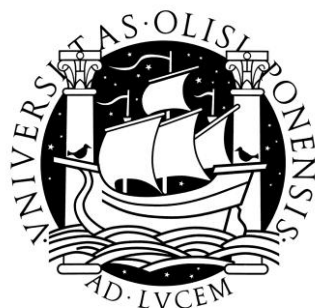


UNIVERSIDADE DE LISBOA

Faculdade de Ciências

Departamento de Informática



MS IPTV AUDIT COLLECTION SERVICES

Nuno Filipe Fernandes Antunes

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2011



UNIVERSIDADE DE LISBOA

Faculdade de Ciências

Departamento de Informática



MS IPTV AUDIT COLLECTION SERVICES

Nuno Filipe Fernandes Antunes

DISSERTAÇÃO

Tese orientada pelo Professor Doutor António Casimiro

MESTRADO EM SEGURANÇA INFORMÁTICA

Dezembro 2011



# Resumo

Microsoft Mediaroom Internet Protocol Television (MS IPTV), uma plataforma de televisão digital, levou o conceito de televisão a uma dimensão totalmente nova. MS IPTV é um sistema onde o serviço de televisão digital é entregue aos clientes usando *Internet Protocol (IP)*, através de uma conexão de banda larga. Com o advento do IPTV começaram a aparecer novas situações relacionadas com a segurança da televisão, uma vez que, a infra-estrutura começou a ganhar complexidade e exposição a uma série de novos riscos. Por esta razão, a segurança numa infra-estrutura de MS IPTV não é apenas mais uma funcionalidade, mas sim uma necessidade. Podemos mesmo dizer que hoje em dia é obrigatório aguçar o engenho para estar um passo à frente dos atacantes, uma vez que estes estão sempre à espera de uma brecha, para comprometer os sistemas. Uma infra-estrutura como o MS IPTV armazena por omissão dados relativos ao comportamento dos utilizadores ao nível dos *logs*, no entanto esta informação só se torna relevante se puder ser consultada e analisada com o objetivo de proporcionar uma compreensão a alto nível sobre os diferentes padrões que estão a ocorrer nos servidores ou no comportamento dos utilizadores, uma tarefa que envolve poderosas técnicas de *data parsing*.

A tese apresenta uma abordagem que combina técnicas de *data parsing*, a fim de analisar os logs relevantes da infra-estrutura de MS IPTV, com o objetivo principal de aumentar a segurança através da investigação dos tipos de informações adicionais que pode ser extraída.

Tentámos assim entender se é possível determinar que tipos de ataques estão a ser perpetrados contra a infra-estrutura MS IPTV, com base na análise dos logs. Como o foco central desta tese está no diagnóstico, propomos uma abordagem para descobrir ataques, onde os logs são verificados para identificar grupos coerentes de ocorrências susceptíveis de constituir ataques que apelidámos de *padrões*. Nos testes, verificámos que a nossa abordagem consegue bons resultados na descoberta de ataques. Os resultados obtidos têm a vantagem adicional de poderem ser integrados na ferramenta de monitorização utilizada pelas equipas de operação dos sistemas da *Portugal Telecom*, o System Center Operations Manager (SCOM).

**Palavras-Chave:** Microsoft Internet Protocol Television, Padrões de ataque; Expressões regulares; Integração, SCOM MEO Collector Security



# Abstract

Microsoft Mediaroom Internet Protocol TeleVision (MS IPTV), one of the platforms for digital TV, took television to an all new dimension level. MS IPTV is described as a system where a digital television service is delivered to consumers using the Internet Protocol over a broadband connection. Since the infrastructure started to gain complexity and exposure to a number of new risks, never envisaged situations related to television security started to appear. For this reason, MS IPTV security is not only a great asset, but also a necessity. Nowadays it is mandatory to sharpen the wit to get ahead of attackers, who are always waiting for a breach to compromise our systems.

MS IPTV log servers collect information about user and system behavior. However, this information only becomes relevant if it can be queried and analyzed with the purpose of providing high-level understanding about the different patterns. This task must comprise powerful data parsing techniques, since MS IPTV is able to generate close to one *terabyte* of logs per day.

This thesis presents an approach that combines data parsing techniques in order to analyze relevant MS IPTV logs, with the main objective to increase security through the investigation of what type of additional information can be extracted from the server log files of a MS IPTV platform. The thesis focus is on diagnosis, trying to understand if it is possible to determine what type of attacks are being perpetrated against the MS IPTV infrastructure.

We propose an approach for discovering attacks, where the application logs are scanned to identify coherent groups of occurrences that we call *patterns*, which are likely to constitute attacks. Our results showed that our approach achieves good results in discovering potential attacks. Our output results can be *integrated* into the MS IPTV monitoring system tool SCOM (System Center Operations Manager), which is an additional advantage over the other monitoring and log management systems.

**Keywords:** MS IPTV, Attack Patterns; Regular Expressions, Integration, SMCS





# Acknowledgments

I would like to thank CMU (Carnegie Mellon University), FCUL (Faculdade de Ciências da Universidade de Lisboa) and PT (*Portugal Telecom*) the sponsor of this program, for their support during the past semesters.

I would like to thank all the professors, for their support and guidance that helped me to take this endeavor to a successful conclusion, in particular my advisor Prof. António Casimiro who never failed to provide me the right advice and review, which I needed to develop my work.

This was the fulfillment of a long awaited dream since I have the notion how important it is to accomplish an international educational program, and Master of Science in Information Technology – Information Security surpassed my expectations.

However, it is my family, which always gave me strength to never give up, the ones that I have to thank for all the affection and support.



*Ana, João and Francisco, they were my lighthouse during this CMU journey.*



# Contents

Resumo.....	I
Abstract .....	III
Acknowledgments.....	V
Contents .....	IX
List of Figures .....	XIII
List of Tables.....	XV
Abbreviations .....	XVII
1      Introduction .....	1
1.1      Motivation.....	4
1.2      Contributions.....	5
1.3      Document structure .....	5
2      Background.....	7
2.1      Security concepts .....	8
2.1.1 Terminology .....	8
2.1.2 Attacks and vulnerabilities .....	9
2.1.3 OWASP Top ten .....	12
2.2      MicroSoft Internet Protocol Television (MS IPTV) .....	12
2.2.1 MS IPTV architecture.....	13
2.2.2 MS IPTV identified threats .....	15
2.3      Monitoring and Log management systems.....	16
2.3.1 OSSIM - AlienVault Unified SIEM.....	16
2.3.2 SALSA.....	19
2.3.3 Feedzai .....	20
2.3.4 SCOM.....	22

2.3.5	Pulso .....	23
2.3.6	ArcSight SIEM .....	24
2.3.7	Summary and comparison of tools .....	26
2.4	Event diagnosis at Portugal Telecom .....	27
3	Data Set Characterization.....	29
3.1	Internet Information Services data set .....	29
3.1.1	IIS logging .....	30
3.1.2	IIS workflow.....	32
3.1.3	IIS HTTP codes .....	34
3.2	Common mistakes regarding log policies.....	34
4	Methodology and Test Cases .....	37
4.1	Methodology .....	37
4.1.1	Attack patterns.....	37
4.1.2	Regular expressions.....	39
4.2	Test cases .....	39
4.2.1	The simulated log .....	40
4.2.2	Real world logs – Scan31.....	41
4.2.3	MS IPTV logs .....	41
5	SCOM MEO Collector Security .....	43
5.1	Problem statement .....	43
5.2	SMCS Architecture .....	44
5.2.1	On-line vs. Off-line approach .....	45
5.2.2	Implementation design .....	45
5.3	Definition of SMCS regular expressions .....	47
5.3.1	A1-Injection attacks.....	49
5.3.2	A2-Cross-Site Scripting attacks.....	51
5.3.3	A3-Broken Authentication and Session Management .....	54

5.3.4	A4-Insecure Direct Object References .....	54
5.3.5	A5-Cross-Site Request Forgery (CSRF).....	55
5.3.6	A10-Unvalidated Redirects and Forwards.....	56
5.3.7	Detecting other relevant attacks.....	56
5.4	Complementing SMCS with Log Parser .....	57
5.4.1	Script Abuse.....	59
5.4.2	Trojan attacks .....	59
5.4.3	DoS and DDoS attacks .....	60
5.4.4	Aggregating Log Parser outputs .....	62
5.5	Reducing false positives .....	62
6	Results .....	63
6.1	Detecting attacks in a simulated test case .....	63
6.2	Detecting real world attacks .....	67
6.3	Detecting MS IPTV attacks .....	70
6.4	SMCS execution overhead .....	72
6.4.1	Parsing times – simulated log.....	72
6.4.2	Parsing Times - scan31 log .....	73
6.4.3	Parsing Times – MS IPTV log.....	73
6.5	SMCS integrating with SCOM .....	75
7	Conclusions .....	77
8	Future Directions.....	81
8.1	Promote SMCS and SCOM integration.....	81
8.2	Reducing false positive rate .....	81
8.3	Automatically trigger response actions.....	81
8.4	Green monitor attacks prevention.....	82
8.4.1	Baseline correlation.....	83
	References.....	85





# List of Figures

Figure 1 - Number of vulnerabilities in Networks, OS and applications .....	2
Figure 2 – OWASP top ten list .....	12
Figure 3 - MS IPTV Microsoft Mediaroom blocks.....	13
Figure 4 – AlienVault architecture .....	17
Figure 5– HLD Feedzai Pulse .....	21
Figure 6 - FeedZai architecture .....	21
Figure 7 – ArcSight architecture.....	25
Figure 8 – Analyzing non-served web request .....	28
Figure 9 – IIS web log example.....	32
Figure 10 - IIS workflow.....	33
Figure 11 – List of attack patterns.....	38
Figure 12 – Simulated log with attack signatures .....	40
Figure 13 – Number of lines simulated.log .....	40
Figure 14 – Number of lines access_log.....	41
Figure 15 – Number of MS IPTV log lines.....	42
Figure 16 – SCMS implementation design .....	46
Figure 17 – Log Parser .....	58
Figure 18 – Number of hits per file name .....	61
Figure 19 – Log Parser browsers chart.....	61
Figure 20 – Log Parser outputs.....	62
Figure 21 – Simulated scenario output (bar chart) .....	63
Figure 22 - Simulated scenario output (pie chart) .....	64
Figure 23 – Simulated log 1LevelReport output.....	64
Figure 24 – Simulated log 2LevelReport output.....	65
Figure 25 – PowerShell SMCS console .....	65

Figure 26 – SMCS output chart for the real world scenario.....	67
Figure 27 – SMCS attacks detected in the <i>scan31</i> (pie) .....	67
Figure 28 – MS IPTV attacks detected.....	70
Figure 29 – SMCS output for MS IPTV logs (pie) .....	70
Figure 30 – MS IPTV Log Parser output console .....	71
Figure 31 – SCOM and SMCS integration .....	75
Figure 32 – SCOM and SMCS Log Parser integration .....	76
Figure 33 – “Green” alerts attack – before the attack.....	82
Figure 34 – “Green” alerts attack – after the attack.....	82
Figure 35 – “Green” attack workflow.....	83

# List of Tables

Table 1 – Current worldwide cyberattacks .....	10
Table 2 – Computer worms that exploit one or more vulnerability .....	11
Table 3 – Tools evaluation.....	26
Table 4 – IIS logging fields .....	31
Table 5 – HTTP codes returned by IIS.....	34
Table 6 – OWASP top ten list 2010 .....	49
Table 7 – SMCS results and simulated log inputted attacks .....	66
Table 8 - SMCS results and Honeynet solutions.....	68
Table 9 – Attacks detected comparison (Honeynet vs. SMCS) .....	69
Table 10 – Simulated log analysis time .....	72
Table 11 – Scan31 log analysis time .....	73
Table 12 – MS IPTV log analysis time .....	73
Table 13 - MS IPTV all logs analysis time.....	74



# Abbreviations

A1 – Injection attacks, according to the OWASP Top 10

A2 – Cross-Site Scripting (XSS) attacks, according to the OWASP Top 10

A3 – Broken Authentication and Session Management, according to the OWASP Top 10

A4 – Insecure Direct Object References, according to the OWASP Top 10

A5 – Cross-Site Request Forgery (CSRF), according to the OWASP Top 10

A10 – Unvalidated Redirects and Forwards, according to the OWASP Top 10

CSRF – Cross-Site Request Forgery

DDoS – Distributed denial of service attack

DoS – Denial of service attack

IIS – Internet Information Services

IPTV – Internet Protocol TeleVision

MEO – *Portugal Telecom* MS IPTV production environment

MS IPTV or Mediaroom – Microsoft Internet Protocol TeleVision

OWASP – The Open Web Application Security Project

ReDoS – Regular expressions Denial of Service

REGEX – Regular expression

SCOM – System Center Operations Manager

SMCS – SCOM MEO Collector Security

VOD – Video on demand

XSS – Cross-Site Scripting

Logs are just data,  
processed and analyzed, they become information  
*Marcus J. Ranum*

# 1 Introduction

Television has become a so significant part of today's life, that something that was earlier just a token of luxury is nowadays a need. We should ask ourselves if we could live without TV (classic TV, computer TV or programs and shows). According to a recent study, the average american television viewer is watching more than 151 hours of television per month [4] (considering that in average a month has 700 hours, we work 140 hours and we spend 1/3 of it sleeping), this number is impressive and it has been increasing year by year. Based on these numbers, it is easy to realize that people daily routines will be affected if something happens to the TV services, since the traffic, weather or stock news became more than commodities, but basic goods.

Satellite, terrestrial and cable TV made people used to a service that is "always on" with reliable quality levels. There is no paying subscriber that will accept frequent interruptions or bad quality images, which leaves the new TV concept - Internet Protocol TeleVision (IPTV) with no space for errors.

This thesis addresses the problem of auditing a specific IPTV service, the Microsoft Internet Protocol TeleVision (Mediaroom or MS IPTV) environment in terms of security. We developed a new set of log parsing techniques, using Microsoft PowerShell scripting language, which can be used by the IT teams to monitor the MS IPTV system proactively. Since no security analysis is performed in a regular basis for the internal MS IPTV systems, we realized how important it is to develop methods that identify and even prevent attacks. This is important, since under certain circumstances, attackers could take control of a significant number of set-top boxes and use them to perform attacks against middleware servers<sup>1</sup> (e.g. denial of service attacks).

We named our project SMCS (SCOM MEO Collector Security), because it uses the best of two worlds. The System Center Operation Manager (SCOM) infrastructure, developed by Microsoft to monitor explicitly their products, and the open source development tools that can be

---

<sup>1</sup> Database middleware, application server middleware, transaction-processing monitors and Web middleware

adapted to fit Microsoft Internet Protocol TeleVision (MS IPTV) needs (e.g. PowerShell, Log Parser, Google charts, Html pages, and others). We will develop SCOM MEO Collector Security to deal with the lack of high-qualified vulnerability researching tools, for a fair price. While this scarcity persists, the defenders will be in disadvantage in terms of protecting their systems against attacks. SMCS provides relevant advances, since these attacks will not only affect the operators' revenue but also its reputation.

This thesis also addresses the fact that the number of attacks is currently so high, that companies do not distinguish which they should deal first, since they are not aware of those that represent the higher risk of causing substantial damages. The data logs that we use cover the MS IPTV 2011 web logs, from different services and applications, what give a reliable picture of what are the current attacks against the MS IPTV infrastructure. Our goal is to document the immediate threats and those that are emerging and could compromise the data, operations and process of MS IPTV.

We also present some descriptions of the most recent attacks and some of the most important discoveries made in the last year (OWASP 2010) [14], on the field of cybernetic security.

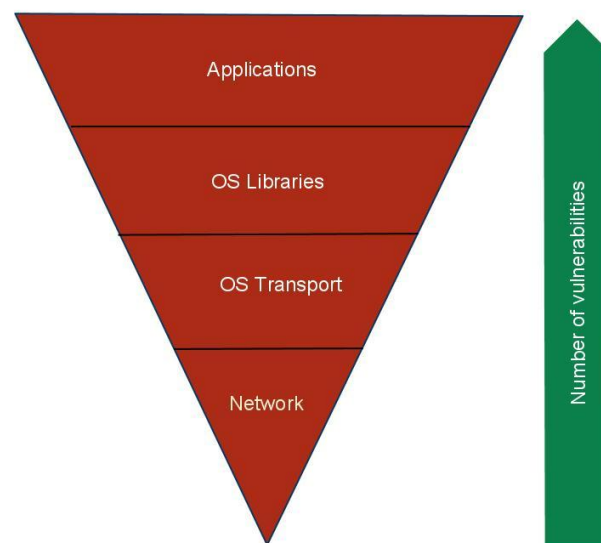


Figure 1 - Number of vulnerabilities in Networks, OS and applications

Recent studies (year 2010), show that the number of vulnerabilities discovered in applications are much higher than those discovered on operating systems, as illustrated in Figure 1. Web applications are one of the preferred attacker targets [5]. For this reason we will concentrate our efforts on this direction.



Our ultimate goal will be to realize how these attacks are performed because only then we can become and train people to be effective MS IPTV defenders.

The thesis relies on MEO, a *Portugal Telecom* MS IPTV production environment and the core business of the operator, which is suffering a migration to a new Microsoft Internet Protocol Television version. These circumstances allied to the fact that production data requires special permission to be retrieved from the platform, restrict the scope of this project to the design and implementation of a security collector that actively audits exposed services, using whatever information is available.

Nevertheless, all the information used in this project is similar to the one used in the MS IPTV systems *of any other operator*, and its application to any production environment requires few conversions.

**Note:** We will use **boldface** characters, when a term is defined, and *italic* characters to focus the reader's attention.

## 1.1 Motivation

IPTV is considered by many [6], one of the emerging technologies that will play an important role in the near future (next decade). Like all new technologies, it brings a large spectrum of paradigms. It is based on a new network solution, through which Internet television services are delivered. It includes live television, time-shift programming and video on demand. All these new services are supported in information systems and all the data generated is saved in logs that are used latter to generate all kinds of alarms and operational data. The Microsoft Internet Protocol TeleVision (MS IPTV) service became the new banner for *Portugal Telecom* and all its reputation is in stake. The growth has been auspicious, as well as the amount of data and events that need to be correlated to know what is happening in the MS IPTV platform.

Security in this new IPTV architecture is also a major concern for the operator that used to work with other architectures and suddenly is forced to adapt to the market impositions. System administrators also have less experience with this new architectural model and due to this, problems are prone to happen. The novelty of this architectural system is also one of the causes of the security faults, owed to all the data parsing and other detection mechanisms that are still poorly tuned to the new IPTV reality. A lot of tuning and manual analysis is still necessary.

Firewalls are an integrant component of the major networks projects and MS IPTV is no exception, nevertheless our experience tell us that, while external systems are closely monitored in terms of security, internal systems tend to be neglected. In the end, what really matters is to find out if attacks are passing beyond the firewalls layer and reaching the application layer (this task is facilitated if we are talking about the web application layer), this is the main issue that we want to help to solve.

Microsoft has learned a lot with past mistakes, and they are coming out with patches faster, this has made MS IPTV architecture more robust and secure. Nevertheless, ports 80 and 443 are usually allowed through firewalls and taking into consideration that a MS IPTV web applications work its way into many components, the only defense is the early detection and patching, since malicious scripts used in attacks are typically heavily obfuscated and evidences carefully deleted in order to evade detection and delay analysis. Web application attacks using brute force password guessing, and web application attacks making use of SQL injection, Cross-Site Scripting and PHP file includes, continue to be popular techniques used for compromising web sites. Traditional protection mechanisms like firewalls are not designed to protect MS IPTV web applications and consequently do not provide adequate defense. The use of firewall

rules is not always. In summary, when we realized the importance of the web services in the MS IPTV platform, we were lead to think that any breakthrough done in terms of security countermeasures will be of utmost importance.

## 1.2 Contributions

The main contribution of our work is to create a collector to audit exposed MS IPTV services. It will help to parse log information around a common objective the detection of attacks that aim to prevent the correct operation of the MS IPTV infrastructure. We called the collector SMCS (SCOM MEO Collector Security), and as far as we know, it is the first log analysis technique that aims to obtain security information from MS IPTV log files.

The main guidelines of SMCS<sup>2</sup> analysis are data logs analysis, data parsing and event diagnosis. *Data logs analysis* because large and complex systems like MS IPTV generate a huge quantity of logs. The first of our contributions is to find the ones that contain valuable information about system security. *Data parsing* since logs must be parsed in search for evidences for attacks. We addressed the most recent and prevalent vulnerabilities and their immediately recognizable attacks. Others can be added, something that is a valuable add-on. *Event diagnosis*, since SMCS present the results in a clear an aggregated form to help the operator diagnose in a more accurate and precise way the attack that happened in the system.

## 1.3 Document structure

The rest of the thesis is organized as follows. We start in Chapter 2 by providing an overview over the context and background of this thesis, where we revise the main concepts associated to security, and then we present an overview of the important blocks of a MS IPTV infrastructure and the already identified threats that can affect their normal operation. The chapter proceeds describing the monitoring and log management systems that we considered relevant to our study, namely OSSIM, SALSA, Feedzai, SCOM, Pulso and ArcSight. Finally, the tools were compared taking into account several parameters that we considered relevant to the analysis. In Chapter 3, we present the reasons that lead us to work with certain logs data sets. We then present an overview over the common errors regarding log policies. Chapter 4 establishes the conceptual model of our approach explaining how we expect to implement our

---

<sup>2</sup> SMCS - symbolizes the integration of the SCOM and MEO systems in the security quest

methodology, using our attack pattern and regular expression strategy. Chapter 5 is dedicated to the solution's architecture and to the implementation of the SCOM MEO Collector Security (SMCS). The solution is implemented accordingly to a workflow presented in section 5.2.2. The chapter proceeds with the description of the regular expressions that will enable a powerful, flexible, and efficient form of text processing for detecting potential attacks. We then present how the integration of the Microsoft Log Parser tool was able to complement SMCS results. The remaining part of the chapter is dedicated to the reducing of false positives rate. In Chapter 6, we present the test cases (simulated, real word and MS IPTV) used to verify if SMCS is able to detect potential attacks. Finally, Chapter 7 is dedicated to the conclusions and Chapter 8 to future work.

## 2 Background

This chapter provides the context and background in which this thesis is inserted. Since our focus is auditing the Microsoft Internet Protocol TeleVision (MS IPTV) environment in terms of security, this chapter starts with an introduction to the commonly used terms and concepts in the security context. The next section presents an overview of MS IPTV architecture blocks and identified threats and their related problems. The final section presents an overview on the monitoring and log management systems, that we consider relevant for our research project.

Internet Protocol TeleVision (IPTV) services are the fastest growing television services in the world today. They are divided into two main flows: the high-definition Linux Android, and MS IPTV. Over 685 companies worldwide are deploying MS IPTV services that will reach over 81 million clients worldwide in 2013 and many more companies are deploying other IPTV solutions [4].

Since the thesis was a suggestion of *Portugal Telecom*, we will focus on their currently deployed MS IPTV solution service, *MEO*. This service requires a great number of servers and databases, when compared with the standard cable networks or free air broadcasts. *Portugal Telecom* choice resided on the fact that MS IPTV represents the future of TV, since it is one of the world most deployed IPTV platforms [5]. Their services are being offered by more than three dozen of the world's leading operators, delivering services to more than seven million consumer households, which corresponds to more than fourteen million set-top boxes deployed throughout the world.

In terms of security, MS IPTV goes to great trouble to ensure it for live TV and VOD<sup>3</sup> services. It employs multiple security measures to ensure that only those subscribers that should have access, can actually access the program.

We will now present some of the mechanisms used to secure the main MS IPTV services. MS IPTV system provides full *DRM*<sup>4</sup>, it encrypts live TV content and VOD with strong encryption

---

<sup>3</sup> VOD - Video on Demand allow users to select the contents they want to watch

algorithms. In terms of *authentication*, MS IPTV uses both client and server authentication via X.509 certificates and server authentication assures that only real MS IPTV servers serve clients. Anyway, the constant technological advances can carry severe threats to the existing security mechanisms and there is where our thesis steps in.

## 2.1 Security concepts

We will now make an introduction to the commonly used terms and concepts, regarding security. Then we will present a brief introduction to the attacks and vulnerabilities and finally the OWASP top ten list.

### 2.1.1 Terminology

Before we continue, it is important to define some of the terminology that is going to be used in the thesis, and in this way mitigate some issues that may arise due to misinterpretation. The following definitions come mainly from [11].

An *attack* is the act of carrying out an exploit. It is also known as a “cyberattack,” if it disrupts the integrity or authenticity of data, usually through malicious code that alters program logic that controls data, leading to errors in output. An attack is the action of causing harm to an information system, for example by unauthorized access or denial of service.

An *attacker* is the person that actually executes an attack. Attackers may range from unskilled individuals leveraging automated attacks developed by others (“script kiddies”), to well-funded government agencies or even large international organized crime syndicates with highly skilled software experts.

An *exploit* is a technique or software code (often in the form of scripts) that takes advantage of a vulnerability or security weakness in a piece of target software.

Vulnerability is defined as a weakness of an asset or group of assets that can be exploited by one or more threats, where an asset is anything that can have value to the organization, its business operations and their continuity, including information resources that support the organization's mission. Vulnerabilities might have different causes, including badly written web

---

<sup>4</sup> DRM - Digital Rights Management is any of the several technologies used to enforce pre-defined policies for controlling access to digital data (such as software, music, movies) and hardware.

applications made by programmers, including the ones considered “expert”. Attackers explore what we call vulnerabilities, a computer or system weakness that allowed them make use of that flaw to reduce the information assurance, in others words reduce the confidentiality, integrity, and availability of computers and systems through an attack.

A *threat* is an actor or an agent that is a source of danger to the system under consideration, or the assets to which it has access. The threat can be a person that abuses the software, a program running on a compromised system, or even a non-sentient event such as a hardware failure. A threat exploits a vulnerability in software to attack it.

An *intrusion* is a malicious activity that triggers a policy violation, making use of a previously discovered vulnerability.

*Honeypots* are traps that appeared as a reaction to the prevalence of security attacks on the Internet. Honeypots initiatives set up information system traps on the Internet, to attract, solicit and monitor malicious attacks. The honeypot data can then be analyzed and mined for valuable information on the patterns behind these attacks, which help in developing defenses against them [12].

An *attack pattern* is a general framework for carrying out a particular type of attack, such as a method for exploiting a buffer overflow or an interposition attack that leverages certain kinds of architectural weaknesses. In these articles, an attack pattern describes the approach used by attackers to generate an exploit against software.

*Risks* capture the probability that a vulnerability will be exploited, as well as the potential damage (impact) that will occur if it is. It is important to note that *risks*, *threats*, and *exploits* are all separate things. *Risks* may be present in the target software, on the target host, or in the broader operational environment of the software.

### **2.1.2 Attacks and vulnerabilities**

We can say that “Kevin Mitnick, Kevin Poulsen, Robert Tappan Morris” are some of the first cyber attackers, since they were responsible for breaching the security barriers of major companies and for the creation of computer worms (e.g. the first computer worm, the “Morris Worm”). Nowadays, the most frequent “family” of attacks, according to OWASP[14], are the “SQL injection” attacks, where a query string parameter that is not filtered or validated, exposes the database contents to the attacker.

Attacks against worldwide systems happen in an almost continuous way. Some examples are presented in Table 1:

Date	Attacks against worldwide systems
Jul 04 2011	<b>Microsoft IIS</b> FTP Globing Functionality causes a remote denial of service vulnerability; Microsoft IIS is prone to a denial-of-service vulnerability affecting the application's FTP server. The following example command: <code>ls "-R p*/./"</code> , causing a DoS attack terminates the affected application, denying service to legitimate users [21].
Jun 18 2011	<b>Sega</b> Japanese video game developer Sega Corp, said on Sunday that information belonging to 1.3 million customers has been stolen from its database, the latest in a rash of global cyberattacks against video game companies. [18]
Jun 06 2011	<b>Sony Pictures</b> confirms hacking group website attack (Security Bytes blog), a hacking activist group which claimed responsibility for attacks against the websites of PBS and Nintendo has breached Sony Pictures [18].
Sept 14 2011	<b>Microsoft IIS</b> Request Header Buffer Overflow Vulnerability, Microsoft IIS is prone to remote buffer-overflow vulnerability. An attacker can exploit this issue to execute arbitrary code within the context of the affected application. Failed exploit attempts will result in a denial-of-service condition. Repeated Parameter Request Denial of Service Vulnerability, Microsoft IIS is prone to a remote denial-of-service vulnerability. An attacker can exploit this issue to force the affected application to become unresponsive, denying service to legitimate users. [23].
Oct 12 2011	<b>Sony Under Attack...Again</b> , Sony Network has detected a large amount of unauthorized sign-in attempts on PlayStation Network (PSN), Sony Entertainment Network (SEN) and Sony Online Entertainment (SOE) services. They discovered these attempts and have taken steps to mitigate the activity. This time less than one tenth of one percent of our PSN, SEN and SOE consumers may have been affected [18].

Table 1 – Current worldwide cyberattacks

From our personal experience, we can say that system administrators have a natural tendency to think in terms of features and functions and neglect security, which leaves attackers with the advantage of having to find only an exploitable vulnerability to achieve the objective of entering the “castle”. Another challenge is the pressure imposed by the market (also called TTM or time to market), that force companies to rush the deployment of their products before



their competitors, at risk of failing their objectives. Time to market led companies to believe that security by obscurity would keep the attackers away. However, looking at an actual list of the current attacks, as presented in Table 1, is enough to realize that this was a very poor assumption.

After looking at the present situation in terms of attacks, we considered important to go back in time, and look into some of major attacks in the history, that caused incalculable damages to companies' computer systems. Some examples are presented on Table 2.

Attacks	Damage caused
<b>Nimda</b>	Was one of the more complex virus/worm released. It infects all versions of Windows (...) as well as Microsoft's IIS. Webworm Nimda scans the Internet for Microsoft IIS Web servers and when a server is found that has open security holes, Nimda enters and modifies random web pages on the server. Nimda is credited with several "firsts" in its infection techniques. It was the first to infect .EXE files, by embedding them into itself as a resource. It also infects Web pages, so unsecured browsers will be infected upon viewing the Web page. Finally, Nimda is the first worm to use any user's computer to scan a network for vulnerable machines behind a firewall to attack (in the past only infected servers did that). [20].
<b>Code Red worm</b>	A computer worm observed on the Internet on July 13, 2001. It attacked computers running Microsoft's IIS web server. It was released on July 13 and the largest group of infected computers was seen on July 19. On this day, the number of infected hosts reached 359,000 [19].
<b>Stuxnet</b>	Microsoft Windows computer worm discovered in July 2010 that targeted industrial software and equipment. While it is not the first time that crackers have targeted industrial systems, it is the first discovered malware that spies on and subverts industrial systems, and the first to include a programmable logic controller (PLC) rootkit [19].

Table 2 – Computer worms that exploit one or more vulnerability

Looking at the devastating effects of these worms, leads us to think what will be the next major computer security threat and, meanwhile, how can we prepare for it. First it is important to understand if some attacks could be identified a priori or, only a posteriori.

Second, in our opinion, log files already allow performing a detailed analysis of attacker's actions, and the only question is how to retrieve that information. We hope with the method employed in the thesis, to be able to retrieve that information. We believe that this is possible since logs showed à posteriori that worms, virus and attackers left fingerprints behind.

### 2.1.3 OWASP Top ten

When we made ourselves the question of what type of attacks should we start looking for, we became overwhelmed with the multitude of answers. To help us with this task, we took as starting point the OWASP (Open Web Application Security Project) top ten document, that lists the most actual and critical web application security flaws [14]. The goal of this document, from this non-profit global organization, is to educate companies and government agencies and help them create more secure web application environments, so it will fit as a glove in our objectives. It is also important to notice that this is not a static attacks list. Figure 2 present side by side the differences between the 2007 list and the updated list (year 2010). For the thesis purposes, we will use the newer one.

In summary, we direct our efforts in a certain direction, based on the OWASP list, showed in Figure 2, but it is important to notice that we are not limiting our scope to it, as we will demonstrate in section 5.3.7 and 5.4.

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)

Figure 2 – OWASP top ten list

## 2.2 MicroSoft Internet Protocol Television (MS IPTV)

In this section, we will present an overview of the important blocks of a MS IPTV infrastructure and the already identified threats that can affect their normal operation.

### 2.2.1 MS IPTV architecture

MS IPTV has the advantage of using a bundle of services and applications, where Microsoft has a solid reputation. The actual architecture of the MS IPTV system implemented by *Portugal Telecom* is illustrated in Figure 3.

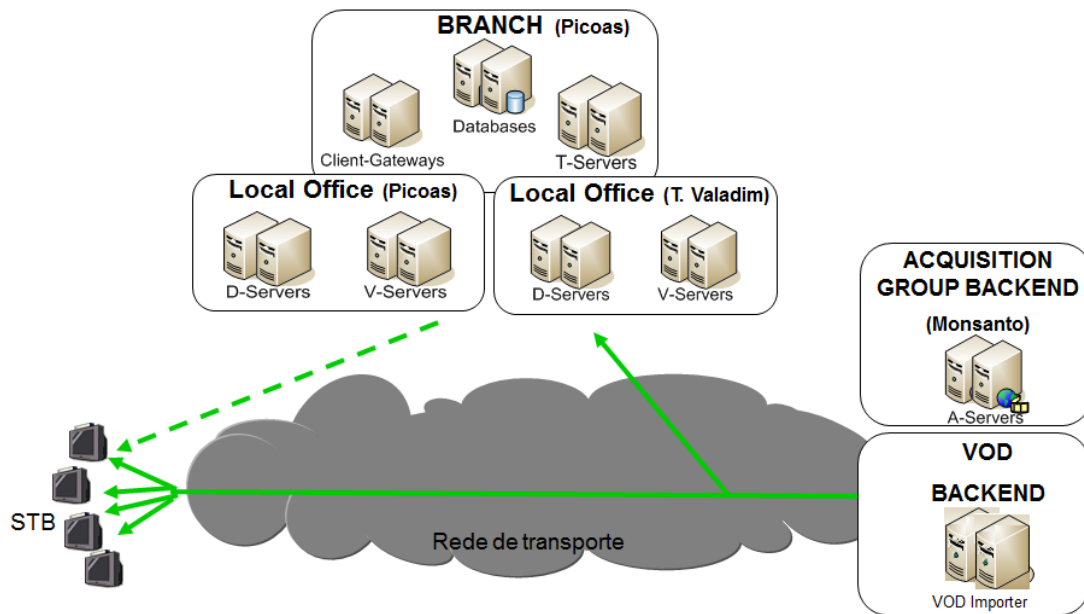


Figure 3 - MS IPTV Microsoft Mediaroom blocks

We will describe the architecture solution that combined all those services and a brief description of the main blocks presented, based on the Microsoft windows "family", with the purpose of providing a television infrastructure.

**Acquisition block** is a group of servers responsible for the television contents, for applying DRM, repacking in RTP<sup>5</sup>, generate PIP<sup>6</sup>, and deliver the contents. They are also responsible for performing streaming multicast for the live TV channels to the entire network and to encode the channels according to a *boundary key* that arrives from the acquisition controllers. In summary, the acquisition subsystem is an installation of hardware and software modules that take live MPEG<sup>7</sup> transport streams as TV data feeds and converts them into RTP streams.

---

<sup>5</sup> Real-time Transport Protocol (RTP) defines a standardized packet format for delivering audio and video over IP networks

<sup>6</sup> Picture-in-picture (Pip) is a lower-resolution, lower-bit-rate, video-only version of a live TV or VOD service that can be used to preview a channel other than the one that the subscriber is currently viewing.

<sup>7</sup> MPEG is a standard method for video compression and audio data compression which permit storage and transmission of movies using currently available storage media and transmission bandwidth

**Branch** is a block of servers composed by a group of servers where the most important for our study are the Client Facing, Server Facing Application, Client Gateways and the Terminal Server servers.

**Client Facing servers** (CFAs) contain the public Web Services from the infrastructure. This group of servers is responsible for sending notifications, authenticating customers, triggering software upgrades of the set-top boxes and the live TV service maps.

**Server Facing Application** (SFAs) contain the internal Web Services platform from the infrastructure. This group of machines is responsible for the management of the Terminal-Servers and interacts with the OSS and BSS layer. It is also responsible, along with the Client Gateways, for the notifications.

**Client Gateways** (CGWs) servers work like a router between the clients and servers from MS IPTV platform. They forward the requests of clients to servers and send notifications such as new versions of EPG<sup>8</sup> and updates for the costumers channel maps.

**Terminal Server** (TSRV) performs RDP<sup>9</sup> functions for applications that customers can access, for example, the “buy channels” menu.

**Local office block** is a block of servers composed by a group of servers where the most important for our study are the Distribution and Video on demand servers.

**Distribution Servers** (D servers) have the function to support the functionality of Instant Channel Change (ICC) and Reliable UDP (RUDP) to services of live TV. Distribution servers also buffer channels (about 10 seconds) and make unicast streaming (RTP / UDP) to customers.

**Video on demand** (VOD) servers have the function to support the functionality of Video-On-Demand. These servers store the content ROV (movies, trailer and posters) and they perform streaming unicast (TCP / HTTP / RTP) directly to the client.

**Monitoring systems block** is a group of servers, responsible for analyzing the alarms received from the different MS IPTV systems. It uses a Microsoft SQL Server engine and network agents that retrieve all the relevant information from the servers.

---

<sup>8</sup> EPG - Electronic program guides

<sup>9</sup> RDP - Remote Desktop Protocol

### 2.2.2 MS IPTV identified threats

Security for the MS IPTV service will rely on the security of the underlying infrastructure. Users and content providers are now exposed to the same security problems faced by any computer user and any Internet service. If a computer worm infects a home PC, then this self-replicating program will try to infect the STB and any other equipment connected, including the head-end equipment. In a sense, there is a logical path between the Internet and the head end. In the past, there was few ways an attacker could disrupt standard TV services. With MS IPTV, that situation is a distant possibility, but a possibility nevertheless. Previously industry practices with the telecom industry show that home-end equipment tends to be configured with minimum-security levels including using a standard password for all equipment or leaving the default values set by the hardware vendor. While this facilitates installation and maintenance, it also creates several security problems, for example the *unauthorized access to elements*.

We will present some of the general MS IPTV infrastructure security threats identified in [2]:

**Denial of service (DoS)** - Attackers could exploit known security problems with the applications or stack implementations, to crash a particular system. Another known approach for DoS is bandwidth consumption or resource exhaustion. Attackers will send a very large number of requests, causing either the networking equipment or the application to stop working due to excessive load.

**Operating system vulnerabilities** - All operating systems have security vulnerabilities that can be used one way or the other to affect the operation of the service, especially in an infrastructure where all the updates have to be tested in Alcatel-Lucent labs, before released.

**Application attacks** - MS IPTV market has several applications currently used by content providers, attackers might discover security problems with the applications and cause problems with the availability of the service.

**Theft or abuse of MS IPTV assets<sup>10</sup>** - something that is possible due to the packet capture and analysis on the home network and IP subnet that will enable the attackers to circumvent the CAS (conditional access systems) to enable access to all contents.

---

<sup>10</sup>Assets - films, tv programs, advertisements...

## 2.3 Monitoring and Log management systems

It's getting harder to sit back and relax relying on passive alerting mechanisms, mainly when dealing with something like the television services, where clients expect providers to actively defend their assets using counter threat operations, better known as *CTO*, to provide a service that is "always on" with reliable quality levels. Typically, log management systems, or *LM*, are the default solution implemented to perform these complex event-processing tasks, or *CEP* [24], but other specific tools can be used.

We will present some of these tools, that process or filter many of the events that are happening across the relevant systems, identifying, aggregating and correlating the most important ones, so real-time actions can be triggered. The default solution, log management, focus on the collection and storage of log files, audit records, audit trails, event-logs, and others, from different types of systems and normally is not concerned with the analysis or content of the data. *SIEM*<sup>11</sup> solutions normally are. What we discovered in practice is that, many log management solutions provide some tools for analysis but rarely reach the level of completeness achieved by *SIEM*. Log management deals with large volumes of log messages, collecting, aggregating and performing long-term retention tasks, but they neglect security analysis. Currently, companies' infrastructures depend more and more on the security information event managers (*SIEMs*), because these tools provide a solution to centralize the storage and interpretation of logs and events with the intention to provide real-time analysis of security alerts generated by other software performing security event manager or *SEM*, in conjunction with the security information managers or *SIM*. The tools that include hardware/software capable of presenting information from multiple sources of security-relevant data, allow companies to react faster, improve efficiency and achieve the ultimate goal of automate compliance. Many are the solutions that a company can choose; open source *SIEMs* that work individually or others that integrate open source tools, research or academic solutions and proprietary solutions. In the next section, we will present an overview of those that we consider relevant for our thesis.

### 2.3.1 OSSIM - AlienVault Unified SIEM

OSSIM, Open Source Security Information Management is designated one of the world's most widely used SIEM [1]. The open source features, and the compilation of different tools

---

<sup>11</sup> SIEM - real-time analysis of security alerts generated by network hardware and applications

associated to the fact that it uses a strong correlation engine, greatly contributed to this fact. As already mentioned the main goal of this software is correlation<sup>12</sup>, thereby providing the opportunity to monitor processes, prioritize events according to their occurrence, detect any threat to the system and finally monitor any security events within the network. AlienVault Unified SIEM is based on OSSIM tool that includes a wide range of security measures, combined together to work as part of a fully functional system. This is accomplished with the distributed architecture shown in Figure 4.

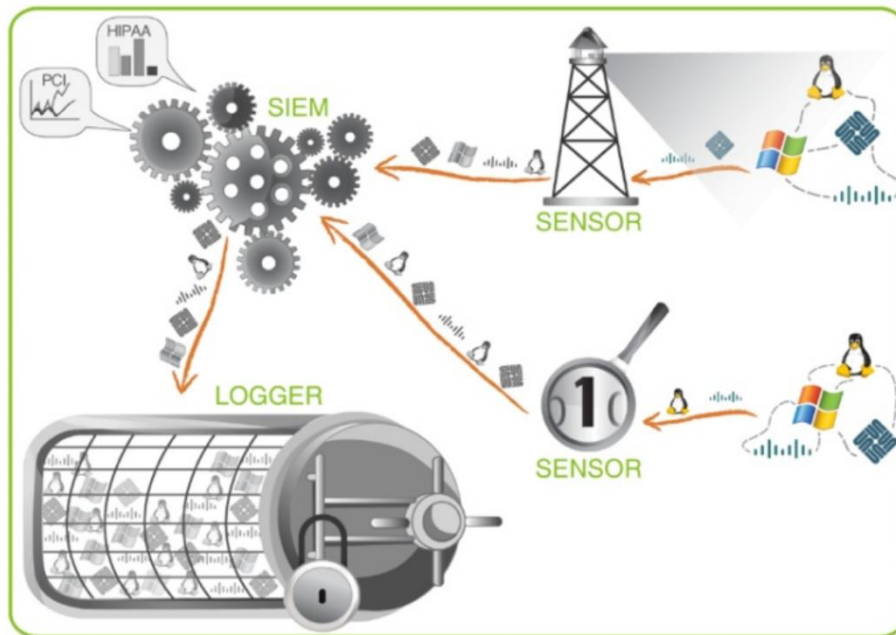


Figure 4 – AlienVault architecture

We will present a brief description of the three AlienVault architecture components observed in Figure 4 (Sensor, SIEM and Logger).

**Sensors** perform the agents role and are responsible for managing the security, in other words to detect and collect information about security incidents. These agents or “sensors” must be distributed throughout the infra-structure depending on the type and volume of information that we intend to collect and analyse.

**SIEM** component provides the tool with the security intelligence and data mining capacities, what enables it to perform risk assessment, correlation, risk metrics, vulnerability scanning, data mining for events and real-time monitoring.

---

<sup>12</sup>Correlation - being able to see every event happening in every system at once in the same form

AlienVault SIEM uses a SQL database to store normalized information, what allows strong analysis and data mining capacities.

**Logger** is the component used to address security. It stores events in raw format in the file system that is later used for forensic purposes.

In summary this tool was optimized for high performance and scalability allowing the correlation of millions of events per day. It uses digital signatures to ensure data integrity and, the *ten to one* compression saves valuable space, because it interoperates with Storage Area Networks (SANs) and Network Attached Storage (NAS). AlienVault Unified SIEM includes a powerful reporting system which may include both historical and real-time information. Built in report examples are (Availability, Security, Vulnerability Analysis). The dashboard functionality is also present, allowing the customizations that are of interest to the user. The real-time policy manager capabilities allow the creation of exceptions, and easily tunes the system behaviour and it is one of the advantages over the tools that only perform historical analysis. Since it includes *Nagios*<sup>13</sup> availability-monitor, it is capable of checking, displaying and reporting hosts and network availability status (e.g. host availability, service availability, service level, system resource utilization; CPU, memory, network interfaces, disk, services running, operating system metrics, and others). In terms of intrusion detection and prevention (IDS and IPS) AlienVault includes *Snort* [28] the most widely used intrusion detection system and, it can be implemented as a sensor or inline as an *IPS*<sup>14</sup>, a configuration that will allow stopping in real-time the selected attacks. Another ability is the anomaly detectors, or *NBA*<sup>15</sup> feature, that will enable the tool to learn by itself and alert when the statistical behaviour deviates enough from what it has learnt as a normal behaviour. It also installs in each system a host *IDS*<sup>16</sup> that will enable the analysis of logs (e.g. windows event logs) and deploy policies and detection rules (e.g. for detecting Malware).

Since AlienVault unified SIEM is an aggregation of other open source tools it has a broad spectrum of activity, it provides immediate protection to critical assets, including detection, prevention and awareness capabilities needed to deal with the security challenges that companies have to deal with nowadays. Since it integrates more than thirty open source tools,

---

<sup>13</sup> Nagios is a powerful monitoring system that enables organizations to identify and resolve IT infrastructure problems before they affect critical business processes.

<sup>14</sup> IPS is an intrusion prevention system

<sup>15</sup> NBA is a network behavior analysis solution

<sup>16</sup> IDS is an intrusion detection system



AlienVault can be installed through an installer package and runs over Debian GNU/Linux operating system, 32-bit and 64-bit editions.

In MS IPTV particular case, a solution like OSSIM will be hard to implement since it relies in *sensors* that need to be installed in the servers that it will monitor. MS IPTV is a closed platform where no type of software or agents can be deployed, so a *non-intrusive security analysis* will be our best option. This is why SMCS that was specifically developed to work under these circumstances is better tailored to fit the MS IPTV security analysis needs. OSSIM is not able to perform attack pattern & regular expressions analysis, by default, hard configuration and scripting has to be performed.

In the end of this chapter we will provide a subsection with the comparison of the tools and also a summary of the parameters used for perform the comparison.

### 2.3.2 SALSA

SALSA is an approach from the Parallel data Laboratory from Carnegie Mellon University. It is an automated system-log analysis tool, that involves examining the logs to trace control-flow and data-flow execution in a distributed system, and the retrieval of state-machine-like views of the system's execution on each node (19). In other words, SALSA picks up log data as-is and works towards building state-machine views and control and data-flow models of the target system's with the purpose of analyzing problems.

Salsa performs an event-based analysis not looking to logs as a purely sources of events, but imposing additional semantics on the events of interest, to identify durations in which the system is performing a specific activity. To demonstrate the Salsa's approach, logs generated by Google's Hadoop<sup>17</sup> were used. One of the major advantages of the SALSA method is that it can uniquely delineate the start and end of key activities in the logs. SALSA also works at the failure diagnosis level and is able to compute the histogram of the durations of its different states in the derived state-machine view. After the study of the behavior of the durations of failure-free hosts and failure-injected hosts SALSA developers hypothesize that failures could be diagnosed by comparing the probability distributions of the durations, for a given state across hosts, assuming that a failure affects fewer than  $n/2$  hosts in a cluster of  $n$  slave hosts.

SALSA it is not a perfect failure-diagnosis algorithm (something that reaches the true-positive rate of 1.0 at a false-positive rate of 0.0), but presents interesting performance results. It

---

<sup>17</sup> Hadoop is an open-source implementation of Google's Map/Reduce framework

presents different views of log data, which can be useful for a variety of purposes, such as visualization and failure diagnosis and also present some ways to diagnose correlated failures by superimposing the derived data-flow models on the control-flow models.

In short, SALSA examines system logs to derive state-machine views of the system's execution, along with control-flow, data-flow models and related statistics.

In MS IPTV particular case, a solution like SALSA will be hard to implement as a single solution for security analysis, nevertheless, SALSA algorithm after some adaptation to the MS IPTV reality, might be valuable to attest SMCS findings. SALSA will not be able to take SMCS place since it is not able to perform attack pattern & regular expressions analysis.

### 2.3.3 Feedzai

When talking about real-time analysis applications we must talk about FeedZai "Pulse". Feedzai was the first company that was born as a startup of Carnegie Mellon-Portugal program and they already won the *European Prize for Intelligent Companies* on the Smart Entrepreneurship Competition of 2010.

They are the developers of "Pulse", a real-time Business Intelligence<sup>18</sup> product created to deal with huge data volumes that are generated by the modern companies and help them to make quick decisions based on real-time information. Feedzai Pulse mission is to provide a clever and high performance way to automatically discover when critical business indicators have unexpected values when compared with historical baselines. This tool can also be applied to single customers that can use it for example to save energy in their houses. Pulse is able to show the home electric, gas and water consumption in real-time and since it is equipped with prediction algorithms it will not only show home consumption as they happen, but also give some estimations and comparisons with other homes with the same profile. For this reason Feedzai Pulse high-level architecture (HLD), presented in Figure 5, shows why it can be considered a real-time event-driven data analytics system. In the HLD we can observe that the interaction with external operational systems is realized through the input and output adapters. Pulse Kernel also designated from PKernel is the core and it is running the event-processing engine, upon it, two modules appear the KPI and the Baseline module. The first allows defining Key Performance Indicators, transparently providing the ability to store them in

---

<sup>18</sup> Business intelligence (BI) mainly refers to computer-based techniques used in identifying, extracting,[clarification needed] and analyzing business data

any database, and the second allows defining reference values to which real-time KPIs are compared.

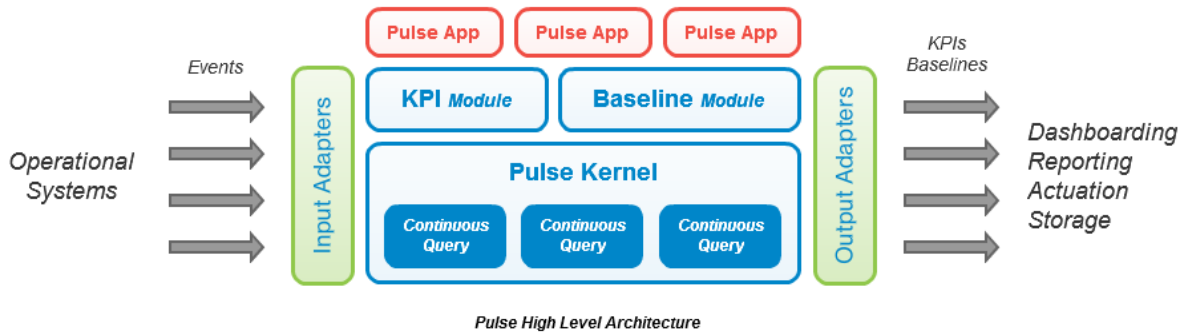


Figure 5– HLD Feedzai Pulse

**Architecture**

Feedzai pulse architecture is composed by different modules that operate in the workflow presented in Figure 6. To better, understand the architecture the modules will be explained using the numbers one to five.

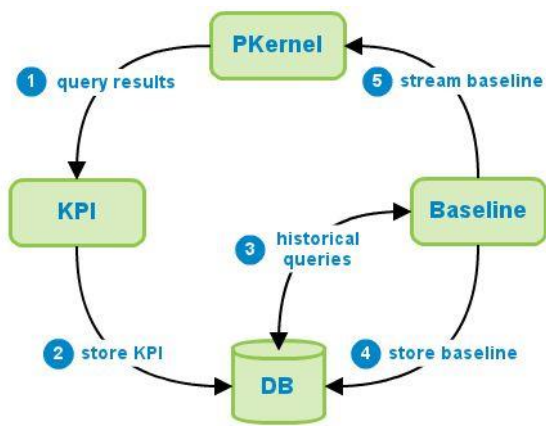


Figure 6 - FeedZai architecture

The KPI (Key Performance Indicator) module, based on KPI definitions, is responsible for submitting the continuous queries into the PKernel engine (data processing kernel).

- 1) KPI module is notified every time new results are produced,
- 2) KPI module validates the results and stores them on the database,

3) The baseline engine, given its baseline definitions, will be responsible for scheduling queries over the KPIs stored in the database,

4) and 5) The baseline engine will validate the query results and not only store them in the database, something that happens for historical purposes, but it will forward them into the PKernel.

### ***Security analysis***

Feedzai “Pulse” is a product that can be integrated in the *SIEMs* group since it allows companies to continuously monitor their businesses, reacting to new information proactively and in real-time. It is able to perform a security real-time analysis, because when an event arrives at the Event Processing engine, it is analyzed by a series of procedures that generate a result. In summary *Pulse* from Feedzai, is a tool that performs complex event processing analysis and integrates real-time data with historical information from huge data volumes.

In MS IPTV particular case, a solution like Pulse will be possible to implement. Nevertheless, the results that Pulse provides are out of the SMCS security analysis scope and, in the other hand, FeedZai Pulse is not able to perform attack pattern and regular expressions analysis.

### **2.3.4 SCOM**

Microsoft chose System Center Operations Manager (SCOM) as the monitoring system to aggregate all the events generated in the different application components of MS IPTV, and issue alarms based on the pre-defined management pack rules<sup>19</sup>. Nevertheless, all the configuration and fine-tuning depend on the client infrastructure, which means that a lot of the customization and authoring, must be done before SCOM is fully operational.

Since MS IPTV is a new technology based on old technology, (e.g. Microsoft SQL<sup>20</sup> Server is used for the databases and Internet Information Services for the web services). They were all bundled in a new concept called Mediaroom or MS IPTV, with the purpose of providing smart TV services. The question here is: can these services be monitored like independent services? On the other hand, does it make sense to analyze the entire infrastructure, knowing à priori that some single services can jeopardize the entire system?

---

<sup>19</sup> Management pack is a set of filtering rules specific to some monitored application.

<sup>20</sup> SQL - commonly expanded to Structured Query Language is a popular computer language used to create, modify, retrieve and manipulate data from relational database management systems

SCOM provides a specific management pack for MS IPTV, but in what concerns security it is not enough. The SCOM module that is more likely to be used in a security analysis is the Microsoft Audit Collection Services (ACS). This module can identify security threats (e.g. hacking and viral activity), auditors and security officers can look for the misuse for regulatory compliance and administrators can track users' activity (e.g. account lockouts).

Nevertheless, SCOM has no built in methodology that enables it to detect attacks, something that, most likely, has to be developed by third party vendors.

### 2.3.5 Pulso

*Pulso* is a *Portugal Telecom* in-house monitoring solution, started to be developed in 2003, which allows the different groups within the company to have the same monitoring tools and indexes.

The *Pulso* platform was initially developed to address the lack of information about the QoS<sup>21</sup> of the internal and critical servers of *Portugal Telecom*. At that time there was the IBM Tivoli solution, because IBM was the company OMG<sup>22</sup> outsourcer, so having a proprietary view, a second vision over the servers QoS, was essential. *Pulso* was then developed based on some open source components and philosophy. Its architecture is made by a central repository and a network of distributed agents, developed in Perl, that gather and forward informational data to the processing and correlation engine. At the present moment *Pulso* has no built-in process to track security breaches, systems OMG quality or the QoM<sup>23</sup>. It can only assure the QoP<sup>24</sup> if some tests on demand were performed against the systems. It is a custom made monitoring application, very difficult to extend, but its graphics are very important to allow the visualization of the status of the overall infrastructure and to drill down and verify the status of each server. In case of any problem affecting the QoS index of the application, the drill down permits to identify which server and component is affecting the index. *Pulso* reports are also very important because they are used in more high-level meetings, with administrators and directors, providing aggregated information about the systems health. In the actual monitoring system scenario, *Pulso* can be categorized in the real-time, predictive and reactive categories, since QoS system metrics are collected every minute and database metrics every five minutes.

---

<sup>21</sup> QoS - Quality of Service

<sup>22</sup> - OMG – Operations and management

<sup>23</sup> QoM - Quality of Maintenance

<sup>24</sup> QoP - Quality of Protection

*Pulso* has some predictive algorithms, for instance to predict when a system partition will become full, and reactive because based on the correlation engine rules it opens alarms via email, SMS and HP OpenView console for the technical teams.

About the programming languages used in *Pulso* we can say that system agents are developed in Perl; oracle agents are developed in PL/SQL; internal core components are developed in Java; web applications are developed in Ruby On Rails; glue code between the different components with no need for performance, are developed in Ruby.

A commercial version of this product is in the forge, *Pulso V5*, the first commercial and customer ready product, and very easy to extend with new functionalities. One of the major advantages over the other monitoring products is the price, since *Pulso* is mostly open source. The company aims to make profit selling consulting services. This latest version of the *Pulso* project creates a model of the target infrastructure, based on a Ruby DSL, defining everything about the operating system, application processes, log files, web servers, databases, and others, from the different vendors (UNIX families and Windows). Agents are automatically generated based on that DSL configuration files.

*Pulso* was not developed to perform attack pattern & regular expressions analysis, so SMCS action is of utmost importance to detect place since it is not able to perform.

### **2.3.6 ArcSight SIEM**

ArcSight is a product that is now part of HP Company, and it was positioned by Gartner, Inc. in the Leaders Quadrant of the 2011 Magic Quadrant for SIEM [30]. It was developed with the objective of providing greater security, better access governance, and faster forensic investigations.

This tool was adopted by the operator, *Portugal Telecom*, with the purpose of performing real-time monitoring, but they soon realized that they were misinterpreting the ArcSight concept. This product is a powerful correlation engine and a data collection database that runs smoothly even when large amounts of information is analyzed. This happens provided that the built in categorizations of ArcSight are used, something that was not happening when PT was using it to act as an IDS system. In a second phase, when ArcSight started to be correctly used to gather multiplatform data, PT was capable of unifying logs into one common format (e.g. Cisco Checkpoint, Linux, Microsoft, and other custom-built applications). It is now possible to perform a plain language analysis that eliminates the need for separate reports for every device type, and the powerful reporting capabilities help in great deal the forensic analysis.

ArcSight presents impressive performance numbers, processing up to hundred thousand events per second, compressing and storing up to thirty-five terabytes of logs, and browsing through tree million records in one second. But most of all, the big win is the increase in the response time, since all these centralizing capabilities enable the really fast discovery of the underling events, something that in the end will supply a better security accuracy.

As we can see in Figure 7, a series of agents are part of the product bundle, but if a connector for a certain proprietary solution does not exist, it can be developed and easily integrated in the operations since ArcSight operates in conformity with the syslog<sup>25</sup> standard. The core engine operates over Java and the database uses Oracle. The new version is planned to use a proprietary database system that will influence the performance (up to three times faster) and the price, since the Oracle licenses are currently one of the reasons for the high price of ArcSight [27].

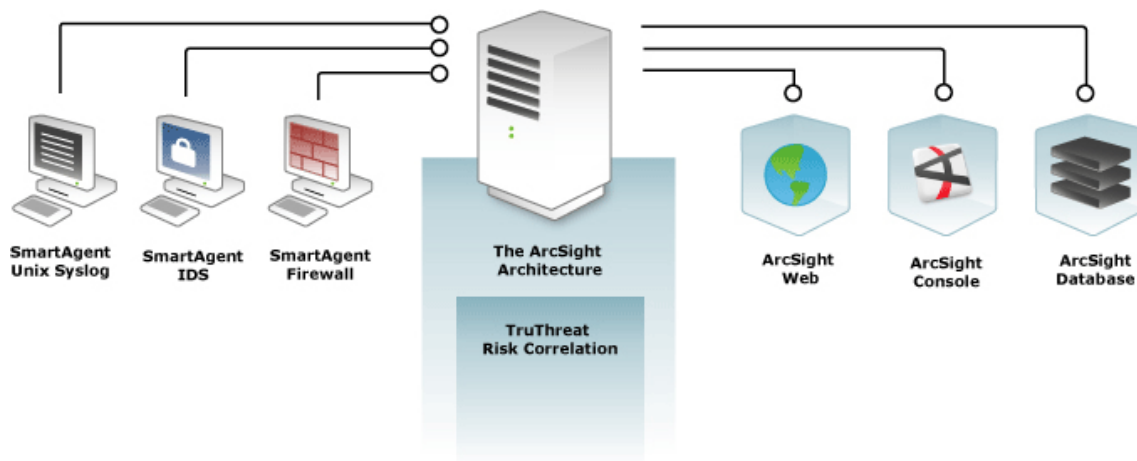


Figure 7 – ArcSight architecture

The ArcSight architecture in Figure 7, presents the Enterprise Security Management (ESM) that is the “brain” of the ArcSight SIEM platform, where the analysis and data correlation is performed. The other components, the smart agents, bring already some built in rules and specifications for certain providers, while the database keeps the results, that are later presented on the consoles.

In terms of requirements, it is a heavy application, which requires a robust server in terms of memory and processor to house the ESM component. Additionally, database server should take into consideration the large amount of disk size necessary to deal with the terabytes of information that ArcSight stores.

---

<sup>25</sup> Syslog is a standard for logging program messages, developed in the 1980s by Eric Allman as part of the Sendmail project.

### 2.3.7 Summary and comparison of tools

Table 3, compares the different tools presented in the previous sections, taking into account several parameters that we considered relevant to the analysis. The parameters used for this comparison were; which *platforms* they were designed to work with, the *event processing* capabilities, what *programming language* they use, what type of *database* they rely on, if they have *built-in* mechanisms to perform *security analysis*, if they incorporate *attack pattern and regex analysis* (this column was important for us to realize that few commercial products have the update and reconfiguration capabilities of the SMCS that we developed), and finally the *type of licensing and pricing*.

Products	Platforms	Event processing	Programming language	Database type	Attack Pattern & Regex analysis	Built-in Security analysis	Pricing and Licensing
OSSIM	Linux	Integrates different tools that can perform real-time and offline event processing	Written in C++ and supports Java and Python scripting languages	MySQL	Not built in, but can be configured	Yes	OSSIM is free of charge to install, certain parts (e.g. Nessus) require a subscription fee of 850€ per year
SALSA	Linux	Aims to target system's logs - performs offline event processing	Java	N/A	No	No	Free university research project
Feedzai	Cross-platform	Specialized in the processing events in real-time, predictive actions	Proprietary language - PulseQL	PostgreSQL	No	Yes	Dual-licensing model: open-source and commercial
SCOM	Cross-platform	Real-time	Proprietary authoring language	Microsoft SQL Server	No	No	Management server 740€, for each agent it can cost from 20€ to 300€
Pulso	Cross-platform	Real-time, predictive and reactive	Agents are developed in Perl, PL/SQL; core components in java	MySQL with InnoDB	No	No	Uses freeware and open source technology
ArcSight	Cross-platform	Real-time and offline event processing	Java	Oracle database	No	Yes	ESM 40k €, and each console or interface 5k€

Table 3 – Tools evaluation



**ArcSight** seems to be the most complete tool to perform security analysis, according to the parameters that we decided to evaluate; nevertheless, it is worthwhile mentioning that it is also the most expensive. After analyzing all these tools, we can conclude that many are the products that companies can use to deal with the problem of the enormous quantity of data that systems are producing and the company needs to analyze. We also realize that none of them is specific for MS IPTV infrastructures, what might compromise the MEO operations and security. This comparative evaluation makes it clear the SCOM Meo Collector Security (SMCS) importance, since it provides built-in security analysis specifically for MS IPTV infrastructures and, very importantly, because it is free of charge and has no license requirements.

## 2.4 Event diagnosis at Portugal Telecom

*Portugal Telecom* (PT) already has systems, implemented with the objective to gather and present information about system health, but they are not enough in terms of security analysis. SMCS has no intention to override them, but to complement their information in terms of identifying security breaches. We present two examples of systems, already in production at PT.

First, System Center Operations Manager (SCOM) that deals with the internal systems events. It assists system administrators to extrapolate about internal systems health, and it is the main MS IPTV alert console. The information provided by SMCS is not provided by SCOM. Therefore, SMCS is complementary. Clearly, it would be very good if all monitoring information could be presented in the same console. For this reason, one of our main goals is to make SMCS security analysis visible in the SCOM console.

Second, *Portugal Telecom* has many tools that analyze network traffic. This type of information allows system administrators to detect abnormal behaviors, but they are not able to extrapolate information about the cause of these behaviors. For instance, in situations like the one shown in

Figure 8, it is possible to know that there is some problem related to the unusual observed load, but it is not possible to determine its cause.

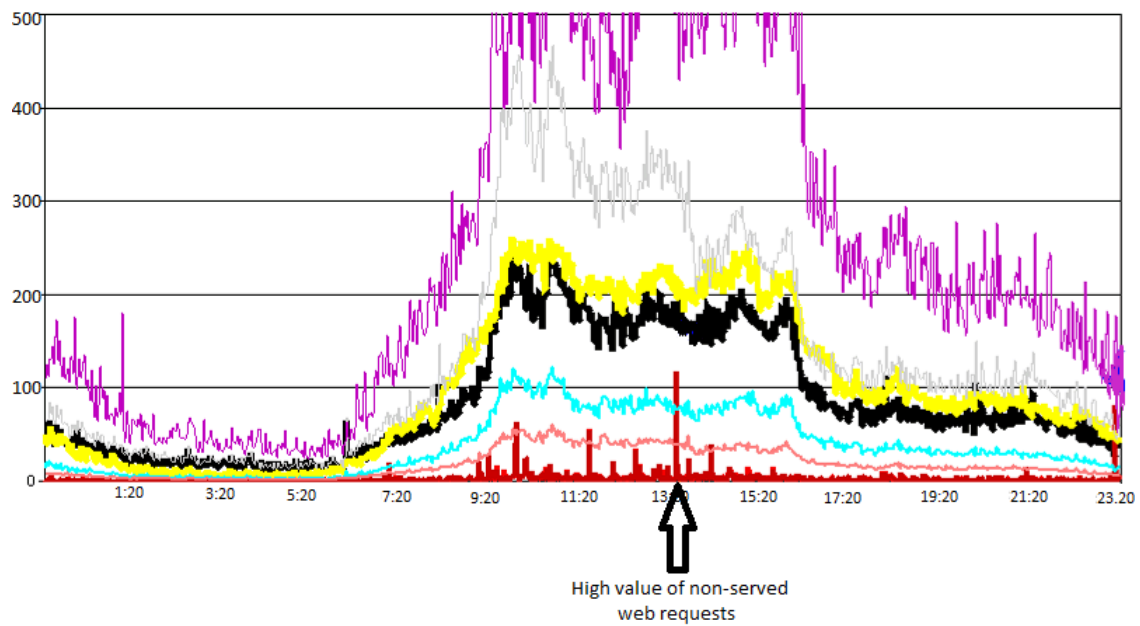


Figure 8 – Analyzing non-served web request

Besides relying on graphs like the one shown in the figure, system administrators will also be able to use information provided by SMCS. Given that SMCS performs a deeper analysis of the log files, hence discovering if and what security breach was the cause of the displayed situation, it provides complementary information to help dealing with the problem.

## 3 Data Set Characterization

In the previous chapter, we provided the context and background of the thesis, and we presented some monitoring and log management systems that base their analysis on the parsing of logs. In this chapter, we will present the reasons that lead us to work with certain data sets, and explain that our approach could be used to any other log data with just a few reconfiguration steps. For this to work, we need the raw material (the log files where we focus our security analysis) to be collected, avoiding the inadvertent mistakes companies usually make concerning log policy. Since some companies tend to jeopardize the way logs are collected, we also refer the common errors regarding log policies.

### 3.1 Internet Information Services data set

At this point, it is important to perform a concise description of the issues that we need to address in terms of security problems in the MS IPTV infrastructure. First, since MS IPTV has an endless number of logging systems it is important to define which are the logs that will give us relevant information about security. Second, given a list of attacks it is important to realize if we can configure the monitoring and log management systems to detect these attacks and develop an algorithm to perform this detection in the most efficient way, always taking in consideration that *SCOM* (System Center Operations Manager) is the operator implemented monitoring system application.

Even though SMCS is not restricted to any type of logs, Internet Information Systems servers (IIS)<sup>26</sup> log files were the inputs used for SMCS. MS IPTV IIS Web server logs are a very important source of information for intrusion detection and in the end an important indicator for the service health, specifically the ones from servers that have external exposed interfaces to the client's network. Despite all, these logs are hard to integrate in a central monitoring system, for example System Center Operations Center (SCOM). One of the goals of the thesis is

---

<sup>26</sup>IIS - Internet Information Services is a web server application created and developed by Microsoft

to retrieve valuable information from the IIS logs that might help to identify attacks or intrusions. All other servers relevant to the MS IPTV operation can benefit with the SMCS analysis. Nevertheless, it is important to notice that SMCS views might not be accurate, since the analysis covers only log data and SMCS has no information about what the systems are actually doing.

### 3.1.1 IIS logging

For all our security analysis process to work, it is important to enable the IIS logging services. This is the first step to prepare for security incidents, logging as much information as possible (by default IIS logging is enabled, but many of the available log fields are not enabled). Enabling full logging will have forensics value in the case of an attack, but on the other hand, system administrators have to be aware of the increase in processing and storage requirements. The fields that are selected to be written in the log should be based on a balance between forensics capabilities and resource allocation.

Information about the log fields and what they can help to identify in terms of *forensic analysis* [9], is shown in Table 4.

IIS Field Name	Description	Used to detect
<b>Date (date)</b>	The date of the request.	Event correlation.
<b>Time (time)</b>	The UTC time of the request.	Event correlation, determine time zone, identify scanning scripts.
<b>Client IP Address (c-ip)</b>	The IP address of the client or proxy that sent the request.	Identify user or proxy server.
<b>User Name (cs-username)</b>	The user name used to authenticate to the resource.	Identify compromised user passwords.
<b>Service Name (s-sitename)</b>	The W3SVC instance number of the site accessed.	Can verify the site accessed if the log files are later moved from the system.
<b>Server Name (s-computername)</b>	The Windows host name assigned to the system that generated the log entry.	Can verify the server accessed if the log files are later moved from the system.
<b>Server IP Address (s-ip)</b>	The IP address that received the request.	Can verify the IP address accessed if the log files are later moved from the system or if the server is moved to a new location.

<b>Server Port (s-port)</b>	The TCP port that received the request.	To verify the port when correlating with other types of log files.
<b>Method (cs-method)</b>	The HTTP method used by the client.	Can help track down abuse of scripts or executables.
<b>URI Stem (cs-uri-stem)</b>	<b>The resource accessed on the server.</b>	<b>Can identify attack vectors.</b>
<b>URI Query (cs-uri-query)</b>	<b>The contents of the query string portion of the URI.</b>	<b>Can identify injection of malicious data.</b>
<b>Protocol Status (sc-status)</b>	The result code sent to the client.	Can identify CGI scans, SQL injection and other intrusions.
<b>Win32 Status (sc-win32-status)</b>	The Win32 error code produced by the request.	Can help identify script abuse.
<b>Bytes Sent (sc-bytes)</b>	The number of bytes sent to the client.	Can help identify unusual traffic from a single script.
<b>Bytes Received (cs-bytes)</b>	The number of bytes received from the client.	Can help identify unusual traffic to a single script.
<b>Time Taken (time-taken)</b>	The amount of server time, in milliseconds, taken to process the request.	Can identify unusual activity from a single script.
<b>Protocol Version (cs-version)</b>	The HTTP protocol version supplied by the client.	Can help identify older scripts or browsers.
<b>Host (cs-host)</b>	The contents of the HTTP Host header sent by the client.	Can determine if the user browsed to the site by IP address or host name.
<b>User Agent (cs(User-Agent))</b>	The contents of the HTTP User-Agent header sent by the client.	Can help uniquely identify users or attack scripts.
<b>Cookie (cs(Cookie))</b>	The contents of the HTTP Cookie header sent by the client.	Can help uniquely identify users.
<b>Referer (cs(Referer))</b>	The contents of the HTTP Referer header sent by the client.	Can help identify the source of an attack or see if an attacker is using search engines to find vulnerable sites.

Table 4 – IIS logging fields

An IIS web log is divided into several fields, as presented in the table, and all can be important when we need to determine if an attacker is trying to compromise our web server. Nevertheless, there are some fields more relevant than others are. For these reason we consider that *URI stem (cs-uri-stem)* and *URI query (cs-uri-query)* should be scanned for security purposes, since in **URI stem** attack vectors can be identified and in **URI query**, malicious data injection can be identified.

The first lines of a web log (Figure 9), describe the server and tell us when this file was generated. Then the important information, the list of the fields in the log file. These fields are the headers of each column. This information is useful to be mentioned, because different web servers (apache, nginx<sup>27</sup>, gws<sup>28</sup>, lighttpd, and others) may record logs in slightly different ways. After that comes the list of requests. A request occurs when a visiting browser says "web service, I need this file and that file." If we look at all those requests together, we will have a moment-by-moment history of every move someone or something makes on a site.

```
#Software: Microsoft Internet Information Services 6.0
#Version: 1.0
#Date: 2011-05-30 00:02:04
#Fields: date time s-sitename s-ip cs-method cs-uri-stem cs-uri-query s-port cs-username c-ip cs(User-Agent) sc-status sc-substatus sc-win32-status
2011-05-30 00:02:04 W3SVC1888260782 10.173.76.5 GET /Default.aspx init=true 443 - 10.219.111.236
Microsoft-IPTV-Client/1.6+(Windows+CE+5.0;+Mediaroom+1.6.25310.38;+MPF+1.1) 403 7 5
2011-05-30 00:02:06 W3SVC1888260782 10.173.76.5 GET /Default.aspx init=true 443 - 10.219.111.236
Microsoft-IPTV-Client/1.6+(Windows+CE+5.0;+Mediaroom+1.6.25310.38;+MPF+1.1) 302 0 0
```

Figure 9 – IIS web log example

By default IIS already logs certain fields, where *URI stem* and *URI query* are included. In our approach, we considered that only the *default logging* options were enabled, something that from our experience corresponds to the reality in of the companies. For example, all the MS IPTV production logs that we observed and studied had only the default options selected.

### 3.1.2 IIS workflow

Since web logs were the elected data source, it is important to understand how the IIS operates at a lower level, since this knowledge is explored in the security analysis performed in the thesis. The IIS workflow puts together the three main stages that IIS has to go through every time it receives and answers a client request. From it, we can understand the

<sup>27</sup> Nginx – vendor Igor Sysoev

<sup>28</sup> Gws – vendor Google

implications of the IIS processing order. Note that data is logged by the IIS before the request is finished. The explanation for this fact is simple: the logs include the total number of bytes sent or the processing time required, and this information is only available when the request is finished. It is also important to realize that the sequence of events are logged based on the time they are finished. If we consider a request *X* that starts at 12:14:01 and ends at 12:14:05 and a request *Y* that starts at 12:14:02 and ends at 12:14:03, we will find the *Y* request logged first then *X* request. This is because request *X* finished after request *Y* what leads us to conclude that the log file sequence is based on the time the request is completed [8].

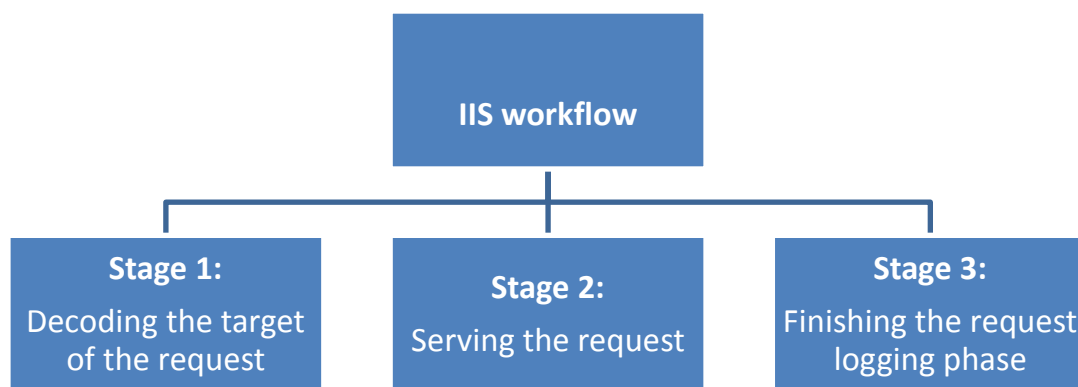


Figure 10 - IIS workflow

At stage1, see Figure 10, the important operation performed is decoding the request. In this phase, the HTTP headers are read and the page or scripts to be called by the web server are identified. For the non-anonymous access, this phase also deals with the authentication. At stage2, IIS checks what kind of processing it must do, which depends on the type of accessed files (e.g. static files, script files, executable files). At stage3, some cleanup work is performed by the IIS and one of the most important operations performed is the logging. At this final stage, all information is available regarding return status of the executed process, the number of bytes sent or the total execution time. Naturally, we chose the end of the last phase to perform the security analysis, because at this point all the valuable information to infer and take conclusions is available.

With this explanation in mind, it is possible to realize why IIS log files are an excellent source of information about attacks and for forensic incident analysis. Unfortunately, these files are stored in plain ANSI text files and IIS workflow itself does not provide any way to forward them to any other system. In this thesis, we will describe SMCS, a helpful tool that we developed to review those logs using a centralized logging infrastructure.

### 3.1.3 IIS HTTP codes

When we try to access content on a server that is running Internet Information Services (IIS) by using HTTP, IIS returns a numeric code that indicates the status of the response. The HTTP status code is recorded in the IIS log. Additionally, the HTTP status code may be displayed in the client browser. The descriptions for most of these codes are self-explanatory. Since we will be working mainly with web logs, Table 5 will be important in terms of understanding the examples presented.

HTTP codes returned by IIS	
Status Code	Condition
1xx	Informational
200	OK. The client request has succeeded.
2xx	Success
3xx	Redirection
4xx	Client error
400	Cannot resolve the request
401.x	Unauthorized
403.x	Forbidden
404.x	File or directory not found.
5xx	Server error

Table 5 – HTTP codes returned by IIS

## 3.2 Common mistakes regarding log policies

From our experience, we notice that some of the most accidental mistakes companies usually make, concern log policy. Logs are produced by default and most companies collect them, but due to the huge volume (number and size), no one looks at them. MS IPTV systems produce many logs that are collected, but in terms of security little information is retrieved from them. In other situations (e.g. network, storage servers, and others), shared solutions (e.g. ArcSight see section 2.3.6), are inspecting logs from perimeter systems, while ignoring the internal MS



IPTV systems. In our work we assume that logs are already being collected into a central repository, but we assume that raw information is provided in terms of security (other systems already look only to “the important stuff”, network logs), but we asked ourselves what about application logs? Ignoring the logs from applications, by only focusing on the perimeter and internal network devices and possibly also servers, but not going “higher up the stack” to look at the application logging is one of the weakness that we intent to overcome with SMCS.

Another common mistake is that the log architecture is often designed before choosing what logs are going to be collected and how security information will be retrieved. A practical example is the SCOM (see section 2.3.4), that Microsoft released with no specific management pack to deal with security issues. We deal with this mistake by building a solution (SMCS) that is able to integrate with the infrastructure that *Portugal Telecom* already uses to monitor the MS IPTV (Microsoft System Center Operations Manager and the centralized MS IPTV logserver).



## 4 Methodology and Test Cases

In the previous chapter, we described the reasons that lead us to work with IIS logs. In this chapter, we will show how we expect to implement our methodology. We start by presenting our strategy for the construction of attack patterns and regular expression, which we will use to search text and data from the millions of log lines of the MS IPTV systems. Then we will present the test cases, identifying the concrete data that is used to demonstrate SMCS capabilities. Finally, we present a section about the already implemented *Portugal Telecom* diagnosis system, which can be used to complement the SMCS security analysis.

### 4.1 Methodology

We now present the strategies used for solving our problem (finding evidences of attacks), the specific components, techniques and procedures (attack patterns and regular expressions). Attack patterns (section 4.1.1), was the first procedure that we used to detect attacks, but as soon as we realized that the regular expression capabilities are enormous (section 4.1.2), we transferred our research efforts into that direction.

#### 4.1.1 Attack patterns

An attack pattern<sup>29</sup> (AP) formally described as an “attack signature”, refers to a special sequence of events that can be distinguished from normal application operation and signal probable instances of specific attacks. For the SMCS operation and detection phases, we assume that an actual, fresh and correct list of attack patterns is provided. An example of attack patterns can be seen in Figure 11.

---

<sup>29</sup> Attack patterns - are a group of expressions that help finding bugs or errors in applications related to computer security

```

null
\0
\00
<script> </script>
<script>
script
nmap
'dir'
\0netstat -a%\n
0xffffffff

```

Figure 11 – List of attack patterns

One of the most trusted repositories of attack patterns is the CAPEC (Common Attack Pattern Enumeration and Classification) that has the objective of providing a publicly available list of AP along with a comprehensive schema and classification taxonomy, and CVE (Common Vulnerabilities and Exposures) a dictionary of publicly known information security vulnerabilities and exposures. The Homeland Security department<sup>30</sup> is the sponsor of both, what helps to increase their credibility. The objective of this initiative is to deploy a public list of AP, which will enable to identify, collect, refine and share the AP among the software community. Both work towards a common objective that is to inform how software may be attacked, because without information it is harder to defend against attacks. They also help to understand the attackers' perspective to software security and this is where AP value became intuitively clear.

One of the potential procedures to deal with the challenge of discovering attacks is the use of attack patterns to detect attacks. APs are often used for testing purposes and are very important for ensuring that potential vulnerabilities are prevented. The AP themselves can be used to highlight areas that need to be taken into consideration for security hardening<sup>31</sup> in a software application. They also provide, either physically or in reference, the common solution pattern for preventing the attack. They help to categorize attacks in a significant way, something that can be used later to discuss problems and solutions. Nevertheless, the patterns will need to be refined until fine-grained results are obtained that are either software vulnerabilities that can be used by the attacker or a backdoors left by the programming team.

---

<sup>30</sup>The US The Department of Homeland Security's Software seeks to reduce software vulnerabilities, minimize exploitation, and address ways to improve the routine development and deployment of trustworthy software products.

<sup>31</sup> Hardening - is the process of securing a system by reducing its surface of vulnerability (e.g. open ports, services, etc.)

### 4.1.2 Regular expressions

Another potential procedure to rise up to this challenge is the use of regular expression (regex) to detect attacks. In this section, we will present the regex concept and their major importance to our thesis, since they can be used in conjunction or to the detriment of the attack patterns presented in the previous section.

Regular expressions, commonly known as regex, constitute a codified method of searching, invented by Stephen Kleene. Regular expressions are made of special text strings that can describe a search pattern. In other words, regex can be compared to “wildcards”. They will be particularly useful in our mission since they will allow us to search text and data from the millions of log lines of the MS IPTV infrastructure, in an expedite way. Using a well-crafted regular expression, we can easily search through a log file (or a million of log lines), for words that match the attack patterns (AP), that we explained in the section 4.1. Nevertheless, regex have some drawbacks since they tend to be easier to write than to read, due to their complex syntax. So, during our work and while implementing SMCS we made sure to “translate into natural language” all used regex (section 5.3). Even with well-constructed regular expressions, there is no guarantee to find all attacks. What we can guarantee is that we will find all attacks whose log traces are matched by the regular expressions.

In order to parse log files in search of log entries that match some of the defined regular expressions or attack patterns we need a search toll. In our framework, SMCS, we used PowerShell<sup>32</sup> which is a scripting language that allows the required searches to be performed.

In summary, SMCS is able to work with both search methods attack patterns and regular expressions, but we recommend the use of regular expressions, since they allow to save space and computational time (e.g. one regular expression might contain hundreds of attacks patterns).

## 4.2 Test cases

When we started to define the test cases, we started with the basics. What is a test case? According to IEEE Standard [3], test cases are defined as “a set of test inputs, execution conditions, and expected results developed for a particular objective”. Therefore, in this

---

<sup>32</sup> PowerShell is Microsoft's task automation framework, consisting of a command-line shell and associated scripting language

section we will present the inputs that we consider and the procedures that we will follow, with the purpose of testing SMCS.

#### 4.2.1 The simulated log

The simulated log is an IIS web log created for SMCS testing purposes only. It is a web log file with evidences of intrusion attempts, so that we can be able to attest if SMCS is working well. In other words, by knowing what kind of attacks have been made and generated evidences in the log file, we can verify if the patterns detected by the regular expressions correctly pinpoint these attacks. If they do, then this means that SMCS did its work as expected.

These intrusions were created with Nessus, Backtrack tools and a Web Server Stress Tool that simulated, among others, a denial of service (DoS) attack against a vulnerable web site. These tools enriched our attack scenario and for this reason, this log has a small size and a high ratio of attacks. We present in Figure 12 a sample of the log file used.

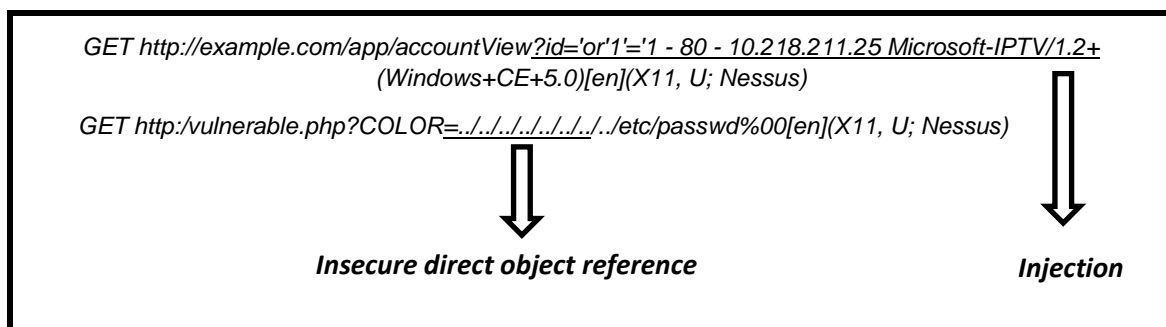


Figure 12 – Simulated log with attack signatures

Looking at an extract of the *simulated log*, we are able to see that Nessus was trying to find out if our web site was able to resist to SQL injection and Insecure Direct Object References attacks. Therefore, we know that at least SMCS has to detect at least these attacks, since they appear in the log file.

With the command from Figure 13, we can retrieve the information that this *simulated log*:

- is a single log file with 76 lines ,
- and it has 12KB of total size.

```
PS C:\tmp> get-content fake.log |measure-object
Count      : 76
```

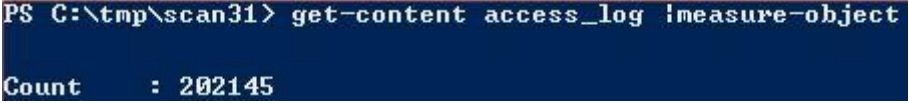
Figure 13 – Number of lines simulated.log

### 4.2.2 Real world logs – Scan31

The real world logs are a collection of logs that we download from the honeypots monitoring and forensics project, *scan of the month*<sup>33</sup> contest [23]. Most of the logs supplied in the *scan of the month* challenges contain real, malicious attacks found in the wild. The purpose of these challenges was to help the security community to develop the forensic and analysis skills to decode real attacks. Therefore, these logs can be used to validate our framework before applying it to real MS IPTV. These scan 31 logs show real world attack signatures, carried out against an apache web server configured as an open proxy<sup>34</sup>, by the Honeynet Project. The concrete logs that we use in our work were part of the challenge called *scan31*, so we called *scan31* to the real world scenario. In summary, these logs are of *utmost importance to our thesis*, since we know for sure they contain different types of attacks, like in the case of the simulated log.

With the command from Figure 14, we retrieved the information that the *scan31* log:

- Is a single log file with 202145 lines,
- and it has 42MB of total size.



```
PS C:\tmp\scan31> get-content access_log |measure-object
Count      : 202145
```

Figure 14 – Number of lines access\_log

### 4.2.3 MS IPTV logs

The available MS IPTV logs that we analyzed comprise logs from the production environment retrieved on the August 28<sup>th</sup> 2011, from the web servers with exposed services to the clients network. When compared with one day logging from MS IPTV, this size could seem small. Nevertheless, the thesis purpose is *to produce a proof* that the chosen approach (SMCS) works, and that we can retrieve security information in bounded time. These logs contain user and company *confidential information*. For this reason, along section 6.3, only non-confidential information (e.g. number of possible attacks found and attack “family”) will be displayed.

---

<sup>33</sup> Scan of the month – These contest had the intention to make competitors to decode obfuscated network traffic, analyze hostile binaries, guess the attacker motives and in the end propose network countermeasures

<sup>34</sup> Proxy - An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients

With the command from Figure 15, we can retrieve the information that these MS IPTV logs correspond to:

- 445 distinct MS IPTV web logs, containing approximately 40 million of log entries,
- which correspond to approximately 10 Gbytes data logs,
- where we do not know *a priori* the attacks.

```
PS C:\tmp\1dayUnzipLog> get-content * |measure-object  
Count      : 39126876
```

Figure 15 – Number of MS IPTV log lines



## 5 SCOM MEO Collector Security

In the previous chapter, we described the methodology, the test cases and data sets that we will use in the security analysis. Along this chapter, we describe the SCOM MEO Collector Security (SMCS). In the first section, we recall the problem statement that motivated us to deal with the security analysis of MS IPTV. Second, we describe the SMCS architecture, the components and their major functions, the evolution phases that it went through, its related problems, and the current solutions. The next section focuses on the operational aspects, as well as on the reasons that made us incorporate the Log Parser tool from Microsoft. Finally, we address how we intend to reduce the false positives rate.

The main question that this thesis sought to answer is: *how relevant can logs be to determine the security of a MS IPTV system?* To answer this question, we do not seek to demonstrate that SMCS detects all the application attacks, but the ones it detects will be illustrative of the useful results that can be achieved.

Since MS IPTV currently has no built in security detection procedures to identify application attacks, developing an efficient parsing methodology will be the target of this thesis. This work intends to aggregate different events, verify if they can have a common cause and then arrive at a high-level approach that can be used to realize what is really affecting the MS IPTV infrastructure. In the end, integrating this approach with the already implemented unified and centralized infrastructure of SCOM, would be the *cherry on top of the cake*.

### 5.1 Problem statement

Let us consider the following simple scenario, of an attacker that is trying to target an *MS IPTV* operator infrastructure. The attacker will gather sufficient information to find vulnerabilities, by doing a ping sweep of the *STB (set-top box)*<sup>35</sup> network followed by port scans. A ping sweep will simply probe each IP address to see which servers are available. Then it could carry out a

---

<sup>35</sup> STB(set-top box) is a device that converts a digital signal into content which is then displayed on the television screen

port scan on each live system with the intention to discover the open port numbers. Now the attacker could simply execute a buffer overflow attack on an open port to get a malicious program to execute. In other words, the attacker could gain administrator access to the server.

These kinds of intrusions are difficult to track and normally the awareness of the attack usually occurs when the attacker is doing something damaging as administrator, therefore, these types of occurrences only get investigated afterwards. Investigating these visible breaches of security is essential, so that there is more information learned than simply that the systems crashed or the network went down. In this scenario, it is well illustrated that the attacker could use the open port on that server as an entry point into the MS IPTV infrastructure, something that he could use to bring down entire operator television services. If no investigation were conducted afterwards, it would not have been known that the abovementioned server was the vulnerable point.

To perform that investigation the following options were available: to use the firewall logs, system logs, or application (e.g. internet information service) logs. We give special attention to the internet information services (IIS) logs, because almost all the external exposed interfaces from MS IPTV, have web services running. Another reason is the increasing popularity that IIS has, among business organizations, something that makes it a popular target among attackers. MS IPTV administrators also have a hard time keeping up a balance between the security patches released for IIS and the product stability. For this reason, it is not hard to find a gap between the recommended and the installed IIS patches, leaving this way an open door for malicious attackers. Attacks performed against IIS web servers are not always evident, since attackers prefer to remain hidden so they can use the server they have attacked, as a launch base for other more severe attacks. This is the reason why it is so important to detect these intrusions and protect the systems against potential attacks, something that we dedicated our efforts to and that lead us to the development of SCOM MEO Collector Security (SMCS).

## **5.2 SMCS Architecture**

During the research phase of the thesis, we had to research and test what type of *log parsing engine* would fit the MS IPTV security needs. Two approaches could be followed (section 5.2.1) to design the SMCS (section 5.2.2). We decided to implement a solution that could be integrated with the infrastructure that *Portugal Telecom* already has (Microsoft System Center Operations Manager and the centralized MS IPTV logserver). Since the approach does not require any modification of the hosted applications, middleware or operating systems it

performs a *non-intrusive security analysis*. We use SCOM agents that are already deployed and reporting events. We also use the logserver that is already collecting and storing all relevant logs from the MS IPTV infrastructure.

Since the security information produced by SMCS will be centralized in the same console already used by the *Portugal Telecom* operations management team, we can say that integration is one of the major advantages of our approach. In the end, it will help to guarantee the success of SMCS, in keeping track of the security threats of the MS IPTV infrastructure, and at the same time following one of the major principles of the company: *systems integration*.

### **5.2.1 On-line vs. Off-line approach**

Initially we intend for SMCS to support an online mode that would analyze the log entries in real time or near real-time. SMCS online mode would analyze the log entries as they were written during system execution, allowing to react more quickly to an attack that was undergoing. One of the issues that could arise using SMCS online mode, would be the runtime overhead that could occur, due to the fact the MS IPTV generates one *terabyte* of logs per day. To implement this online mode the logs needed to be analyzed using an incremental technique.

However, the issues raised by this approach would move us away from the fundamental problem of log analysis, and we considered that as a first step it would be more reasonable to deal only with offline analysis. In offline mode, the analysis can be performed concerning a past log. The SMCS can, for instance, analyze an input log of big dimensions and apply powerful parsing methods with no impact on the production environment. Consequently, this is a considerable slower analysis depending on the size of the log, since the analysis will be performed over the entire log and not only over the incremental information like in the online mode. This mode is useful to evaluate issues that happened in the past, since all data is already collected and kept in a LogServer (repository server).

### **5.2.2 Implementation design**

SMCS has several components that can be executed in parallel or in sequence, as Figure 16 depicts. It also illustrates the data flow for the security event processing analysis.

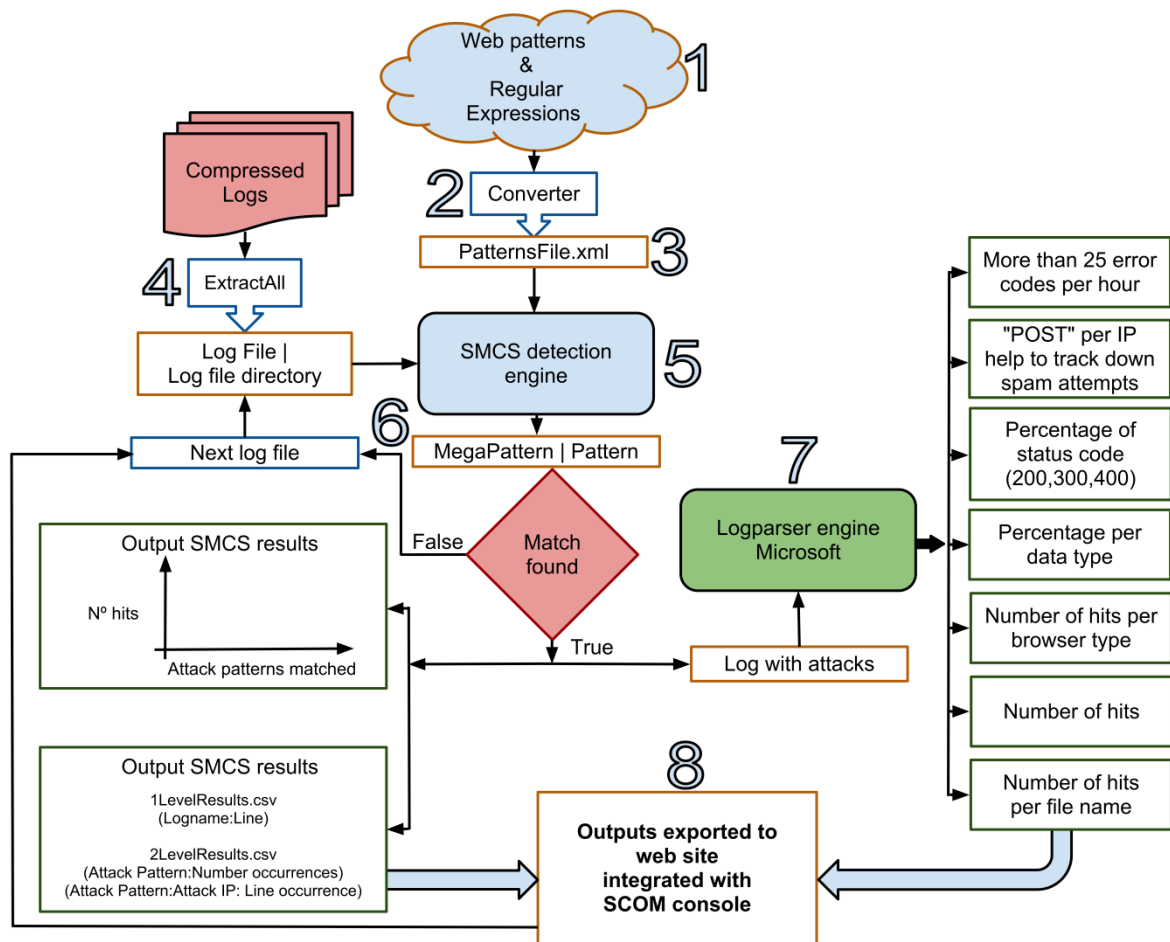


Figure 16 – SCMS implementation design

### Components description:

In order to facilitate the implementation design explanation numbers from 1-8 were used to identify the core SMCS components (Figure 16). The components are the following:

- 1) Web patterns & regular expressions** – are the source material that we use to parse logs in search for attack evidences.
- 2) Converter** – this is a PowerShell script developed to convert new attack patterns, into a format that our parsing engine can use.
- 3) PatternsFile.xml** – this xml file as the web patterns or the regular expressions in the correct format.
- 4) ExtractAll** – this is a PowerShell script developed to deal specifically with the *Portugal Telecom* MS IPTV logs. PT stores the logs in a common LogServer but to save space during the transfer phase they are compressed. We need to perform this extract operation and this script

was optimized to extract all files inside directories and subdirectories. This component was designed to work at the log database level, or another storage server since this is an operation that can run in parallel or before the parsing activity, and also due to the load that it could cause to a production system.

**5) SMCS detection engine** – this block is the center of all our implementation. We developed a parsing script using PowerShell language to read the logs and search for matches of the attack patterns or regular expressions. This parsing script can be applied to any *IIS log* that uses the default settings, but can rapidly be configured to any other type of logs including firewall and network logs.

**6) MegaPattern** – this is the name we gave to the result of the operation of joining all patterns with an 'or' expression. This megapattern is useful to perform the first level search. This is a faster process since the only thing retrieved is information about the log file name and line, of files that contain evidences of potential attacks.

**7) Log Parser** – Microsoft does not officially support this tool and does not really promote its use, but this is a utility that really works and it is the "*Swiss army knife*" of the log parsing utilities available.

**8) Output results** – The SMCS comprises two forms of presenting the detected attacks: chart and logs. The most relevant outputs are then combined into a dynamic web page that can be integrated into the SCOM operational console. We divided the outputs into two main categories: the SMCS detection engine outputs (chart of attack patterns matched versus the number of hits, and *1LevelReport* and *2LevelReport* results), and the Log Parser engine outputs (charts and logs with extra information that help to classify the attack)

## 5.3 Definition of SMCS regular expressions

We took the OWASP top ten of attacks, illustrated in Table 6, as a starting point for defining a set of regular expressions to serve as the basis for the execution of SMCS. However, we note that only with training will we be able to program SMCS to be fully compliant with the specifications of our environment and our business. Therefore, we should evaluate *each security risk by ourselves*, focusing on the threat agents, security controls, and business impacts in our MS IPTV infrastructure. Table 6, list the top ten attacks of OWSAP, which we address in detail in the subsequent sections.

OWASP Top ten	Application Security Risks
<b>A1-Injection</b>	Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
<b>A2-Cross-Site Scripting (XSS)</b>	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser, which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
<b>A3-Broken Authentication and Session Management</b>	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities.
<b>A4-Insecure Direct Object References</b>	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
<b>A5-Cross-Site Request Forgery (CSRF)</b>	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
<b>A6-Security Misconfiguration</b>	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many are not shipped with secure defaults. This includes keeping all software up to date, including all code libraries used by the application.
<b>A7-Insecure Cryptographic Storage</b>	Many web applications do not properly protect sensitive data, such as credit cards, SSNs, and authentication credentials, with appropriate encryption or hashing. Attackers may steal or modify such weakly protected data to conduct identity theft, credit card fraud, or other crimes.
<b>A8-Failure to Restrict URL Access</b>	Many web applications check URL access rights before rendering protected links and buttons. However, applications need to perform similar access control checks each time these pages are accessed, or attackers will be able to forge URLs to access these hidden pages anyway.
<b>A9-Insufficient Transport Layer Protection</b>	Applications frequently fail to authenticate, encrypt, and protect the confidentiality and integrity of sensitive network traffic. When they do, they sometimes support weak algorithms, use expired or invalid certificates, or do not use them correctly.

### A10-Unvalidated Redirects and Forwards

Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Table 6 – OWASP top ten list 2010

In this thesis, we used regular expression that will enable us to detect, A1, A2, A3, A4, A5, A10 and also some other relevant alarms (e.g. brute force password attacks, ReDos, *Code-Red* signature, and others). We will not handle attacks from A6 to A9 from the OWASP table because, A6-Security Misconfiguration is something that has to do with infrastructure configurations, which cannot be detected in the logs. There is also no trace in the web server logs how the sensitive information is stored. For the A7-insecure cryptographic storage and the A8-failure to restrict URL access, mainly because it has to do with applications that are not always protecting page requests properly due to developers that must include the proper code checks, and they forget. Finally, we will not detect A9-Insufficient Transport Layer Protection, since this is out of the scope of SMCS.

In the following subsections, we address the attacks presented in Table 6, referring only to the ones that we considered in the development of the SMCS platform.

### 5.3.1 A1-Injection attacks

This has been considered one of the most devastating attacks ever (OWASP 2010 number one), because of its dynamic and of flexible deployment. Injection attacks are similar to an *“armored tank”* that it will run over the opponent, injecting code that enables it to propagate until the opponent is destroyed.

#### 5.3.1.1 Example scenarios

In this example scenario, we demonstrate how an attacker might be able to perform an injection attack.

If the IIS application uses untrusted data (data from a HTTP request that is not sanitized<sup>36</sup>) in the construction of the following SQL call, **String query = “SELECT \* from account WHERE customID= ” + request.getParameter(“id”) + ” ”**, the attacker can send an HTTP request

---

<sup>36</sup> Sanitized data – validate the inputs searching for danger characters and removing then

containing whatever value for the “id” parameter (e.g. 'or ' 1='1) and therefore perform arbitrary ( in this case it will try to get all accounts from the database, instead of getting only one account as supposed - we call this a *tautology* attack).

Another example, where the attacker uses the same weakness to invoke special stored procedures, <http://www.IPTV.pt/app/accountView?id=' or '1='1>. This HTTP request will allow him take ownership of the database.

### 5.3.1.2 Regular expressions to detect injection attacks

Regular expressions are almost like a programming language, which allows parsing text. We will present now some of the regular expressions that we define for searching patterns that may indicate the presence of *injection attacks*.

**(?i)exec(\s|\+)+(\s/x)p\w+**

This query tests the keyword required to run the stored or extended procedure (EXEC), it looks for one or more whitespaces, the letter “sp” or “xp” that identify a stored or extended procedure and, finally, one or more alphanumeric or underscore characters to complete the name of the procedure. This regular expression identifies subroutines used to access a relational database system also called stored procedures (e.g. *EXEC sp\_storeprocedure.sp*).

**(\')/(\w%27)/(\-\-)/(\%23)/(\#)**

This query tests the single quote or the double-dash that attackers use for jumping out the original SQL statement, (something that has to be done for the SQL injections to work) and, finally, it looks for the asterisk sign. In all these searches it also looks for the hex equivalent versions<sup>37</sup>. This regular expression will identify occurrences of the type ' ;-- ; #, used for SQL “jumping”.

**(?i)(\%3D)/(\=)/[^\n]\*(\%27)/(\')/(\-\-)/(\%3B)/(\:)**

This query looks for the equal sign followed by zero or more non-newline characters and then the single-quote or a double-dash or the semi-colon. It also looks for their hex equivalent. This regular expression will identify occurrences of the type SQL tautologies *1=1--* or script in IMG tags *<IMG SRC=javascript:alert('SRC')>*.

---

<sup>37</sup>Hex equivalent - is a technique used to encode certain characters or strings within the URL.



**(?i)\w\*(\%27)(\')(s|+|\%20)\*(\%6F)/o/(\%4F)(\%72)/r/(\%52)**

This regex searches for the single-quote followed by the SQL keyword 'or'. This regex starts by looking for zero or more alphanumeric or underscore characters, then it looks for zero and one or more white spaces, then for the single-quote and finally for the word 'or' upper or lower case. In all the above mentioned searches it also looks for the hex equivalent. This regular expression will identify SQL tautology that might look like *1' or '2'='2*

**(?i)(\%27)(\')(select|union|insert|update|delete|replace|truncate)**

Since not only the keyword 'or' is commonly used in attacks, but other keywords such as UNION are used to combine the result from multiple SELECT statements into a single result set, the above regex looks for the single quote and the SQL keywords. This regular expression will identify occurrences like *'delete', 'union,* and others.

**(?i)(\|/%00/system\(|eval\(|'\|\\)**

This regex looks for the pipe used in commands for putting the *stdout*<sup>38</sup> of one program into the *stdin*<sup>39</sup> of another and this can be abused to execute another command. Then it looks for the null character (detects poisoned NULL byte attacks %00). Then it looks for the *system( )* or *eval( )* functions that execute an external program in certain languages like Perl and PHP. Function *eval( )* is an useful function that evaluates a string as another language code. Finally, it looks the backtick operator or for the backslash that can be used as escaping characters. It also looks for the hex equivalent. This regular expression will identify occurrences like *system( )* or *eval( )*.

### 5.3.2 A2-Cross-Site Scripting attacks

Cross-Site Scripting (XSS) was during the year 2010 and according to the OWASP, the second most dangerous *application security* risk exploited. However, in terms of *web applications*, XSS was the most prevalent. This flaw occurs when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. There are three known types of XSS flaws: 1) reflected, 2) stored, and 3) DOM based XSS.

---

<sup>38</sup> Stdout - Standard output stream

<sup>39</sup> Stdin - Standard output stream

### **5.3.2.1 Reflected XSS (or Non-Persistent)**

The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser [15].

### **5.3.2.2 Stored XSS (or Persistent)**

The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user.

### **5.3.2.3 DOM-Based XSS**

In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as JavaScript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible. Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit a vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "drive-by hacking." In many cases, the attack can be launched without the victim

even being aware of it. Even with careful users, attackers frequently use a variety of methods to encode the malicious portion of the attack, such as URL encoding or Unicode, so the request looks less suspicious [15].

#### 5.3.2.4 Example scenarios

In this example scenario, we demonstrate how an attacker might be able to perform a Cross-Site Scripting attack. If the IIS application uses the HTML code `<input name='credit card' type='TEXT' value='' + request.getParameter("CC") + "">`, and an attacker is able to provide any value for the "CC" parameter, then we could do something like the following. He could use `'<script>document.location='http://www.XSS.com/cookie.cgi?foo='+doc.cookie</script>'`, that will cause the victim's session ID to be sent to the attacker's website, allowing the attacker to take over the victim's current session.

#### 5.3.2.5 Regular expression to detect XSS attacks

**(?i)(\%3C|<)(\%2F|\/)\*[a-z0-9\%]+(\%3E|>)**

This regex checks for opening angle bracket. Then it looks for the forward slash for a closing tag. Then it looks for the alphanumeric string inside the tag. It finalizes looking for closing angle brackets. In all the above-mentioned searches, it looks for their hex equivalent. This regex will detect attacks of the type `<script>alert('XSS')</script>`.

**(?i)(\%3C|<)(\%69|i|(\%49))(\%6D|m|(\%4D))(\%67|q|(\%47))(\%5C|\\|'|")+(\%3E|>)**

This regex checks for opening angle bracket. Then it looks for the letters 'img' in vary combinations of ASCII. Then for any character other than new line following the `<img` and, finally, for the closing angle bracket. In all the above-mentioned searches, it looks for their hex equivalent. This regular expression will identify occurrences of the type `<img...>`, since XSS attacks may be conducted without using `<script></script>` tags. Other tags will accomplish the exact same effect, for example `<img>; <body>; <b>; <meta>`.

**(?i)(javascript|vbscript|expression|applet|script|embed|object|iframe|frame|framest)**

This regex checks for each keyword inside the brackets. The ("|") denotes an *or* expression. This regex is able to find a wide spectrum of Cross-Site Scripting attacks (e.g. `<script>alert(document.cookie);</script>` or `<iframe src=http://hc.org/ scriptlet.html <`). Nevertheless,

the keywords used need to be carefully chosen to achieve equilibrium between the false positives and the attacks found.

### 5.3.3 A3-Broken Authentication and Session Management

When an attack of this kind is perpetrated, there are two notable evidences in log files. The first evidence shows up if new, preset or invalid session identifiers are accepted from the URL or in the request (in this case this is called a *session fixation attack*). The second occurs if session identifiers or any portion of valid credentials are exposed in URLs or logs. For this reason, a URL string should not use sensitive data like session IDs, user names, and passwords since URLs are stored in cache and it might be possible for other users to access this user's account information.

#### 5.3.3.1 Regular expressions to detect Broken Authentication and Session Management attacks

***(?i)login\?.\*(userId|password)=.***

This regex checks for the *login* followed by a question mark. It also searches for the parameter *userID* and *password*. This regex identify occurrences of the type *https://iptv.net/login?userid=bob&password=4321*.

***(i?);jsessionid=.***

This regex can also be used to search for the *jsessionid* parameter in the URL.

### 5.3.4 A4-Insecure Direct Object References

Since applications sometimes expose internal objects (e.g. files and directories, URLs, database keys and other database objects such as table name), attackers might manipulate unauthorized objects if the proper access controls are not in place. This type of attack can be identified in logs, when certain files names or endings are called, mainly if these are dangerous files (e.g. *passwd*, *shadow* or *cmd*). Another good evidence that we can look for is the dot-dot-slash (*../..*) used to cross directories and perform the so-called, *path traversal attack*.

#### 5.3.4.1 Example scenarios

To illustrate what we have just talked about, consider the following two examples ***%2e%2e%2f*** and ***%252e%252e%255c***, which represent respectively the dot-dot slash in an encoding format.

#### 5.3.4.2 Regular expression to detect Insecure Direct Object References

**(?i)(\.(%|25)2E)(\.(%|25)2E)(\/|(%|25)2F|\\|(%|25)5C)**

This regex can be used to detect encoded requests that use the dot-dot-slash syntax (e.g. *GET /scripts/..%25c../winnt/system32/cmd.exe?/c+dir*).

#### 5.3.5 A5-Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery attacks are one of the most spread attacks in web applications making use of one of the most basic html features, the link. It might be the less well known, but it is an equally dangerous cause of the Cross-Site Scripting (XSS) attack.

##### 5.3.5.1 Example scenarios

If a web site requires links like, *https://www.msiptv.com/tv.php?channel=25&toclient=12*, a CSRF attack might happen. In other words, any application that relies on requests based only on credentials submitted as session cookies is vulnerable to these type of attacks.

There are numerous ways in which an end-user client (set-top box or MEO online client) can be tricked into loading information from, or submitting information to, a web application. In order to execute an attack, we must first understand how an attacker could generate a malicious request for our victim to execute. If we consider the following example (userA wishes to buy a video on demand from an MSIPTV platform and the request generated looks similar to **GET http://example.com/transfer.do?acct=MEO&amount=5 HTTP/1.1**. Then, if a malicious userB decides to exploit this CSRF vulnerability using userA as a victim, he can construct the following URL that will transfer the userA amount to the userB: **<a href="http://example.com/transfer.do?acct= userB&amount =500"> View my Pictures! </a>**. Assuming that userA is authenticated with the service, when he clicks the link that he received by mail or by picture (in the online sharing pictures service), a transfer of five hundred euros will occur to the account of userB. Using this method, userA might realize that a transfer has occurred, so userB might try to hide this attack using for example a *zero byte image*.

##### 5.3.5.2 Regular expression to detect CSRF

**(?i)https?:\\/\\\*.\*\\(login|overview|transfer|amount)\\.**

This regex can be used to detect Cross-Site Request Forgery attacks of the type, **, where an attacker is trying to force an end user to execute unwanted actions on a web application in

which he/she is currently authenticated. However, this regex might also catch legitimate actions, so this regex should only be used in specific situations (e.g. CSRF attack suspicions).

### 5.3.6 A10-Unvalidated Redirects and Forwards

Unvalidated Redirects and Forwards is a threat that might be detectable with SMCS. The issue here is to know for sure what is the name of the *redirect* page, so we are able to build a regex that will detect if an attacker is trying to create a malicious URL that redirects users to a malicious site that performs phishing and installs malware.

#### 5.3.6.1 Example scenarios

The attacker injects into a vulnerable web site, a malicious URL that redirects users to a malicious site *evil.com*, **`http://www.iptv.com/redirect.jsp?=evil.com`**. In this scenario, the application has a page called *redirect.jsp*, which takes a single parameter named *url*.

In another scenario, the application uses *forward* to route requests between different parts of the site. To facilitate this, some pages use a parameter to indicate where the user should be sent if a transaction is successful. In this case, the attacker creates a URL that will pass the application's access control check and then forward the attacker to an administrative function that he/she would not normally be able to access: **`http://www.iptv.com/boring.jsp?fwd=admin.jsp`**

#### 5.3.6.2 Regular expression to detect unvalidated Redirects and Forwards

**`(?i)((\%2F)|\/)(redirect|transfer|fwd|relay)`**

This will detect malicious URLs like the ones presented in the previous example scenarios.

To *prevent* unvalidated redirects or forwards attacks, sometimes it will be enough to avoid using redirects and forwards. But if they are used, it is recommended that any destination parameter be a mapping value, rather than the actual URL or portion of the URL, and that server side code translate this mapping to the target URL [14].

### 5.3.7 Detecting other relevant attacks

Although the top ten OWASP rank gives us the most relevant list of attacks, this is just a small spectrum of the attacks that might be perpetrated against our infrastructure. Most of the attackers do not use deterministic methods, and for this reason the next sections will present some additional techniques that we incorporated into the parsing syntax of SMCS.

**(?i)^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{4,8}\$**

This regex will try to match a string with numbers and letters upper or lower cases with the length of four to eight, the standard requirements for passwords. This will be important if passwords are not encrypted and appear in the log files in clear text, or if an attacker is trying to force an application with http requests that include passwords, this regex will be able to find these occurrences.

**(?i)^http://.\*/default\.ida?**

This regex match the *Code-Red* signature. This worm exploits vulnerability in the indexing software distributed with IIS, and then it will spread itself using a common type of vulnerability known as a buffer overflow.

**(?i)^((4\d{3})|([51-5]\d{2})|(6011)|([7\d{3}]))-?\d{4}-?\d{4}-?\d{4}|3[4,7]\d{13}\$**

This regex will go through the code and find credit card inputs. This regex matches major credit cards including, Visa (length 16, prefix 4), MasterCard (length 16, prefix 51-55), Discover (length 16, prefix 6011), American Express (length 15, prefix 34 or 37). All 16 digit formats accept optional hyphens (-) between each group of four digits [17].

## **5.4 Complementing SMCS with Log Parser**

To complement the information retrieved by SMCS, we looked into some log analysis tools, to understand their potentialities and limitations in terms of security analysis that can leverage the need of innovation in this field. We found that the information retrieved by the *Log Parser* from Microsoft, is helpful in terms of complementing the SMCS, in terms of detecting potential attacks. To dig into web logs looking for evidences of intrusions can be an overwhelming task, especially in MS IPTV infrastructures that can have tens of thousands of clients, which can rapidly generate gigabytes of information logs. To avoid losing precious time trying to locate the suspicious activity Microsoft provides the Log Parser tool. Figure 17, presents in a graphical manner the type of logs that this tool can analyze and the possible output formats.

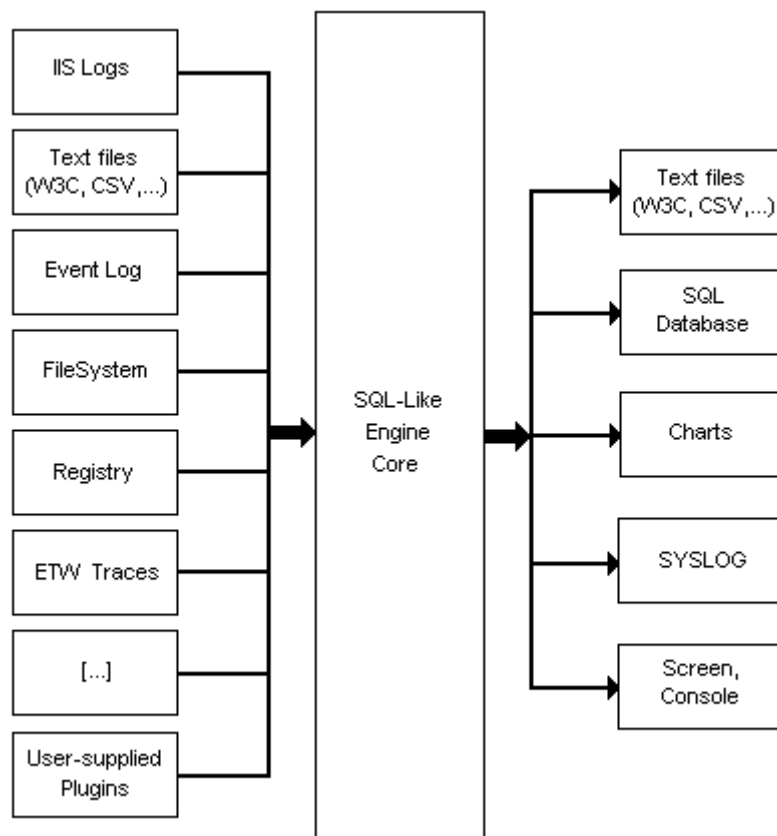


Figure 17 – Log Parser

Log Parser consists of three components that are the *input engine*, which reads IIS logs, text files, and others, the *SQL query engine*, and the *output engine*, which writes text files, charts, and others. When investigating intrusions, we have to analyze logs from many sources, possibly not being compatible with the other. One of the great advantages of Log Parser is that it can accept almost any common log format and output it into one format of our choosing.

Log Parser has a generic application over web logs and *our contribution* will be to develop, using Log Parser semantics, an expedite way to confirm a vulnerability in MS IPTV logs. The combined results (SMCS &Log Parser) will help to complement and substantiate the results obtained only with SMCS.

We will assume that SMCS pointes the compromised logs, where attacks are spotted. Only these logs go through the Log Parser analysis phase, as illustrated in the SMCS workflow of Figure 16.

In the following sections we describe the several queries that we used with Log Parser.



### 5.4.1 Script Abuse

**logparser -rtp:-1 -o:w3c "SELECT c-ip, COUNT(\*) AS [Requests] INTO match.script.log FROM \*\SuspLog\\*. \*WHERE cs-uri-query IS NOT null and STRCNT(TO LOWERCASE(cs-uri-query),'script')> 0 GROUP BY c-ip ORDER BY [Requests] DESC"**

This query will look for the *script* expression in the suspected log and it returns a list of IP addresses that may be making a hacking attempt by passing a site address or a call to a malicious script.

**logparser -rtp:-1 -o:w3c "SELECT c-ip, COUNT(\*) AS [Requests] INTO match.http.log FROM \*\SuspLog\\*. \*WHERE cs-uri-query IS NOT null and STRCNT(TO LOWERCASE(cs-uri-query),'http://')> 0 GROUP BY c-ip ORDER BY [Requests] DESC"**

This query will look for the *http://* expression in the suspected log.

### 5.4.2 Trojan attacks

To spot Trojan attacks using Log Parser from Microsoft, two queries must be executed this time against the web site that we suspect that has suffered the attack. First we use the following query to list the twenty newest files on the web site:

**logparser -i:FS "select top 20 path, CreationTime from \*\SuspSite\\*. \*order by creation time desc" -rtp:-1**

Second, we use a query to list the twenty most recently modified files in the web site. These files should match the list of the files that we expect to find in the web folder:

**logparser -i:FS " select top 20 path, LastWriteTime from \*\SuspSite\\*. \* order by LastWriteTime DESC" -rtp:-1**

If the above queries are not enough, we might consider that the attacker was careful enough to delete all Trojan files when finished. In this case, the files will not exist but there will be log entries showing successful requests for those files. To identify these log entries we use the following query to list all files that have resulted in 200 HTTP status codes.

**logparser "select distinct to lowercase (cs-uri-stem) as url, Count() as Hits from \*.log where sc-status=200 group by URL order by URL" -rtp:-1**

Now we make a careful review of this list and make sure that each item listed is part of the web application. We can be particularly alert for files such as *nc.exe*, *tini.exe*, *root.exe*, *cmd.exe*, *upload.asp*, *aspexec.asp*, and others, that are typically used in trojan attacks.

### 5.4.3 DoS and DDoS attacks

A **Denial of Service** (DoS) attack is a malicious effort to keep authorized users of a website or web service from accessing it, or limiting their ability to do so. A **Distributed Denial of Service** (DDoS) attack is a type of DoS attack in which many client systems (e.g. client set-top boxes), can be used to harm the web based services. A hypothetical form of DDoS, could be performed by a massive number of client set-top boxes used to perform upgrade requests or just requesting any other operation precisely at the same time, overflowing the web servers with requests to the point where they cannot serve legitimate clients. To detect these occurrences we need to analyze web log files, It will be possible to infer that the systems are being targeted by DDoS attacks, looking at *suspicious size, contents and syntax*.

Another form of recognizing a DoS attack will be to search for several repeated requests from one single IP. We could identify these occurrences using the following query:

```
logparser -rtf:-1 "SELECT c-ip, COUNT(*) AS [Requests] INTO spam.txt FROM *.log WHERE cs-method = 'POST' GROUP BY c-ip ORDER BY [Requests] DESC"
```

We can also try to identify the different IPs with similar packet requests, and then inspect the ones with a very high volume. This information can be retrieved from the logs with the following query:

```
LogParser "SELECT TOP 20 cs-uri-stem, COUNT(*) AS Hits.NameFiles INTO hits.name.gif FROM *.log GROUP BY cs-uri-stem ORDER BY Hits.NameFiles DESC" -chartType:Bar3D -groupSize:640x480 -view:off
```

This query, will output a chart with the requested information, that is presented in Figure 18.

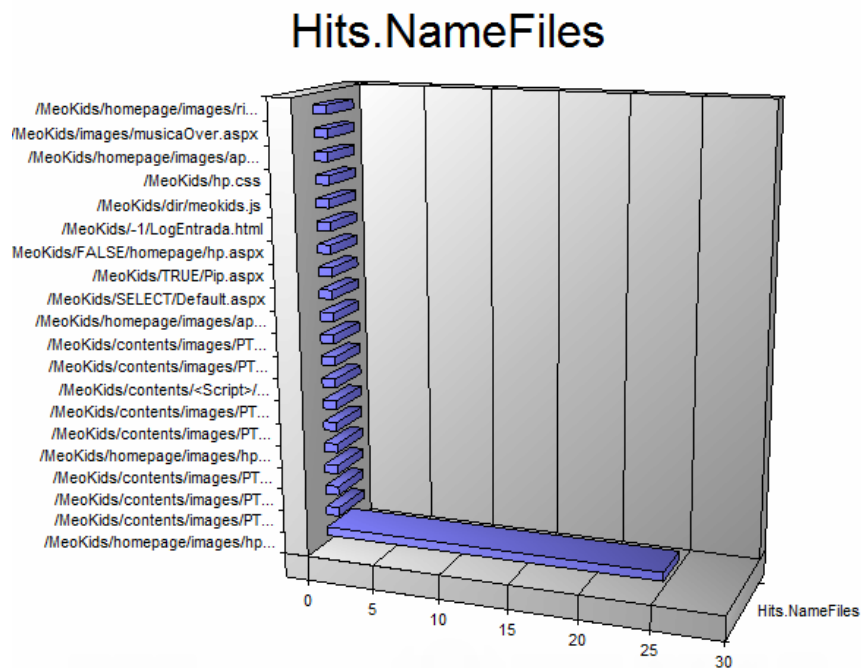


Figure 18 – Number of hits per file name

We can look at the number of browsers accessing our system, since MS IPTV will mainly attend set-top boxes requests (Windows CE operating system), all other operating systems accessing the MS IPTV services should be analyzed.

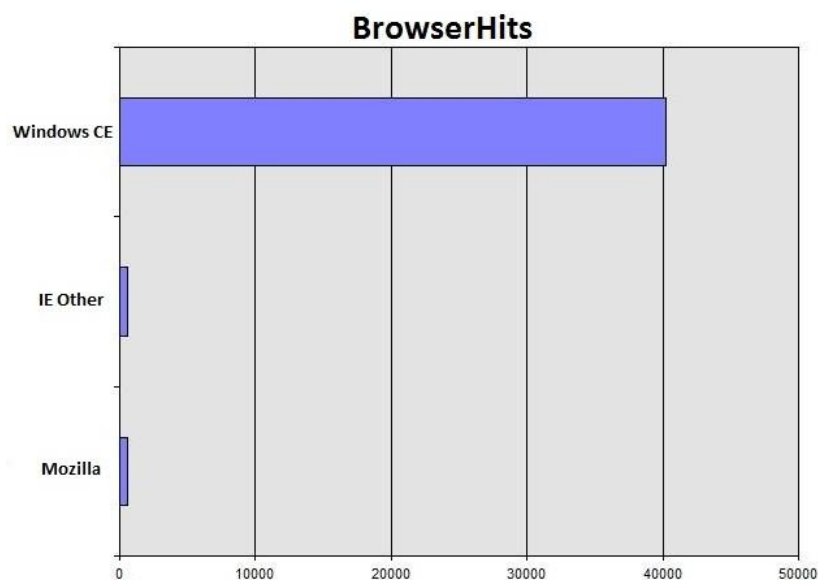


Figure 19 – Log Parser browsers chart

Since the only requests should come from Internet Explorer for Windows CE, which is the default operating system of the MEO clients, this chart is of a particular importance. In the example, of the Figure 19 the browser bar IE and Mozilla might indicate some attempt to

emulate a set-top box. The equipment's running these different browsers might be trying to attack the MS IPTV infrastructure [25].

#### 5.4.4 Aggregating Log Parser outputs

The results of these Log Parser queries are combined into a web page. The intention is to create a *dashboard* view of the compromised logs. This single web page will be useful for integration purposes with the SCOM console.

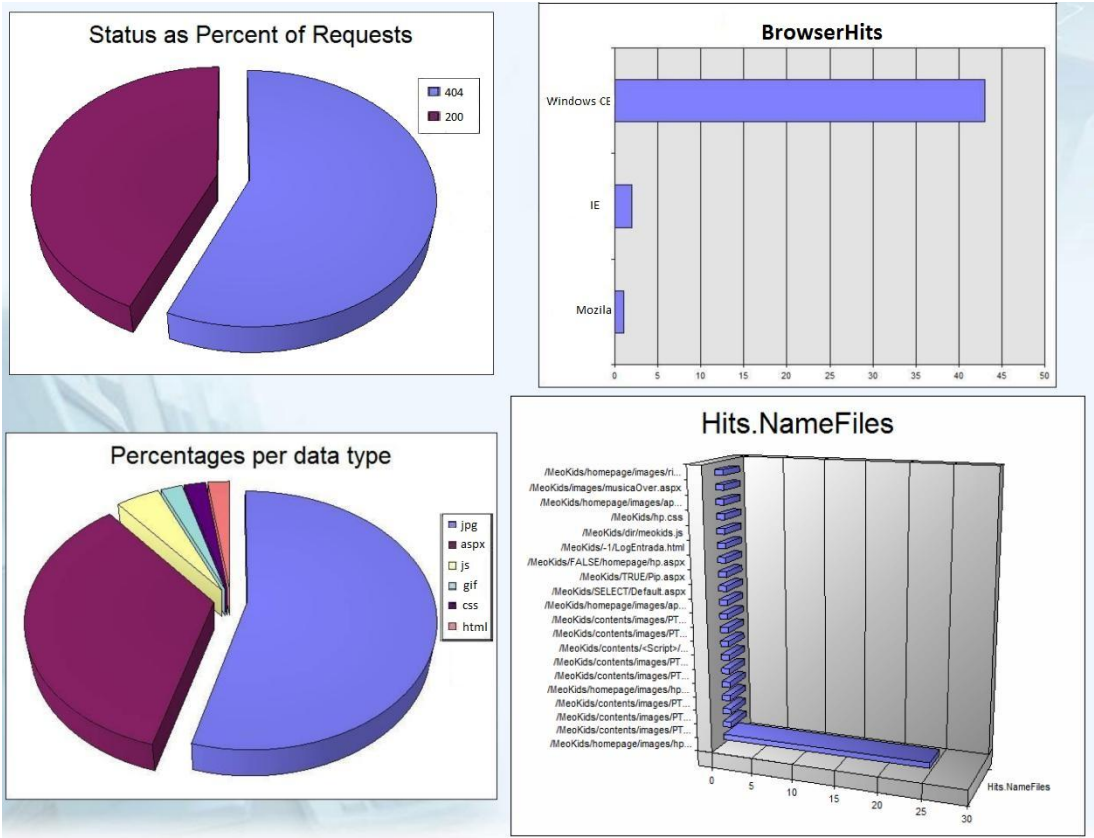


Figure 20 – Log Parser outputs

### 5.5 Reducing false positives

The strategy chosen is one that we believe is inclined towards detection. It will result in a high detection rate at the cost of more false positives. The strategy is to mark all identified malicious requests detected with the regex or with the attack patterns as malicious. Then, analyze the output result searching for all false positives, which will be marked as benign until there are no obvious false positives left. Either this search process could be made manually or using scripts, in our work we have not yet included specific solutions. We left this as future work, as mentioned in section 8.2.

## 6 Results

In the previous chapter, we described the SCOM MEO Collector Security (SMCS) architectural components, and explained how Log Parser tool helps to complement the security information provided. In this chapter, we describe the results obtained when using SMCS to parse the different data sets described in section 4.2. The results show that SMCS is able to detect signs of potential attacks.

### 6.1 Detecting attacks in a simulated test case

The first log that SMCS is going to analyze is the *simulated.log*, which has only 76 lines and 12KB of size. Nevertheless, almost every line has an attack signature. We already described how this log was generated (section 4.2.1), so we will now describe the attacks found by SMCS. The output results shown in Figure 21, reveal the “family” of attacks that were detected in this log, accordingly to the OWASP scale [14]. In this chart, we present the number of attack found per “family” type.

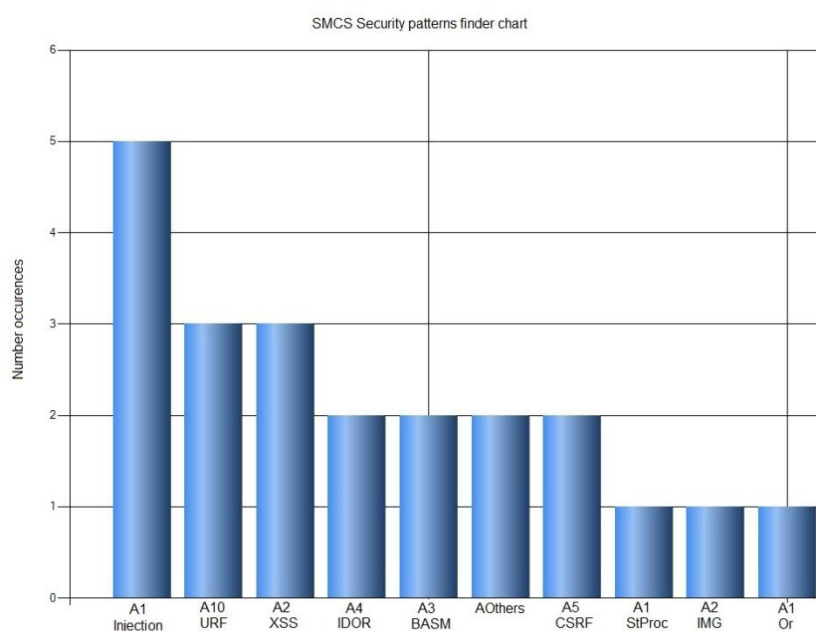


Figure 21 – Simulated scenario output (bar chart)

Examining the chart, we notice that SMCS detected many attacks. This is obviously expected, and is due to the fact that this is a simulated scenario. Recall that this log was generated with real active scanners that performed all these attacks. SMCS is also able to present the same results using another graphical format, a pie chart, as the one presented in the Figure 22. This representation provides a very clear image of how attack families occurrences relate to one another (e.g. A1 “family” of attacks was the one with a higher incidence of occurrences, followed by the A10 and A2 “families”).

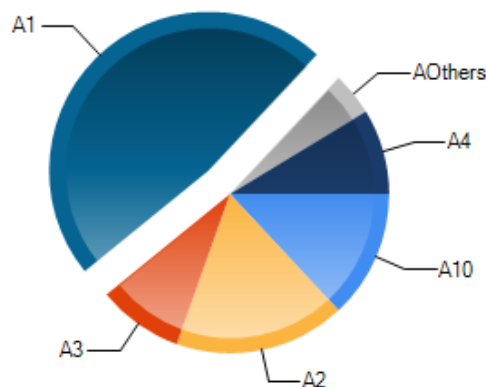


Figure 22 - Simulated scenario output (pie chart)

Now that we have a graphical perspective of the SMCS discovered attacks, we can look into more detailed information, since SMCS is able to generate two levels of *output log reports*, (section 5.2).

The *1LevelReport* output is illustrated in Figure 23. It is the smallest output, but the information that it presents already gives a good idea of which log files shown signs of attacks, and the log line where the attacker left his fingerprint.

Log Text
Possible attack on -> logname:line (simulated.log:3)
Possible attack on -> logname:line (simulated.log:5)
Possible attack on -> logname:line (simulated.log:7)
Possible attack on -> logname:line (simulated.log:8)
Possible attack on -> logname:line (simulated.log:9)
Possible attack on -> logname:line (simulated.log:11)
Possible attack on -> logname:line (simulated.log:12)
Possible attack on -> logname:line (simulated.log:14)
Possible attack on -> logname:line (simulated.log:15)

Figure 23 – Simulated log 1LevelReport output

The *2LevelReport* output is illustrated in Figure 24. This output includes two sections; in the first, we can observe the attack pattern or regex that was matched during the search process

and the number of occurrences. To complement the information we add a second section where more information is provided about the attack pattern or regex matched, followed by the IP address of the attacker and finally the line where the attacker left his fingerprint.

Log Text	
AP or Regex: Number occurrences	
(\?)(\%27)(\)(\)(select union insert update delete replace truncate);1	
(\?)(\%2F)(\)(\)(redirect transfer fwd relay);3	
(\?)(\%3C)(\<)(\%2F)(\)(\)*[a-z0-9\%]+(\%3E)(\>);3	
(\?)(\%3C)(\<)(\%69)(\%49)(\%6D)(\%4D)(\%67)(\%47)(\^n)+(\%3E)(\>);1	
(\?)(\%3D)(\=)(\)[^n]*(\)(\%27)(\-\-)(\%3B)(\);):2	
(\?)(\)(\%27)(\-\-)(\%23)(\#);2	
(\?)(\.(%25)2E)(\.(%25)2E)(\.(%25)2F)(\.(%25)5C);2	
(\?)(\%00)(system\)(eval\)(\)\%5	
(\?)(w*(\%27)(\)(\)(s\+ \%20)*((\%6F)(o)(\%4F)(\%72)(h)(\%52));1	
(\?)(^(?=.*d)(?=.*[a-z])(?=.*[A-Z]).{4,8}\$;1	
(\?)(login\?.*(userid password)=.;1	
(\?)(?);sessionid=.;1	
AP or Regex: Attack ip; Line occurrence	
(\?)(\%27)(\)(\)(select union insert update delete replace truncate);213.13.16.243;3	
(\?)(\%2F)(\)(\)(redirect transfer fwd relay);111.13.16.243;19;111.13.16.243;21;111.13.16.243;25	
(\?)(\%3C)(\<)(\%2F)(\)(\)*[a-z0-9\%]+(\%3E)(\>);111.13.16.243;11;111.13.16.243;12;111.13.16.243;19	
(\?)(\%3C)(\<)(\%69)(\%49)(\%6D)(\%4D)(\%67)(\%47)(\^n)+(\%3E)(\>);111.13.16.243;12	
(\?)(\%3D)(\=)(\)[^n]*(\)(\%27)(\-\-)(\%3B)(\);):111.13.16.243;5;111.13.16.243;15	
(\?)(\)(\%27)(\-\-)(\%23)(\#);213.13.16.243;3;111.13.16.243;5	
(\?)(\.(%25)2E)(\.(%25)2E)(\.(%25)2F)(\.(%25)5C);111.13.16.243;8;111.13.16.243;17	
(\?)(\%00)(system\)(eval\)(\)\%5	
(\?)(w*(\%27)(\)(\)(s\+ \%20)*((\%6F)(o)(\%4F)(\%72)(h)(\%52));111.13.16.243;9;111.13.16.243;11;111.13.16.243;12	
(\?)(^(?=.*d)(?=.*[a-z])(?=.*[A-Z]).{4,8}\$;111.13.16.243;23	
(\?)(login\?.*(userid password)=.;111.13.16.243;14	
(\?)(?);sessionid=.;111.13.16.243;15	

Figure 24 – Simulated log 2LevelReport output

To perform a forensic analysis, the operator can now dig into the correspondent log file or use the *SMCS PowerShell* console (Figure 25) where all the compromised log lines from each log file are displayed. This is another SMCS important output format.

```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\CMU\2Semestre_tese\PS> .\SMCS1.ps1
Possible attack on -> logname:line (simulated.log:3)
A iniciar análise de segundo nível na linha - (n:3) 2011-08-06 11:03:18 W3SVC732792582
-IPTV/1.2+(Windows+CE+5.0) meokids.app.iptv.telecom.pt 200 0 0 62
Possible attack on -> logname:line (simulated.log:5)
A iniciar análise de segundo nível na linha - (n:5) 2011-08-06 11:03:17 W3SVC732792582
ntView?id=or'1'=1 - 80 - 10.218.211.25 Microsoft-IPTV/1.2+(Windows+CE+5.0) meokids.app.iptv.t
(A1)SQLI-http://example.com/app/accountView?id=or'1'=1 -
Possible attack on -> logname:line (simulated.log:7)
A iniciar análise de segundo nível na linha - (n:7) 2011-08-06 11:03:17 W3SVC732792582
p?COLOR=C:\\ftp\\upload\\exploit
Possible attack on -> logname:line (simulated.log:8)
A iniciar análise de segundo nível na linha - (n:8) 2011-08-06 11:03:17 W3SVC732792582
p?COLOR=../../../../../../../../etc/passwd%00
(A1)CodeInjection-http://vulnerable.php?COLOR=../../../../../../../../etc/passwd%00
Possible attack on -> logname:line (simulated.log:9)
A iniciar análise de segundo nível na linha - (n:9) 2011-08-06 11:03:17 W3SVC732792582
p?COLOR=C:\\notes.txt%00
(A1)CodeInjection-http://vulnerable.php?COLOR=C:\\notes.txt%00
Possible attack on -> logname:line (simulated.log:11)
A iniciar análise de segundo nível na linha - (n:11) 2011-08-06 11:03:17 W3SVC732792582
einside</Tag>
(A2)Html.tags-http://[tag]withsomecodeinside</tag>
```

Figure 25 – PowerShell SMCS console

In summary, SMCS was able to detected attacks in the simulated log, specifically attacks from the top ten OWASP project (A1, A2, A3, A4, A5, A10), and also other evidences of possible attacks (*e.g. code-red signatures, password and credit card abuse*). This first analysis was important for two reasons. First, to validate if our methodology to detect attacks works and secondly, this simulated log is important in terms of testing the attack patterns and regular expressions that we developed, in terms of their accuracy to detect attacks.

With the intention to determine the degree of credibility that we can achieve with the SMCS security analysis, we compared the SMCS outputs with the existing attacks in the simulated log.

SMCS results		Simulated log	
Attacks type	Nº of occurrences	Attacks type	Nº of occurrences
A1-Injection	7	A1-Injection	7
A2-XSS	4	A2-XSS	4
A10-URF	3	A10-URF	3
A4 -IDOR	2	A4 -IDOR	2
A5-CSRF	2	A5-CSRF	2
A3-BASM	2	A3-BASM	2
AOthers	2	AOthers	2
Total	22	Total	22

Table 7 – SMCS results and simulated log inputted attacks

According to this table, SMCS obtained excellent results. SMCS detected all the attacks that real active scanners performed against this simulated scenario. This is obviously expected, and is due to the fact that this is a scenario built for testing purposes, where we knew the attacks that were performed against the vulnerable web site.



## 6.2 Detecting real world attacks

To detect real world attacks we used *scan31*, one of the many “Scan of the Month” challenges from the Honeynet project [23]. We choose specifically this scan, because it includes a web log that contains some signs of abuse. The important aspect of this security analysis is that we *know a priori* that this log have a considerable number of attack signatures. Our mission will be to identify/classify them, using SMCS. In the following charts, we present the number of attacks found per “family” type.

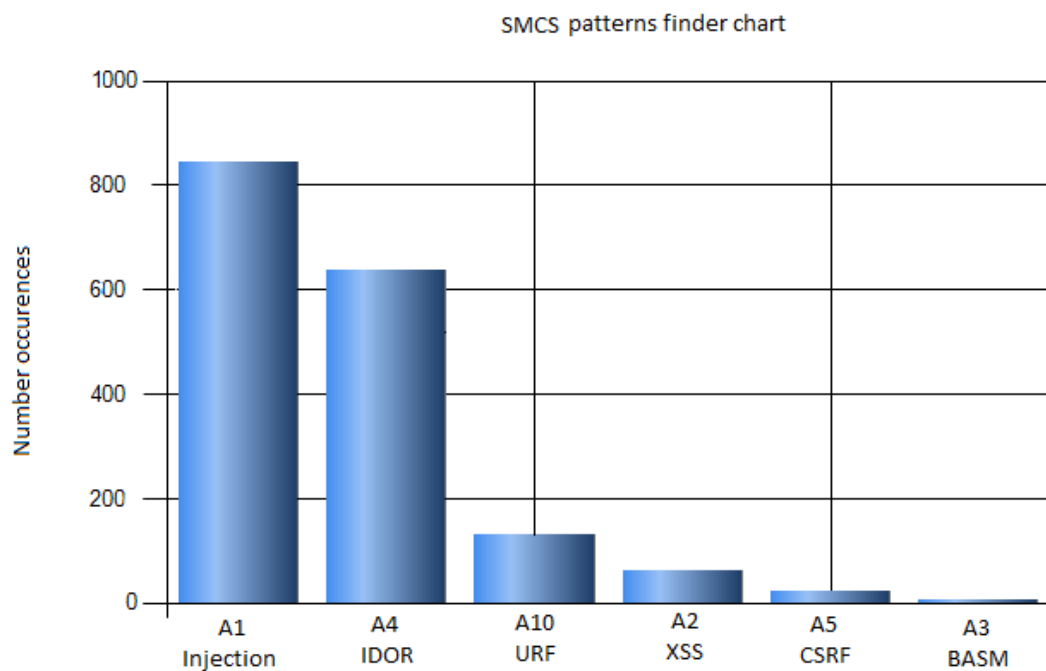


Figure 26 – SMCS output chart for the real world scenario

All the attack types detected with SMCS in the *scan31* are also illustrated in the Figure 27, another graphical representation that provides a very clear image of how attack families occurrences relate to one another.

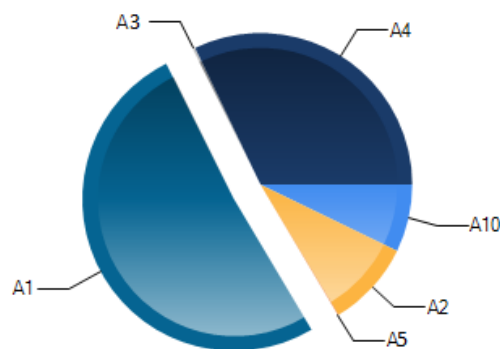


Figure 27 – SMCS attacks detected in the *scan31* (pie)

We now present a table where we put side by side the results obtained with the SMCS *1LevelReport* and *2LevelReport* and the *Honeynet solutions* for *scan31* challenge.

SMCS results		Honeynet solutions	
Attacks type	Nº of occurrences	Attacks type	Nº of occurrences
A1-Injection	830	Nessus	733
A4 -IDOR	623	CGI abuse attacks	567
A10-URF	114	Misc attacks	82
A2-XSS	41	Remote file access attacks	54
A5-CSRF	24	Firewall attacks	21
A3-BASM	5	Firewall attacks	7
Total	1629	Total	1464

Table 8 - SMCS results and Honeynet solutions

In Table 8, we must take into consideration that we are using the OWASP scale to categorize the attacks and the Honeynet solutions uses a different categorization schema. Even though the values are not identical, we can conclude that SMCS is able to detect the “family of attacks” described in the Honeynet solutions.

Many evidences of potential attacks were found and since it would not be practical to describe all of them, we pick some arbitrary examples and gather them on Table 9. This table presents the log line identified as containing the attack, the solution provided by the Honeynet with the attack type and the SMCS regex “family” that detected the potential attack.

Scan31 log line	Honeynet solutions	SMCS regex family	Solutions match
"GET /class/mysql.class HTTP/1.1"	This is a SQL <b>injection</b> attack	Detected with the <b>A1</b> injection regex	Yes
GET /%00/ HTTP/1.1" 404 280 "- " "Mozilla/4.75 [en]	This is a poison with <b>NULL byte</b> attacks	<b>A1</b> -injection attacks (Null byte regex)	Yes

"GET http://us.edit.companion.yahoo. com /config /slv4_done?.src= ym&.act=4&.intl= us&.partner= &.region= &.dlsrc=ym&.done http://f600.mail.yahoo.com<SCR IPT%20language=JScript>	<b>XSS</b> is used in a web application to maliciously gather data from a user	<b>A2</b> -Detected with XSS regex	Yes
GEThttp://edit.yahoo.com/config ?login'=german_1975ar&passwd =german&.version	<b>Brute force</b> password attack	<b>A3</b> - Detected with BASM regex	Yes
GET/_vti_bin/..%255c../..%255c.. /..%255c../winnt/system32 /cmd.exe?/ c+dir HTTP/1.0	Request made by a machine infected by the infamous <b>Nimda</b> worm	<b>A4</b> -Detected with IDOR regex	Yes
GET/default.ida?XXXXXXXXXXXXX XX u0078%u0000%u00=a HTTP/1.0" 200 566	The request shows an attack by the worm <b>Code Red</b>	<b>AOthers</b> -Code Red regex detected this occurrence	Yes
"GET/cgibin/includes/hnmain.inc .php3?config[incdir]=http://xxxxx xxxxx/HTTP/1.1" 404 312 "-" "Mozilla/4.75 [en](X11, U; Nessus)"	<b>Nessus</b> attacks	<b>AOthers</b> -Nessus regex detected this occurrence	Yes

Table 9 – Attacks detected comparison (Honeynet vs. SMCS)

In summary, with the intention to determine if the SMCS security analysis was able to detect potential attacks, we compared SMCS outputs with the results posted by the Honeynet project [8] for the scan of the month (*scan31*). The results obtained are good, even though the solutions from the Honeypot appear in terms of the attack causes, (e.g. Nessus, firewall, etc.) and with SMCS, they are presented in terms of the attack “family” (e.g. A1, A2, etc.). Looking at Table 8 with the numerical solutions, we are able to find similarities in the numbers presented. In relation to Table 9 we were able to demonstrate that even with different designations, SMCS was able to discover the attacks present in the *scan31* from the Honeynet project.

### 6.3 Detecting MS IPTV attacks

In this section, we describe our findings when we applied the SMCS analysis to the MS IPTV production logs. As we stated in section 4.2.3, these are MS IPTV production web logs, so we do not know a priori if they contain attacks. In the following charts, we present the number of attacks found per “family” type.

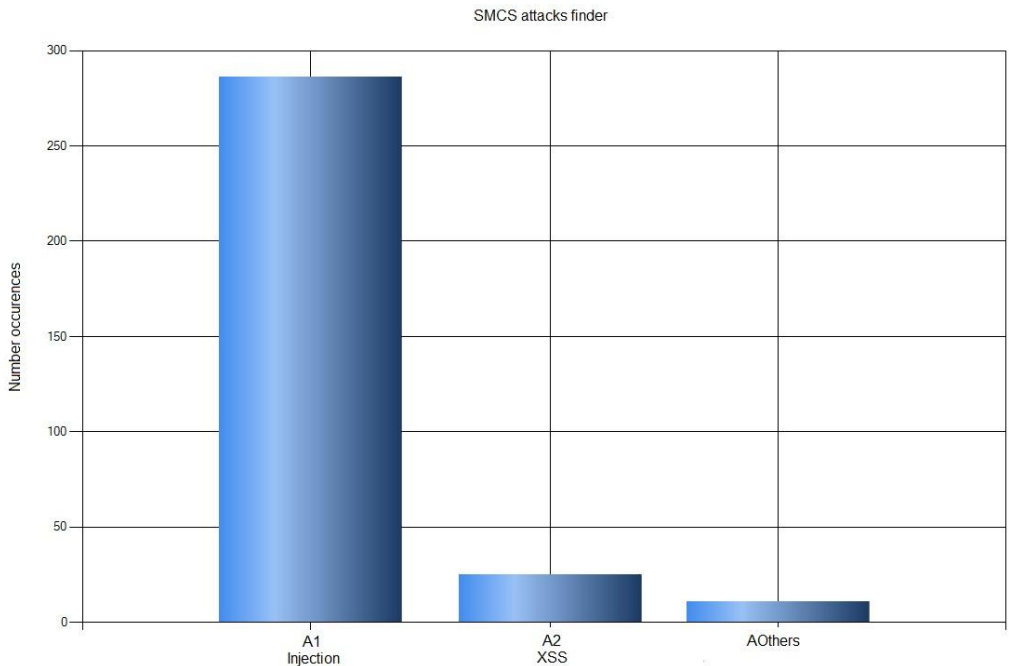


Figure 28 – MS IPTV attacks detected

The types of attacks detected with SMCS were, A1 injection, A2 XSS and AOthers. Next output presents an overview of the results obtained from the MS IPTV logs analysed, using another graphical representation that provides a very clear image of how attack families occurrences relate to one another.

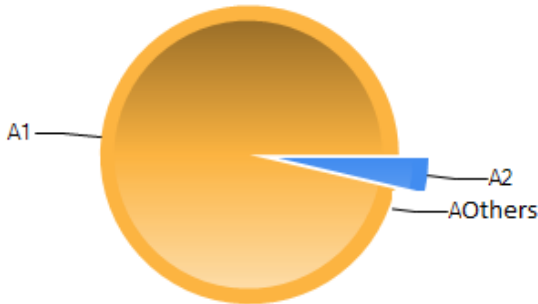


Figure 29 – SMCS output for MS IPTV logs (pie)

Following the SMCS workflow, the next step is to run the pre-defined *Log Parser* scripts, over the suspected logs. The output results are presented in Figure 30, exactly as we see them in the web site created with the information from the Log Parser console.

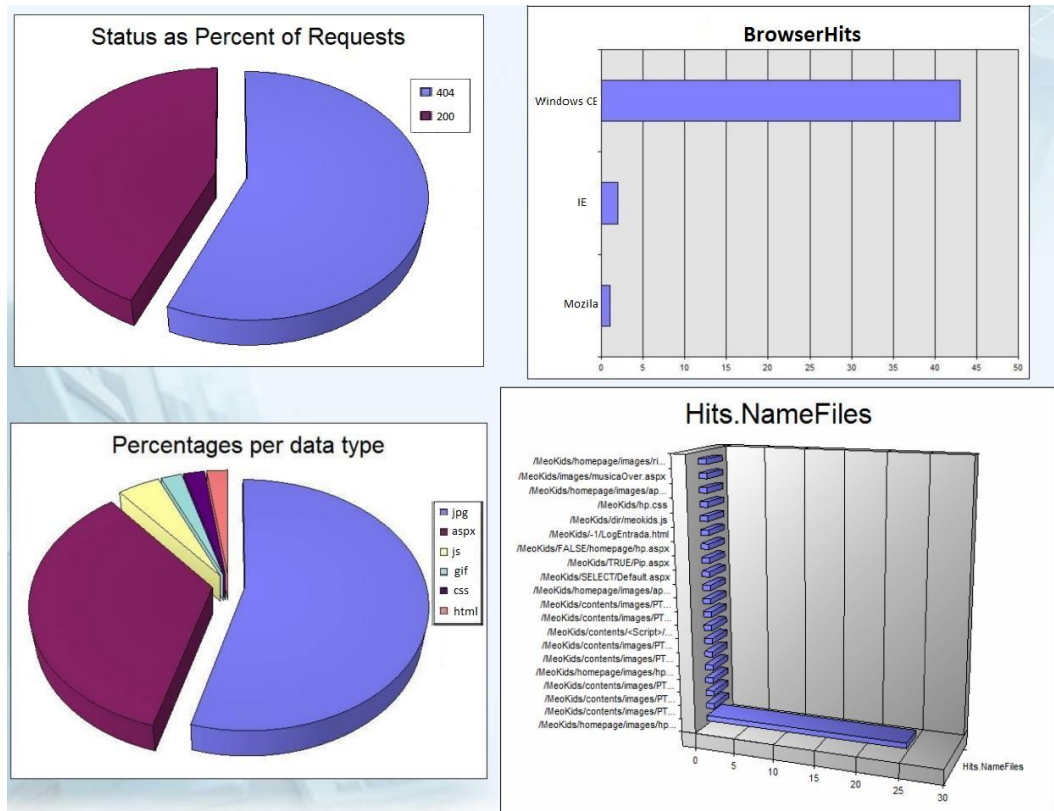


Figure 30 – MS IPTV Log Parser output console

From the SMCS chart outputs, we can state that A1-Injection attacks were the predominant “family” of attacks found. From the Log Parser outputs, is possible to infer from the *status as percent of request* chart that the 404 errors (HTTP Not Found), had a high percentage of occurrences among the logs with potential attacks investigated. In the *BrowserHits* chart, we might want to analyze all the requests coming from other operating systems different from the one of the set-top boxes (Windows CE). In the *Percentages per data type* and *Hits.Namefiles* charts, we can detect if suspicious file types were being accessed within the compromised web logs. After running through all the phases from the workflow, we were able to isolate a few hundreds of log lines (Figure 28) that might indicate possible attacks. At a first glance this might seem an exaggerated number, but it is important to realize that our set of logs contain over forty millions of lines, and that no other solution is looking at these log lines in terms of MS IPTV security.

In this scenario we are using limited and confidential logs, from which we do not know the semantics. For this reason we performed just a superficial analysis over the potential attacks and found what we suspected to be a *SQL injection* attempt and a *credit card abuse attack*. However, many more false alarms were also found. We believe that with some small regular expressions configurations and false positives adjustments, we might drastically reduce the detected number of potential attacks, until only real attacks are left. It was never on the scope of the thesis to perform an exhaustive search over the *Portugal Telecom* MS IPTV web services, looking for attacks. We already demonstrated that SMCS is able to perform this, during the *simulated* and the *real world* scenario, section 6.1 and 6.2.

This scenario was important to realize the applicability of SMCS in a MS IPTV production environment. SMCS needs few adjustments to the MS IPTV production reality and we believe that the methodology used could be a great asset to help defend MS IPTV systems from future attacks. We intend with SMCS to equip *Portugal Telecom*, with a mechanism that performs security analysis specifically over the MS IPTV logs infra-structure, in bounded time.

## 6.4 SMCS execution overhead

To determine if “SMCS executes within bounded time”, we present a summary of the average time that SMCS needs to parse the *simulated log*, the *real world log*, and an arbitrarily chosen *MS IPTV log*. Finally the average time that SMCS takes to parse the complete set of MS IPTV logs.

### 6.4.1 Parsing times – simulated log

Table 10, displays the parsing times obtained for the first test case, the *simulated log*. We record the time that SMCS took when searching for two top OWASP attacks (*Injection* and *XSS* attacks), and the time that it takes to look for all the attacks.

	SMCS taken to complete the security analysis	
Simulated log	Level 1	Level 2
A1 - Injection	2 seconds	5 seconds
A2 – XSS	1 second	3seconds
All attacks	3 seconds	7 seconds

Table 10 – Simulated log analysis time

The results were very good, since we notice that the output results appeared in an almost instantaneous way. This was the expected behavior, since the log analyzed was very small (approximately 76 lines).

### 6.4.2 Parsing Times - scan31 log

Table 11, displays the parsing times obtained for the second test case, the *scan31* log. The log analyzed contains 202145 log lines and it has 42MB of size as described in section 4.2.2.

	SMCS time to complete the security analysis	
Scan31 log	Level 1	Level 2
A1 - Injection	5 minutes 15 seconds	39 minutes 11 seconds
A2 – XSS	42 seconds 35 seconds	4 minutes 20 seconds
All attacks	7 minutes 44 seconds	54 minutes 34 seconds

Table 11 – Scan31 log analysis time

This time a larger number of injection attacks were detected and has expected the time SMCS took to parse, increased. We also notice that SMCS *Level1* was considerably quicker to finalize in comparison with the *Level2*. This behavior was also expected since the *Level2* retrieves even more complete information about the potential attacks (e.g. log line, ip of the attacker).

### 6.4.3 Parsing Times – MS IPTV log

Table 12, displays the parsing times obtained for the third test case, the *MS IPTV* logs. For this test case, *one random log (ex11041.log* with 10MB of size) was selected to be a MS IPTV log representative. From our experience, this will be approximately the time SMCS will take to analyse each MS IPTV production log, for the same conditions present in this scenario.

	SMCS time to complete the security analysis	
MS IPTV “ <i>ex110411 log</i> ”	Level 1	Level 2
A1- Injection	35 seconds	1 minute 30 seconds
A2-XSS	8 seconds	12 seconds
All attacks	55 seconds	1minute 58 seconds

Table 12 – MS IPTV log analysis time

Based on the results of Table 12, we can say that it would be possible to use this solution in a real world scenario, since we are able to parse production logs in an acceptable amount of time. We also confirmed our expectations, that the *execution time* of SMCS depends on the input log size and on the number of attacks found.

Finally, in Table 13 we present the parsing times obtained for the *complete set* of logs from *MS IPTV*. The *complete set* consists of 445 distinct web logs that totalize 40 million of lines and sum up to 10 GB of data.

MS IPTV all logs	Level 1	Level 2
All attacks	58 minutes 34 seconds	2 hours 34 seconds

Table 13 - MS IPTV all logs analysis time

Taking into consideration that the SMCS performance was not on the scope of this thesis, the time values found for the *complete set* of MS IPTV logs are good. These time values lead us to consider that security reports can be generated in bounded time by the SMCS solution, for an MS IPTV infrastructure.

Note: It is important to take into account that these execution times were obtained in a system with the following characteristics:

- Operating system: Windows 7 (Business) x64,
- CPU: One physical processor of 64 bits,
- Physical memory: 3888 MB Total.

This is not a dedicated server, it is a personal laptop that was running SMCS in parallel with other applications.

We believe SMCS performance can be improved with distributed systems, since SMCS engine can execute multiple instances in parallel. Therefore, we could set multiple autonomous servers connected through a computer network, to interact with each other in order to achieve a common goal: divide the log parsing activity for multiple instances, or even force some instances to run only the SMCS *Level1*, while others might use the results obtained by the previous iteration, to execute the *Level2* analysis.

We also test and concluded that execution times could be reduced if SMCS runs in a *dedicated server* with higher computational power (memory and CPU). This conclusion is based on the



tests performed in the FCUL lab, with slower machines that returned higher execution times for the same test cases.

## 6.5 SMCS integrating with SCOM

To achieve our integration goals we needed to find a way for SMCS to communicate with the System Center Operations (SCOM). Using the commands:

**eventcreate /L Application /T WARNING /SO SMCS /ID 998 /D "SMCSAlert"**

**eventcreate /L Application /T ERROR /SO SMCS /ID 999 /D "SMCSError"**

We are able to generate an occurrence in the event viewer of the machine that will be hosting the SMCS engine. A SCOM agent will then capture this event viewer occurrence and will generate a security ticket that will be solved by an operator or system administrator.

These commands create two distinct severity levels. The *alert level*, used to inform that a new SMCS security report has been generated, and the *error level* used every time that SMCS finds relevant information that requires immediate attention.

Figure 31, illustrates a SMCS security reports integrated into the MS IPTV SCOM console. The *Portugal Telecom operators'* team will be able to access security information regarding the last SMCS analysis.

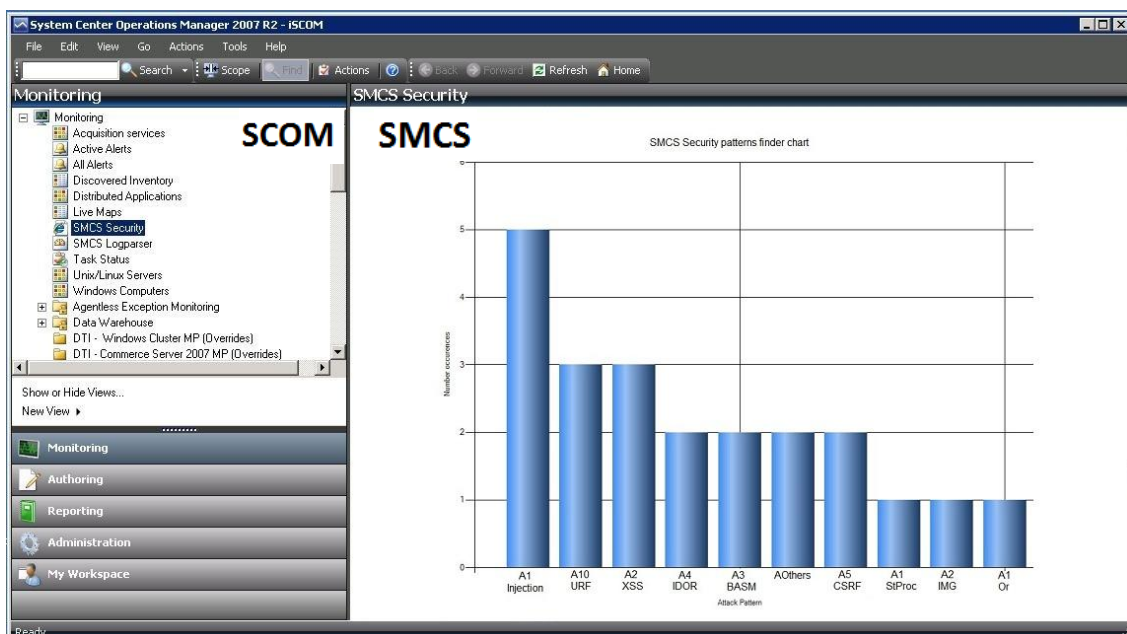


Figure 31 – SCOM and SMCS integration

Integrating the Log Parser complementary results into SCOM console, was one of the other functionalities achieved (Figure 32).

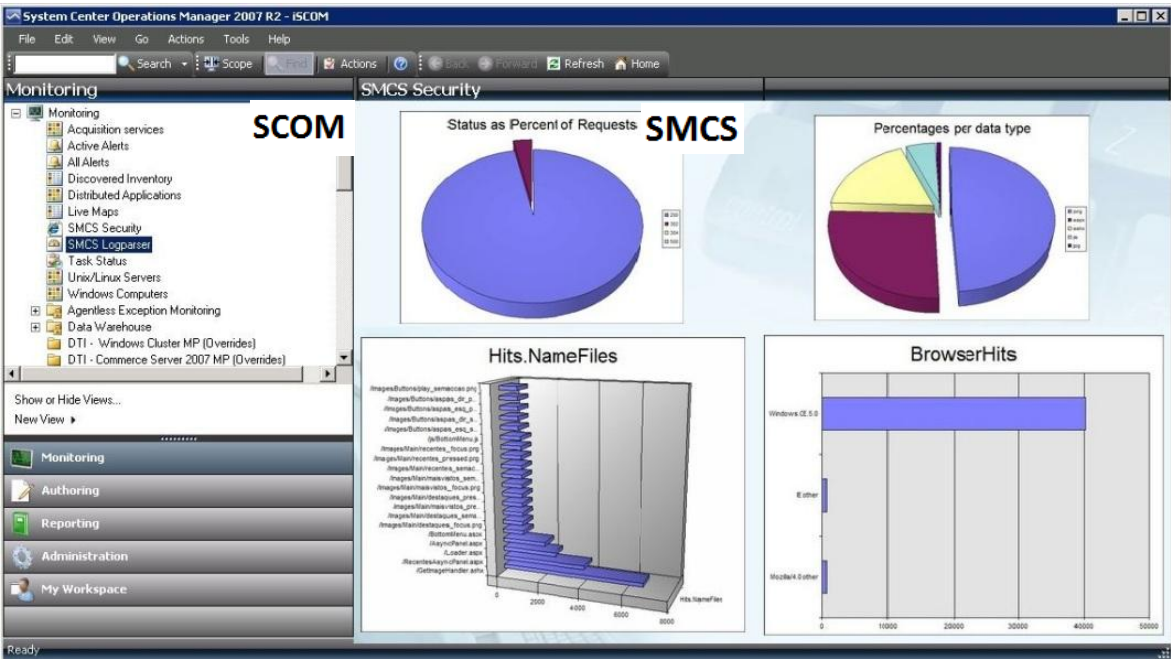


Figure 32 – SCOM and SMCS Log Parser integration

We can say that we already took the first steps into *our integration objectives goal*, since we were able to integrate into a single console (SCOM console) the security analysis outputs generated by SMCS.

## 7 Conclusions

In this thesis, our proposal was to build a collector to audit exposed MS IPTV services with the intention to analyze if some technology areas of Microsoft Internet Protocol TeleVision (MS IPTV) might be vulnerable (e.g. web services). This fact allied to the immaturity and lack of established best practices for MS IPTV systems, can provide an opportunity for illegitimate activities, since new services are attractive targets for malicious hacking attempts. In the end, the main issue that we want to help solve, is to find out if attacks are passing through the firewall layer and reaching the application layer. This is something easier to accomplish if we are dealing with web applications, since ports 80 and 443 are usually allowed through firewalls.

The thesis starts from a motivation based on some cutting edges of society facts where studies attest that TV services are today more than commodities, but basic goods. MS IPTV was left with no space for errors. The main concerns are therefore prevention and detection and with this in mind, we compared some of the most relevant monitoring and log management systems (e.g. ArcSight, Feedzai, SCOM and others.). We soon realized that none of them aims to obtain security information from MS IPTV log files, so we developed a collector and named it SMCS (SCOM MEO Collector Services).

Instead of focusing only on parsing log files, we decided to build a toll end-to-end, i.e. to develop a component for each of the tasks described in the SMCS implementation design. In the process, we acquired a deeper knowledge of the requirements of such a system and gathered some ideas of what directions should be taken in the future to improve the results obtained. Naturally, we can use the information gathered with SMCS and correlate it with other systems behavior and focus in a future work for enhancing detection. Another important decision was, to follow the regular expression approach, since the advantages are considerable for example, one regular expression might contain hundreds of attacks patterns.

To identify potential attacks, we built two report levels, two chart types and the SMCS PowerShell console. These features facilitate the evaluation of the malicious occurrences.

Presently, this evaluation has to be performed using shared company solutions (e.g. ArcSight) that were not developed specifically for the MS IPTV systems.

In the introduction, we presented three main properties to characterize the contributions of our application. SMCS performs:

- data logs analysis – the results of the tool identify not only the logs reports that contain valuable information about MS IPTV system security, but also different metrics can contribute to clarify what happened;
- data parsing – the application engine is able to parse huge quantity of logs in bounded time in search for evidences of potential attacks. This is a mostly automatic process, since a *converter* was developed to populate the parsing queries (i.e. attack patterns) and an *extractall* process was developed to prepare all the logs to be parsed. The only exception is the regular expression creation, since this process requires manual parameterization of the attacks to parse.
- event diagnosis - the tool includes reports, charts and a PowerShell console that allows users to execute the tool and to visualize the output results. These results are presented in a clear and aggregated form to help the users to diagnose in a more accurate and precise way the attack that happened in the system.

SMCS is compliant with the OWASP (Open Web Application Security Project) scale that ranks the most relevant attacks, listing the most actual and critical web application security flaws in a scale from one to ten. We decided to focus on very specific but relevant aspects of MS IPTV security and that is why we dedicate a section to explain the role of web servers in MS IPTV infrastructures and how attacks against them could be detected.

To demonstrate that our search and match algorithm is effective and robust, we generate a *simulated*, a *real world* and a *MS IPTV* test case. The most important results were the ones obtained with the web logs from the Honeynet project (*scan31*), because they made possible to prove that our approach is valid, and that attacks can be identified and categorized in terms of the OWASP scale. This is due to the fact that, the attacks performed against *scan31*, were known a priori.

The results obtained with the test cases, allow us to be confident about the execution of SMCS in the MS IPTV production environment. SMCS is able to display a list of the top matching attacks identified on the logs in bounded time. This means that *Portugal Telecom* can benefit

from the SMCS outputs, since they will show the most relevant and recurring security problems happening in the MS IPTV infrastructure.

We were also able to achieve our *integration* objectives, since we integrate into a single console (SCOM lab console) the security analysis outputs generated by SMCS. In the end, it will help to guarantee the success of SMCS, in keeping track of the security threats of MS IPTV infrastructure, and at the same time to be compliant with one of the company major goals, *systems integration*. Nevertheless, further work has to be done, before we fully integrate SMCS into the production environment, as described in section 8.1.



## 8 Future Directions

### 8.1 Promote SMCS and SCOM integration

As we demonstrated in section 6.4.1, we can say that we already took the first steps into *our integration objectives goal*, since we were able to integrate into a single console (SCOM console) the security analysis outputs generated by SMCS. However, is important to note that all integration tests were performed in a laboratory environment, and for this integration to become a reality in a production environment, acceptance tests must be conducted to determine if the requirements of *Portugal Telecom* are meet.

### 8.2 Reducing false positive rate

The term false positive, is a broad and somewhat vague term that describes a situation in which SMCS triggers an alarm, when there is no malicious activity or attack occurring. These occurrences are also known as "false alarms".

While analyzing the output results from MS IPTV logs, we realized that one of the problems that need to be address are the *non-malicious alarms*, generated through some real occurrence that are non-malicious in nature. They result from the fact that we mark all requests detected with the regex or with the attack patterns, as malicious. The solution lies in, analyze the output result searching for false positives (manually or using scripts,) which will be marked as benign until there are no obvious false positives left. This will be part of a "learning phase" that SMCS will have to go through in order be fully compliant with the specifications of our environment and our business.

### 8.3 Automatically trigger response actions

As we have seen in the workflow from Figure 16, given a specific detection scenario, we immediately trigger some analysis. An error event is created for SCOM to trigger an alarm and the Log Parser scripts are executed against the suspected logs to obtain a deeper security

analysis. In the future, with the inclusion of refined regular expressions and attack patterns, we will be able to achieve a better granularity and a lower false positive rate. Upon a detection of an attack, necessary response actions could be defined, and once a similar attack were identified these actions could automatically be triggered.

## 8.4 Green monitor attacks prevention

Some future work could be done to develop a way to make SMCS detect, what we discovered to be, an attack against the current monitoring infrastructure. In a hypothetical but also feasible attack, a recurring script could make use of some tools created to help the system administrator to clear alarms in a flooding situation<sup>40</sup>, and run this task in a periodic way. This will make all alarms appear in a healthy state, even the critical ones (see Figure 33).











	Source		Name	Resolution State	Created	Age
<b>Severity: Critical (3)</b>						
	stg-omgw-02.ipt...		Health Service Heartbeat Failure	New	5/13/2011 6:43:16 PM	3 Days, 16 Hour...
	stg-omgw-01.ipt...		Health Service Heartbeat Failure	New	5/11/2011 4:43:17 PM	5 Days, 18 Hour...
	MSSQLSERVER		Run As Account does not exist on the target sy...	New	4/18/2011 10:47:50 AM	29 Days, 51 Min...
<b>Severity: Warning (3)</b>						
	STG-SCOM-01.i...		Operational Data Reporting failed	New	5/14/2011 10:00:25 PM	2 Days, 13 Hour...
	Microsoft(R) Wi...		NTFS - Delayed Write Lost	New	5/14/2011 12:18:53 AM	3 Days, 11 Hour...
	STG-SCOM-01.i...		Workflow Runtime: Failed to run a WMI query	New	5/12/2011 12:08:31 PM	4 Days, 23 Hour...
<b>Severity: Information (2)</b>						
	STG-CSFB-02.ip...		Cache Hit Ratio	New	4/18/2011 11:08:23 AM	29 Days, 31 Min...
	stg-csfb-01.iptv...		Cache Hit Ratio	New	4/18/2011 11:08:22 AM	29 Days, 31 Min...

Figure 33 – “Green” alerts attack – before the attack











	Source		Name	Resolution State	Created	Age
<b>Severity: Critical (3)</b>						
	stg-omgw-02.ipt...		Health Service Heartbeat Failure	New	5/13/2011 6:43:16 PM	3 Days, 16 Hour...
	stg-omgw-01.ipt...		Health Service Heartbeat Failure	New	5/11/2011 4:43:17 PM	5 Days, 18 Hour...
	MSSQLSERVER		Run As Account does not exist on the target sy...	New	4/18/2011 10:47:50 AM	29 Days, 51 Min...
<b>Severity: Warning (3)</b>						
	STG-SCOM-01.i...		Operational Data Reporting failed	New	5/14/2011 10:00:25 PM	2 Days, 13 Hour...
	Microsoft(R) Wi...		NTFS - Delayed Write Lost	New	5/14/2011 12:18:53 AM	3 Days, 11 Hour...
	STG-SCOM-01.i...		Workflow Runtime: Failed to run a WMI query	New	5/12/2011 12:08:31 PM	4 Days, 23 Hour...
<b>Severity: Information (2)</b>						
	STG-CSFB-02.ip...		Cache Hit Ratio	New	4/18/2011 11:08:23 AM	29 Days, 31 Min...
	stg-csfb-01.iptv...		Cache Hit Ratio	New	4/18/2011 11:08:22 AM	29 Days, 31 Min...

Figure 34 – “Green” alerts attack – after the attack

<sup>40</sup> Flooding situation – thousands of alarms impossible to manage



Workflow of the green monitor attack:

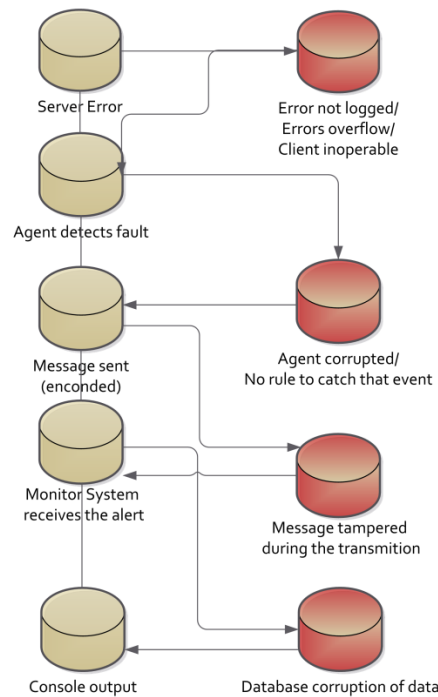


Figure 35 – “Green” attack workflow

This type of attack will be particularly dangerous since the operational teams will relax in a presence of the “green alerts” that normally indicates that everything is fine with the systems. Figure 34 shows a console that continues to report green healthy states for all the agents but in the reality, two of the nodes were down.

#### 8.4.1 Baseline correlation

Some future work could be to expand SCOM MEO Collector Security (SMCS) functionalities, to compare the information retrieved from the current logs with a baseline retrieved when the infrastructure was working as expected (optimal operational state). It will be able to detect problems by the correlation of the current logs with past-diagnosed problems, an approach that will complement the approaches presented during this thesis. First, each input log line will be decomposed and analyzed for similarities with the top match historical log lines and secondly SMCS will show the corresponding similar lines highlighting the portions that match exactly the inputted lines as long as a match score between the two lines. As result, SMCS will not only present a graphical visualization from the most important log lines but it will also provide the selection of the top matches. This baseline correlation will then help to diagnose the problems for the most relevant MS IPTV exposed services.



## References

- [1] AlienVault\_Unified\_System\_Description  
URL: [http://alienvault.com/docs/AlienVault\\_Unified\\_System\\_Description\\_1.0.pdf](http://alienvault.com/docs/AlienVault_Unified_System_Description_1.0.pdf), July 2011
- [1] Alexandre Silveira Pupo, Anderson De Salve, André Felipe de Oliveira Fernandes, Germano Packer, Guilherme Marinheiro Chaves, Sandra Regina Borges de Salve, Thomaz Fischer Levy. In *Os Maiores Riscos de Segurança Cibernética*, pages 5-6, 2009
- [2] David Ramirez. In IPTV Security - Published by John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, pages 141-181, 2008
- [3] Institute of Electrical & Electronics Engineers, Standard 610 (1990), reprinted in IEEE Standards Collection: Software Engineering 1994 Edition.
- [4] Nielsen. Television, Internet and Mobile Usage in the U.S.  
URL: <http://i.cdn.turner.com/cnn/2009/images/02/24/screen.press.b.pdf>, July 2011
- [5] Lawrence Harte, Hannah Morgan, Robert Belt. IPTV Industry Statistics.  
URL: <http://www.iptvmagazine.com/stats.html>
- [6] IPTV - Emerging Technology Trends. Via Technology Forum  
URL: [http://www.via.com.tw/en/downloads/presentations/events/vtf2006/VTF2006\\_EM-SPLInnotech-NandanKundetkar.pdf](http://www.via.com.tw/en/downloads/presentations/events/vtf2006/VTF2006_EM-SPLInnotech-NandanKundetkar.pdf)
- [7] Doug, Eric, Tina, Dophine, Clore. Honeybot Scan31, challenge response.  
URL: <http://old.honeynet.org/scans/scan31/sub/>
- [8] Rainer Gerhards. IIS Workflow Described  
URL: <http://www.securityprone.com/it/networksystems/spn-21-20030403IISWorkflowDescribed.html>, July 2011
- [9] Mark Burnett, Forensic Log Parsing with Microsoft's LogParser  
URL: <http://www.symantec.com/connect/articles/forensic-log-parsing-microsofts-logparser>, July 2011
- [10] Open Web Application Security Project (OWASP). ReDoS  
URL: [https://www.owasp.org/index.php/Regular\\_expression\\_Denial\\_of\\_Service\\_-\\_ReDoS](https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS), September 2011
- [11] Barnum, Sean and Amit Sethi, Cigital.Attack Pattern Glossary  
URL: <https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/590-BSI.html>, August 2011
- [12] Ying Zhu. Attack Pattern Discovery in Forensic Investigation of Network Attack  
URL: <http://faculty.uoit.ca/zhu/papers/jsac11-draft.pdf>
- [13] Monitor extremely slow or problematic scripts (ASP hangs, not responding web)  
URL: <http://iismonitor.motobit.com/help/iistrace/monitor-extremely-slow-or-problematic-scripts-asp-hangs-not-responding-web.htm>
- [14] Top 10 2010-Main, open web application security community  
URL: [https://www.owasp.org/index.php/Top\\_10\\_2010-Main](https://www.owasp.org/index.php/Top_10_2010-Main), October 2011
- [15] Improper Neutralization of Input during Web Page Generation ('Cross-Site Scripting')  
URL: <http://cwe.mitre.org/data/definitions/79.html>. August 2011

- [16] Roger Meyer, Detecting attacks on web applications from log files Sans Institute  
URL: [http://www.sans.org/reading\\_room/whitepapers/logging/detecting-attacks-web-applications-log-files\\_2074](http://www.sans.org/reading_room/whitepapers/logging/detecting-attacks-web-applications-log-files_2074)
- [17] Steven Smith, Regular expression library  
URL: <http://regexlib.com/Search.aspx?k=credit&c=-1&m=-1&ps=20>, October 2011
- [18] Sony Corporate Information  
URL: <http://www.sony.net/SonyInfo/News/Press/201110/11-1012E/index.html>, October 2011
- [19] Stuxnet, Code\_Red Wikipedia  
URL: <http://en.wikipedia.org/wiki>, July 2011
- [20] DaBoss, Vtutor virus types, Nimda  
URL: <http://www.cknow.com/cms/vtutor/nimda.html>, July 2011
- [21] Kingcope, SecurityFocus  
URL: <http://www.securityfocus.com/bid/36273/info>, July 2011
- [22] The HoneyNet Project. Scan of the moth – Scan31  
URL: <http://old.honeynet.org/scans/scan31/>, October 2011
- [23] Microsoft IIS Repeated Parameter Request DoS, Symantec  
URL: [http://www.symantec.com/security\\_response/vulnerability.jsp?bid=43140](http://www.symantec.com/security_response/vulnerability.jsp?bid=43140), August 2011
- [24] Complex Event Processing (CEP)  
URL: <http://domino.watson.ibm.com/comm/research.nsf/pages/r.datamgmt.innovation.cep.html>, August 2011
- [25] Meo Shell  
URL: <http://videos.sapo.pt/hugordias>, October 2011
- [26] ArcSight Product review  
URL: [http://www.arcsight.com/articles/SCmagazine\\_ArcSight\\_Product\\_Review.pdf](http://www.arcsight.com/articles/SCmagazine_ArcSight_Product_Review.pdf), August 2011
- [28] Snort  
URL: <http://www.snort.org/>, August 2011
- [29] Exploits of a Mom carton  
URL: <http://xkcd.com/327/>
- [30] Magic Quadrant for Security Information and Event Management (SIEM)  
URL: <http://www.arcsight.com/library/download/GartnerMQ2011>, August 2011
- [31] Strategies to Reduce False Positives. Symantec  
URL: <http://www.symantec.com/connect/articles/strategies-reduce-false-positives-and-false-negatives-nids>,  
November 2011