the same instant events,
“log file 1” events come before than “log file 2” events, and so on. And last, “log file 2” events of a next instant are processed if “log file 1” events belongs to a posterior time instant.

For example, if “log file 1” has an event in $t_0$ and its next event occurs in $t_3$, and “log file 2” has events in $t_0$, $t_1$, and $t_2$, than the result will be: first the “log file 1” $t_0$ event, than the “log file 2” $t_0$ event. Since the “log file 1” does not have an event in $t_1$, the next entry will be the “log file 2” $t_1$ event. Same applies to “log file 2” $t_2$ event. Finally it is processed the last entry of “log file 1”. Since the log files have no more events, the integration process reaches the end. Fig. 3.17 presents an example showing how time line sorting can be made.

![Figure 3.17: Time Line Sorting example](image)

It is important to make some considerations about the events time recording. In a single system, it is reasonable to assume that sequential events will be recorded in the same order that events occurred. Even in a system with multi-core processors or multiple processors, typically one entry is written to a log file at a time. For this reason, it can be considered that in a single system, events occur in sequence over time.

On the other hand, in a distributed environment some of these principles does not apply, because some conditions are naturally relaxed. In a single system there is only one clock. In a distributed system each component has its own clock, which means that it can exist little discrepancies between a pair of components. To minimize the time differences between components, it is used a time server to help to synchronize clocks among components.

In addition, it is possible the existence of network delays. These facts can cause the accurate definition of the events sequence over time virtually impossible, unless the implementation can count with some kind of extra mechanism, for example, vector clocks[16]. Usually, systems does not have that sophisticated methods because it implies that all systems comply with the same rules, and clients, for example, are autonomous so it is not possible make them use such methods.

Proceeding in environments with these limitations, and without extra mechanisms it only can be made a best effort. That means that it can be assumed the log entry sequences as the occurred sequences in each system, and it can also be assumed that clocks are synchronized so there is no significant diversion between events in different components.

It is important to remember that once there is no control over all involved components, the “real” sequence of events can differ from the recorded sequence along the components, so this work only considers the recorded data and does not aims to go further.

---

23In a normal situation the discrepancy does not exceed some fractions of a second, once each component synchronizes its internal clock with the time server several times a day.
3.4.3 Reduction Process

Reduction is an important step in the process improvement. The objective is to reduce the data set used in the process, consequently reducing the number of generated combinations in sequential pattern discovery and, of course, the memory consumption and the time needed to process the data. This efficiency helps to generate simplified results without loss of knowledge, and gives rise to new data incorporation once the process becomes lighter.

Reduction consists in reducing, whenever is possible, the number of items (or labels) in baskets construction. Reducing the number of items in some baskets will naturally reduce the number of generated combinations in the mining phase, and can be made in two stages, semantically and statistically.

Semantic reduction can be achieved by analyzing the information representation in taxonomy diagrams (see Sec. 3.3). As an example, it can be considered the name server taxonomy, Fig. 3.5, “queries” brunch has the “view” field with two possible values, “internal-in” and “external-in”, defined according to the source request was originated from the local network or not. In the same brunch is represented the “source IP” field in “client information” defined according to the source request address, as the “view” field, but more specific. So the “view” field is contained in “source IP” field originating this way a duplication of information. These two fields are redundant because they contain the same information, and these two fields can be joint together in one single field.

Grouping fields is implemented in the baskets construction, Sec. 3.4.2.1, modifying the baskets to have a concatenation of the two redundant fields.

For example, the name server query log message in Fig. 3.12 would be represented as “1299715200 aaa.bbb.ccc.ddd (ns_info=query,ns_src_ip=MAILSERVER, view=internal-in_AND_ns_dst_addr=international, ns_resource_record=A, ns_recursive_desired=+)”.

The “AND” string can be used to easily identify a label concatenation, but the labels can be simply concatenated.

Statistic reduction can be achieved by analyzing the frequency relation between the occurrences of pairs of labels, trying to find strong associations in pairs of labels. The high frequency pair candidates must be ultimately semantically analyzed to ensure that a reduction can be made, and the pair is not originated only by the nature of the data.

Association rules emerge as a natural method to discover associations between pairs of items. Depending on the application domain, typically association rules methods uses relatively low values for support and confidence. Support defines the coverage of the method. The lower the support is, more items are considered frequent, so more items are included in the rules generation process. Confidence defines how strong the rule is. The higher the confidence is, more occurrences of the consequent exist where the antecedent is verified. In common association rules applications, higher confidence values tend to reveal information that is already known. Lower confidence values tend to carry low information significance.

To avoid information loss redundant fields are join together. However, in this particular situation one of these fields can be simply eliminated because one field is contained in the other field. “source IP” field is more specific that “view” so it reasonable to keep “source IP” field and simply drop “view”, without loss of information. Note that this is is a particular case.
This application domain uses a different parameters configuration in association rules. To evaluate the existence of redundant items it is necessary to apply an algorithm for association rules, for example Apriori\(^25\)[2], with a low support value to have a convenient coverage, and high confidence to get strong associations.

The specific values are always dependent of the nature of the available data to be processed. Reasonable values can be, for example, 5% for support and 95% for confidence.

After the association rules algorithm execution, the output result is analyzed in search of rules with the highest confidence, starting with 100% and looking for rules pairs where \( A \Rightarrow B \) and \( B \Rightarrow A \), stating that a \( A \Leftrightarrow B \) implication exists, so the involved labels have tendency to be redundant.

Evaluation of the rule pairs requires attention and domain knowledge to separate the rules that represent a redundancy, and rules that were originated only by the nature of the data.

To illustrate the difference between the two kind of pairs it is presented an example. Considering the gateway taxonomy, Fig. 3.5, a rule pair stating \((gw\_service\_name = junos\_dns\_udp) \Rightarrow (gw\_rule\_name = NS)\) with 100% confidence and in complement \((gw\_rule\_name = NS) \Rightarrow (gw\_service\_name = junos - dns - udp)\) with the same confidence value points a redundancy between the “service name” field and the “rule name” field, because the network configuration is domain knowledge, it is known that only the name server is a target of a significant number of DNS requests.

On the other hand, it is possible to have \((gw\_rule\_name = allow - ping) \Rightarrow (gw\_dst\_if = pub)\) with 100% confidence and \((gw\_dst\_if = pub) \Rightarrow (gw\_rule\_name = allow - ping)\) with the same confidence value for a certain period of time. Like the previous described situation, it would point to a redundancy of these two labels, but with domain knowledge it can be concluded that this rules pair depends of the data formation, because in the analyzed time period, only international clients contacted the public servers subnet through ICMP protocol. If other time period would be chosen, it could exist both local and international clients requests, for example.

It is possible to have more elaborate relations that are not represented by a simple pair, instead, relations can be pointed by a conjunction of implications containing difference confidence values forming a set to have a wider implication. For example, considering the gateway taxonomy, Fig. 3.5, it is possible to have a set of rules as follows:
\(\text{(gw\_src\_ip = international, gw\_service\_name = junos - dns - udp) } \Rightarrow (gw\_dst\_if = pub)\)
and \(\text{(gw\_src\_ip = local, gw\_service\_name = junos - dns - udp) } \Rightarrow (gw\_dst\_if = pub)\), both rules with high confidence. This can be explained by domain knowledge, once it is known that the name server lies on the public network, connected through the gateway public network interface.

Name server responds to different origin requests, so the corresponding rules pair with the form:
\(\text{(gw\_dst\_if = pub, gw\_service\_name = junos - dns - udp) } \Rightarrow (gw\_src\_ip = international)\)
and \(\text{(gw\_dst\_if = pub, gw\_service\_name = junos - dns - udp) } \Rightarrow (gw\_src\_ip = local)\) will have a very different confidence\(^26\), which could induce to primary reject the rules set.

\(^{25}\)It is beyond the scope of this work to discuss association rules algorithms and its different characteristics.

\(^{26}\)It is important to note that the confidence value must be reduced to output associations with lower
Analyzing associations in a wider way points out the (not so obvious) possibility of grouping together the “source ip” and the “source interface” labels. This must always be confirmed with domain knowledge and verifying that in fact the source interface or the destination interface contains implicit knowledge and are related with the components that the interface connects to.

It is important to observe that the analyzed baskets are built from log file entries. Because of this, baskets have proper characteristics, different from, for example, shopping baskets that can have any combination of elements. Log files baskets does not have pure random combinations. Baskets of one server does not contain labels from other servers.

To illustrate the involved constraints it can be noted that it does not exists a basket with a “gw” label and a “ns” label at the same time. To exemplify, a basket containing these two labels at the same time (ns_info=query, gw_reason=unset) is not possible because gateway labels does not occur in name server log files.

This factors makes the confidence measure more important than the lift in this particular application. In common association rules applications, such high values are not used, because rules with such values usually reveals knowledge that is already known, and the lift express the utility of the discovered rule by making the ratio between the rule and the independence grade of the antecedent and the consequent. Because of the log files baskets constraints, the lift is not what this method looks for, once the objective is to discover redundant rule pairs.

3.5 Mining Phase

This work uses sequential pattern mining to discover sequential patterns in the network usage. The discovery process is made using the GenPrefixSpan algorithm, an adaptation of the PrefixSpan designed to deal with gap constraints, that means that it allows to discover patterns which the elements are contiguous.

This algorithm follows a pattern-growth approach instead of a candidate generation and test or Apriori philosophy[5].

confidence but yet related with rules with higher confidence.
4. Case Study

Case study uses log data from the network services in Rede das Novas Licenciaturas (RNL) at Instituto Superior Técnico (IST) from Universidade Técnica de Lisboa (UTL), in Lisbon, Portugal.

RNL is part of Departamento de Engenharia Informática\(^1\) (DEI) and is responsible to support the necessary services and software used by students in their academic journey.

RNL has the web page with information to students and the public, institutional services and public services. Institutional services comprise a discussion forum and authentication services. Public services comprise a public FTP service with collaborative open source projects. For example, Eclipse platform or Gentoo Linux distribution.

This chapter is organized as follows: Sec. 4.1 presents the network configuration of the infrastructure, Sec. 4.2 presents the services that provided the data for this work, Sec. 4.3 presents the results analysis.

4.1 Network Configuration

The RNL network is divided in six segments, as presented in Fig. 4.1. Each segment is a subnet with specific organizational constraints and security policies. These subnets are the Admin network composed by the system administrators workstations, laptops and other related material like the office printer, the Priv network composed by private services like the backup system and remote logging system, the GIA\(^2\) network mainly composed by laptop accesses for professors and computer science researchers, and the DMZ\(^3\) network composed by services with less restrictive access and security policies, for example, linux remote login service for computer engineering students. There are also two other subnets, the Labs network composed by laboratory workstations used by students, and the Pub network composed by public services.

The Admin, GIA and Labs networks are in its essence composed by workstations or laptops and does not provide public services to the Internet. For this reason these networks are not considered by this work.

The Priv network is composed by private services, what defines an isolated environment without public services available from the Internet. For this reason, this network is not present in this work.

The DMZ is composed by specific services, important only to the organization, for example, dedicated development systems for specific disciplines. For this reason, this network also is not considered by this work.

\(^1\)Computer Engineering Department
\(^2\)IDE is acronym to Grupo de Inteligência Artificial (Artifitial Inteligence Group), is historically called GIA
\(^3\)DMZ is acronym to Demilitarized Zone [14]
4.2 Components and Services

4.2.1 Servers

This work analyzes some public services in the Pub network, mining log files from the following computers:

1. Alpha, the gateway. This computer is the router and firewall of the infrastructure. It does not provide a public service by itself, but its work is critical to the infrastructure availability and security policies enforcement.

2. NS & NS2, the name servers. These computers are responsible for domain name resolutions.

3. FTP, the file transfer protocol server. This computer is responsible for the sharing of public files and some open source projects collaboration. For example, sharing public releases of the Eclipse IDE\(^4\) platform or the Gentoo Linux distribution.

4.2.2 Services

Services mined are the Domain Name System (DNS) and the File Transfer Protocol (FTP).

1. The Gateway is a Juniper Networks SRX series appliance[37].
   The gateway logs information only considers the Cartesian product between the Pub network interface, the Labs network interface and the Internet uplink interface. This means that information between two networks that are not representative for this work are not considered. For example, network traffic information between Admin network and DMZ network are not registered by the gateway.

2. DNS service is provided by BIND service running in a hardened Gentoo GNU/Linux operating system.

3. FTP service is provided by Pure-ftp and Apache services running in a hardened Gentoo GNU/Linux operating system.

RNL infrastructure has a remote logging system. This system is composed by a computer named Logaritmo running a logging factory via Stunnel[69]. Public servers, like Name Servers and FTP Server, mirrors its log files in Logaritmo as security measure. If any service is compromised and the attacker can have super user access, than the attacker can change or delete log information. The mirror copy will not be easily accessible to an attacker once it is

\(^4\)Integrated Development Environment
4.3. EFFICIENCY ANALYSIS

in a more restrictive network segment. It also works as a backup if the servers suffers some kind of catastrophe.

Each server has a directory subtree with mirrored log files in the remote logging system. This remote logging system makes logging gathering easy because log files are stored in a centralized place, avoiding the necessity of retrieving log files from several different servers.

4.2.3 Applied criteria

This work lays on network communications between computers over the Internet. The Internet uses the 4th version of the protocol called IPV4\(^5\), but the shortage of IPV4 free addresses leads to the design and use of the 6th version of Internet protocol called IPV6\(^6\). IPV6 is not an extension of IPV4. Instead, it is a new different protocol, incompatible with IPV4. IPV6 has been implemented in recent years, but it is not as comprehensive as IPV4 yet, and because of this aspect IPV6 log entries were not considered.

To label the network addresses, it was made a refinement in the “local” label. The “national” and the “international” label follows the same criteria as in Sec. 3.4.2.3.

The “local” label was detailed in: ”NS” for the primary name server, ”NS2” for the secondary name server, ”COMSAT” for the mail server, ”FTP” for the FTP server, ”WWW” for the web server, and local for the remaining addresses in the local network. Since this work goal was to research of sequential patterns in clients network interaction with public services, these were the relevant servers to identify.

IP addresses where classified using IP Address Location from \textit{http://www.ipaddresslocation.org/}.

4.3 Efficiency Analysis

The analysis was made in a sample with one day of retention time, containing 18665 sequences. Tests were made in a Intel dual core at 2.8GHz computer with 1GB of RAM running Ubuntu Server Linux with a 32bit kernel version 2.6.38.

The preprossessing was implemented in Perl using regular expressions to output sequencies from log files and the Apriori and PrefixSpan algorithms were implemented in Java.

This case study focus on service log files only, leaving out the local log files, for example, kernel messages. Mail servers and authentication servers were focus of interest to this work, but could not be included in this work due privacy concerns.

In the gateway, it was mined the routing service log files with 350081 entries.

In the name servers, it was mined the \textit{BIND} log files with 787424 from NS and 57761 from NS2, to total 845185 entries. Since the name servers are a replicated service, the two servers were considered the same as a single server.

\(^5\) Acronym to Internet Protocol Version 4
\(^6\) Acronym to Internet Protocol Version 6
In FTP server, it was mined the Pure-ftpd log files with 10918 entries, the Apache log files with 58925 entries, and the Rsync log files with 3204 entries, to total 73047 entries.

### 4.3.1 Evaluation Before Reduction

Items frequencies are represented in an histogram form presented in Fig. 4.2, for a 10% minimum support value. Fig. 4.2 shows the frequency of each label value, sorted from the most frequent item, starting from left, ending in less frequent items on the right.

Before the reduction phase, Fig. 4.3 presents the number of discovered sequential patterns. Fig. 4.3 starts with low support values from left to higher support values on the right. At 10% minimum support there were discovered 889361 sequential patterns.

![Figure 4.2: Data Frequency Histogram Before Reduction](image)

![Figure 4.3: Number of Sequential Patterns Before Reduction](image)

Fig. 4.4 presents the correspondent processing time to each support value presented in Fig. 4.3 respectively, starting from the lower support values from the left to higher support on the
4.3. EFFICIENCY ANALYSIS

right. At 10% minimum support the processing time was 06 hours, 55 minutes and 12sec.

Figure 4.4: Time Before Reduction

It can be noted that for support values higher than 50%, the time consumption is dominated by data readings and the processing time is not relevant.

4.3.2 Data Reduction

Data reduction is an important step in the process once it allows a significant reduction in the number of generated candidate sequences.

Analyzing the application of association rules, Fig. 4.5, in the name server taxonomic representation of the information, Appendix .1, allows the aggregation of the “view” field with the “source IP” field.

These association rules statistically confirm the semantic analysis made.

 eject for the processing time was 06 hours, 55 minutes and 12sec.

Figure 4.4: Time Before Reduction

It can be noted that for support values higher than 50%, the time consumption is dominated by data readings and the processing time is not relevant.

4.3.2 Data Reduction

Data reduction is an important step in the process once it allows a significant reduction in the number of generated candidate sequences.

Analyzing the application of association rules, Fig. 4.5, in the name server taxonomic representation of the information, Appendix .1, allows the aggregation of the “view” field with the “source IP” field.

These association rules statistically confirm the semantic analysis made.

The gateway analysis is more complex, and contrarily to the name server that could be reduced only by a semantic analysis, the gateway needs a complementary analysis. Appendix .1 presents a taxonomic representation of the gateway information, and Fig. fig:gateway-association-rules show the rules that allows the aggregation of the “service name” with the “rule name” fields, and the “destination IP” with the “destination interface”.

Data reduction slightly increased the alphabet size from 4107 items to 4113 items, because some different label values are combined in a new label, but when the frequency is calculated, some of the new labels are not frequent and in consequence are not included in next integrations of the algorithm.

Figure 4.5: Name server association rules for data reduction

The gateway analysis is more complex, and contrarily to the name server that could be reduced only by a semantic analysis, the gateway needs a complementary analysis. Appendix .1 presents a taxonomic representation of the gateway information, and Fig. fig:gateway-association-rules show the rules that allows the aggregation of the “service name” with the “rule name” fields, and the “destination IP” with the “destination interface”.

Data reduction slightly increased the alphabet size from 4107 items to 4113 items, because some different label values are combined in a new label, but when the frequency is calculated, some of the new labels are not frequent and in consequence are not included in next integrations of the algorithm.
4.3.3 Evaluation After Reduction

After reduction phase, some improvements can be noted.

Fig. 4.7 presents an histogram representation of the items frequencies after the data reduction for a 10% minimum support value, sorted from the most frequent item, starting from left, ending in less frequent items on the right.

Figure 4.7: Data Frequency Histogram After Reduction

After the reduction phase, Fig. 4.8 presents the number of discovered sequential patterns, starting with low support values from the left to higher support values on the right. At 10% minimum support there were discovered 15493 sequential patterns.

Fig. 4.9 presents the correspondent processing time to each support value presented in Fig. 4.8 respectively, starting from the lower support values from the left to higher support on the
4.3. EFFICIENCY ANALYSIS

Figure 4.8: Number of Patterns After Reduction

right. At 10% minimum support the processing time was 01 hour, 35 minutes and 35 seconds.

Figure 4.9: Time After Reduction

It can be observed that for support values higher than 40%, the time consumption is dominated by data readings and the processing time is despicable, which represents a reduction comparing with data before the reduction step.

4.3.4 Comparative Analysis

To highlight the improvements achieved with data reduction, there are presented the number of discovered sequential patterns and corresponding processing time.

Fig. 4.10 shows a comparison between the number of patterns represented in pairs of bars, with the left bar representing the number of sequential patterns before the reduction, and the right bar representing the number of sequential patterns after the reduction.

For 10% of support the improvement was from 889361 sequential patterns to 15493 sequential patterns, a reduction of 98%.

Preprocessing time is constant, dominated by the regular expressions processing, and is less than three minutes.

Fig. 4.11 shows a comparison between the time consumption by two time lines. The upper
CHAPTER 4. CASE STUDY

Figure 4.10: Reduction Impact in the Number of Patterns

time line represents the time spent after the data reduction, the lower time line represents the time spent after the data reduction, here the efficiency was improved by 77%.

Figure 4.11: Reduction Impact in Time

4.3.5 Scalability Analysis

Scalability was tested for 30% support, trying to process a week, starting 11th of May of 2011. The process could only process two days of logs, running out of memory during the mining phase.
Fig. 4.12 shows the number of sequences over time, with a rough average increase of 16 thousand entries per day.

![Figure 4.12: Nr. of Sequences Over Time](image)

Fig. 4.13 shows the number of baskets over time, with a rough average increase of 926 thousand entries per day.

![Figure 4.13: Nr. of Baskets Over Time](image)
Fig. 4.14 show the total parsing time, with a rough average increase of 4 minutes of parse time per day. Total parse time corresponds to the initial parsing consisting in the baskets identification, the blocks integration so the entries follow the temporal order and the baskets extraction and output to me mined.

Figure 4.14: Parsing Duration Over Time

Fig. 4.15 show the number of discovered patterns along two consecutive days of logs.

Figure 4.15: Nr. of Discovered Sequential Patterns

Fig. 4.16 show the time consumption to discover the corresponding patterns.

Figure 4.16: Time of Sequential Patterns Discovery

The mining algorithm run out of memory for more than two days of log data. The algorithm was tested for the same data set with support values varying from 30% to 95%. The average running time before memory errors was about ten minutes.
4.3. **EFFICIENCY ANALYSIS**

4.3.6 **Qualitative Analysis**

Qualitative analysis evaluates the quality of discovered patterns for 20% of support and 1 day of log data. Analyzed data only contains patterns with the maximum length of two interactions. Fig. 4.17 show the proportion of the distribution in terms of number of baskets in sequences.

![Figure 4.17: Sequences Length Proportion](image)

Short length sequences is a direct consequence of the reduced number of mined servers beside the typical interaction between clients and server.

Reduction process is highlighted by the keyword “and” that connects two distinct labels used before the reduction process. For example, considering the pattern:

“(alpha_session=close, alpha_dst_ip=comsat and alpha_dst_if=pub, alpha_dst_port=25, aplha_src_ip=international and alpha_src_if=world, alpha_reason=tcp_fin, alpha_service_name=junos-smtp and alpha_rule_name=comsat, alpha_protocol=tcp)”

It can be seen the aggregation of the label “alpha_dst_ip=comsat” with the label “alpha_dst_if=pub”. Despite the reduced number of services mined, this frequent pattern show a strong evidence of TCP connections from the world to port 25 of the mail server, with clean connection closed. Even with only one element and without other data from the mail server, this pattern reveals yet useful information.

Not all patterns reveals useful information. For example, considering the pattern:

“(aplha_src_ip=international and alpha_src_if=world)(alpha_session=close)”

It can be observed that it is frequent to have an international connection followed by a session close event. Even with two interaction steps, this pattern reveals poor information.

Important frequent patterns with more than one basket can be found. For example, considering the frequent pattern:

“(ns_info=query, ns_src_ip=international and view=external-in) (aplha_src_ip=international and alpha_src_if=world, alpha_dst_port=53, alpha_protocol=udp)”

It can be observed that it is frequent to have a resolve request from the world to the name
server, confirmed by the gateway with the information about the request origin and protocol used.

It is pertinent to speculate about the relation that this pattern together with the first pattern evaluated could be confirmed if mail server data was also mined. Analyzed patterns does not show any direct relation about the frequent name server resolve requests and the mail server connections from the world. In order to have more rich results, it would be necessary to add more data to the process and refine the CRISP-DM cycle, if possible.
5. Conclusion

Data mining in its pure sense, is a blind process, agnostic to the application domain, so the information extracted is highly dependent the way the data is looked through.

Each domain has its own needs so the processes used must be adjusted to provide information as rich as possible, which implies addition of domain knowledge and process adaptation to the domain.

In this work, some parameters must be defined, like support threshold, even if its value depends on the total number of observed items, in example, a value that handles 8% of the total events, because even with such values, it can generate a huge list of results so a human being can not extract the desired results.

This process is an off line process so the parameters can be adjusted in each iteration in a way to improve the quality of the results.

It was shown that in order to be able to process a reasonable amount of data, it is necessary to apply reduction to eliminate redundant data, avoiding time consumption processing data that will not bring relevant new knowledge.

Patterns quality is highly dependent of the used data, and the number of mined services. The more comprehensive the data is, the higher quality the sequential patterns are. It was shown that, patterns reveals interactions between mined services, and are also able to point some possible relations in other connected services.

As a future complement of this work, it can be considered extend this process to incorporate available SNMP information like, for example, load information of the server when some important data is processed. Such information is a valuable complement, as well as world events like security advisories, to provide end user information about the circumstances of important events.

This work have some characteristics of an integration project, like reading and processing text log files. This process is not efficient, but there is no better way to do it once there is no control over its components. The universal output format of the majority applications is the human readable text format because it was built to interact with humans.

Looking through the future, if a wider integration necessity grows, probably it will be possible to have an application programming interface (API) that provides the data without the normal process to be presented in a human readable text format. It eliminates two time consuming steps, one with formatting the information to a human readable format and other in a reverse way taking human readable formatted information to a structured way again. Of course, one important concern can not be neglect, that is a semantic information complement, vital to information comprehension and integration.
Appendices
.1 Case Study Taxonomies
.1. CASE STUDY TAXONOMIES

This appendix presents taxonomies defined to accomplish case study. These taxonomies are instances of the generic taxonomies in the process.
Figure 3: Case Study FTP Server Taxonomy
2. DETAILED SCALABILITY

.2 Detailed Scalability
This appendix presents detailed data about the processed data in the Chapter 4.

Fig. 4 show the number of processed baskets over time with a rough average of 926 thousand entries per day. The left bar represents the initial number of entries in log files, the middle bar represents the number of discarded entries and the right bar represents the resultant entries. Discarded entries are log entries that does not comply with the mining criteria or entries that are not relevant to the process.

Figure 4: Nr. of Processed Baskets Over Time

Fig. 5 shows the time consumption of the blocks integration process. This process consists of the integration of the entries from different log files according the time line.

Figure 5: Time of Blocks Integration in a Time Line
Fig. 6 shows the time of baskets extraction and sequences construction for each sequence identifier.

Figure 6: Time of Baskets Extraction

Following charts presents detailed data about the number of processed entries and the time spent processing these entries, respectively.

Fig. 7 shows the number of processed entries in the gateway with a rough average increase of 246 thousand entries per day.

Figure 7: Nr. of Processed Baskets Over Time in the Gateway
Fig. 8 show the time consumption of baskets processing in the gateway.

![Gateway Baskets Time](image1)

Figure 8: Baskets Extraction Time in the Gateway

Fig. 9 shows the number of processed entries in the primary name server with a rough average increase of 505 thousand entries per day.

![Nr. of Processed Baskets in the Name Server (primary)](image2)

Figure 9: Nr. of Processed Baskets Over Time in the Primary Name Server
2. DETAILED SCALABILITY

Fig. 10 shows the time consumption of baskets processing in the primary name server.

![Name Server (primary) Baskets Time](image)

Figure 10: Baskets Extraction Time in the Primary Name Server

Fig. 11 shows the number of processed entries in the primary name server with a rough average increase of 63 thousand entries per day.

![Nr. of Processed Baskets in the Name Server (secondary)](image)

Figure 11: Nr. of Processed Baskets Over Time in the Secondary Name Server
Fig. 12 shows the time consumption of baskets processing in the secondary name server.

Figure 12: Baskets Extraction Time in the Secondary Name Server

Fig. 13 shows the number of processed entries on the FTP service in the FTP server with a rough average increase of 16 thousand entries per day.

Figure 13: Nr. of Processed Baskets Over Time on FTP Service in the FTP Server
2. Detailed Scalability

Fig. 14 shows the time consumption of baskets processing on the FTP service in the FTP server.

![Figure 14: Baskets Extraction Time on FTP Service in the FTP Server](image)

Fig. 15 shows the number of processed entries on the HTTP service in the FTP server with a rough average increase of 93 thousand entries per day.

![Figure 15: Nr. of Processed Baskets Over Time on HTTP Service in the FTP Server](image)
Fig. 16 shows the time consumption of baskets processing on the HTTP service in the FTP server.

![FTP (http service) Baskets Time](image)

Figure 16: Baskets Extraction Time on HTTP Service in the FTP Server
Bibliography


[25] Behzard Mortazavi-Asl Helen Pinto Quiming Chen Umeshwar Dayal Jian Pei, Jiawei Han and Mei-Chun Hsu. Prefixspan: Mining sequential pattern efficiently by prefix-projected pattern growth. Technical report, Simon Fraser University and Hewlett-Packard, 2001.


[34] Jiawei Han Ming-Syan Chen and P. S. Yu. Data mining: an overview from a database perspective. Technical report, National Taiwan University, 1996.


