UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Hardening an Open-Source Governance Risk and Compliance Software: Eramba

Miguel Francisco Duarte Pinheiro Ramos Chaves

MESTRADO EM SEGURANÇA INFORMÁTICA

Dissertação orientada por:
Prof. Doutor Mário João Barata Calha

2020

Acknowledgments

First of all, I would like to thank my brother Francisco, mother Deolinda, father Francisco and grandfather Manuel for guiding me through life.

To Doctor Domingos Coiteiro, Doctor Nuno Simas and the whole team of neurosurgery of Hospital Santa Maria for saving my life in 2015.

A huge thank you to my girlfriend Matilde for motivating me this year to reach my goals through love, strength, patience, support, determination, and encouragement.

I would like to express my gratitude to Professor Mário Calha. More than being my counsellor, Professor guided and inspired this project by giving relevant advice, observations and always being available to help and contribute to this project's success.

Thank you to all my friends from Faculdade de Ciências da Universidade de Lisboa especially Rui Pereira, Francisco Castel-Branco, Gonçalo Miranda and Daniel Pires.

Thank you to André, Sebastião, Rafael and *Little Tooth.*

Thanks to André Ginja and Paulo Rugeiro.

Thank you to all my friends from Saint Cross. Thank you to Tomás Bastos, my dude and my sidekick. Thank you to all my friends from Colégio Manuel Bernardes.

From EY, huge thanks to everyone from the Cybersecurity team, especially to Ricardo Dionísio for guiding me on this project and giving me the liberty to follow the path of the project as I wanted; thank you to Rita Cunha; Sérgio Sá; Manuel Oliveira; José Rafael Monteiro and Pedro Monzelo for giving me the motivation and time to work on this project despite everything being on fire every day.

Resumo

Lições históricas como Chernobyl, Fukushima ou o colapso da ponte de Mississípi revelam a vital importância da gestão de risco. Para além de saber gerir o risco, as empresas têm de desenvolver planos para se precaverem e oferecerem resiliência a qualquer ameaça que possam enfrentar, desde desastres naturais e terrorismo a ciberataques e propagação de vírus. Estes planos são denominados de planos de continuidade de negócio. A crucialidade destes planos e a introdução de novas leis como Lei Sarbanes-Oxley, Diretiva Europeia 2006/43/EC VIII e recentemente do Regulamento de Protecção de Dados geraram uma maior preocupação e sensibilidade nas empresas em aglomerar todos estes processos de governança, risco e conformidade (*GRC*). *GRC* integra a implementação da gestão de risco, planos de continuidade de negócio, conformidade com as leis e boas práticas de auditoria externa e interna. As empresas necessitam de uma ferramenta que ofereça uma visão global da Governança, Risco e Conformidade. No entanto, estas ferramentas são por norma dispendiosas, o que faz com que pequenas e médias empresas não tenham meios para suportar o custo. Consequentemente, estas empresas tendem a adoptar ferramentas de código aberto, como SimpleRisk, Envelop ou Eramba. Apesar de suportarem o *GRC*, existem vários problemas com as aplicações deste tipo, como a falta de manutenção, problemas de migração, dificuldade de escalabilidade, a necessidade constante de fazer atualizações e a grande curva de aprendizagem associada.

A Ernst & Young agora conhecida como EY oferece serviços de *Consulting*, *Assurance*, *Tax* e de *Strategy and Transaction* para ajudar a resolver desafios mais difíceis dos seus clientes e criar valor. Para se preparar para uma futura auditoria, um cliente da EY pertencente ao sector bancário procura ser certificado em ISO/IEC 27001 e ISO/IEC 22301, referentes a Sistema de Gestão de Segurança de Informação (SGSI) e Sistema de Gestão de Continuidade de Negócio (SGCN), respectivamente. Adicionalmente, o cliente visa migrar a sua infraestrutura no local para uma infraestrutura na cloud. Com todos estes fatores em conta, a EY recomendou uma ferramenta de código aberto de *GRC* chamada Eramba.

Esta tese propõe um estudo profundo das vulnerabilidades que o Eramba pode oferecer assim como uma solução para as resolver através de armazenamento em nuvem. Seguindo uma metodologia de *pentesting* chamada PTES para o estudo de vulnerabilidades foi possível identificar dez vulnerabilidades sendo quase todas de baixo nível. A metodologia PTES recomenda o uso de adoção de modelo de ameaças de modo a perceber como os processos estão correlacionados, onde estão armazenados dados importantes, quais são os principais ativos e como é processado um pedido na aplicação. Para fazer esta modelação foi seguido uma metodologia proposta pela Microsoft nomeada de STRIDE, esta metodologia é uma mnemónica para *Spoofing*, *Tampering*, *Repudiation*, *Information Disclosure*, *Denial of Service* e *Elevation of Privilege*. A Microsoft propõe um modelo de ameaças em quatro passos: modelação do sistema através de *Data Flow Diagrams*; encontrar ameaças e consequentemente classificá-las através da nomenclatura STRIDE; endereçar ameaças mitigando e eliminando-as e validar se cada uma foi realmente endereçada com sucesso. De modo a endereçar estes dois últimos passos e para conjugar com os requisitos da empresa de migração para armazenamento na nuvem foi desenvolvido uma solução de tornar o Eramba num *container* para então usufruir da orquestração de containers que é o *Kubernetes*. Como resultado, a partir do trabalho desenvolvido é possível que qualquer organização adapte esta solução de *GRC* e consiga hospedar na nuvem sem enfrentar dificuldades. Este trabalho proporcionou analisar a viabilidade da ferramenta Eramba a longo prazo por qualquer organização e perceber se este é escalável.

Palavras-chave: Governança, Risco e Conformidade, Continuidade de Negócio

Abstract

Historical lessons such as Chernobyl, Fukushima or the collapse of the Mississippi bridge showcase the vital importance of risk management. In addition to managing risk, companies must develop plans to safeguard against and offer resilience to any threat they may face, from natural disasters and terrorism to cyber-attacks and the spread of viruses. These plans are called business continuity plans. The cruciality of these plans and the introduction of new laws such as the Sarbanes-Oxley Act, European Directive 2006/43/EC VIII and recently the Data Protection Regulation have generated greater concern and sensitivity in companies, leading them to agglomerate all these governance, risk and compliance processes (GRC). GRC integrates the implementation of risk management, business continuity plans, law compliance and good external and internal auditory practices. Companies need a tool that provides an overall view of Governance, Risk and Compliance. However, such tools are usually expensive, which means that small and medium-sized companies cannot afford the cost. Consequently, these companies tend to adopt open source tools such as SimpleRisk, Envelop or Eramba. Despite being compliant with GRC, there are several problems with applications of this type, such as lack of maintenance, migration problems, difficulty in scalability, the constant need to make updates and the large learning curve associated.

Ernst & Young now known as EY offers Consulting, Assurance, Tax and Strategy and Transaction services to help solve more difficult challenges for its clients and create value. To prepare for a future audit, an EY client within the banking sector seeks to be certified in Business Continuity and Information Security. Additionally, the client aims to migrate its onsite infrastructure to a cloud infrastructure. With all these factors in mind, EY has recommended an open source tool called Eramba.

This thesis proposes an in-depth study of the vulnerabilities that Eramba can face as well as a solution to solve them through cloud storage. Following a pentesting methodology called PTES for the study of vulnerabilities it was possible to identify ten vulnerabilities, almost all of which are low level. The PTES methodology recommends the use of a threat model in order to understand how processes are correlated, where important data are stored, what are the main assets and how a request is processed in the application. To make this modeling was followed a methodology proposed by Microsoft named STRIDE, this methodology is a mnemonic for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege. Microsoft proposes a four-step threat model: modeling the system through Data Flow Diagrams; finding threats and consequently classifying them through STRIDE nomenclature; addressing threats by mitigating and reducing them and validating whether each one has actually been successfully addressed. In order to address these last two steps and to combine them with the company's requirements for migration to cloud storage, a solution has been developed to turn Eramba into a container to then make use of orchestration that is the Kubernetes. As a result, from the work done it is possible for any organization that is an EY customer to adapt this solution and be able to host in the cloud without facing difficulties. This project also provided an overview to analyze if Eramba is secure and scalable.

Keywords: Governance, Risk and Compliance, Business Continuity

Contents

List of Figures

List of Tables

List of Acronyms

ARP - Address Resolution Protocol
COSO - Committee of Sponsoring Organizations of the Treadway Commission
CREST - Council for Registered Ethical Security Testers
CSP - Cloud Service Provider
CSRF - Cross-site Request Forgery
CVE - Common Vulnerabilities and Exposures
DNS - Domain Name System
DoS - Denial of Service
EY - Ernst and Young
FEDRAMP - Federal Risk and Authorization Management Program Penetration Test
GDPR - General Data Protection Regulation
GKE - Google Kubernetes Engine
GRC - Governance Risk and Compliance
IAM - Identify Access Management
IDS - Intrusion Detection System
ISMS - Information Security Management Systems
ISO - International Organization for Standardization
ITIL - Information Technology Infrastructure Library
IT - Information Technology
JSON - JavaScript Object Notation
NHS - National Health Service
NIST - National Institute of Standards and Technology
OCEG - Open Compliance and Ethics Groups
OSINT - Open Source Intelligence
OS - Operating System
OWASP - Open Web Application Security Project
PCI DSS - Payment Card Industry Data Security Standard
PTES - Penetration Testing Execution Standard
RBAC - Role Based Access Control
SEC - Securities and Exchange Commission
SLA - Service Level Agreements
SOX - Sarbanes-Oxley Act
SSL - Secure Sockets Layer
TLS - Transport Layer Security
VPC - Virtual Private Cloud
WAF - Web Application Firewall

Chapter 1

Introduction

1.1 Motivation

In business, every decision becomes a potential risk and making the right one is what drives progress.

If a driver is approaching a yellow light he must choose if he goes through or not. By going through he has to accelerate, risking an accident or a fine. Every time a farmer plants corn he is facing the risk that the corn may not grow due to a drought or that there will not be demand for it. Day after day, companies' headquarters are in physical risk, as fire or gas explosions are a common hazard that can happen to a building. However, enterprises often have prevention plans to mitigate the occurrence of these risks such as fire-extinguishers and emergency exit plans.

Considering that risk can be described as the effect of uncertainty on objectives [34], calculating risk is not an exact science.

Consequently, to properly handle risk, managers must identify, analyse and control threats to the organization's capital and earnings, in order to then implement risk-reducing actions and assess how risk changes over time. This is risk management. Taking into account this paradigm, risk management plays a vital role in protecting an organization's assets [53].

New regulations, cultures and values of organizational life demand for systematic risk assessment and competent management, since managing liability is a challenging and continuous process within any organization.

Since the mid-1990s, risk management underwent a dramatic expansion [62][58]. If before it was regarded simply as a field of management control, risk management gained momentum as it became a tool for companies to benchmark. History lessons such as Chernobyl [77], Fukushima [2] and the Mississippi River Bridge Collapse [52] are some of many events that could have been mitigated if there had been proper, well-established risk management. Terrorism, cyberattacks, power outage and network failures frequently cause more harm than expected due to the lack of a risk management plan. For example, in 2017, National Health Service in the United Kingdom was hit by WannaCry [26], a global ransomware attack that caused more than 19,000 appointments and surgeries to be cancelled. The Department of Health had previously developed a plan to respond to cyber-attacks, however, as it had never been tested before, it was not clear how to proceed and what actions should be taken (see Investigation:WannaCry cyber attack and the NHS [26]).

On the other hand, in 2013, the offices of Cantey Technology, an IT company that hosts servers for more than two hundred clients, were caught on fire due to a lightning strike [71]. Every cable and piece of hardware was destroyed, yet their clients did not notice any difference in the service provided. This is because five years before the fire, Cantey had decided to implement a business continuity plan and move all of their clients' servers to a remote datacenter with continuous back-up.

These past happenings, the intensification of auditing and control processes are the blueprints of what risk management is today.

In addition to risk management, Governance Risk and Compliance (GRC) have a substantial role in protecting company assets [25].

The Open Compliance and Ethics Group (OCEG) defines Governance Risk and Compliance as "a capability to reliably achieve objectives (governance) while addressing uncertainty (risk management) and acting with integrity (compliance)" [4].

Governance describes the establishment of policies and the continuous monitoring of their implementation, setting the background for risk management. Compliance guarantees that the organization meets the requirements of the boundaries established in organizational values, policies and legal requirements.

With the emergence of recent internal and external factors such as government reforms Sarbanes-Oxley Act [78], European directive 2006/43/EC VIII [21] and General Data Protection Regulation [56], GRC has become considerably relevant and continues to move up priority-wise in the industry's agenda. As a result of these legislation, society is more risk-averse than ever [65].

Therefore, decision-making skills are pivotal to thrive and improve better deployment. GRC systems are a need for the business community, as they supply essential tools for continuous growth, wise decision-making and better understanding of the full scope of risk [25].

According to OCEG latest GRC Maturity Survey [4], organizations that fail to fully integrate GRC functions consequently struggle to remain aware of the full scope of embedded risk. The survey also reports that the GRC Maturity level is below the recommended, since companies overlook the efficiency of skillfully integrated risk management, business continuity and compliance.

Due to the necessary legal requirements established by the recent government reforms, frequent human intervention is required for the systems to be fully compliant and have effective controls and policies [7]. Nevertheless, this current process is resource-consuming and fails to accommodate new needs that may arise from market trends.

The primary intent of GRC systems is to maintain sensitive data, automate risk management across the organization and communicate the organization's risk posture to internal and external groups.

The latest Forrest Waver report [49] displays a chart for enterprise governance, risk and compliance platforms. This chart is based on who has the strongest strategy as well as the strongest current offering, displaying market leaders and their differences.

Recently, Gartner also published a magic quadrant [6] for enterprise governance, risk and compliance platforms. This quadrant is based on the quality of risk, audit, compliance, policy and regulatory management. The quadrant also displays who the market leaders are and how they distinguish from one another.

Since GRC platforms are the basis of risk management and business continuity, they are a need of the current market.

Forrest Waver and Gartner's leaders have tools and technology that are able to fully identify risks across the organization while offering enhanced compliance training, strategic company levels, awareness programs and highly detailed reports. However, they are costly and some companies cannot afford them.

Small and medium-sized enterprises tend to adopt open-source software due to its lower cost and price-value ratio. Additionally, companies with higher budgets, tend to adopt open-source software for then do a post-analysis of their methodologies and verify which software suits them the most. Systems such as Eramba, Simple Risk, Envelop are examples of the long-established open-source software used by these companies. However, open-source software bears various issues such as lack of maintenance, difficult scalability, constant need for updates and the high learning curve associated.

These various software usually require external security vulnerability assessments also known as pentesting. Pentest is a methodology with the purpose to circumventing the security function of a system [16]. It aims to find security weaknesses that may be exploited or not at some extent. Additionally, these pieces software consume plenty of resources and does not tend to offer any portability or scalability as their normal setup is done via a virtual machine.

EY (Ernst & Young) is one of the largest professional services networks in the world. It offers the expertise to capitalize and grow clients' business through four services lines - assurance, consulting, tax and transaction advisory services. EY Portugal also assists clients in providing a methodology along with recommended technologies.

To prepare for a forthcoming audit, a client requested EY advisory services. Due to privacy reasons its name will remain confidential and will be treated as "organization". The organization belongs to the banking sector and handles millions of dollars per day. This organization plays a crucial role in the economy, so it is important that its operations are resilient and that the effects of any disruptions in its services are minimised, to maintain confidence in the financial system and the satisfaction of customers, shareholders and other stakeholders. As a bank, there are innumerable critical processes for the business. These processes host numerous applications used by millions of people daily across the globe. Some of these applications use legacy technologies which handle sensitive and personal data. Therefore, it is necessary to administer these technologies in a subtle approach. Currently, the organization seeks to be certified in Information Security and Business Continuity for a specific bank process. Additionally, this organization also aims to migrate its infrastructure from on-premises to cloud computing and wants to start with a minor application.

Besides EY being a Big4 (alongside with KPMG, PwC and Deloitte) and having an extensive client portfolio, EY aims to provide a GRC solution accessible to everyone so that this solution can be tested, and improve EY's GRC processes with external support.

EY Portugal for this particular client, endorsed an open-source software, Eramba, as their main GRC solution, since it accommodates all sorts of client's needs for a fraction of the leaders' cost.

However, given the limitation of open source software of these sort, the questions arose: How secure is Eramba? How could Eramba be scalable in the future in case needed? These questions will make the scope of this project.

1.2 Goals

The role of governance risk and compliance is not truly achieved without having a proper and secure software to manage it. This project's main goals are to perform a penetration test to further asses the security of said software and then provide solutions to increase it.

Overall, the goals of the proposed solution are:

- Assess vulnerabilities of Eramba.
- Exploit those vulnerabilities in a controlled environment.
- Supply patches/fixes for those vulnerabilities.
- Analyse feasible solutions for those vulnerabilities, including for on-premises and cloud environments.

Additionally, there is also a need for Information Security of the tool itself, as Eramba contains plenty confidential information of an organization. This information may range from critical assets, vulnerabilities, and implemented controls.

1.3 Work Plan

The purpose of this section is to outline what the work plan for this dissertation is and is described on Figure 1.1.

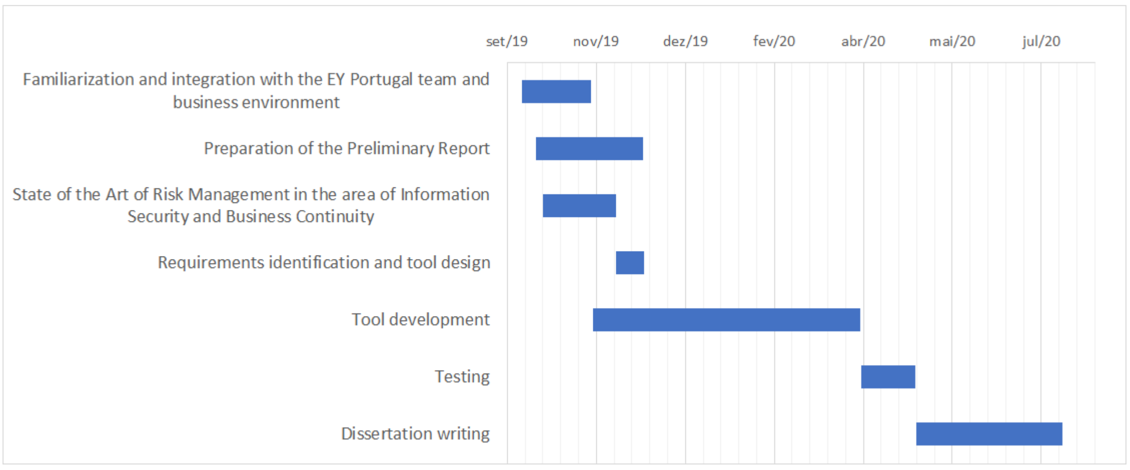


Figure 1.1: Work Plan

The work done was divergent of the work planned. The work planned was related to the development and programming of new features in Eramba. In contrast, the scope of work elaborated on security components of Eramba created a solution to fix vulnerabilities found. Additionally, due to the COVID-19 pandemic, the thesis's delivery and its deadline were postponed for two months.

1.4 Contributions

The contributions of the developed work are:

- Vulnerabilities were mitigated in line with the client's requests.
- A new Eramba setup was developed.
- An automated Eramba deployment was provided in the cloud environment.
- The foundation for a future Eramba-as-a-Service was established.

1.5 Document Structure

The remainder of the document is structured in the following sections:

- Chapter 2 - Related Work - all the work studied that led the solution proposed.
- Chapter 3 - Vulnerability Analysis of Eramba - the vulnerabilities found on Eramba and their exploitation.
- Chapter 4 - Eramba-as-a-Service - the establishment of an Eramba-as-a-Service solution.
- Chapter 5 - Conclusion and Future Work.

Chapter 2

Related Work

The purpose of this chapter is to outline: what are the methodologies of pentesting; what are the main motivations and regulations for the need of Governance Risk and Compliance (GRC); what is GRC; what are the good practices of risk management; how risk management complements GRC, how to handle it, and finally what makes a GRC software valuable and desirable. Additionally, it is presented a brief study of what cloud computing, containers and the Kubernetes technology is.

2.1 An overview of Pentesting

No system is one hundred per cent secure. With the growth of Web Applications, it is vigorously complex to guarantee security while developing such applications [16].

Web Applications can either be static, dynamic, e-commerce, portal web applications, animated web applications flash-based or a content management system. Therefore, while developing such applications, there are different challenges that the developers have to face. Security-wise, those challenges are securing the database, accessing management or guaranteeing the safety of the user. There are also other technical threats such as cross-site scripting, phishing, cross-site request forgery, shell injection, session hijacking and SQL injection. Said challenges and threats created the need for someone to simulate an attack on the applications. This is known as penetration testing or pentesting.

The roots of pentesting date back to the 1970s with the appearance of tiger teams on the computer scene [28]. Tiger teams are teams of individuals highly specialized on problem-solving. Sponsored by the Department of Defense (DoD) of the United States of America, teams of crackers attempted to break the security of computers' systems and find security issues to eventually apply a patch. Although DoD sponsored most of these teams, in the 1970s IBM spent 40 million dollars to raise awareness and address computer security. However, tiger teams were not as effective as expected and their efforts were just the beginning of analysing security flaws in the computer scene.

Presently, there are countless definitions for the term Pentesting, NIST defines it as "A test methodology intended to circumvent the security function of a system [54]." or as "A method of testing where testers target individual binary components or the application as a whole to determine whether intra or intercomponent vulnerabilities can be exploited to compromise the application, its data, or its environmental resources." [54]

Pentesting involves the use of a variety of manual and automated techniques to simulate an attack on an organisation's information security arrangements [8].

Overall, Pentesting allows the simulation of authorized cyberattacks with the main purpose of finding vulnerabilities and their exploitation.

There are three types of penetration testing:

- Black Box Penetration Testing;
- White Box Penetration Testing;
- Gray Box Penetration Testing.

The colours refer to the amount of access a threat actor has to the source code. It is an extremely hard task to test an application or a program to full extent and find all single errors, however, it is possible to find most of them through these three types. Only one of the types are chosen for an application. Black box testing analyses the application as a single black box where there is no prior knowledge of how the program behaves, its internal structure, design or implementation or how it reacts to certain inputs. White Box testing is a testing strategy that allows the tester to view the internal structure of an application. This allows the tester to follow every step of the control flow graph, therefore, testing every possible input. Gray Box testing is a combination of both black box and white box testing. In this type of test, only a part of the internal structure, design and implementation is known.

2.2 Pentesting Methodologies

There are different approaches on how to correctly pentest an application: it depends from the application type, the access to it, the level of knowledge of the pentester and the limitations associated with the technology.

2.2.1 Penetration Testing Execution Standard (PTES)

Penetration Testing Execution Standard is a standard created with business and security service in mind. It was conceived due to the lack of penetration testing in the industry back in 2009 [76]. Even though eleven years have passed, this standard is still relevant and still used. Each phase depends on the previous one and the results that are generated from it.

PTES [76] defines penetration testing in seven phases:

- Pre-engagement Interactions. The first phase is the arrangement phase, before the pentest is conducted. This phase comprehends a bureaucratic phase ranging from approval of documents, meetings convened and attended as well as the tools that are going to be used. In this phase, the Rules of Engagement (RoE) are also established. These rules aim to protect both the client and the pentester meaning that the systems will not be subject of needless risk nor the pentest will face any legal action or fine.

- Intelligence Gathering. This is the reconnaissance phase. It is all about gathering as much data as possible, passive or active, from external sources of the target systems. This data ranges from social media websites, email and cellphone related devices. It also makes use of OSINT (Open Source Intelligence) using search engines, job posting and reports that have valuable data for the pentest. There are three levels of Information Gathering, these levels identify how mature the application is. Level one is a one-click button information and can be gathered via automated tools; level two is done via using the automated tools from level one plus manual analysis; lastly, level three is the most advanced and requires heavy analysis, most likely a full team working on it and requires a vast number of hours to gather the information. Moreover, Intelligence Gathering also introduces social engineering spanning, from staff impersonation via cellphone, studying of a social media profile and phishing. There are three steps included in the Intelligence Gathering phase: Covert Gathering, Footprinting and Identification of Protecting Mechanisms. The Covert Gathering phase covers a physical environment that sometimes may be required. Actions such as wireless scanning, dumpster diving and physical security inspections are covered in this phase.

  On the other hand, footprinting focuses on the direct and indirect interaction with the target to gain data from a perspective external to the organization. Lastly, the identification of protection mechanisms is a fundamental step to a successful conduct a pentest. It is expected that applications have cryptographic functions in their protocols, Firewalls, Web Application Firewall (WAF), Intrusion detection system (IDS), closed port protocol, a suitable patch management environment but sometimes these simply are not possible due to a lapse in memory or hardware/software restrictions.

- Threat Modeling. Upon gathering information from the previous phase, threat modelling aims to understand how the business works (Business Process Analysis), how the processes are correlated, where the important data is stored (Business Asset Analysis), what are the important assets, in what kind of infrastructure the application is built on and what are the third parties at stake. After comprehending said factors, the pentester can now simulate an accurate attack to the application.

- Vulnerability Analysis. This is the process of cross-referencing the weaknesses identified, the information obtained during the intelligence gathering along with ports scanned, CVE records and DNS records into a single entry point to define the scope of the pentest and the extent of the vulnerability. There are two types of vulnerability testing: active assessment and metadata analysis. Active Assessment involves direct interaction with the component being tested through the use of vulnerability scanners and passive assessment through traffic monitoring.Traffic monitoring is to review, analyze and manage the traffic of the network to find a specific issue or have a better understanding of the network. Metadata analysis is the process of looking at the information contained in any file, ranging from last modified, owner of the file and filesize.

- Exploitation. This phase consists of identifying attack vectors passable through its security controls and choosing the one with the highest probability to have a larger impact on the organization according to the Business Asset Analysis. Then, it will be established a priority line according to which attack vectors should be explored: priority is given to the ones with highest probability and highest impact, followed by the ones with highest probability and lowest impact (and vice-versa), and lastly, attack vectors lowest impact and lowest probability.

  After exploiting the attack vector with highest probability of having the greatest impact, the ones with lower probability will follow so that all attack vectors are analyzed according to their priority.

- Post Exploitation. The exploitation of a system is just the tip of the iceberg. After the exploitation, the pentester must understand what information is available from said exploitation by determining the value of the machine compromised. The value of the machine ranges according to what data assets are stored in it and how likely is the machine to compromise other machines in the same network. Upon deciding the value of the machine, the pentester should be able to identify critical infrastructures and be capable of targeting sensitive data with high impact to the organization.

- Reporting. Reporting is the last phase of PTES standard. In this phase, the pentester will create a report with two sections: one is the objective of the conducted pentest and the other a detailed technical report. The first section must contain the background, overall posture, risk ranking, general findings and the recommendation summary to fix the vulnerabilities found. The second section which is the technical report should include an introduction, the information gathered, the assessment of the vulnerabilities, exploitation of said vulnerabilities, post-exploitation, the exposure and conclusion.

The macro-steps described for this methodology are illustrated in Figure 2.1.

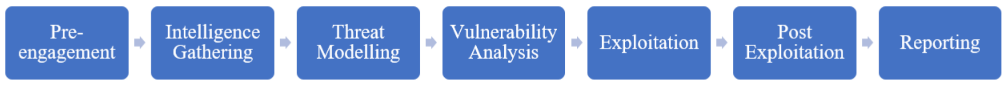


Figure 2.1: Penetration Testing Execution Standard.

2.2.2 Federal Risk and Authorization Management Program Penetration Test (FEDRAMP)

United States of America (USA) federal agencies are required by law to protect all federal information that has been added to cloud services. FEDRAMP is an US program that provides federal agencies with standards for security assessment, authorization and monitoring for cloud-based products and service. This methodology is in compliance with NIST SP 800-115 Technical Guide to Information Security Testing and Assessment, NIST SP 800-145 The NIST Definition of cloud computing, NIST SP 800-53 Security and Privacy Controls for Federal Information Systems and Organizations, IST SP 800-53A Assessing Security and Privacy Controls in Federal Information Systems and Organizations: Building Effective Assessment Plans. It was designed by the government of the USA and it was explicitly conceived for Cloud Service Provider (CSP). FEDRAMP organizes the methodology according to different targets, each having a different weight, in order to conduct an effective penetration test:

- Web Application and API.
- Mobile Application.
- Network.
- Social Engineering.
- Simulated Internal Attack.

According do FEDRAMP [23], three steps are carried out for each target: Information Gathering and Discovery, Exploitation and Post-Exploitation. Afterwards, there is a fully detailed report with the scope of the target system, the attack vectors addressed during the penetration test, the timeline of said activity, tests performed and results, findings, evidence and the access paths. This report must be included in the Security Assessment Report (SAR). Posteriorly to said report, the next pentest should be scheduled within 12 months. According to this methodology, all pentest activities should be assessed by third party organizations with proven proficiency and capability of maintaining the flow of the methodology.

There is a different workflow to be followed according to each step. The first step, Information Gathering and Discovery.

- Web Application/API information gathering - firstly, a deep internet search should be made regarding the target application to identify relevant public information, application architecture, account roles, authorization bounds, all user-controlled inputs. The mapping of all content with the functionality must be done and also a web vulnerability scanning to said target should be made.
- Mobile Application information gathering - firstly, equally to web application, there should be an internet search to identify publicly available information, map all its content and functionality and identify all permissions requested by the application.
- Network Information gathering - there should be Open Source Intelligence (OSINT) activities: an enumeration and inventory of live network endpoint and availability, fingerprint operating system and networks, as well as performing vulnerability identification.

- Social Engineering Information gathering is performing internet searches to identify CSP personnel of interest responsible for target system management.
- Simulated Internal attack information gathering is to perform a scoping exercise with the CSP to determine potential attack vectors and also perform vulnerability identification.

The second step, exploitation:

- To exploit Web Application/API, the pentester must perform activities such as authentication and session management, authorization, application logic and input validation.
- To exploit a Mobile Application, the privileges associated with the application must be identified, as well as the information stored on device, the level of encryption, and the type of information that is stored in cache and logs.
- With the intent of gaining access to the network target, first the attack scenarios should be identified, presented to the CSP and the approval for the attack is required. If approved, the pentester must attempt to elevate his privileges and afterwards document his results.
- A successful social engineering exploitation will target the employees of the Cloud Service Provider whom are responsible for the management of the system.
- After identifying the attack vectors, a simulated internal attack should be pursued to exploit all of the vectors to full extent. The main objective of this attack is to simulate a breach of the corporate assets. The tester should also be able to escalate administrative privileges through the CSP workstation image.

In the post-exploitation step, the tester will explore the vulnerabilities found during the last step. The main goal of these activities is to demonstrate the impact of the said exploitation while accessing different endpoints and accessing sensitive data, controls and infrastructure. The ease of post-exploitation is susceptible to the privilege given to the tester and the technologies used by the CSP.

The last step is reporting. The report must include the scope of the target system, the attack vectors addressed, the timeline for assessment activity, tests performed, results, finding, evidences, access paths. There should also be scheduled a new pentest within 12 months.

The macro-steps described for this methodology are illustrated in Figure 2.2.

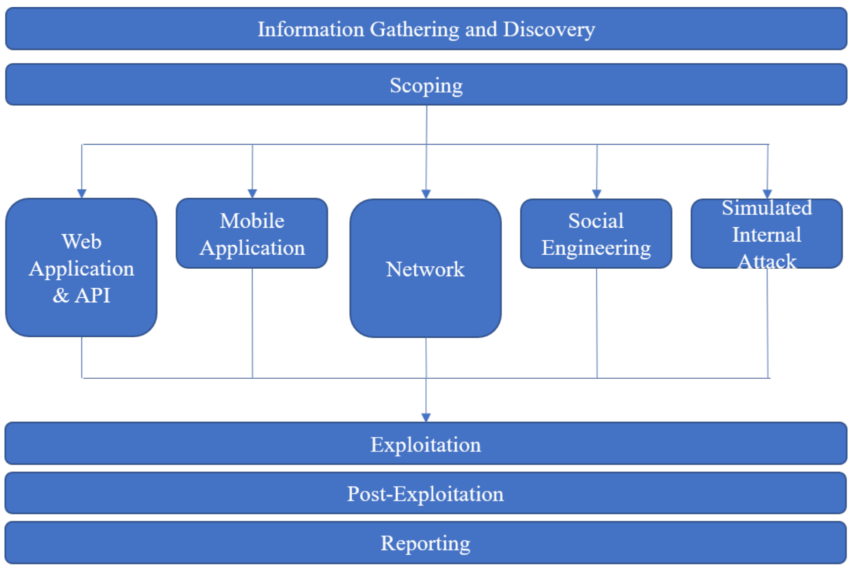


Figure 2.2: FEDRAMP Penetration Testing.

2.2.3 Council for Registered Ethical Security Testers (CREST) Penetration Test

CREST Penetration Test approaches pentest in three different steps. It provides practical advice on how to pursue an effective penetration test. This guide was developed and based on other industry standards of pentesting namely Centre for Protection of National Infrastructure (CPNI), Open Web Application Security Project (OWASP), Open Source Security Testing Methodology Manual (OSSTM) and PTES. The three steps are:

- Prepare for penetration testing (Preparation).
- Conduct penetration tests enterprise-wide (Testing).
- Carry out appropriate follow up activities (Follow up).

In the preparation phase there are seven key-steps that should be pursued. They are: the maintenance of a technical security assurance framework, the establishment of a penetration testing governance structure, evaluation of the main drivers that conduct to a pentest, identification of target environments, the outlines of the tests and they aim to achieve them, the requirements specifications and the selection of the appropriate suppliers.

On the second phase there are nine key-steps that should be pursued and some of these may follow a repetitive cycle. These steps are: the agreement of test style and type, identification of testing constraints, the scope statement, establishment of a management assurance framework, implementation of management control processes, the use of an effective testing methodology, conduct sufficient research and planning, identification and exploitation of vulnerabilities and in the end reporting of key findings.

In the follow up phase there are six actions that should be followed: remediation of the weakness, addressing the root causes of weaknesses, initiation of the improvement program, evaluation of penetration testing effectiveness, building of lessons learned and creation and monitoring of action plans.

The macro-steps described for this methodology are illustrated in Figure 2.3.

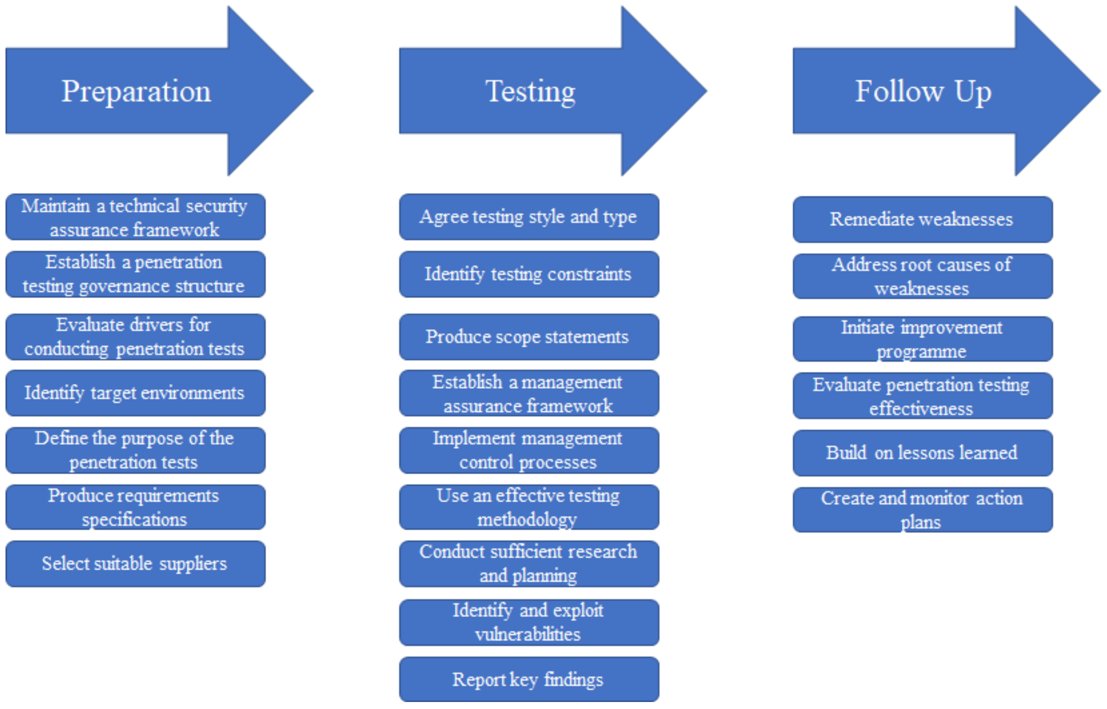

Preparation
Maintain a technical security assurance framework
Establish a penetration testing governance structure
Evaluate drivers for conducting penetration tests
Identify target environments
Define the purpose of the penetration tests
Produce requirements specifications
Select suitable suppliers
Testing
Agree testing style and type
Identify testing constraints
Produce scope statements
Establish a management assurance framework
Implement management control processes
Use an effective testing methodology
Conduct sufficient research and planning
Identify and exploit vulnerabilities
Report key findings
Follow Up
Remediate weaknesses
Address root causes of weaknesses
Initiate improvement programme
Evaluate penetration testing effectiveness
Build on lessons learned
Create and monitor action plans

Figure 2.3: CREST Penetration testing framework.

2.2.4 Payment Card Industry Data Security Standard (PCI DSS) Penetration Testing Guide

Payment Card Industry is a council that strengthens payment account data security by ensuring that the cardholder data environment (CDE) is maintained in a secure IT system through the guides of Data Security Standard. CDE is defined as “the people, processes,and technology that store, process, or transmit cardholder data or sensitive authentication data" [59].

Data security standard has a requirement titled “Payment Card Industry Data Security Standard (PCI DSS) Requirement 11.3 Penetration Testing”. It addresses obligatory penetration testing to the security systems and processes for all applications that support financial transactions through credit cards.

To guarantee a compelling and prosperous pentest, PCI DSS methodology provides three phases, each one with distinct activities:

- Pre-engagement. First, it is vital to inform all parties involved regarding what types of testing are going to be performed, how they will be performed and what is the target. The organization is responsible for defining the CDE and providing all sort of documentation to the pentester. In this phase, the rules of engagement should be agreed on to ensure that the tester does not exceed the scope of the test. These rules encompass the time of the testing, security controls established, presence of legacy systems and what steps should be taken to not threaten the environment. It should also be defined as the success criteria setting the limits of the penetration test. It is also required to do a review of past threats and vulnerabilities in the previous 12 months. Since most systems have an IDS and WAF the test should be executed to avoid these protection entities. There is also an Approved Scanning Guide with a section titled "Scan Interference" that covers how to actively protect the system during testing.

- Engagement. Each environment has different testing approaches. The organization should supply login credentials to the tester to allow the tester to assess the security of the application layer and the roles assigned to the credential. The organization should also guarantee that the role that the tester is using has all the roles applicable to fully explore the security of the application layer to full extent. The pentester must verify that there is a network segmentation and that all LANs are isolated from the CDE. Upon accessing the cardholder data, the tester commits to immediately notify the organization and provide the documentation of the conducted test for them to follow the steps reviewed and find a patch for it.

- Post Engagement. After the engagement activities, both associations must guarantee that all the exploration paths of the vulnerabilities were found and fixed. Additionally, after the patch has been deployed, the tester must retake all the steps and test if the vulnerability still exists. The last step should be cleansing of the environment, revamping credentials of the tester and tools used and guaranteeing that rules of engagement were followed.

The macro-steps described for this methodology are illustrated in Figure 2.4.

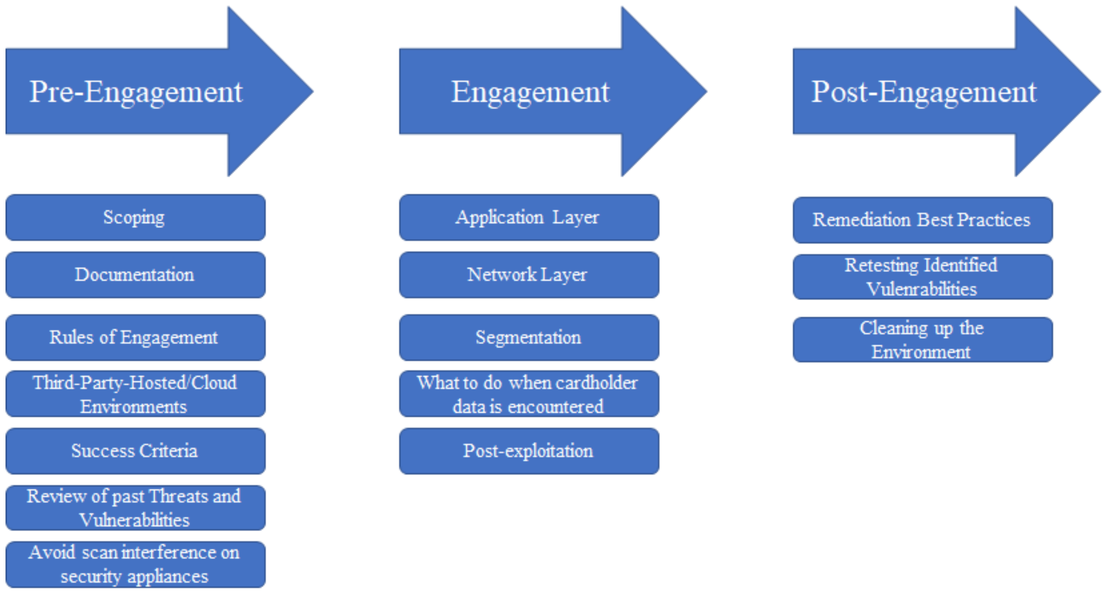

Pre-Engagement
Scoping
Documentation
Rules of Engagement
Third-Party-Hosted/Cloud Environments
Success Criteria
Review of past Threats and Vulnerabilities
Avoid scan interference on security appliances
Engagement
Application Layer
Network Layer
Segmentation
What to do when cardholder data is encountered
Post-exploitation
Post-Engagement
Remediation Best Practices
Retesting Identified Vulenrabilities
Cleaning up the Environment

Figure 2.4: PCI DSS Penetration testing framework.

2.2.5 Chosen methodology

The Table 2.1 shows the differences between the pentesting methodologies studied throughout this chapter. They follow an identical path, however, their environments differ: FEDRAMP is followed usually by US agencies and is only used in Cloud. PCI-DSS is a particular pentesting methodology as it only works in a specific environment that is cardholder data. Despite the fact that CREST and PTES methodologies can be followed in any environment, CREST methodology is more certification-guided, meaning that the pentester who follows this methodology aims to have the application certified. In contrast, the pentester who follow PTES seeks to find vulnerabilities and exploit them. For this project, the PTES methodology was chosen due to its flexibility, as some minor phases can be eliminated and still produce a valuable pentest. Additionally, PTES also allows for selecting any network tool and threat modeling framework.

Methodology	Steps	Used for
PTES	Pre-Engagement Intelligence Gathering Threat Modelling Vulnerability Analysis Exploitation Post-Exploitation Reporting	Any environment
FEDRAMP	Information Gathering and Discovery Exploitation Post-Exploitation	Cloud environment
CREST	Preparation Testing Follow Up	Any environment
PCI-DSS	Pre-Engagement Engagement Post-Engagement	Cardholder data environment

Table 2.1: Penetration Testing methodologies

2.3 Most common Attacks in Web Applications

Open Web Application Security Project (OWASP) is an organization dedicated to web application security. It is built on a community that functions independently to improve the security of software. OWASP provides documentation, tools, videos, forums and conferences all to achieve security amongst web applications.

OWASP supplies an awareness document of the top ten most common web application security risks, being the latest release of said document in 2017 [57]. This document is based primarily on a survey filled by over five hundred security professionals across the globe congregating data from hundreds of vulnerabilities. Those vulnerabilities are:

1. Code Injection. These attacks subsist on sending untrusted data to be interpreted, using a form or other submission method data to a web application. Among all types of code injection, the most common attack is the injection of an SQL query into a text form. If security measures are not taken, the SQL code could be executed resulting in data leaks or compromising of the system. This type of attack can be prevented by validating or sanitizing submitted data, or even adding controls to the database itself.

2. Broken Authentication. Vulnerabilities in authentication systems could lead a threat actor to gain access to user accounts or even accounts with privileged administration. Some of the strategies used to mitigate this type of vulnerability go through the use of 2-factor authentication (2FA), limitation of the number of successive authentication attempts or the implementation of weak passwords-checks.

3. Sensitive Data Exposure. In case web applications do not adequately protect sensitive data contained within them, a threat actor can gain access to that data and thus use it for malicious purposes. The exposure of sensitive data can be minimized by encrypting data, storing passwords using salt hash functions as well as disabling caching features for sensitive information.

4. XML External Entities (XXE). This is a type of attack that affects applications that analyze XML data. This data may refer to external entities, such as a storage unit, which can lead to data being sent to unauthorized external entities. The best way to prevent this type of attack is by using other, less complex data formats, like JSON, or at the very least, disable the use of entities in an application that uses XML, and also avoid serialization of sensitive data.

5. Broken Access Control. An access control system defines who can access particular information or a functionality. If there is a break in this control of access, a threat actor can perform tasks as if he was a privileged user. For example, an application can allow you to change between different authenticated accounts by changing part of the URL, without making any additional verification changes to ensure secure access control of the application. The web may resort to the use of authentication tokens, which must be presented each time a privileged order is placed.

6. Security Misconfiguration. This is the most common vulnerability on this list and is often the result of default settings that are used or error messages that are displayed too much. For example, an application may reveal vulnerabilities in the present errors that are too descriptive. This risk can be mitigated by removing any unused functionality from the code and ensure that only generic error messages are shown.

7. Cross-Site Scripting (XSS). This type of vulnerability exists when web applications allow additional code in an URL path or page that will be viewed by a third party. A threat actor can execute a script in the victim's browser, using the URL of a page that at first sight would be trusted. To mitigate this type of vulnerability, in addition to the use of more modern

development frameworks, untrusted HTTP requests should be ignored, and a validation and sanitizion of content generated by users should be conducted.

8. Insecure Deserialization. This is a threat to web applications that serialize and deserialize data frequently. The exploitation of this vulnerability is the result of deserialization of data from unreliable sources, which may lead to consequences like denial of service or remote code execution. There are methods to identify this type of attack, monitor and validate deserialization, however, the only safe way to avoid this type the attack is by forbidding the deserialization of data from unstrusted sources.

9. Use of components with known vulnerabilities. Web application development usually follows a framework and the use of libraries. Some threat actors look for vulnerabilities on these components so that they can carry out attacks. Some of the most common elements are used by thousands of web applications, which leads to the discovery of a vulnerability in one of these components. To minimize the risk of using parts with known vulnerabilities, these should not be used and, if used, they have to be under constant monitoring as well as have the latest version of said libraries.

10. Insufficient Logging and Monitoring. Most of web applications don't take the steps needed to prevent a data breach. Plenty of studies show it takes about two hundred days to detect a breach after it happened which means that threat actors have two hundreds days to tamper, damage, extract or destroy data from the system before any type of response. With efficient logging, monitoring and a incident response plan, it is possible to reduce this time opportunity.

2.4 Laws and Regulations

2.4.1 Sarbanes-Oxley-Act

Sarbanes-Oxley-Act (SOX) [78] was the catalyst and the main driver of the need for a Governance Risk Compliance solution. After numerous scandal frauds, on July 30 of 2002 the U.S congress passed a law to protect investors and accounting firms from said scandals. This act set a precedent for all U.S companies:

- SOX act imposed criminal penalties and high fines for companies or individuals who attempt to defraud.
- Senior Executives have to sign financial reports statements to validate their accuracy, therefore becoming personally responsible.
- Companies are accountable to hire an external auditor to audit the accuracy of said reports, which must have a section for auditor opinion. This auditor must be in compliance with pre-established requirements.

- Companies have to fully describe their internal controls and how are they are being applied.
- All codes of conduct must be Sarbanes-Oxley compliant.
- Security and Exchange Commission (SEC) is now fully capable and has the authority to censure any suspicious actions from individuals, brokers, auditors or executives.

Due to this regulation, companies had to immediately respond by creating several controls to be compliant with the act [25]. However, these controls were addressed individually as audits came along and not as a comprehensive business process. Facing such issues, companies namely PriceWaterHouseCoopers in 2004 decided to address these questions by combining every business process through a universal perspective naming it Governance, Risk and Compliance.

2.4.2 European directive 2006/43/EC VIII

European directive 2006/43/EC VIII [21] belongs to European Company Law and Corporate Governance and it covers audit and accounting regulations in the European Commission. This directive acts like Sarbanes-Oxley-Act law in Europe, as it covers the responsible for approving statutory auditors and audit firms. A statutory audit is a required review of the accuracy of the financial reports made by external sources.

2.4.3 General Data Protection Regulation

General Data Protection Regulation [56] (GDPR) was issued on the 27th of April of 2016 on the European Union (EU) Official Journal. This new regulation introduced new regulatory requirements for the protection of individuals and the processing of personal data and their free movement. It was strictly applied directly to all the 28 Member States, without the need for any transposition legislation. GDPR encompasses all the data treatment processes, from the data controllers to the data processors.

A data controller is the entity that determines the purpose of processing personal data. The entity could be a natural or legal person, a company, public authority, or other agency. Data controllers determine the purpose and means of the processing of personal data. The Data processor is the natural or legal person, public authority, or another body that processes personal data on behalf of the data controller. Data processors do not determine the purpose of processing data. They must only process data in the way determined by the data controller.

GDPR is designed to protect personal data today and in the future. Its Objectives include:

- Provide a robust set of rules to protect the fundamental rights and freedoms of individuals in the EU.
- Enable the free movement of personal data within the EU.
- Harmonize data protection legislation across EU member states.

GDPR applies to all organizations that are established in the EU and process personal data in the context of that establishment. It also applies to organizations established outside of the EU if they process data on individuals in the EU when offering them goods and services, or monitoring their behaviour.

GDPR defines two types of different data:

- Sensitive data is the personal information that reveals a person's racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, health, or sexual orientation. Sensitive personal data also includes genetic data or biometric data. It requires a higher level of protection. Sensitive personal data also includes personal data relating to criminal convictions and offences as well as data that may facilitate identity theft or payment fraud (like social security files, financial account numbers, credit card details and government identification numbers).

- Personal Data is any information relating to an identified or identifiable natural person.

2.5 Governance Risk and Compliance

The Open Compliance and Ethics Group (OCEG) defines Governance Risk and Compliance as "a capability to reliably achieve objectives (governance) while addressing uncertainty (risk management) and acting with integrity (compliance)" [4].

As the sales world is shifting, the business of the future will require constant risk monitoring and reviewing. By controlling actions holistically and seeking to enhance efficiency, GRC provides an integrated comprehensive view of the risk of all key business units.

Acts such as Sarbanes-Oxley-Act, European directive 2006/43/EC VIII and General Data Protection Regulation require constant risk monitoring due to tighter regulations imposed by the government. To manage all the gathered information, it is crucial to have a system that has the essential tools for continuous growth, wise-decision making and a better understanding of the full scope of risk. The main focus of GRC is to maintain sensitive data and manage risk across the whole organization while being able to communicate risk posture to internal and external groups.

This section will focus on the most relevant GRC software according to the Forrester Wave report [2.5]. All the leaders from the reports are in line with Governance Risk and Compliance and they all share the following modules:

- Access Control is the foundation of access governance and aims to regulate access risk and thwart fraud by automating the administration of user access, applications, processes and data against the wrongful risk use. The need for quality access government is clear, since access control is a powerful tool to help organizations automate the process of managing authenticated users to solely access what they are approved to, while maintaining the perseverance of the whole system and the segregation of duties coherent. Access controls grant the ability to lower potential internal fraud while improving user experience and increasing

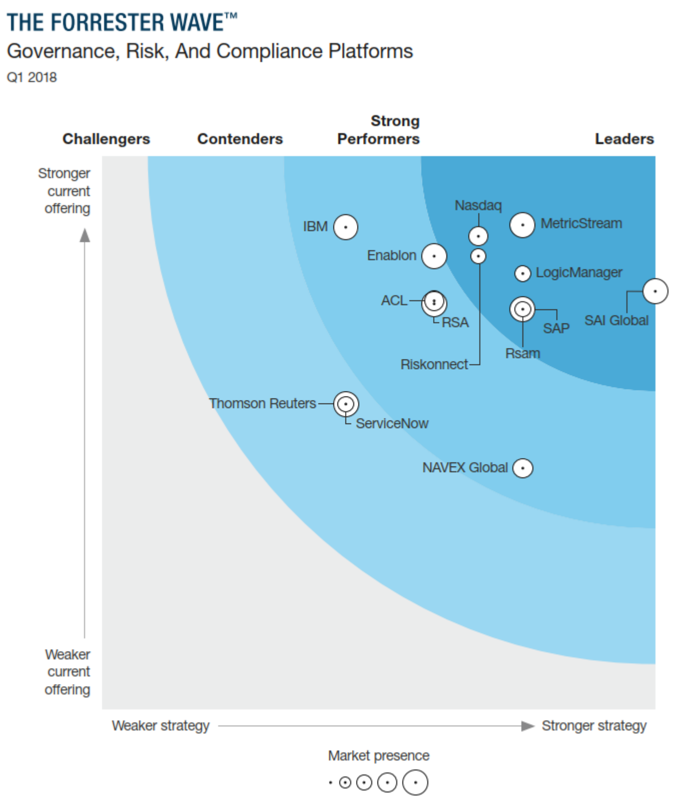


Figure 2.5: Forrester Wave chart for Enterprise Governance, Risk and Compliance Platforms.

productivity. This module conjointly permits the unceasingly monitorization of transactions against current policies in order to detect anomalies and prevent cash leakage.

- In some solutions, Access Control acts as two modules, Application Access Controls Governor (AACG) and Enterprise Transaction Controls Governor (ETCG). This latest module provides the ability to track access from users by creating "continuous controls", granting real-time monitoring and segregation of duties to ensure regulatory requirements and corporate security

- Process Control grants the effectiveness of the controls and ongoing compliance. These controls allow an enterprise to endlessly access existing controls in real-time, in order to analyze if they are being applied properly thorough respective business areas, and if they are aligned with each risk prevention measure. Process Control enables a company to define controls to apply to risks, assets and all business processes. It works as a journal of the enterprise strategy to address risk management according to regulatory requirements. It provides an integrated way to reduce time and costs necessary to understand regulatory requirements.

  Process Control supports companies having regulatory challenges with library management, according to the life cycle of policy management, forcing them to design, publish, implement and track policies through the organization. It also supplies an agile analysis of regulatory compliance from distinct perspectives. Through a complete guideline, it is possible to audit IT risks and understand if they are being mitigated accordingly.

- Risk Management regulates risk crosswise the organization, incorporating risk experts from diverse lines of business and facilitating the execution of risk assessment, in order to reduce its cost. It helps the risk management team to address the specific challenges around operational risk, offering key solution features such as key risk indicator, single data repository, business intelligence, decision support and loss event management. Through dashboards, risk indicators, audit trailing and statistical analysis, risk management provides the capital range that needs to be allotted to address the company's operational risk, by collecting the data that might be relevant to run these capital models. It follows the established flow of ISO 31000 [34]. It also provides mechanisms to trace, monitor and review the progress. Lastly, risk management provides a full analysis of whether the organization is compliant with all the Information Security standards such as ISO/IEC 27001, Information Security Forum (ISF) and National Institute of Standards and Technology (NIST). This solution allows to complete risk management related activities such as the assessment of assets, threats, existing controls, vulnerabilities and impacts.

- Audit Management (AM) is often incorporated into risk management, and it supplies internal auditors with an exclusive view of the whole GRC. AM combines every department, giving the auditor a comprehensive perspective of each department compliance management activities to provide an automated auditing procedure.

  With the engagement activities, annual planning and work paper management that AM offers it is possible to understand the full scope of risk and how to proceed before each audit. This solution is Sarbanes-Oxley Act compliant, providing mechanisms to review and monitor financial compliance obligations. It enables the processing and harmonization of the audit environment, since audits are based on templates. With these templates, Audit Management compiles reports from audit outcome activities in order to analyze the results.

2.6 Eramba

Governance Risk and Compliance is often not supported by centralized software that incorporates these three fields. Many companies offer only one category of the three. For example, Oracle provides a solution to deal with Risk management named ERP Risk Management exclusively. This solution has embedded AI techniques to analyze risk through the organization; IBM offers a solution to deal with Compliance called IBM RegTech to handle regulatory monitoring and Compliance within the financial organization; IT governance can be handled with a solution from BWise that allows creating a comprehensive, accurate and holistic view of IT operations and assets through the organization. These examples are just a portion of software used in the industry. There are many more, ranging from free open source software to more costly options.

Some companies that can not afford costly GRC systems, opt to choose Eramba as their solution. With an ergonomic user interface, it offers eight different categories with six different modules that interact with each other. It also provides numerous functions (that are listed on the table A.1) each with the description of its main purpose. All these functions are the blueprints to be in accordance with governance, risk and compliance.

Eramba splits the main modules into two different categories, "Problems" and "Solutions". Problems are the cause of an organization implementing a solution. If there is no problem, there is no need for a solution. Problems may vary from future audit (compliance management) and critical assets with no controls (risk management) to the understanding the correct flow of the data (data flow analysis). Solutions are implemented after all the problems are well-established and there is a clear need for them. The Figure 2.6 demonstrates how the flow is distributed between each problem and solution.

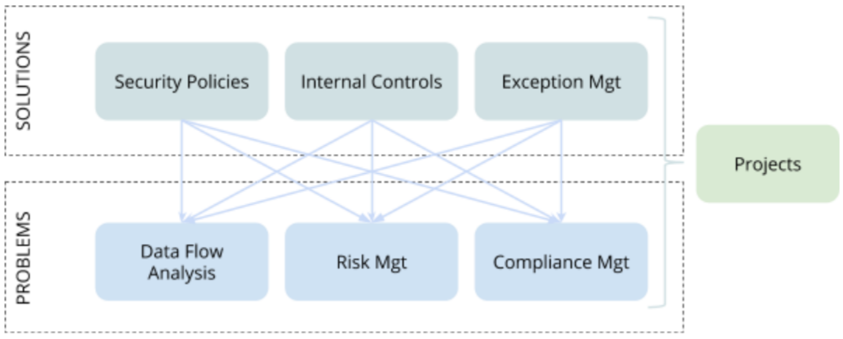


Figure 2.6: Eramba Problems and Solutions [19].

In Eramba there are some modules that are required by default and some are optional. There are six different modules in total: Organization, Risk Management, Asset Management, Control Catalogue, Compliance Management and Security Operations.

- Organization Module. In this module it is possible to describe organisation business units (For example Finance, IT, Human Resources) along with their core processes, all the liabilities embedded in the scope of the GRC program (such as GDPR, Brand Reputation, Financial and Accouting Obligations) and all the third parties involved (PCI-DSS, ISO).

- Risk management Module. This is one of the most important modules as it gives an overview of how assets' risks are managed across the organization. Besides managing all risks across the organization, this module also provides a method to manage risk exceptions.

- Asset Management Module. This module provides asset identification, along with data flow analysis.

- Control Catalogue Module. Controls in risk management are pivotal, as they are able to reduce or even mitigate the risk. In this module are described all internal controls, business continuity plans and security policies.

- Compliance Management Module. This model allows the management of all exceptions to compliance, the insertion of compliance package with their due requirements (ISO/IEC 27001 for example) and its full analysis.
- Security Operations Module. This is an extremely essential module as it guides the definition of improvements across the organization. It is also possible to record and manage security incidents which can then be linked to controls, assets and third parties.

2.7 Threat modelling

According to NIST, "Threat modeling is a form of risk assessment that models aspects of the attack and defense sides of a particular logical entity, such as a piece of data, an application, a host, a system, or an environment. The fundamental principle underlying threat modeling is that there are always limited resources for security and it is necessary to determine how to use those limited resources effectively [69]".

In short, threat modelling is a process by which it is possible to determine which threats are important to an application and find entry points where defences may be lacking. To establish a threat modelling it is vital to determine potential entry points (this depends on the type of system), protected resources or assets and to examine and describe data flow paths. With threat modelling, it is possible to find security issues by identifying and mitigating possible threats.

Examples of different methodologies for threat modelling are STRIDE, P.A.S.T.A, Trike, VAST and OCTAVE.

The STRIDE methodology was described by Microsoft and currently is one of the most mature threat-modelling methods [73]. It is an mnemonic that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege:

- Spoofing is the act of impersonation something or someone else.
- Tampering is the process of modifying data or code.
- Repudiation is to claim the performance of an action.
- Information Disclosure is the exposure of information to someone not authorized to see it.
- Denial of Service is the constant denial of service to users.
- Elevation of Privilege is the ability to gain access without authorization.

The STRIDE model was designed as a mnemonic framework to provide guidance while developing software, and identify possible threats and attacks of said software.

Shostack [1] describes a four-step framework to address threat modelling:

- Model System - Understanding how the system works by modelling it using Data Flow Diagrams (DFD).

- Find Threats - Classification and enumeration of threats using STRIDE.
- Address Threats - Addressing threats by mitigating and reducing threats.
- Validate - Correctly verify if said threats were in fact mitigated.

Also known as threat model diagrams, DFD are graphic representations to portrait how the flow of data is conducted through the whole system. This flow may be external or internal, thus being possible to analyse the data boundaries of the system. Trust boundaries are where entities with different privilege interact with others entities. These trust boundaries isolate trustworthy and untrustworthy elements.

According to Shostack, the elements of a DFD are external entities, trust boundaries, data storage, processes, data flow and multi-processes each with different notation represented in Figure 2.7.

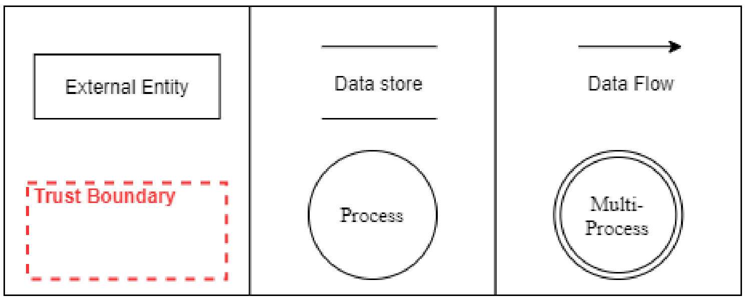


Figure 2.7: Data Flow Diagram Elements.

There are different variants to perform a STRIDE-based threat modeling: STRIDE-per-element, Stride-per-Interaction and DESIST. STRIDE-per element process applies STRIDE to each element.

STRIDE-per-Interaction finds data flow at an intersection of a trust boundary. It finds threat at origin, destination and interaction in a Data Flow. To create a STRIDE-per-interaction, first create a table of elements, interactions and potential threats, afterwards make a DFD, then extract the data flow at the intersection of trust boundary, subsequently enumerate threats and then lastly create a table of the comparison result. DESIST is an acronym for Dispute, Elevation of privilege, Spoofing, Information disclosure, Service denial, and Tampering. In DESIST, Dispute replaces repudiation and Service denial replaces Denial of Service.

2.8 Cloud Computing

"Cloud computing is the delivery of computing services—servers, storage, databases, networking, software, analytics, intelligence and more—over the internet (the Cloud), enabling faster innovation, flexible resources, and economies of scale" [51].

Cloud services provide distinct attributes and considerations from traditional On Premises service. It provides a safer and more secure environment to host a web application such as Eramba.

Some services described may be more important and relevant than others, however each one plays their part:

- High availability: The ability to keep the service up and running for long periods of time with little to no downtime is fundamental to an application that should be running 24/7.
- Elasticity: The ability to automatically or dynamically increase or decrease resources as needed. Although not common, this may be vital if many users need to access an application at the same time.
- Fault tolerance: The ability to remain up and running even in the event of a component or service is no longer functioning.
- Disaster Recovery: The ability to recover from an event which has taken down a Cloud service.
- Scalability: The ability to increase or decrease resources given any workload.
- Agility: The ability to react quickly. Cloud services can allocate and deallocate resources quickly. They are provided on-demand via self-service, so vast amounts of computing resources can be provisioned in minutes. There is no manual intervention in provisioning or deprovisioning services.
- Global reach: The ability to reach audiences around the globe is fundamental when Eramba is used at a multinational level.
- User Latency capabilities: If users are experiencing slowness with a particular Cloud service, they are said to be experiencing some latency. Cloud services have the ability to deploy resources in datacenters around the globe, thus addressing user latency issues.
- Predictive cost considerations: It is possible to predict costs for each service hosted in Cloud thus creating a budget to host Eramba annually or monthly.
- Technical skill requirements and considerations: Cloud services can provide and manage hardware and software for workloads. A user can be expert in the application they want to run without having the need for specialized skills to build and maintain the underlying hardware and software infrastructure.
- Security: Cloud providers offer a broad set of policies, technologies, controls, and expert technology skills that can provide better security than most organizations can otherwise achieve. This is essential to improve the security of Eramba.

The Cloud model provides three different types of configuration.

- Public Cloud: A public Cloud is owned by the Cloud services provider (CSP). It provides resources and services to multiple organizations and users, who connect to the Cloud service via a secure network connection, typically over the internet. With a public Cloud, there is no local hardware to manage or keep updated, as everything runs on the Cloud provider's hardware.

- Private Cloud: A private Cloud is owned and operated by the organization that uses the resources from that Cloud. The organization creates a Cloud environment in their own data center, and provides self-service access to compute resources to users within their organization.
- Hybrid Cloud: A hybrid Cloud combines both public and private Clouds, allowing critical applications to run in the private Cloud and other non-critical on the public Cloud.

Furthermore, Cloud Computing offers three different types of service Cloud, each one with different levels of controls and flexibility.

- Infrastructure as a service (IaaS): IaaS is the most basic category of Cloud Computing services. With IaaS, users rent IT infrastructure servers and virtual machines (VMs), storage, networks, and operating systems from a Cloud provider on a pay-as-you-go basis. It is an instant computing infrastructure, provisioned and managed over the Internet.
- Platform as a Service (PaaS): PaaS provides an environment for building, testing, and deploying software applications. The goal of PaaS is to help create an application as quickly as possible without having to worry about managing the underlying infrastructure. For example, when deploying a web application using PaaS, it is not required to install an operating system, web server, or even system updates. PaaS is a complete development and deployment environment in the Cloud, with resources that enable organizations to deliver everything from simple Cloud-based apps to sophisticated Cloud-enabled enterprise applications. Resources are purchased from a Cloud service provider on a pay-as-you-go basis and accessed over the Internet.
- Software as a Service (SaaS): SaaS is software that is centrally hosted and managed for the end customer. It allows users to connect to and use Cloud-based apps over the internet. SaaS is typically licensed through a monthly or annual subscription.

To sum up, the Figure 2.8 summarizes each model. Each one contains different levels of managed services and their usage relies on how responsibilities are distributed in the company. With these different models, it is now possible to virtualize operating systems in the Cloud. A notable approach to virtualize OS are containers.

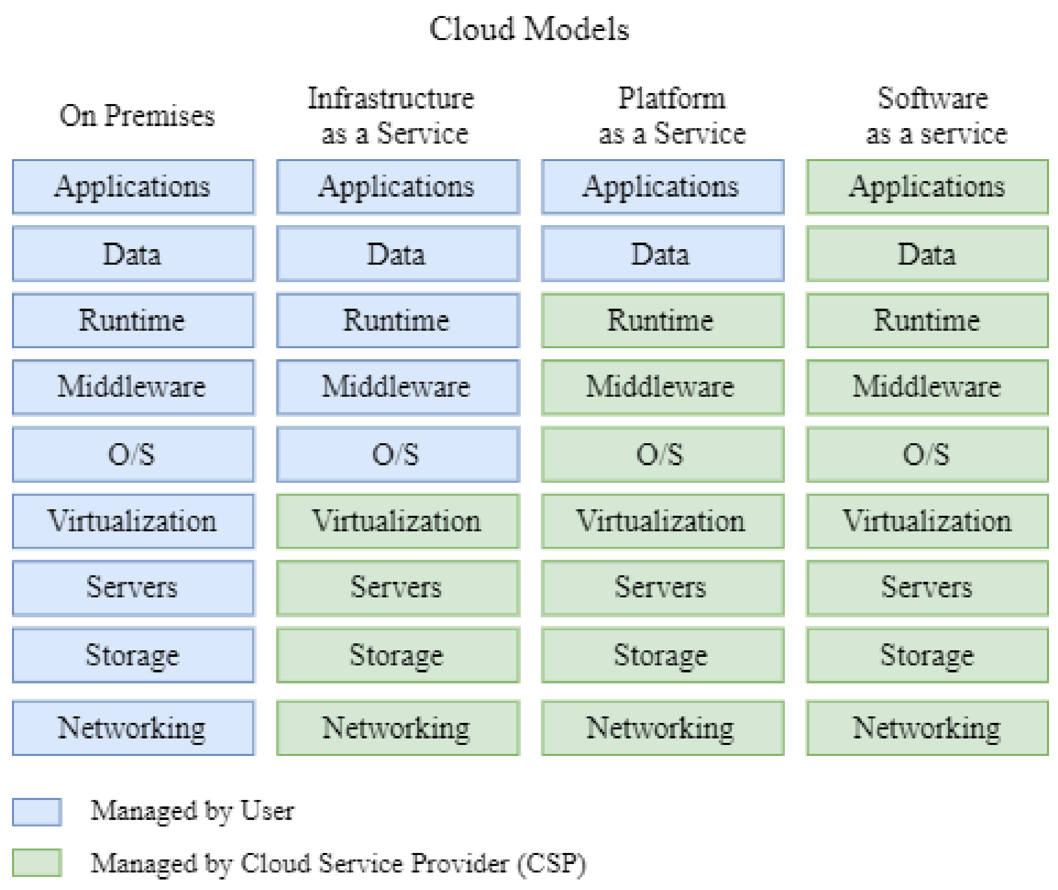


Figure 2.8: Cloud Models

2.8.1 Requirements for Cloud Computing Architecture

There are three architectural requirements when migrating to Cloud Computing services [67]: Cloud service provider requirements, organization requirements, user requirements. The Cloud service provider requires a highly efficient and reliable architecture to support organization infrastructures. This architecture must be able to virtualize services, support fault tolerance (systems continue to operate in the event of failure of some components) and provide storage mechanisms to handle and store as much data as needed. All these usually are supplied via cheap tariffs with on-demand support.

The organization requires the capacity to host their business services model via Cloud models discussed in Section 2.8. As the CSP provides interoperability and scalability, these services are easily adaptable to the CSP infrastructure. The user requires a simple interface with a low learning curve associated and a self-learning capability that allows comprehending the extra features that Cloud Computing brings, such as pricing, metering, and service level agreements (SLA). User privacy is now enhanced, as encryption and decryption operations increase the stability and usability of Cloud services.

These requirements are directly dependent on each other. CSP must be aware of transparency for billing, data governance, and user privacy; Otherwise, the user will not trust the CSP any-more. Data security issues must be fixed and disclosed to CSP clients, or organizations will no longer trust the provider to host their applications. Additionally, CSP features such as security, compliance, reliability and SLA will impact the organization's performance.

2.9 Containers

Containers are an encapsulation of an application with all its dependencies [48]. They are an isolated environment contained in a server that, unlike virtual machines, shares a single system kernel. Each one has their own processes for services, network interfaces and mounts.

A container virtualizes the underlying OS, thus no longer requiring an OS per application. In essence, the container isolation allows the container image to perceive all the underlying components (storage, RAM, CPU and networking connection) for itself.

Since a container holds an isolated instance of an OS, it enables a lightweight and efficient deployment regardless of whether the environment is in an On Premises, private or hybrid Cloud. It also enables developers to not have to be concerned with installation and configuration issues.

Additionally, by having a text file with all the commands required to assemble an image and its dependencies, the software and its dependencies became less vulnerability-prone as the patch only requires minor changes to the file.

The Table 2.2 establish the traits of containers versus virtual machines as well as the exposure of high-value arguments for container implementation.

Containers	**Virtual Machines**
Multiple containers can run on a single physical or virtual machine (high density), enabling isolation at the process level, therefore, providing additional isolation features such as namespaces, cgroups, and other kernel capabilities	A few VMs can run together on a single physical machine (low density), relying on complete isolation of VMs for security
Containers share the same kernel as their Docker Host	Each VM has its own OS and the physical resources are managed by an underlying hypervisor
Containers leverage standard inter process communications mechanisms, such as signals, pipes, sockets, and so on, for networking. Each container gets its own network and storage stack.	For networking, VMs can be linked to virtual or physical switches. Hypervisors have buffer for I/O perfomance improvement, NIC bonding, and so on.

Table 2.2: Containers versus Virtual Machines [22].

Containers also provide innumerable advantages versus a virtual machine as they occupy less space and have shorter boot-up time. The containers have a better performance as they are hosted in a single container, easy to scale up, highly efficient, easily portable and the data volumes can be shared and reused among multiple containers.

The ideal environment is to have containers provisioned on virtual machine hosts, to utilize advantages of both technologies. Use the benefits of virtualization to easily provision or decommission machine hosts as required, and at the same time make use of containers to easily provision application and quickly scale them as needed.

There are varied Linux containers solution (LXC) such as Docker, LXC, LXD, Rocket and Warden.

2.9.1 Docker

Docker was used to migrate Eramba to a new environment due to a considerable amount of reasons: the reproducibility (a Docker container operates the same way in any machine), isolation (discussed on sub-section 2.9.1), security (if one container is compromised the others remain unaffected), the Docker hub (a hosted repository with official images from components providers), continuous integration and support, ease of use and a favorable environment manager which allows to have separate containers for testing, development and production as well as facilitated deployment. Additionally, Docker has an extensive community with plenty of documentation which provides insights of a improved implementation.

Docker is a software development container platform on a host machine that enables to build and run containers using Docker components and services. The Docker engine, also known as Docker daemon, implements the blueprint that provides the specification for container images, run time and network connectivity. The migration to Docker is also an effortless task as it can run in any machine that runs Docker.

Docker images are sets of layers with metadata in JSON format. To successfully run a Docker container, it is only required to build the Dockerfile and use Docker command build to run it.

In order to conceive multiple containers, Docker provides Docker compose. Docker Compose is a solution that allows the creation of multiple containers in a coordinated way, making use of a file in YAML format for the configuration of each of these elements and the networks used in communication between them.

With this new environment, containers face different security issues than the usual virtual machine implementation.

Docker security [17] relies on four major areas: isolation of processes at the userspace level, the attack surface of the Docker engine, the hardening of the kernel and how it interacts with other containers and loopholes in container configuration.

- Isolation: Docker containers are started by default with a restricted set of capabilities due to their dependency of Linux kernel features such as namespaces, control groups (cgroups), and network interface. With namespaces, processes running in a container can not see any other process running in the host system or other container. Also, each container has their own network and storage stack, meaning that there is no interference with other containers. On the side of namespaces, cgroups, besides managing memory, CPU and the disk used by the container, guarantee that a single container can not bring down the host by exhausting one of those resources. All in all, these capabilities turn the root and the non-root binary into an access control system, meaning that the root of the container is still the root of the host but with less capabilities and privileges.

- Docker daemon attack surface: Docker engine requires root privileges, thus implying that only trusted users should be allowed to control the Docker daemon. As Docker allows sharing directory between the host and guest containers without limiting the access rights of the container, it is possible to start the container in the host's root directory. This means that the guest can alter the host file systems without any restrictions.

- Hardening of the kernel: Host hardening involves reducing the attack surface by removing unnecessary features or settings. Measures such as a policy for putting all Docker objects in same domain or creating separate partition for containers, audit Docker files and logs improve the host hardening. Security modules such as AppArmor, SELinux, GSREC, Seccomp, along others, also support host hardening by allowing users to specify file paths to a binary, specifying permissions they have or custom actions to be taken when a system call is called.

- Loopholes in container configuration: When configuring the Dockerfile it is important to set the restart policy correctly, otherwise the container can enter in a loop cycle.

2.9.2 Container registries

Container registries are stored locations where it is possible to push and pull containers' images. Docker Hub is the most used container registry as it supplies an extensive amount of container images provided by third party (open-source, software vendors) or official sources. Docker container registry has direct integration with Gitlab [27], so, with the correct environment, it is possible to have Docker images hosted on Gitlab. With the continuous integration and continuous delivery (CI/CD) environment provided by Gitlab shown with the script, the Docker image is always built from Dockerfile whenever there is a change in the file and the file is pushed to the Git. This is a simple script that creates an environment to fully deploy a container.

```
stages:
    - build
registerContainer:
    stage: build
    image: Docker:stable
    services:
        - Docker:18-dind
    variables:
        TAG: latest
    script:
        - echo $CI_REGISTRY_PASSWORD | Docker login
          $CI_REGISTRY -u $CI_REGISTRY_USER --password-stdin
        - Docker build -t "$CI_REGISTRY_IMAGE:$TAG" .
        - Docker push $CI_REGISTRY_IMAGE
```

Listing 2.1: Docker CI/CD integration script.

By having an application containerized, there is now the opportunity to scale the application up in pursuance of reaching organization users worldwide. Therefore, there is an urge for container orchestration.

2.9.3 Container Orchestration

With this new environment of portability and reproducibility that are containers, the opportunity surges to scale containerized applications across Cloud Computing service. As containers run similarly everywhere, scaling up the service as business needs demand and building services across multiple machines without dealing with burdensome network settings becomes a straightforward challenge with the help of orchestration. Container orchestration defines the relationship between containers: how will they scale, how they connect to the world and where do they come from. It also provides redundancy and availability, allocation of resources between each container, load balancing of service and health monitoring [48]. A container orchestration solution easily allows to deploy thousands of instances with a simple command.

Tools such as Docker Swarm from Dockers, Mesos from Apache or Kubernetes from Google guarantee that containers maintenance, management and escalation work as a smooth operation.

Kubernetes was the tool chosen for this project, due to the fact that despite the CSP selected was Google, all other major CSP also have native support for it. The huge ecosystem Kubernetes has and also its documentation and attractiveness made it the ideal tool for this project.

2.10 Kubernetes

Kubernetes, also known as k8s, is "a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation" [44]. It provides tools to auto-scale, roll, deploy, compute resource and manage volumes across containerized applications. Like containers, k8s are designed to run anywhere, being able to be run on a data center, a single machine, public Cloud, a hybrid Cloud or a private Cloud. Kubernetes enforces implementation concepts of how containers and networks are organized. There are some essential concepts that must be understood to fully comprehend Kubernetes.

A Kubernetes cluster consists of a set of nodes. They are the foundation of Kubernetes. When running Kubernetes, a cluster is being executed. It is a set of nodes grouped together, meaning that even if one node fails, the application is still accessible from the other nodes.

A node is a physical or virtual machine on which Kubernetes software is installed. It is considered as a worker machine where containers will be launched by Kubernetes. There are master nodes and regular nodes. A master node controls and schedules all the activities in the cluster. The master is responsible for deciding nodes operation such as scheduling workloads, managing life cycle, scaling and upgrading. It also manages network and storage for the workloads. The master watches over the nodes in the cluster and is responsible for the actual orchestration of containers. It is required to have more than one node to maintain the application, if for some reason the only node fails.

When installing Kubernetes on a system, different components are installed: API server, etcd, kubelet, a container runtime, different controllers and a scheduler.

The API server acts as the front end for Kubernetes, with all the task management being accomplished through this server.

Etcd is a distributed high accessibility reliable key value store used by Kubernetes to store all data used to manage the cluster. It is responsible for implementing logs within the cluster to ensure that there are no conflicts between the master.

The scheduler is responsible for distributing work on containers across multiple nodes. It looks for newly created containers and assigns them to nodes.

The controllers are responsible for noticing and responding when nodes or endpoints go down. For instance, one important controller is the replication controller. It guarantees that at any time, there is always a homogeneous set of pods up and available. Another important controller is the controller manager, responsible for gathering and sending data to the API server.

The container runtime is the underlying software that is used to run containers according to their environment, which in this situation is Docker containers.

Kubelet is the Kubernetes' node agent that runs on each node in the cluster. The agent is responsible for making sure that the containers are running on the nodes as expected. It report activities back to the master, such as pod and node health, as well as liveness probe. It is also responsible for starting and running containers scheduled on that node.

Kubectl is the command-line tool that allows to run commands against Kubernetes clusters. As Kubernetes is an HTTP REST API, kubectl's main job is to carry out HTTP requests to the Kubernetes API. It allows to deploy applications, inspect and manage resources as well as view cluster logs.

Networking in Kubernetes is slightly different than Docker networking. In Docker, containers have a private subnet and can not communicate directly with containers in different hosts without port forwarding or proxy. In Kubernetes, containers within a pod share the same virtual IP address and port, which means they can find each other through localhost. Meaning, it is no longer required any network translation operation.

Other important concepts regarding Kubernets are Pods, Labels and Service.

- Pods are groups of one or more containers that are deployed and scheduled together. The deployment has a shared storage/network and instruction on how to run the containers. Typically, the containers in a pod work together to provide a service. Additionally, Kubernetes itself provides other containers to monitor and log the services. Nodes are the workers that host the Pods. When a worker node dies, the pods on it are also lost.

- Labels. Labels are key-value pairs used to describe identifying characteristics of the object. They identify attribute of objects that are meaningful and relevant to users. They can be used to describe, for instance, if the environment is in testing phase or in production phase.

- A service is a REST object and an abstract way to expose an application running. Via label selectors, a service can be connected to pods. Services provide the layer of abstraction required so that applications do not need to know the details of the service they are calling, and the application code only requires the name and port of service of the database in order to call it.

Securing a Kubernetes environment

Security-wise, Kubernetes approaches security as a defense in depth and distinguishes four different layers of security which are the 4C [45]. Cloud, Clusters, Containers and Code. Each layer builds for the other, meaning that Code represents the central layer, followed by Containers, which are then followed by Clusters and finally Cloud. Cloud security differs from CSP to CSP as each has different features, however, most of them operate in a similar manner. Google Cloud Platform security options are discussed in Section 4.3.

To secure a cluster there are a few steps that can be taken [46]. It is required to control the access to the Kubernetes API, as everything in a Kubernetes cluster is controlled via an API. Regulating who can access the API and its actions is fundamental. To correctly monitor API access, a TLS should be used for all traffic. The authentication mechanism used must match the cluster size: after correctly authenticated, each API call is expected to pass the authorization phase. As Kubernetes has a RBAC integrated component, every user has a role assigned with permission of get, create or delete with specified resources.

Kubelet allows unauthenticated access to the API and it exposes powerful endpoints that grant control over the node and container. To monitor the access to the kubelet there are some additional settings such as x509 client certificates and bearer tokens. Additionally, kubelet allows to delegate the access to the kubelet API, meaning that not everyone who is authenticated and with authorization can access some resources. Kubelet has a set of different policies that limit access, usage and additionally monitor how those resources act on the cluster, for example, network policies to restrict which pods in other namespaces have permission to access other pods or pods within their namespaces. With these policies it is also possible to restrict Cloud metadata provided by the CSP and to control to which nodes pods may access.

Pods definition may contain an additional field which is securityContext: this field allows to specify what kind of Linux user can run the pod. By default, specific kernel modules are loaded when needed, for instance, when a filesystem is mounted, to prevent this it is possible to configure a file to disable those modules.

It is also possible to protect cluster components (etcd) from future compromise. Gaining access to the etcd back-end for the API is an attack surface, so it is recommended to use strong authentication mechanism such as TLS client certifications and to isolate etcd servers behind a firewall. It is recommended to enable audit log despite being a beta feature as of the time of writing, it is also recommended to disable features that are in alpha stage however this differs from the risk appetite of the organization, rotate credentials frequently to reduce the lifespan if an attacker gets access. Moderate third party integration to cluster before enabling them. Also Kubernetes will encrypt all traffic through the etcd database.

Container security was already discussed in the Section [2.9]. All in all, it is recommended to scan the container for known vulnerabilities to figure out what we are dealing with, to sign container images to verify the authenticity of said images. Also it is important to monitor who can access the container and follow the principle of least-privilege.

Code layer is the most vulnerable attack surfaces of the four layers. The developer must build the application with security in mind following popular frameworks, OWASP Top 10, get an external security audit, access over TLS only, encrypt everything by default, analyse statically the code and use automated tools to verify if the application is secure.

Chapter 3

Eramba Solution Analysis

In this chapter, the Penetration Testing Execution Standard (PTES) [76] methodology was followed. The decisions of why PTES was chosen have already been discussed in the previous chapter (2.2.5).

On this section, while using PTES a full vulnerability analysis of Eramba was persecuted.

Although it is a seven-phase methodology, threat modeling, vulnerability analysis, exploitation, and post-exploitation were studied at a comparably large extent versus pre-engagement and intelligence gathering. The reporting phase was not done.

Pre-engagement and intelligence gathering had lesser attention since both are minor on this project and did not influence the pentest's result. The reporting phase was not conducted because the vulnerability analysis and post-exploitation provided enough information to serve as a report.

As PTES was fulfilled in a restrained laboratory - access to the organization's private network via a VPN with full access to Eramba in a subnet dedicated to it - the Rules of Engagement (RoE) were considerably low. Hence, the pre-engagement interactions were almost non-existent. Additionally, there was access to many tools, which made it unclear which tool should be chosen. Furthermore, the intelligence gathering was done at a level-two degree ("level two is done via using the automated tools from level one plus manual analysis") since there were no limits imposed to physically access the machine where Eramba was hosted in, nor limitations on the virtual machine.

The choice of STRIDE for threat modelling is motivated due to several reasons: it is one of the most used threat-modeling methods in the industry; it is a technical approach to cyber threats; it is comprehensive and analyzes security properties such as authentication, authorization, confidentiality, integrity, nonrepudiation and availability against each system component; it provides a clear understanding of the impact of a component vulnerability on the entire system and helps ensure system security as the component level [42].

The threat modelling was built following Adam Shostack framework [1] using Microsoft methodology named STRIDE to find, rank and enumerate threats. On this phase a study of how the application is built, what are the main assets and how the authentication mechanisms work was compassed.

The vulnerabilities analysis phase and exploitation were made together and are discussed in the same Section, Section 3.2. This was because right after the vulnerability was found, an ex-

ploitation was provided for it. In some cases, some vulnerabilities are false positives, meaning that the network tool used provided an alert for the vulnerability found. However, with manual studying, it was proven that there was no vulnerability.
Since Eramba has over three hundred enterprise users and over ten thousand community downloads, plentiful of pentests were persecuted by different teams [15] [60], meaning that the vulnerabilities found were low level and easily fixable in the next chapter. Additionally, the post-exploitation phase provides guidelines to improve Eramba security and is also affiliated with the next chapter.
It was not possible to cover all the tests on this document as there were multiple attempts making use of different network tools to find different types of exploits, vulnerabilities and flaws - tools such as metasploit which is a "pentest platform that enables to find, exploit and validate vulnerabilities" [41]. With metasploit there was an attempt to gain a reverse shell via PHP and a corrupted PDF file using "adobe_utilprintf" payload; With dirbuster, which is a "mutlithreaded java application designed to brute-force directories and files names on web/application servers" [41] there was an attempt to mass list as directories on the web application but unsuccessful; With Nikto which is an "Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files/programs" [41] some tests were made but nothing too conclusive, and with SQLmap, which is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers" [41] plenty of injections were tried but with no return. However, as the Acutenix report shows [15], Eramba does not have any injectable fields so it was not possible to manipulate the database. Even though there is an Acutenix report showing no alerts, the researcher also tried to detect cross-site scripting opportunities using XXSer [41] which is an "automatic -framework- to detect, exploit and report XSS vulnerabilities in web-based applications" [41].

3.1 Eramba threat model

To diagram Eramba, a STRIDE-per-element was chosen, as it shows how certain threats are more prevalent than others according to each element. This variant focuses on a set of threats against each data flow diagram element described on Table 3.1. The threat model of Eramba was then designed and is described on the Figure 3.1.

Data Flow Diagram Element	**S**	**T**	**R**	**I**	**D**	**E**
External Entity	X		X			
Process	X	X	X	X	X	X
Data Flow		X		X	X	
Data Store		X		X	X	

Table 3.1: STRIDE-per-Element.

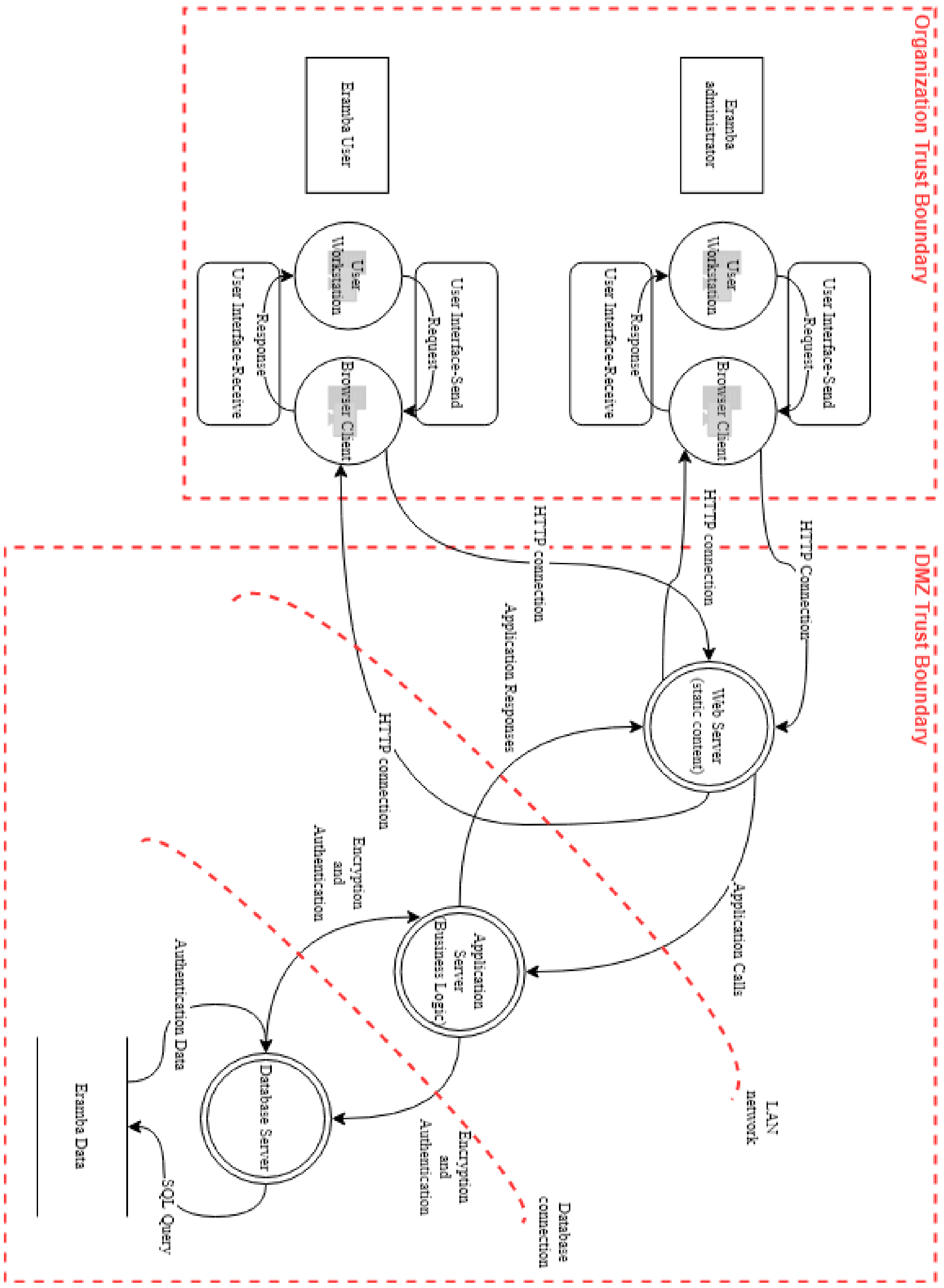


Figure 3.1: Threat Model.

It is also fundamental to understand the underlying architecture of Eramba, the business logic behind it and the database. What is the path an HTTP request goes through when clicking on the URL? What language was the application written in? What is the ideal configuration and how is the authentication done? These next subsections will focus on how Eramba is built. All these configurations are based on Eramba's guidelines provided by their website [20].

Model View Controller

Model View Controller (MVC) is the standard for assembling the code of interfaces with the user. It handles output, how data is shown and its appearance. Additionally, it also handles input, it selects the view and the data to display, makes validations, makes the necessary changes in the model and decides what happens next. [43]. MVC models guarantee separation of business logic from data and presentation layers. As MVC is a standard in web application architectures, there are numerous frameworks that facilitate this process, namely CakePHP, Laravel and CodeIgniter. CakePHP was used to create this application. The Figure 3.2 simplifies how an HTTP request is handle in Eramba.

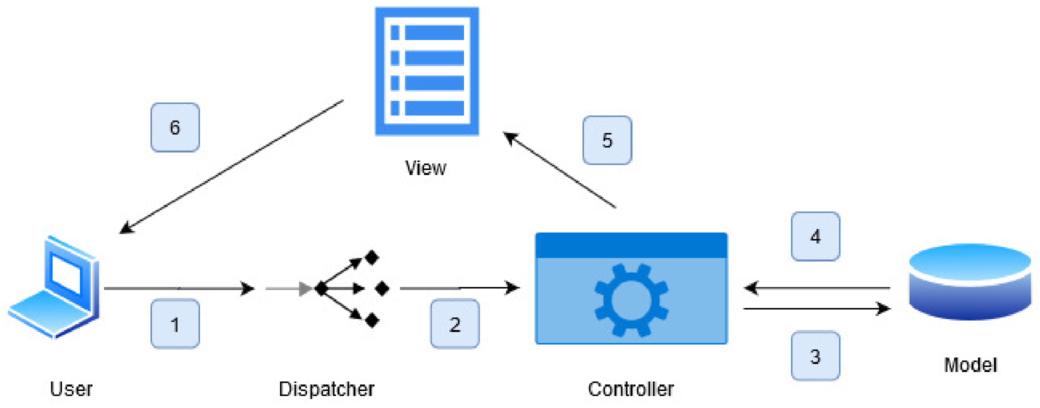


Figure 3.2: Eramba Model View Controller.

Upon requesting a page or resource from Eramba, the request will go to the dispatcher; The dispatcher locates and loads the correct controller; The controller, which is the connection between view and the model, will also communicate with the model to process the data requirements. After collecting the data, the request is sent back to the controller which will redirect to view. Lastly, the view generates the output to the user terminating the request cycle. Eramba is a REST web application. A REST API consists of an assembly of interlinked resources. REST API operations allow being called from any HTTP client, thus meaning that for Eramba to work it is only required a workstation with an HTTP client.

Configuration

Eramba runs using Apache, PHP, a Database and a Linux Server. It provides two different techniques to configure it: Either by the source code provided in a ZIP file, or a virtual machine (VM) to host the software.

To configure the VM, it is required to have an Open Virtualization Format (OVF) compatible Virtual Machine. The PHP version must be 7.1, 7.2 or 7.3; MySQL version should be higher than 5.6.5; MariaDB, which is a open source relational databases, should be 10.x version; the Apache version should be higher than 2.2 and OpenSSL should be higher than 0.9.x.

Database Structure

Eramba database is built on MySQL and it has 346 tables. When configuring the database there are some settings that should be set, namely the maximum allowed packet setting. This value indicates the maximum size of one packet sent by MySQL and must be equal to two hundred units. InnoDB is the engine used to manage MySQL database system. The innodb lock wait timeout, which is "the length of time in seconds an InnoDB transaction waits for a rowlock before giving up", should be at least set for two hundred units as it is the ideal time before of a search within a row in a database. A rowlock means that the SQL will only lock the affected row and not the entire table of a database when executing the delete operation.

Authentication and Access Management

As Eramba is expected to have multiple users, each one with different permissions, it uses a Role-based access control system (RBAC). The main objective in guaranteeing an access control based on profiles is to guarantee that the users of a given system do not have indiscriminate access to the features and information. With the use of this type of access control, users can only access the resources required through the profiles assigned to it. It is a simplified management of authorisations while adding flexibility in specifying and ensuring compliance with appropriate protection policies. Users can be assigned of a certain profile according to their responsibilities and qualifications, and in addition, profiles can easily be assigned or removed. Profiles can also be changed without the need to change the access infrastructure as well. With the use of RBAC, the decisions of who can access which information are directly related with the user role in organization.

RBAC means that each profile will be associated with a set of operations, and users are appropriately assigned certain profiles [24].

For instance, a user may be associated with one or more profiles, and a certain profile may be associated with one or more users. Profiles can be created according to the users' function in the organisation, and the set of operations associated with a profile will restrict users to that same set of operations.

This kind of access control is effective for: systems which treat sensitive information, systems that support the specification of competencies to perform certain tasks, systems that highlight the specification of rules which avoids conflicts of interest and systems that promote the principle of minimum privilege.

In Eramba, users are given a role by the administrator, each one with different permissions. These permissions may range from add, delete and edit content. Eramba has five different portals for the users to interact with, each one providing different functions and each having their own authentication options and limitations:

- Main Portal;
- Awareness Portal;
- Online Assessments Portal;

- Account Review Portal;
- Policy Portal.

The nucleus module of Eramba is dictated by the "Main Portal". All other portals are used when specific functions for each user are enabled. Thus, each portal has its own authentication option and limitation associated with it.

There are four ways of authenticating in Eramba:

- Lightweight Directory Access Protocol (LDAP) - It is an application protocol used over an IP network to manage and access the distributed directory information service. The purpose of a directory service is to provide a systematic set of records organized in a hierarchical structure [72].
- Local Password - This authentication is done via the user and password plus salt stored in the database.
- Google OAuth - Google OAuth is an API that uses OAuth protocol to authenticate and authorize users to log in [30].
- Security Assertion Markup Language (SAML) - SAML is an XML-based framework that allows identity and security information to be shared.

Despite the method of authentication, any user is required to have an account. The authentication process for Eramba is described in the Figure 3.3.

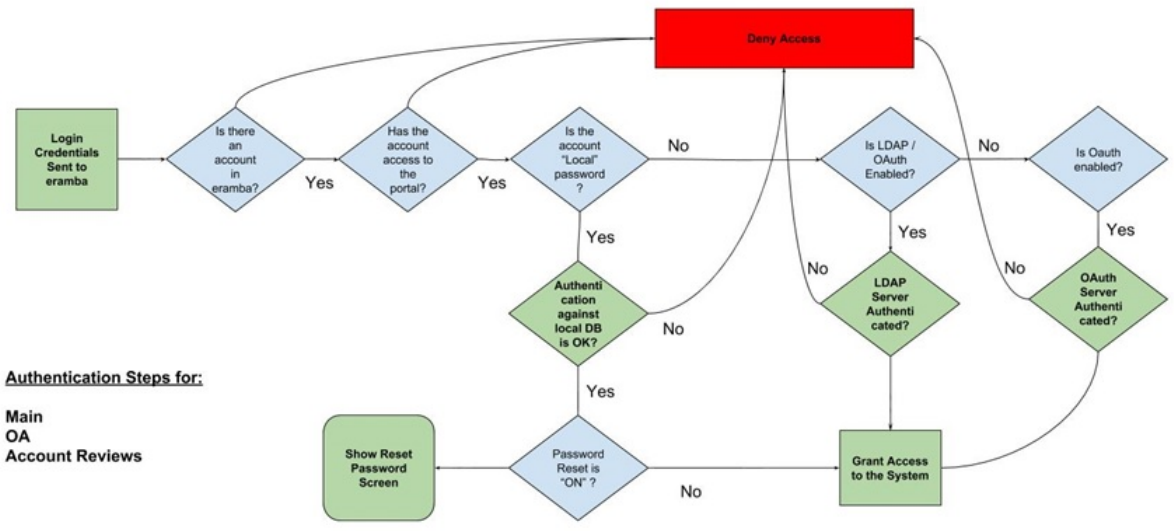

Deny Access
Login Credentials Sent to eramba
Is there an account in eramba?
Yes
Has the account access to the portal?
Yes
Is the account "Local" password ?
No
Is LDAP / OAuth Enabled?
No
Is Oauth enabled?
Yes
Yes
Yes
No
No
Authentication against local DB is OK?
No
LDAP Server Authenticated?
OAuth Server Authenticated?
Yes
Authentication Steps for:
Main
OA
Account Reviews
Show Reset Password Screen
Password Reset is "ON" ?
No
Grant Access to the System

Figure 3.3: Authentication Process [20].

3.2 Vulnerabilities Analysis and Exploitation

Vulnerabilities assessment allows to maintain an awareness of the vulnerabilities of an application while providing knowledge to the IT team. Additionally, it also permits to quantify the risk in the long run to then mitigate it. Governance Risk and Compliance may seem to contain non-sensitive information but if there is no correct due diligence by upper management, there is a vast possibility for bigger issues. Usually the main core information of a GRC is risk information and what kind of controls there are to mitigate or transfer them. A correctly placed threat actor may use this information wisely to compromise the organization.

The most notorious method to identify web application vulnerabilities is to follow the list named OWASP TOP 10. This list is a industry standard to ensure that a web application is the most secure possible. On this chapter, two vulnerabilities from TOP 10 were found: Use of components with known vulnerabilities and an overall security misconfiguration of the environment the app was configured in. However there are some other notable vulnerabilities that should be assessed besides TOP 10 Section [2.3]. Web applications are built on server-side and client-side, each one with different vulnerabilities and threats. The server side may open several ports for the acceptance of requests and communication between components; there could be lack of patching; no system hardening; no firewall, and others. Usually, server side attacks attempt to compromise and breach data that the applications are present on.

Client-side is typically served through a user interface, commonly known as UI. Client-side attacks tend to target software such as web browsers, email clients and other type of applications.

In this Section, using network tools mainly of open-source nature, we describe threats found in Eramba that its users may face. Those users may face these threats if Eramba is not configured correctly, either client-side, server-side or if the underlying infrastructure is not protected.

According to Adam [1], STRIDE threats may violate securities' properties such as confidentially, authentication, integrity, non-repudiation, availability and authorization. The Table 3.2 demonstrates which properties STRIDE may violate.

Threat	Property Violated
Spoofing	Authentication
Tampering	Integrity
Repudiation	Non-repudiation
Information Disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

Table 3.2: STRIDE securities properties violated.

3.2.1 Brute-Force Attack

A brute-force attack is a cryptographic attack that relies on guessing all the possible combinations of the pair username:password. This is a trial and error approach with the hopes of eventually achieving the correct username and password.

There are numerous methods to try brute-force attack. For this attack Burp Intruder [61] was used. Burp Intruder is a component from Burp suite, which is a very popular toolkit used for pentesting web applications. Burp Intruder is a fuzzing tool: it works by taking the HTTP request, modifying the request by changing a set of values through an input point (username and password, for example) and automatically issuing the modified request to the server. The output success is recognized by the content length and status code. With Burp Intruder there are four type of attacks: Sniper, Battering ram, Pitchfork and Cluster bomb. Each attack varies according to the payload (data transmitted) number and position assigned to it.

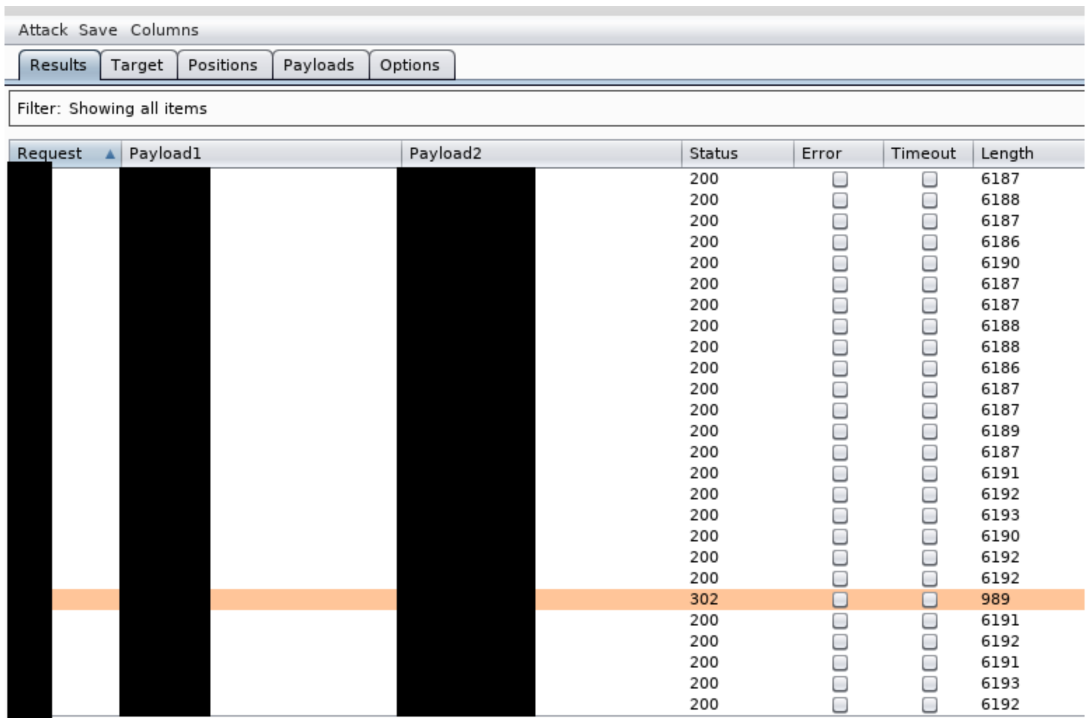
Sniper Attack uses a single set of payloads. It individually targets each payload position for iteration.

Battering ram also uses a single set of payloads, however it iterates through the payload and places the same payload into all defined positions at once.

Pitchfork attack uses multiple payloads. The attack iterates through the multiple payloads used for each different position.

Cluster bomb attack was the one used and it also uses multiple payloads. On this attack is possible to assign a payload for each different position: one for the username position and the other for the password position, looping through all possible combinations between them.

The payload used for this attack was a common word text file with 14,341,564 passwords used by the industry, named "rockyou". The Figure 3.4 shows the attack was successful: It is possible to conclude this due to the status code and length of the request response.



Request	Payload1	Payload2	Status	Error	Timeout	Length
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6188
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6186
[redacted]	[redacted]	[redacted]	200	☐	☐	6190
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6188
[redacted]	[redacted]	[redacted]	200	☐	☐	6188
[redacted]	[redacted]	[redacted]	200	☐	☐	6186
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6189
[redacted]	[redacted]	[redacted]	200	☐	☐	6187
[redacted]	[redacted]	[redacted]	200	☐	☐	6191
[redacted]	[redacted]	[redacted]	200	☐	☐	6192
[redacted]	[redacted]	[redacted]	200	☐	☐	6193
[redacted]	[redacted]	[redacted]	200	☐	☐	6190
[redacted]	[redacted]	[redacted]	200	☐	☐	6192
[redacted]	[redacted]	[redacted]	200	☐	☐	6192
[redacted]	[redacted]	[redacted]	302	☐	☐	989
[redacted]	[redacted]	[redacted]	200	☐	☐	6191
[redacted]	[redacted]	[redacted]	200	☐	☐	6192
[redacted]	[redacted]	[redacted]	200	☐	☐	6191
[redacted]	[redacted]	[redacted]	200	☐	☐	6193
[redacted]	[redacted]	[redacted]	200	☐	☐	6192

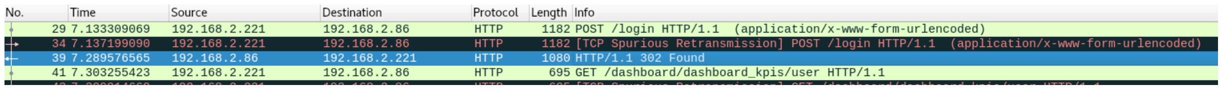
Figure 3.4: Brute force attack results using Burp Intruder.

Since the HTTP status code is 302, it means that the page has been found and is redirecting to the mainpage of Eramba which translates to a successful log in session. Also, it is possible to verify that the length of the request response is different from the others since the logged in page leads to no errors at all.

3.2.2 Address Resolution Protocol (ARP) Spoofing

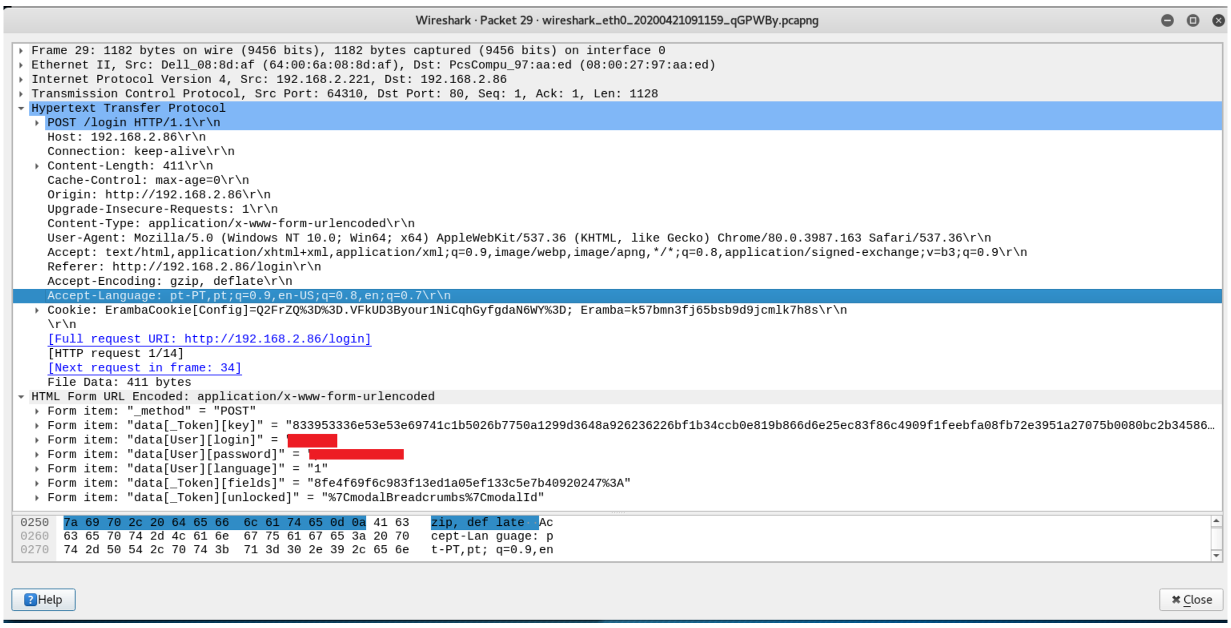
ARP Spoofing also known as ARP poisoning is a sniffing technique. The threat actor sends falsified ARP messages over a local area network to link a threat actor's MAC address with the IP address of a legitimate computer or server on the network. After the linkage is successful, the threat actor is able to perform a man-in-the-middle attack (MITM) and sniff any traffic going between the target and the internet. If sensitive information such as credit card details or passwords are sent in plaintext the target is going to face difficulties. By doing a MITM attack and sniffing the network after an ARP poison, the threat actor is able to intercept the credentials of an account to successfully log in to Eramba environment. The ARP poisoning was possible using Ettercap, a free and open source network security tool for MITM attacks and the capture was done via Wireshark. The figure 3.7 shows a successful ARP spoofing attack. The process to complete ARP poisoning was the following:

1. Set up a malicious host (Kali linux) in the same network as the target host (web browser).
2. Start ARP spoofing using the Ettercap tool.
3. Start Wireshark to sniff packets that flow through the network.
4. Analyze Wireshark capture.



No.	Time	Source	Destination	Protocol	Length	Info
29	7.133309069	192.168.2.221	192.168.2.86	HTTP	1182	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
34	7.137199090	192.168.2.221	192.168.2.86	HTTP	1182	[TCP Spurious Retransmission] POST /login HTTP/1.1 (application/x-www-form-urlencoded)
39	7.289576565	192.168.2.86	192.168.2.221	HTTP	1080	HTTP/1.1 302 Found
41	7.303255423	192.168.2.221	192.168.2.86	HTTP	695	GET /dashboard/dashboard_kpis/user HTTP/1.1

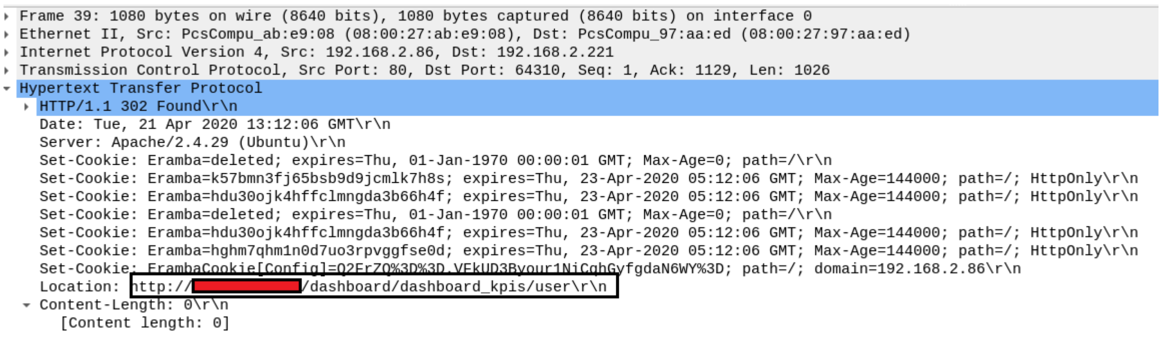
Figure 3.5: HTTP header that show the requests from the client IP to the Eramba host.


Wireshark · Packet 29 · wireshark_eth0_20200421091159_qGPWBy.pcapng
Help
Close

```
▸ Frame 29: 1182 bytes on wire (9456 bits), 1182 bytes captured (9456 bits) on interface 0
▸ Ethernet II, Src: Dell_08:8d:af (64:00:6a:08:8d:af), Dst: PcsCompu_97:aa:ed (08:00:27:97:aa:ed)
▸ Internet Protocol Version 4, Src: 192.168.2.221, Dst: 192.168.2.86
▸ Transmission Control Protocol, Src Port: 64310, Dst Port: 80, Seq: 1, Ack: 1, Len: 1128
▾ Hypertext Transfer Protocol
  ▸ POST /login HTTP/1.1\r\n
    Host: 192.168.2.86\r\n
    Connection: keep-alive\r\n
  ▸ Content-Length: 411\r\n
    Cache-Control: max-age=0\r\n
    Origin: http://192.168.2.86\r\n
    Upgrade-Insecure-Requests: 1\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.163 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
    Referer: http://192.168.2.86/login\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: pt-PT,pt;q=0.9,en-US;q=0.8,en;q=0.7\r\n
  ▸ Cookie: ErambaCookie[Config]=Q2FrZQ%3D%3D.VFkUD3Byour1NiCqhGyfgdaN6WY%3D; Eramba=k57bmn3fj65bsb9d9jcmlk7h8s\r\n
    \r\n
    [Full request URI: http://192.168.2.86/login]
    [HTTP request 1/14]
    [Next request in frame: 34]
    File Data: 411 bytes
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "_method" = "POST"
  ▸ Form item: "data[_Token][key]" = "833953336e53e53e69741c1b5026b7750a1299d3648a926236226bf1b34ccb0e819b866d6e25ec83f86c4909f1feebfa08fb72e3951a27075b0080bc2b34586…
  ▸ Form item: "data[User][login]" = [redacted]
  ▸ Form item: "data[User][password]" = [redacted]
  ▸ Form item: "data[User][language]" = "1"
  ▸ Form item: "data[_Token][fields]" = "8fe4f69f6c983f13ed1a05ef133c5e7b40920247%3A"
  ▸ Form item: "data[_Token][unlocked]" = "%7CmodalBreadcrumbs%7CmodalId"

0250  7a 69 70 2c 20 64 65 66  6c 61 74 65 0d 0a 41 63   zip, def late  Ac
0260  63 65 70 74 2d 4c 61 6e  67 75 61 67 65 3a 20 70   cept-Lan guage: p
0270  74 2d 50 54 2c 70 74 3b  71 3d 30 2e 39 2c 65 6e   t-PT,pt; q=0.9,en
```

Figure 3.6: Log in request from client to Eramba host.



```
▸ Frame 39: 1080 bytes on wire (8640 bits), 1080 bytes captured (8640 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_ab:e9:08 (08:00:27:ab:e9:08), Dst: PcsCompu_97:aa:ed (08:00:27:97:aa:ed)
▸ Internet Protocol Version 4, Src: 192.168.2.86, Dst: 192.168.2.221
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 64310, Seq: 1, Ack: 1129, Len: 1026
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 302 Found\r\n
    Date: Tue, 21 Apr 2020 13:12:06 GMT\r\n
    Server: Apache/2.4.29 (Ubuntu)\r\n
    Set-Cookie: Eramba=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/\r\n
    Set-Cookie: Eramba=k57bmn3fj65bsb9d9jcmlk7h8s; expires=Thu, 23-Apr-2020 05:12:06 GMT; Max-Age=144000; path=/; HttpOnly\r\n
    Set-Cookie: Eramba=hdu30ojk4hffclmngda3b66h4f; expires=Thu, 23-Apr-2020 05:12:06 GMT; Max-Age=144000; path=/; HttpOnly\r\n
    Set-Cookie: Eramba=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/\r\n
    Set-Cookie: Eramba=hdu30ojk4hffclmngda3b66h4f; expires=Thu, 23-Apr-2020 05:12:06 GMT; Max-Age=144000; path=/; HttpOnly\r\n
    Set-Cookie: Eramba=hghm7qhm1n0d7uo3rpvggfse0d; expires=Thu, 23-Apr-2020 05:12:06 GMT; Max-Age=144000; path=/; HttpOnly\r\n
    Set-Cookie: ErambaCookie[Config]=Q2FrZQ%3D%3D.VFkUD3Byour1NiCqhGyfgdaN6WY%3D; path=/; domain=192.168.2.86\r\n
    Location: http://[redacted]/dashboard/dashboard_kpis/user\r\n
  ▾ Content-Length: 0\r\n
      [Content length: 0]
```

Figure 3.7: Packet that show the login was successful.

3.2.3 Session Hijacking

Cookies are small pieces of data used to store information on the users' web browser. These are used to store stateful information and improve ease of access on the user part. A common use of cookies is to identify an associated user with a multi interaction session in a website. The user logs into a website and the server sends a session cookie also known as session token. This cookie will either go in an HTTP header or will be explicitly included in a hidden field. Cookies have three purposes: Session management, personalization and tracking.

A session hijacking attack consists of a man in the middle attack exploiting a logged in session and changing the values of the HTTP request.

A session hijacking happens when a threat actor steals a valid cookie and reuses it to impersonate the user. This attack is a man-in-the-middle attack (MITM) and this was only possible due to it being an unencrypted communication. This attack was done using Burp Suite.

3.2.4 Unencrypted Communication

Eramba default configuration allows insecure HTTP communications. This means that every single piece of data and information transmitted over the HTTP channel by users is eavesdrop prone. There are different methods to disable this, being the most known SSL/TLS. SSL/TLS ensures a safe communication between client and the server over the transport layer, namely TCP. Client and server exchange parameters for a secure session - in this negotiation both peers authenticate using asymmetric key pair and certificates x.509 of each public key. Authentication of the client implies a mutual authentication, however the opposite (client-side authentication) does not imply mutual authentication. Server authentication is done implicitly, via verification of the server certificate; it then verifies if the public key of certificate matches. Client authentication is accomplished in a different way. It authenticates with a certificate verify message that is accomplished via all messages exchanged of the protocol until this point [3].

Overall, SSL/TLS allows client-server applications to communicate across a network in a way designed to prevent eavesdropping, tampering and man-in-the-middle. This protocol, when used over HTTP is named HTTPS and uses port 443 to communicate. Although SSL/TLS are fairly secure, there is still a widely known vulnerability that myriad devices suffer from: the Heartbleed bug. This bug occurs with OpenSSL cryptographic software library. Overall, it allows stealing information encrypted by SSL/TLS. All things considered, despite having SSL/TLS configured, it is important to configure it correctly not to have this vulnerability. In certain circumstances, a threat actor may suit himself to a man-in-the-middle position that allows him to monitor, track and record network traffic between a user and the application, to obtain all the user-supplied information, particularly sniff user credentials. Additionally, a threat actor is able to modify the traffic and misuse the application as intended.

3.2.5 Cross-site Request Forgery

A cross-site request forgery (CSRF) is a confused deputy attack, a program tricked by a threat actor into misusing its authority. It inherits the privileges and identity of the end user and forces the user to execute unwanted actions in a vulnerable site in which he/she is authenticated. A CRSF exploits the relationship between the client and the server. In theory, everything that the server uses to establish trust with a browser is what allows a CSRF to happen. This means that it requires some sort of HTTP request, cookies or other authentication method. Along side with an authentication method, it also requires a web form or request with predictable parameters so that a pentester can craft the request. CSRF are not possible in GET requests that change server state [3]. To prevent CSRF attacks it is recommended to use random nonce tokens per session when a user logs in and stores data. For any subsequent page that checks the session data, if the nonces token do not match, then no other requests are forwarded. The framework CakePHP enables CSRF protection by simple adding "CsrfComponent" to the code. OWASP ZAP allows to generate a Eramba CSRF proof of concept (PoC). By clicking on "Generate anti-CSRF test FORM", a new tab will open with CSRF PoC which contains POST parameters and values from the HTTP request. Then, these values can be adjustable by a threat actor and he can change them however desired.

3.2.6 Use of components with known vulnerabilities

Web applications typically use numerous commercial libraries and open source software developed by third parties, namely as authentication and session management, communications libraries, cryptographic libraries, report generation, logging libraries and many others. OWASP also lists "Use of components with known vulnerabilities" as its number nine vulnerability, therefore these security details should be taken seriously. Those vulnerabilities in libraries usually are identified by a developer, vendor or a pentester but sometimes, by the time they are identified, plenty of web applications were developed using said libraries. Periodically, when some vulnerabilities are found on these libraries, a patch is uploaded to fix the vulnerability but only in the next version. Web applications must include mechanisms that ensure that libraries with known vulnerabilities are not used or that at least are used with limitations. Eramba uses plenty of different web components from third parties to build its system.

Almost every single application has known vulnerabilities from third parties, either from Jquery, Bootstrap, Apache, MySQL etc. If not monitored or tracked, these can be a backdoor to the application. Some vulnerabilities may lead to minor impacts while others may lead to major data breaches. This is why it is crucial to correctly configure and monitor the application as well as apply a policy for patch management process. Depending on the asset, this risk may be important to evaluate and prioritize. When using known vulnerabilities components, it is important to research the flaws, weaknesses, understand if the application uses the components at the full extent and also distinguish their nested dependencies. It is also critical to establish to what degree can the application be vulnerable and to balance if it is really required to use those components. To prevent these classes of vulnerabilities, the developer should monitor databases of known vulnerabilities such as Common Vulnerabilities and Exposures (CVE), National Vulnerability Database (NVD) or a community-driven open-source vulnerability database (VULDB). Topping that, the developer should also subscribe to email alerts for security vulnerabilities related to components used, and only obtain components from legitimate and secure sources, verifying if the checksum is official. Some vulnerabilities are easy to find, while others require tremendous skill level to craft the request. The most effective prevention is to not use any kind of libraries made by third parties. Besides being an unrealistic scenario, this approach would introduce other type of vulnerabilities. Therefore, development teams of web applications must define a process that allows to safely use libraries developed by third parties. Eramba uses three components with known vulnerabilities: Jquery with version 2.1.4, Jquery UI dialog 1.14 and bootstrap 3.3.7. These components will now be studied to understand if their usage presents any true positive or false positive.

jQuery 2.1.4

JQuery is a fast, small, and feature-rich JavaScript library. It makes concepts like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers [39]. The jquery version that Eramba uses is the 2.1.4 and according to CVE 2015-9251 [9], "is vulnerable to Cross-site Scripting (XSS) attacks when a cross-domain Ajax request is performed without the dataType option, causing text/javascript responses to be executed" [9]. This happens due to the failure of sanitizing user-input value. A threat actor may take advantage of this vulnerability to execute code in a user browser.

jQuery 1.14 UI dialog

Jquery-UI is a library required to manage UI elements in Jquery. This library is known for having a varied set of user interface interactions, effects, widgets and themes, all built on Jquery [40]. As many applications, Eramba also adopted Jquery-UI to create its UI components. According to CVE-2016-7103 [10], this library has a known vulnerability. This vulnerability requires specific circumstances to happen, mainly that the web page has a dialog component. The UI component dialog allows to inject XSS content into "closeText" parameter of a dialog function. The proof of concept is shown on the script [3.1] for this vulnerability [11]. However, there are no webpages with dialog components, thus we can consider this a false positive.

```
 <!DOCTYPE html>
<head>
<title>XSS in closeText option of component ui dialog</title>

<script src="https://code.jquery.com/ jquery-2.1.4.js"></script>
<script src="https://code.jquery.com/ui/1.11.4/ jquery-ui.js"></
    script>
<link rel="stylesheet" type="text/css" href=" http://code.jquery.
    com/ui/1.9.1/themes/base/jquery-ui.css">
<script>
    $(document).ready(function () {
        $('#dialog').dialog({ closeText: '<script>
alert("XSS in here")<\/script>' });
    });
</script>
</head>
<body>

<div id="dialog" title="Dialog Title">Content here!</div>

</body>
</html>
```

Listing 3.1: XSS closeText PoC.

Bootstrap 3.3.7

Bootstrap is described as a HTML, CSS, JavaScript open-source framework used to design mobile-first responsive projects. It enables developers to quickly build responsive websites, supplying the developer with numerous advantages such as responsive grids, responsive images, navigation bars, menu dropdowns, progress bars, etc. Additionally, Bootstrap also requires jQuery to function. Eramba uses Boostrap version 3.3.7, which is outdated and with certain conditions vulnerable to XSS. According to CVE-2018-14040 [12], this vulnerability affects the data-target attribute and occurs due to lack of validation of user-supplied input. Data-target enables web developers to control a widget in JavaScript. A threat actor can exploit this vulnerability by persuading a target user to follow a malicious link. However, this vulnerability only occurs when the data-target attribute relies on external data accompanying the page where others users besides the threat actor are affected. If data-target attributes are made of hardcoded html text, it is not considered an issue. The PoC of this vulnerability is shown on the script [3.2]. The environment for the vulnerability to occur is not present in any of Eramba pages, therefore we can conclude that this vulnerability is also a false positive.

```
  <!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/
    jquery.min.js"></script>
<script src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/
    bootstrap.min.js"></script>
</head>
<body>

<button data-toggle="collapse" data-target="<img src=x onerror=
    alert(0)>">Test</button>
</body>
</html>
```

Listing 3.2: Bootstrap vulnerability PoC.

3.2.7 Excessive administrator privileges

One of the most common authentication methods is via password, due to its simplicity, ease of use and convenience. However, in order to set up a user account in Eramba, the administrator has the full power. For any user that intends to create an account, they must ask the administrator for it. Then the administrator must go to the access management portal and create an account with the user details. Those details are: username; password; email; which groups and portals the user has access to and if any REST API is enabled. This means that the password is no longer unique, since the administrator has to set it up. Additionally, when the user tries to log in for the first time, Eramba requires the user to change the password, yet the password can be the same as the one administrator set for the first time which is not a best practice according to security standards. To sum up, the administrator is the core of access management in Eramba and can fully compromise it if he so wishes.

3.2.8 Weak Cryptographic Algorithm

According to NIST [54], a Cryptographic algorithm is a "well-defined computational procedure that takes variable inputs that may include a cryptographic key to provide confidentiality, data integrity, authentication and/or non-repudiation". There are several encryption and hashing algorithms that are no longer secure. Hashing algorithms are vital for web applications as they provide secure passwords in a fast way, through low-use of compute power using a one-way function. Eramba uses Bcrypt algorithm to store users password in the database. Bcrypt is a key derivation and adaptive function based on Blowfish cipher that can encrypt data up to 512 bits. It also incorporates a salt mechanism to protect against rainbow table attacks and the option to change salt round values - the higher the value, the more time is required to brute-force the resulting hash. As it has the option to configure numbers of rounds, it is also a slower hash but a more strengthened key.

Blowfish cipher is a 16-round Feitsel cipher and a particularly fast symmetric-key block cipher. It has a 64-bit block size and variable key length (according to input) from 32 bits up to 448 bits.

Usually, passwords are not stored directly into a database as it would be easy to compromise the system in case the database was compromised. To solve this problem, passwords are mutated via a one-way function which then produces a hash [3]. Therefore, passwords inserted by the users are mutated via said function and then the resulting hash is compared with stored database hash. Besides this mutation, password are sometimes transformed using a random value designated by salt. This salt is used in conjunction with the password for the first time the password is inserted and then the one-way function is applied. Lastly, the resulting value is stored for future use in the database. Typically one-way function is given by: $f(salt, password) = salt|MutatedPassword$.

With a tool like Hashcat [31] which has Bcrypt support, it is possible to perform a dictionary attack (trying all words in a list), a combinator attack (concatenating words from multiple word lists), a brute-force attack, a mask attack (trying all characters from given charsets) or a hybrid attack (combining world lists masks). When having full access to the database, a threat actor may use these tools to compromise a user password and the required time only depends on graphics and the computer processing unit.

3.2.9 Denial of Service

A Denial of Service (DoS) attack attempts to overwhelm and exhaust application's resources, making said application slow or unresponsive to legitimate users. A Distributed Denial of Service (DDos) attack aims to make a server, service or infrastructure unavailable. The attack can take many forms: a server bandwidth overload to make it unavailable or a depletion of the machine's system resources, preventing it from responding to legitimate traffic. Denial of Service can be a useful probing technique as well an attack in itself. A lone DDos attack impact is usually temporary although it can be exceedingly costly to the victim. However, a DDos can preclude far more damaging attacks. A system that fails under a DDos attack can be quite informative on what parts of the system have been neglected or less invested in. When the web application starts to fail, it is important to understand if database queries failed before the website. If so, this means that the database server or the website software is most likely being neglected. If the website itself just times out static pages, it means that the hosting server may have issues or the software hosted in it is under specced. The error handling may also not have been very well executed and bad error handling throwing error may provide more information. At the time of a DDoS attack, a series of requests are sent at the same time from various points on the web. The intensity of this "crossfire" makes the service unstable, and in the worst case, unavailable.

DoS and DDoS attacks can be split into five different categories [18]:

- Network device level: These attacks may occur due to software deficiency such as bugs or hardware exhaustion.
- OS level: The attack takes advantage of protocol implementation techniques.
- Application Level: In this kind of attack, the threat actor studies the application to find application flaws that allow him to exhaust resources of the victim.
- Data flood: In data flooding attacks, the threat actor attempts to over fatigue the network bandwith by sending extremely large amounts of data to process. Attacks such as SYN Flood, UDP Flood, HTTP flood or ICMP flood are common examples of this attack as they are used to overload the network with data packets.
- Protocol Feature attack: Attacks based on protocol features exploit flaws of protocol implementation or bugs of installed protocol. Common examples of these attack are Smurf Attack, SYN, UDP and ICMP.

To assess the studied vulnerability, a SYN Flood attack was performed. This attack was chosen because it is an extremely powerful and straightforward attack. SYN flood, as discussed previously, is a data flood attack [5]. It exploits the TCP connection known as three-way handshake, using one of the TCP headers named as synchronize flag. The threat actor sends repeated SYN packets using spoofed addresses to the same host. Then, the host acknowledges the request (ACK) from, supposedly, multiple connections and attempts to establish connection with each one. When trying to establish connection, the host sends SYN-ACK and waits for a reply from the client, however, since it is a spoofed IP address, it never sends back a reply, leaving the host waiting for the ACK. During this time, the host can not close down the connection and before the connection times out it receives another SYN packet. This means that an immense number of connections are left open and eventually the host overflows. To perform this attack, a network tool named Hping was used [32]. Hping is a command-line TCP/IP packet generator that can be used for firewall testing, advanced port scanning, networking testing, manual path discovery, remote OS fingerprinting and other network related actions. Through Hping and a set of commands, it was possible to perform a successful SYN Flood attack on Eramba as seen on Figure 3.8. It must be noted that this procedure was done in a safe laboratory environment, in a sandbox, with no harm to the company. To effectively make a distributed denial of service attack, a botnet would be necessary.

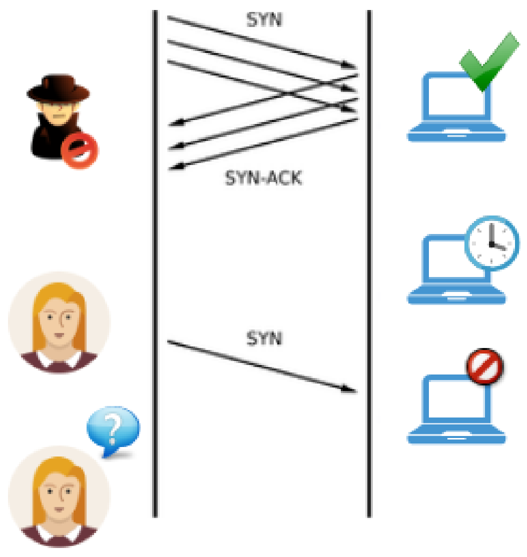


Figure 3.8: SYN Flood Attack.

3.2.10 Password Field automatic

The log-in form contains the password field with auto-complete enabled. This is both a server and a client side issue. According to CWE-200 [14], this is exposure of sensitive information to an unauthorized actor, as the product may expose sensitive information to an actor that is not explicitly authorized to access the information. Despite being a minor vulnerability, it still may compromise the security of the company if misused. A threat agent may capture a user's credentials if it gains control over the user's computer. Additionally, if a user finds another application with XSS vulnerability it is possible to capture all browser-stored credentials. From the client-side, the user must configure his browser options to enable auto-filling in forms. From the application it self, the HTML form must include the attribute "autocomplete=off" within the form tag in order to prevent all form fields. However, some browsers may ignore this atribute, even though it is required for web application to get PCI compliance.

All in all, this should not be a problem if two factor authentication is correctly implemented. The access management already discussed provides a comprehensive overview of how the authentication process works in Eramba. Additionally, it is also possible to add an extra step of authentication with Google Oauth, as discussed in in Authentication and Access Management 3.1.

3.3 Post-exploitation

The next step of the Shostack framework is to go through the list of threats and address them. There are four actions to be taken against each threat:

- Mitigating threats.
- Eliminating threats.
- Transferring threats.
- Accepting the risk.

The main focus should be to mitigate the threats, however this action may be costly, involving complex work, consuming extensive resources or requiring constant monitoring. Since the objective of this project is to provide a solution both for **security** and **scalability** of the Governance Risk and Compliance software that is Eramba, the actions to be taken will be discussed in the following chapter as most of the vulnerabilities found are immediately mitigated/eliminated/transferred while using cloud computing to host. The Table 3.3 illustrates the result of STRIDE. It was not possible to map "Weak Cryptographic Algorithm" (3.2.8) to the table as it does not fit any of the fields. If in the future new vulnerabilities are discovered in components, these may or may not be mapped with STRIDE.

INTERACTION	**S**	**T**	**R**	**I**	**D**	**E**
Brute-Force Attack	■					
ARP Spoofing	■			■		
Session Hijacking	■					
Unecrypted Communication	■			■		
CSRF	■		■	■		■
Use of components with known vulnerabilities	■	■	■	■	■	■
Excessive Administrator privileges	■			■		
Weak Cryptographic Algorithm						
Denial of Service					■	
Password Field Automatic				■		

Table 3.3: STRIDE iteration result.

The Table 3.4 provides the standard mitigation applicable to STRIDE threats. In the next chapter, cloud computing offers a solution for almost all of the threats, as when hosting with a CSP we are mostly transferring the risk to the CSP. With the study of Cloud Services we were able to map these countermeasure with said services thus providing this table.

Threat	**Countermeasures**	**Cloud Services**
S	Strong authentication.Do not store secrets in plaintext. Do not pass credentials in plaintext over the wire. Protect authentication cookies with Secure Sockets Layer (SSL).	VPC Network; IAM Managed Certificates
T	Data hashing and signing. Digital signatures. Strong authorization. Use tamper-resistant protocols across communication links. Secure communication links with protocols that provide message integrity.	Managed Certificates
R	Create secure audit trails. Use digital signatures.	Managed Certificates
I	Strong authorization. Strong encryption. Secure communications links with protocols that provide message confidentiality Do not store secrets in plaintext.	By default
D	Resource and bandwidth throttling techniques. Validate and filter input.	Cloud Armor; Load Balancer
E	Follow the principle of least privilege and use least privileged service accounts to run processes and access resources	N/A

Table 3.4: STRIDE Threats and Countermeasures mapped with Cloud Services.

Chapter 4

Eramba Cloud Solution

In this chapter, a solution for the scalability problem of Eramba is presented. This chapter also provides an agenda to mitigate most of the security vulnerabilities discussed in the previous chapter [3]. Additionnaly provides the foundation for an Eramba-as-a-Service Through Cloud computing and containerization, almost all of the vulnerabilities were mitigated in conjunction with future vulnerabilities that may have appeared on the hardware side. When considering mitigation techniques, the host and network configuration were considered as the main entry point to mitigate. The countermeasures referenced in the Table 3.4 are easy to implement and monitor and also provide on-going support via any Cloud Service Provider (CSP). It is important to differentiate the application and the infrastructure needed to host Eramba.

This solution provides vulnerability fixes to the current software (community version 2019) and provides on-going solutions to the hardware that the software is hosted on. Additionally, as it was provided an Eramba container as a solution, any underlying OS infrastructure issues are mitigated. Furthermore, using Kubernetes as the main tool for container orchestration, the clear choice for the CSP was Google Cloud Platform. With all advantages Google Cloud offers, namely less costly tariff prices, integration with open source software and easy integration with hybrid Cloud, the deployment is a straightforward task and it has a smooth learning curve associated with it. Google was also the founder of Kubernetes which further supports this choice. The solution offers an almost one-click setup of Eramba, as the new setup is extremely intuitive. It is also the first step of what could potentially be a new future for the infrastructure of the organization since it is going to be the first application built on Cloud hosting computing and not On Premises. The X-as-a-Service is the future and is seen as a facilitator for the users and system administrators, as the users do not notice any difference and the latter do not have to spend plentiful of hours trying to set up and scale up the application as the user-base increases.

4.1 Requirements Gathering

In order to develop a container solution to further orchestrate it on Kubernetes, it is required to combine hardware, capable of supporting the desired functionalities with a system compatible and desirable. For that there are a set of requirements that must be established.

The set of requirements is summarised in the table [4.1].

Requirements	Description
R1	The solution must allow any organization's user to access the software worldwide at any time.
R2	The system must be portable.
R3	The system must be compatible.
R4	The system must be transparent.
R5	The system must be fault tolerant.
R6	The system must be scalable.
R7	A new entry must be registered through all replicas.
R8	The organization's DNS must be able to associate a Google static IP to a custom organization domain.
R9	The load balancer must be capable of balancing the load as required.
R10	The load balancer must be able to associate more than one service at the time.
R11	The response time must be under 5s.
R12	The operational cost must be under 10€ per day.
R13	The system must have a SLA uptime of 99.95%.
R14	The system must be able to protect itself from security threats namely DDoS and Brute-force attacks.
R15	The solution must have a renewable TLS/SSL certificate.
R16	The hosting machine must have at least two GB of RAM for master nodes and two CPU cores.
R17	Machines that host the worker node used must have at least one GB of RAM and one CPU core.

Table 4.1: Requirements Table

4.2 Eramba Container

To fulfill requirements number two (portability) and number three (compatibility), a container solution was designed for Eramba. The new solution for Eramba is an adapted version from an Eramba image pulled from Docker Hub [47]. This image from Docker Hub was not suitable for Kubernetes integration. In order to function properly, it was required to create an extra file for global configurations of databases values; create a script to make new directories needed, and another additional script was required to build database entries. All this defeated the purpose of a containerized solution as it is not a one-click solution. Consequently, some adjustments were made and the new docker file is shown on appendix B. All database values are the default one and do not represent the true value. This now translates into a one-click solution.

The Figure 4.1 shows the new architecture of Eramba containerized. This image is also open-source and will have significant modifications when migrating to Kubernetes.

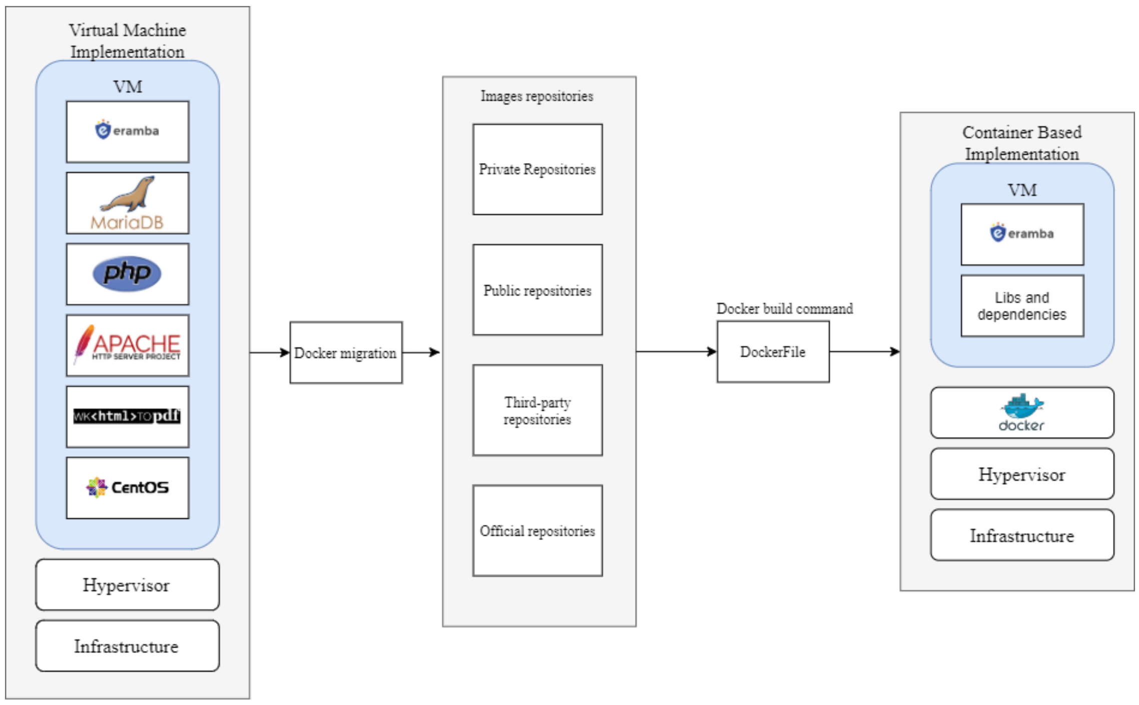


Figure 4.1: Eramba containerized.

All Eramba dependencies are now written on a single text file that is Dockerfile which can be visualized on section B. Dependencies such as MariaDB, PHP, Apache, wkhtmltopdf and a Linux operating system are no longer required to be installed on a virtual machine. Eramba is now a light-weighted portable application that is capable of running in any machine with Docker installed. To further orchestrate this solution across Cloud Computing services, an architecture was designed based on Google Cloud Platform.

4.3 Architecture of Eramba as Service

To meet the remaining requirements the following architecture has been designed and proposed. This architecture aims at ensuring that requirements are met as well as promoting a smooth approach to the services that the cloud offers. Google Cloud offers mechanisms for fault tolerance, scalability and transparency by default so, requirements number one (accessible worldwide); number four (transparency); number five (tolerance); number six (scalability) and number seven are met.

4.3.1 Google Kubernetes Engine

Google Kubernetes Engine is Google´s fully managed Kubernetes platform, which is an enterprise-grade platform for stateful and stateless containerized apps. It offers managed Kubernetes and comprises a high availability control plane that Google collects and operates, and nodes, which hold the pods and the connected Google Cloud services (discussed previously). There are two ways to interact with GKE: using kubectl or the Cloud console. GKE simplifies platform operations with load balancing auto-scale, auto-upgrade, and auto-repair features. It is secure by default, including data encryption and vulnerability scanning of container images. Additionally, it supports integrated Cloud monitoring with infrastructure applications and Kubernetes' specific views.

Eramba Kubernetes object runs in a GKE Zonal Container Cluster, which is the foundation of GKE. This cluster consists of at least one control plane, and one or more machines called nodes created during the cluster creation process. The control plane includes the Kubernetes API server, scheduler, storage, and core resource controllers. The control plane is responsible for deciding what runs on all cluster nodes. This includes scheduling, workloads, managing networks, storage, life-cycle, and upgrades all in an automated way. GKE offers high availability (HA) and scaling. For availability, it is possible to choose between two types of clusters: zonal and regional. Regional cluster is better suited for HA because they have multiple control planes across various zones in a region. On the other hand, zone cluster has one control plane in a single zone. GKE provides four types of auto-scaling for workloads and infrastructure. Workloads: Horizontal pod auto-scaling (HPA) for adding and removing based on utilization metrics like CPU and memory; vertical pod autoscaler for sizing pods Infrastructure: cluster autoscaler for adding and removing nodes based on the scheduled workload; node auto-provisioning for dynamically creating new node pools with nodes according to the need of users' pods.

As stated, GKE is secure by default: It offers automatic data encryption and the OS images deployed are Google certified. It is possible to access clusters only via private IP, and it provides a robust identify, access management and role-based access controls. Moreover, GKE offers trusted networking. Global VPC allows to connect and isolate clusters; Load balancing allows to deploy public services behind a single domain; Cloud Armor provides protection against Layer 7 load balancing (high-level application layer) and DDoS attack.

4.3.2 Google VPC network

Google Virtual Private Cloud network is a virtual version of a physical network, implemented inside Google's production network. It provides connectivity to instances used by GKE and distributes traffic from google Cloud external load balancer to backends. Traffic to and from instance can be controlled with network firewall rules, and network administration can be secured using IAM roles. While using containers, Eramba solution makes use of VPC network by creating external IP addresses accessible through the internet.

VPC Firewall Rules

Since VPC firewall rules deny all network traffic by default, the backend instance must allow connections from the external load balancer. These firewall rules define the allowance of HTTP traffic from 130.211.0.0/22 and 35.191.0.0/16 IP (both subnets for load balancers) to reach backend instances or endpoints passing the health check. This operation must be done manually since

GCP does not do it automatically, and Kubernetes service requires a 200 HTTP response to fully function. For these rules to take effect, a LivenessProbe and a ReadinessProbe are required in the Eramba Kubernetes solution file. Liveness probes are needed for the kubelet to know when to restart a container, and the Readiness probe defines when a container is ready to start accepting traffic.

4.3.3 Google Load Balancer

Google Load balancer distributes user traffic across multiple instances of applications. By spreading the load, load balancing reduces the risk for applications to experience performance issues. Google Cloud Load Balancer is a fully distributed software, and it offers multiple features that were used in the Eramba solution: Single IP address to serve as the frontend; Automatic intelligent autoscaling for backends; External load balancing to reach Eramba application from the internet; Layer 7-based load balancing to add content-based routing decisions focused on attributes, such as the HTTP header and the uniform resource identifier. For this solution, an external HTTP(S) load balancer was used. An external HTTP(S) load balancer distributes traffic coming from the internet to the Google VPC. Additionally, alongside an ingress C.2 object (an object that defines rules for routing HTTP(S) traffic to the application running in a cluster) associated with the service created from the Eramba solution, it is then possible with Google Managed Certificates to redirect traffic via 443 port to have secure communications between the load balancer and the client. Moreover, an external load balancer provides DDoS protection crucial for the Eramba Kubernetes Solution. All of Google Cloud proxy-based external load balancers inherit DDos Protection from Google Front Ends, which are part of Google's production infrastructure. It is also possible to enable Google Cloud Armor to enhance DDoS protection.

4.3.4 Google Custom Resource ManagedCertificate

Google Self Managed Certificates uses Let's Encrypt Certificate Authority to provide the ability to automate the certificate issuing and renewal using Automatic Certificate Management Environment (ACME) protocol. It is a Google custom resource that specifies the domain that the SSL Certificate will be created for. The solution shows how Eramba SSL certification is created via a ManagedCertificate. Since it is a free certificate, it expires after three months, but it is automatically renewed. Additionally, to create a Google Self Managed Certificate, it is required to have a DNS domain from the organization to configure the DNS record for said domain, in order to map the IP address of the load balancer created.

4.3.5 Google Cloud Armor

Google Cloud Armor offers protection to applications and infrastructure from DDoS attacks. It provides built-in defenses against infrastructure DDoS attacks. As it protects some of the world's biggest websites like Google Search, Gmail, and Youtube, it is sufficient to cover an application like Eramba. Additionally, Google Cloud Armor also provides predefined rules to defend against OWASP TOP 10 attacks, namely cross-site scripting, and SQL injection. If any of those vulnerabilities are found, the system is well-prepared. Moreover, Cloud Armor also offers Web Application Firewall (WAF) services that can be integrated with the organization's modsec rules.

4.3.6 Container Optimized Images (COOS)

COOS are images that are designed by Google to run Docker containers strictly. They provide a smaller attack surface for provident attacks, and they are locked down by default, meaning that they include firewall and security settings by default and benefit from automatic updates. As the COOS root filesystem is always mounted as a read-only, it is possible to run containers quickly, efficiently, and securely.

4.3.7 Cloud Identify Access Management (IAM)

Cloud IAM is a Google Cloud unified system for managing access to resources and assigning permissions for users and services to access those resources. Cloud IAM is designed for organizations with plenty of projects and users. As Eramba marks the new step for the organization's digital transformation, IAM will unify control for access for all Cloud projects and resources in one place in the future.

4.3.8 Security Command Center

Security Command Center is a built-in security control that can help to prevent, detect, and respond to threats in GCP. With the command center, it is possible to generate cryptographic keys, create data loss prevention policies, and manage built-in web security scanner for future threats.

4.3.9 Architecture Overview

The following diagram 4.2 provides an overview of the Architecture of a Eramba cluster in GKE with all Cloud features discussed in the previous section 4.3.

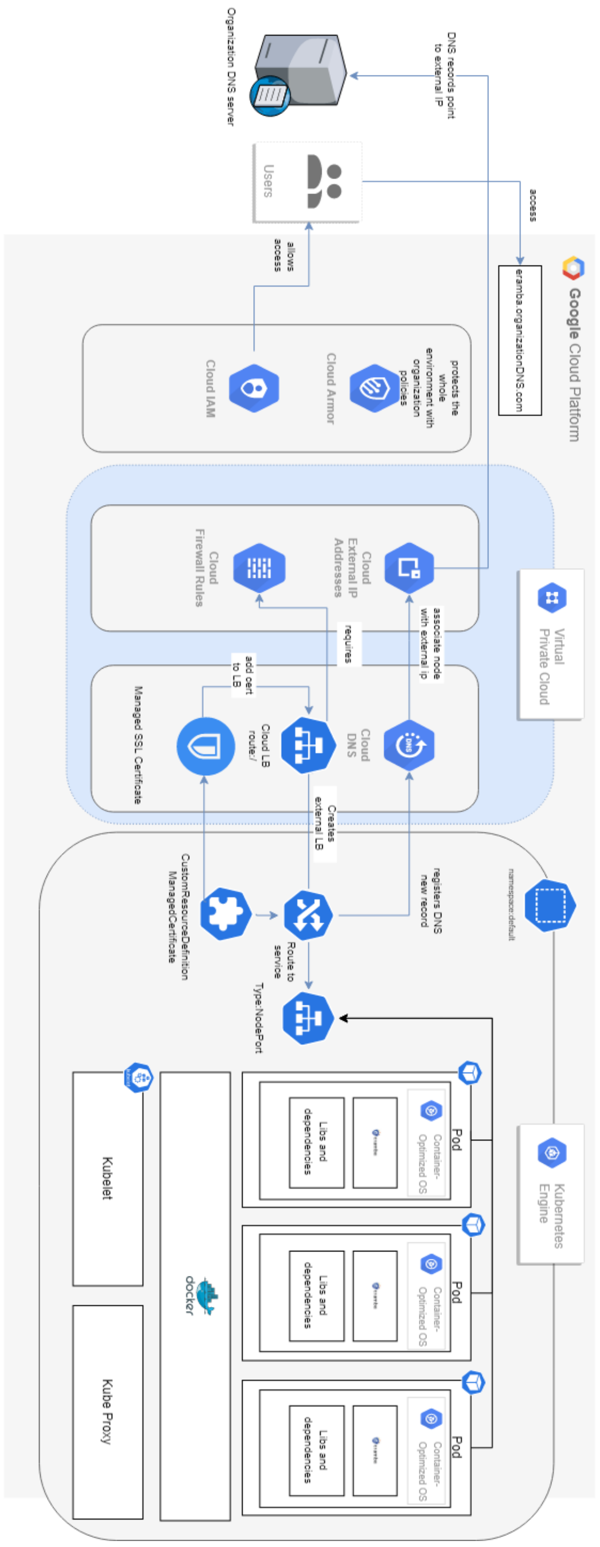


Figure 4.2: Eramba Kubernetes Solution architecture.

4.4 Kubernetes Solution Design

Kubernetes objects are persistent entities that are expressed in an YAML file. It is only required to apply information via an YAML file to kubectl to launch a kubernetes instance. Then, kubectl converts the information from YAML file to JSON and makes an API request. YAML files have some optional parameters values and some others are required. Values such as apiVersion (version of kubernetes API used on the project), kind (kind of object created, either a pod, service or a deployment), metadata (data that helps to identify unique objects, namely name string, UID and namespaces) and spec (state desired for the object) are needed. Other values are optional such as status or annotations.

The proposed solution is expressed in a YAML file that is listed in appendix [C.1]. The architecture is shown on the Figure [4.2], and it illustrates how a user can connect to Eramba from any public IP (accepted in organization' DNS). The load balancer will effectively distribute the request as needed.

This Kubernetes object has particular nuances: the replica number is adjustable and it can be as high as needed; the MySQL user and MySQL password are default for documentation purposes; the image is hosted on a private registry in GitLab being only accessible by its owner.

The features mentioned in Section 4.3 add new security measures that mitigate most of the security issues found in the previous chapter.

4.5 Implementation

This section will provide an agenda to deploy and implement the new solution offered. For a user to access Eramba, there are several steps that take place. The user must have access to organization policies to allow the user's IP to access the application; if the user doesn't have permission, he will be denied access. Additionally, the organization must have a Google Cloud Platform account and a DNS server. When Google Cloud Platform access is created, there are a series of steps to ensure that the application is fully operational and secure:

- The first operation is to enable the cluster API to start the Google Kubernetes Engine (GKE) and create a cluster. GKE offers optional features to ensure the security of the cluster, namely: the validation of the authenticity of OS and kernel modules by enabling secure boot; intranode visibility to observe data flowing between pods and nodes; placement of the Kubernetes API on a private network of VPC; placement of the node pool on a private network; enabling shielded GKE nodes.

- The organization must then reserve a static external IP address to redirect it to the organization' DNS domain attributed to Eramba.

- Subsequently, a ManagedCertificate custom resource definition must be created to specify the domain that the SSL certificate will be created for; This can be done via the deployment of the YAML file described in appendix **??**.

- Then, the service and deployment on C.1 must be applied to get up the pods and running. To guarantee that the certificates will work, the type of nodes must be set to nodeport.

  The ingress file should be configured to match the static IP address and domain to be associated with. This will create an External HTTP(S) LoadBalancer.

- Afterwards, an external domain from the DNS will be linked to a static external IP address, e.g.: `https://www.eramba.organizationDNSdomain.com` All these steps are described in 4.2.

4.6 Results

According to section 3.3, there are four actions that can be taken against each threat. With the proposed architecture, some vulnerabilities are eliminated and others are accepted. The Table 4.2 maps what feature(s) will eliminate the vulnerability found.

	Cloud Feature	VPC Network	Load Balancer	Managed Certificate	Cloud Armor	IAM
Vulnerability						
Brute-force Attack		X	X		X	X
ARP Spoofing		X	X	X		X
Session Hijacking		X	X	X	X	X
Unencrypted Communication		X		X		
CSRF		X		X	X	X
Denial of Service		X	X		X	

Table 4.2: Mapping Google Cloud features with Vulnerabilities found.

By using this Cloud Features, "Brute-force" (3.2.1); "ARP Spoofing" (3.2.2); "Session Hijacking" (3.2.3); "Unencrypted Communication" (3.2.4); "CSRF" (3.2.5) and "Denial of Service" (3.2.9) threats are all eliminated. "Use of components with known vulnerabilities" (3.2.6) threat does not need any kind of mitigation since all vulnerabilities found were considered false positive. "Excessive administrator privileges" (3.2.7), "weak cryptographic algorithm" (3.2.8) and "Password Field Automatic" (3.2.10) threats were all accepted as the potential loss from them is not enough to warrant spending money to avoid them as it would require a team of developers to reprogram some features of the application.

This proposed architecture will meet all requirements established in Section 4.1. The Table 4.3 maps what feature(s) will meet each requirement.

	Cloud Feature	GKE	VPC Network	Load Balancer	Managed Certificate	Cloud Armor	IAM	COOS
Requirement								
R1		X					X	
R2		X						
R3		X						
R4		X						
R5		X						
R6		X						
R7		X						
R8			X				X	
R9				X				
R10				X				
R11		X	X	X	X			
R12		X	X	X	X	X	X	X
R13		X	X	X	X	X	X	X
R14						X		
R15					X			
R16								X
R17								X

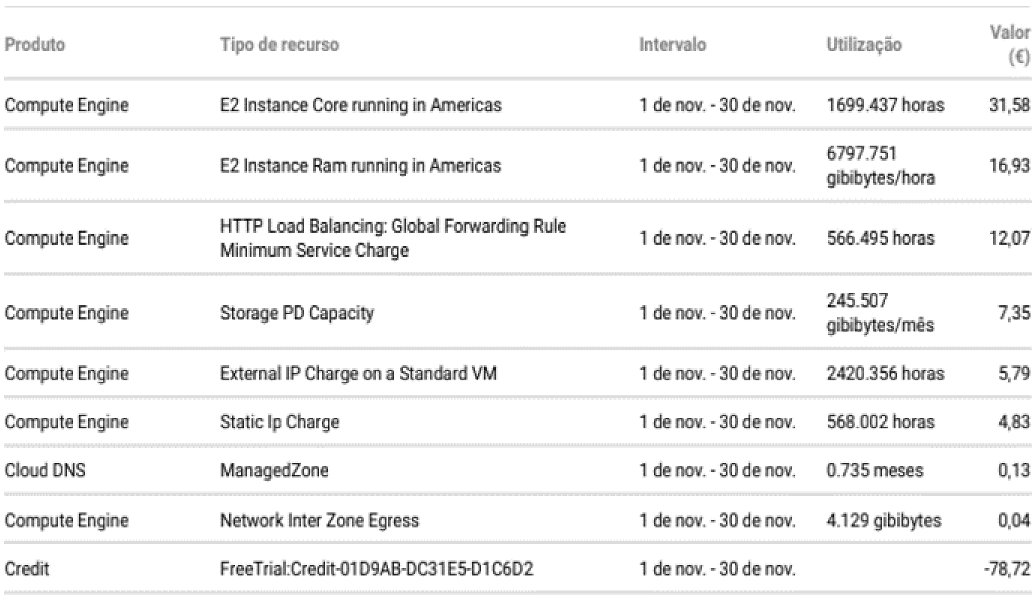
Table 4.3: Mapping Google Cloud features with requirements met.

Google Kubernetes Engine provides a cluster architecture that supports portability, compatibility, fault tolerance and auto-scalability with Cloud TPU (Tensor Processing Units). With Google VPC Network using Cloud DNS, it is possible to associate an external domain to a static google IP. Google Load Balancer allows to distribute load as demand requires and supports multiple backend services. With metrics to measure access time, it is possible to conclude that each average requests takes about 2,5 seconds, however, some heavier requests took as much as 5 seconds. The costs are less than 10€ per day (roughly 6€) and as more services are hosted in the backend services the expense will dilute through all the services.

The GKE offers an SLA up to 99,5% for zonal clusters which reflects exactly the requirement needed. The solution also inherits the protection from Cloud Armor which is ideal for DDoS, brute-force attacks and other attacks mentioned through this project. Google Custom Resource Managed Certificate offers SSL protection and allows to automatically renew certificates every three months. Lastly, Container Optimized solutions provide more than two GB of ram and more than two CPU cores of usage.

4.7 Discussion

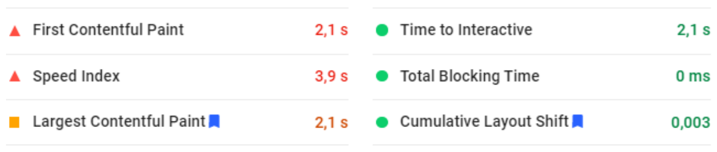
The implementation proposed is not entirely ideal as not all threats are mitigated. Although it is not possible to calculate the costs of hosting Eramba locally, it is possible to presume that it is cheaper to host Eramba locally than in the Cloud. This cost happens because Eramba is the only application hosted in the Cloud. When more applications are migrated to the Cloud, the cost of hosting will be diluted as all the Google Cloud features can be shared. Additionally, with this implementation, the hardware supporting the application is no longer a problem. There is automatic scaling and node management (either to maintain node health or vulnerability) is done automatically, with logs being able to be directly sent to the SIEM. Moreover, when trying to perform the attacks discussed in chapter 3, the threat actor will automatically be blocked by the CSP and, depending on the attack performed, possibly face a fine or even jail time. They may face a fine or even jail due to the CSP having tied relationship with governments who seek to protect the privacy and security of CSP's users. This adds a new abstract layer of security as monetary damage or jail time frighten and dissuade threat actors from trying any attack. The hosting of Eramba, according to Google Cloud bill management, costs close to about 2,5€ per day which translates into 78€ per month. Although meeting requirement number twelve (The operational cost must be under 10€ per day), for the P&L (profit and loss) structure of the organization it is an extra accentuated cost. The discriminated cost is described in the Figure 4.3.



Produto	Tipo de recurso	Intervalo	Utilização	Valor (€)
Compute Engine	E2 Instance Core running in Americas	1 de nov. - 30 de nov.	1699.437 horas	31,58
Compute Engine	E2 Instance Ram running in Americas	1 de nov. - 30 de nov.	6797.751 gibibytes/hora	16,93
Compute Engine	HTTP Load Balancing: Global Forwarding Rule Minimum Service Charge	1 de nov. - 30 de nov.	566.495 horas	12,07
Compute Engine	Storage PD Capacity	1 de nov. - 30 de nov.	245.507 gibibytes/mês	7,35
Compute Engine	External IP Charge on a Standard VM	1 de nov. - 30 de nov.	2420.356 horas	5,79
Compute Engine	Static Ip Charge	1 de nov. - 30 de nov.	568.002 horas	4,83
Cloud DNS	ManagedZone	1 de nov. - 30 de nov.	0.735 meses	0,13
Compute Engine	Network Inter Zone Egress	1 de nov. - 30 de nov.	4.129 gibibytes	0,04
Credit	FreeTrial:Credit-01D9AB-DC31E5-D1C6D2	1 de nov. - 30 de nov.		-78,72

Figure 4.3: Google Cloud Cost per day.

To meet requirement number eleven, Google PageSpeed Insights was used to report the performance of the page. This search engine considers different scenarios including mobile and desktop connection and type of content display and response time. The results are shown on Figure 4.4 for desktop usage - these results can be slightly improved if CSS and jQuery scripts are moved to an inline version, however it is not worth the high cost.



▲	First Contentful Paint	2,1 s	●	Time to Interactive	2,1 s
▲	Speed Index	3,9 s	●	Total Blocking Time	0 ms
■	Largest Contentful Paint	2,1 s	●	Cumulative Layout Shift	0,003

Figure 4.4: Google PageSpeed report results

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The work described in this document was developed in the scope of the Project in Information Security, subject of the completion of the Master in Information Security at Faculdade de Ciências da Universidade de Lisboa. This work was developed during an internship at EY Consulting in Portugal.

The goals of the work were based on pentesting an open Governance Risk and Compliance software and analyzing its vulnerabilities. Additionally, another objective was to reduce the large amount of resources consumed by virtual machines hosting the application on a physical server. Furthermore another objective was to simplify the setup and configuration of Eramba to turn it as light-weighted as possible in order to provide process isolation. Thus providing greater usage of resources via Containers.

As the organization owns different offices worldwide, the software needed to be accessible in any part of the world reasonably quickly. It was required to provide container orchestration to offer the option to scale up and down based of the demand, plus provide the option to auto-scale as needed. With container orchestration it was also provided fault tolerance.

The organization also required to switch their infrastructure on-premises to Cloud computing due to their abundant advantages such as high availability; elasticity; fault tolerance; disaster recovery; scalability; agility; global reach; user latency capabilities; predictive cost considerations; technical skill requirements and security. The Cloud Service Provider chosen was Google Cloud mainly due to their raw integration with kubernetes.

The first phase of the project was dedicated to research existing pentesting methodologies that are widely employed by the specialists, afterwards a study was persecuted of laws and regulations that may influence GRC such as SOX, European Directive 2006/43/EC VIII and GDPR. Moreover, a study of Governance Risk and Compliance purpose was conducted. After this study a full analysis of a GRC tool named Eramba was made.

On the second phase, the Governance Risk and Compliance tool was pentested and the threat model of Eramba was designed. Following each step of the threat model framework proposed by Adam Shostack: model the system, find threats, address threats and validate them.

Consequent to verifying that Eramba is secure to a really high degree the third and final phase was conducted: a migration of the virtual machine Eramba hosting environment to a light-weighted environment scalable as the demand required. To provide redundancy and availability worldwide relatively quick a popularized container orchestration tool known as Kubernetes was used. By creating a Kubernetes object via an YAML file it was possible to simplify the deployment of an extensive resource usage application that is Eramba.

The result of this work can be seen as a particular solution to a vulnerability assessment and a method to mitigate most of the vulnerabilities assessed whilst providing a scalable and light-weighted environment. Overall, it was an hardening of the application Eramba. As Cloud computing is becoming the new normal, this project provides the first step for a digital transformation of the organization to fully migrate its infrastructures to the Cloud.

5.2 Future Work

Regarding future work to be developed, the first step should be to implement a pentest team to follow vulnerabilities disclosed by the community and fix future vulnerabilities that may appear on Eramba immediately. The second step is to assemble a team of Cloud Engineers to monitor Eramba and the next organization's application deployed in the Cloud. Cost-wise, it is only beneficial to host Eramba in Cloud computing if other applications share Google Cloud services provided in this solution. Furthermore, Cloud Engineers should also contact Google to have access to fully perform attack simulations in this environment.

Additionally, Cloud Engineers are required because Google Cloud also discloses vulnerabilities found in containers. When developing this project, a High-security vulnerability described in CVE-2020-14386 [13] was discovered. This vulnerability was one in the Linux Kernel and allowed the container to escape to obtain root privilege on the host node.

The work developed assumes that the organization has a big enough number of users to justify Cloud hosting. Moreover, the architecture described must be changed to support when more applications migrate to the Cloud as minor web services can make use of the same load balancer created.

Bibliography

[1] ADAM SHOSTACK. *Threat Model: Designing for Security*. Microsoft, 2014.

[2] AGENCY, I. A. E. The Fukushima Daiichi Accident Report by the Director General. *Director General* (2015), 1–222.

[3] ANDRÉ ZUQUETE. *Segurança em redes informáticas*. FCA, 2012.

[4] APPROACH, H. O., AND METRICS, E. 2017 GRC Metrics Survey. Tech. rep., OCEG, 2017.

[5] BOGDANOSKI, M., SHUMINOSKI, T., AND RISTESKI, A. Analysis of the SYN Flood DoS Attack. *International Journal of Computer Network and Information Security 5*, 8 (2013), 15–11.

[6] CALDWELL, F., SCHOLTZ, T., AND HAGERTY, J. Magic Quadrant for Enterprise Governance , Risk and Compliance Platforms. *October 3*, July (2011), 1–19.

[7] CERULLO, V., CERULLO, M. J., RACZ, N., WEIPPL, E., SEUFERT, A., ALEKSIC, A., LJEPAVA, N., RISTIC, M., CHENG, D. C., LIM-CHENG, N. R., VILLAMARIN, J. B., CU, G., LIM-CHENG, N. R., AND DE, N. An ontology based framework to support multi-standard compliance for an enterprise. *International Journal of Advanced Computer Science and Applications 224*, 12 (2004), 456–466.

[8] CREASEY, J. A guide for running an effective Penetration Testing programme. *Crest*, April (2017), 1–64.

[9] CVE. Cve-2015-9251, 2015. [Online; accessed 30-January-2020].

[10] CVE. Cve-2016-7103, 2016. [Online; accessed 30-January-2020].

[11] CVE. Cve-2016-7103, 2016. [Online; accessed 30-January-2020].

[12] CVE. Cve-2018-14040, 2016. [Online; accessed 30-January-2020].

[13] CVE. Cve-2020-14386, 2020. [Online; accessed 24-September-2020].

[14] CWE. Cwe-200, 2006. [Online; accessed 15-April-2020].

[15] DAVIS, C. D., SWANSON, C. A., ZIEGLER, R. G., CLEVIDENCE, B., DWYER, J. T., AND MILNER, J. A. Executive Summary Report. *The Journal of Nutrition 135*, 8 (2005), 2014S–2029S.

[16] DE JIMENEZ, R. E. L. Pentesting on web applications using ethical - Hacking. *2016 IEEE 36th Central American and Panama Convention, CONCAPAN 2016*, 503 (2017).

[17] DOCKER. Docker security, 2020.

[18] DOULIGERIS, C., AND MITROKOTSA, A. DDoS attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks 44*, 5 (2004), 643–666.

[19] ERAMBA CONTRIBUTORS. Basic grc relationships in eramba, 2019. [Online; accessed 30-October-2019].

[20] ERAMBA CONTRIBUTORS. Eramba website, 2019. [Online; accessed 30-October-2019].

[21] EUROPEAN PARLIAMENT AND THE COUNCIL. Directive (EU) No 43/2006 of the European Parliament and of the Council. *Official Journal of the European Union* (2006).

[22] FALLIS, A. *Learning Docker*. Packt Publishing, 2013.

[23] FEDRAMP. FedRAMP Penetration Test Guidance. *Security* (2017), 1–34.

[24] FERRAIOLO, D., CUGINI, J., AND KUHN, D. R. Role-based access control (RBAC): Features and motivations. *Proceedings of 11th Annual Computer Security Application Conference pages* (1995), 241–248.

[25] FRIGO, M. L., AND ANDERSON, R. J. A Strategic Framework for Governance, Risk, and Compliance. *Strategic Finance 90*, 8 (2009), 20–61.

[26] GENERAL, A. Investigation : WannaCry cyber attack and the NHS. Tech. Rep. April 2018, UK Government, 2019.

[27] GITLAB. Container registry, 2020.

[28] GOLLMANN, D. *Computer Security (2nd Edition)*. O'Reilly Media, 2006.

[29] GOOGLE KUBERNETES ENGINE. Kubernetes engine, 2020. [Online; accessed 30-July-2020].

[30] HARDT, D. The OAuth 2.0 Authorization Framework. RFC 6749, RFC Editor, October 2012.

[31] HASHCAT. Hashcat, 2020.

[32] HPING. Hping, 2020.

[33] IBM. IBM OpenPages GRC Platform Watson Financial Services. Tech. rep., IBM, 1996.

[34] ISO31000 Risk management — Principles and guidelines, 2009.

[35] ISO/IEC 22301 Societal security — Business continuity management systems — Requirements, 2009.

[36] ISO/IEC 27002 Information technology — Security techniques — Code of practice for information security controls, 2009.

[37] ISO/IEC 27005 Information technology — Security techniques — Information security risk management , 2011.

[38] ISO/IEC 27001 Risk management — Information technology - Security techniques - Information security management systems - Requirements, 2005.

[39] JQUERY. Jquery, 2020. [Online; accessed 30-January-2020].

[40] JQUERY-UI. Jquery, 2020. [Online; accessed 30-January-2020].

[41] KALILINUX. Kali linux penetration testing tools, 2020. [Online; accessed 30-November-2019].

[42] KHAN, R., MCLAUGHLIN, K., LAVERTY, D., AND SEZER, S. Stride-based threat modeling for cyber-physical systems. *ISGTEurope* (09 2017), 1–6.

[43] KRASNER, G. E., AND POPE, S. T. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal Of Object Oriented Programming 1*, 3 (1988), 26–49.

[44] KUBERNETES. Kubernetes concept, 2020.

[45] KUBERNETES. Kubernetes security, 2020.

[46] KUBERNETES. Securing a cluster in kubernetes, 2020.

[47] MARKZ0R. Eramba image, 2020.

[48] MARTIN, R. C. *Using Docker*. O'Reilly, 2011.

[49] MCCLEAN, C., HAYES, N., AND MURPHY, R. The Forrester Wave™: Governance, Risk, And Compliance Platforms. Tech. rep., Forrest Waver, 2014.

[50] MEIER, J., MACKMAN, A., VASIREDDY, S., DUNNER, M., ESCAMILLA, R., AND MURUKAN, A. *Improving Web Application Security : Threats and Countermeasures*, vol. 2003. Microsoft Press, 2003.

[51] MICROSOFT. What is cloud computing?, 2020.

[52] NATIONAL TRANSPORTATION SAFETY BOARD. Collapse of I-35W Highway Bridge Minneapolis, Minnesota National Transportation Safety Board. Tech. rep., US government, 2007.

[53] NIST. SP800-30r1 Guide for Conducting Risk Assessments. *NIST Special publication*, September (2011).

[54] NIST. Nist, 2019. [Online; accessed 30-November-2019].

[55] ORACLE. Oracle ® Enterprise Governance, Risk and Compliance. Tech. Rep. May, Oracle, 2013.

[56] OTTO, M. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation – GDPR). *International and European Labour Law 2014*, March 2014 (2018), 958–981.

[57] OWASP. Top 10 web application security risks, 2020. [Online; accessed 30-January-2020].

[58] PALFREYMAN, D. *Organised Uncertainty: Designing a World of Risk Management*, vol. 14. Oxford, 2010.

[59] PCI-DSS. Data Security Standard. *Security*, April (2015), 139.

[60] PENTEST ERAMBA PARTNER. Partner pentest eramba, 2020. [Online; accessed 30-June-2020].

[61] PORTSWIGGER. Burp suite, 2020. [Online; accessed 3-March-2020].

[62] POWER, M. The End of Enterprise Risk Management. *Research Gate*, January 2007 (2016).

[63] PROGRAM, C. BWise ® Compliance Management. Tech. rep., Nasdaq, 2016.

[64] RACZ, N., AND SEUFERT, A. A.: A frame of reference for research of integrated governance, risk &compliance (GRC. *International Conference on Communications and Multimedia Security* (2014), 106–117.

[65] RACZ, N., WEIPPL, E., AND SEUFERT, A. Governance, risk & compliance (GRC) software - An exploratory study of software vendor and market research perspectives. *Proceedings of the Annual Hawaii International Conference on System Sciences* (2011), 1–10.

[66] RACZ, N., WEIPPL, E., AND SEUFERT, A. Integrating IT governance, risk, and compliance management processes. *Frontiers in Artificial Intelligence and Applications 224* (2011), 325–338.

[67] RIMAL, B. P., JUKAN, A., KATSAROS, D., AND GOELEVEN, Y. Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach. *Journal of Grid Computing 9*, 1 (2011), 3–26.

[68] RISK, R. Risk Management and Compliance Software Platform Financial and Reputation Risk. Tech. rep., SAI Global, 2017.

[69] SCARFONE, K. Draft NIST Special Publication 800-154 Guide to Data-Centric System Threat Modeling Draft NIST Special Publication 800-154 Guide to Data-Centric System Threat Modeling. *NIST 800-154* (2019).

[70] SCHOLER, S., AND ZINK, O. SAP governance, risk and compliance. Tech. Rep. April, SAP, 2009.

[71] SCOTT CAVE. A real life example of how a business continuity plan can save your business,, 2019. [Online; accessed 15-October-2019].

[72] SERMERSHEIM, J. Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511, RFC Editor, June 2006.

[73] SHEVCHENKO, N., CHICK, T. A., RIORDAN, P. O., SCANLON, T. P., AND WOODY, C. Threat Modeling : a Summary of Available Methods. *Research Report*, July (2018), 26.

[74] SRIRAMYA, P., AND KARTHIKA, R. A. Providing password security by salted password hashing using Bcrypt algorithm. *ARPN Journal of Engineering and Applied Sciences 10*, 13 (2015), 5551–5556.

[75] SURVEY, T. I. S. O. THE ISO SURVEY OF MANAGEMENT SYSTEM CERTIFICATIONS – 2018 – EXPLANATORY NOTE Background The ISO Survey of Certifications is an annual survey of the number of valid certificates to. *ISO*, September (2019), 2017–2018.

[76] TEAM, T. P. The Penetration Testing Execution Standard Documentation. *PTES* (2017), 223.

[77] THE, A. R. B. Y., NUCLEAR, I., ADVISORY, S., ATOMIC, I., AND AGENCY, E. *The chernobyl accident: Updating of INSAG-1*. International Nuclear Safety Advisory Group, 1993.

[78] UNITED STATES CODE. Sarbanes-oxley act of 2002, pl 107-204, 116 stat 745. Codified in Sections 11, 15, 18, 28, and 29 USC, July 2002.

Appendix A

Eramba modules and functionalities

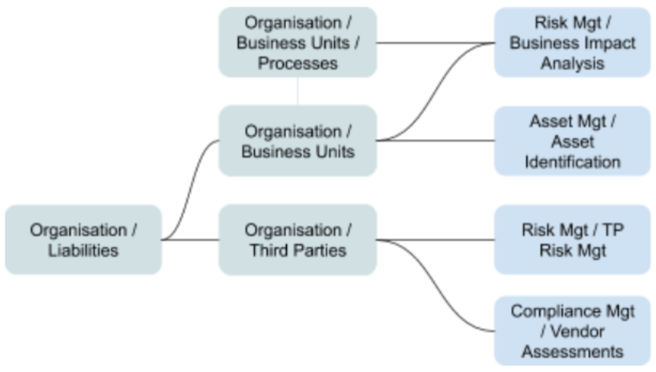

Organisation / Business Units / Processes
Risk Mgt / Business Impact Analysis
Organisation / Business Units
Asset Mgt / Asset Identification
Organisation / Liabilities
Organisation / Third Parties
Risk Mgt / TP Risk Mgt
Compliance Mgt / Vendor Assessments

Figure A.1: Organization Module [19].

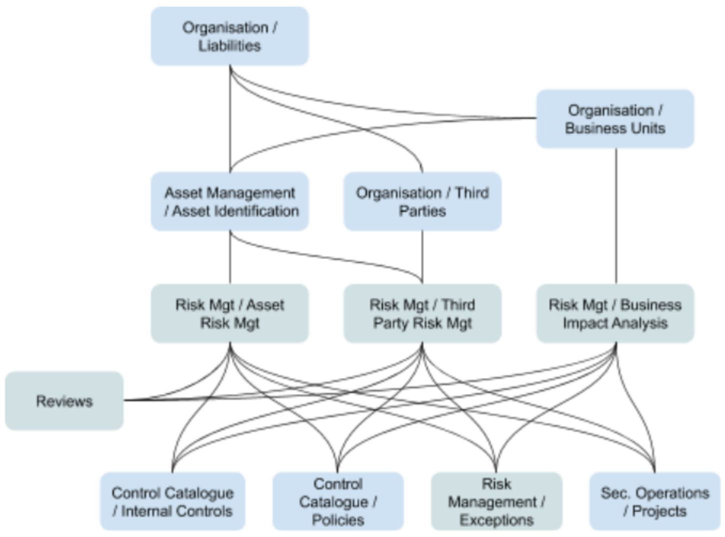

Organisation / Liabilities
Organisation / Business Units
Asset Management / Asset Identification
Organisation / Third Parties
Risk Mgt / Asset Risk Mgt
Risk Mgt / Third Party Risk Mgt
Risk Mgt / Business Impact Analysis
Reviews
Control Catalogue / Internal Controls
Control Catalogue / Policies
Risk Management / Exceptions
Sec. Operations / Projects

Figure A.2: Risk Management Module [19].

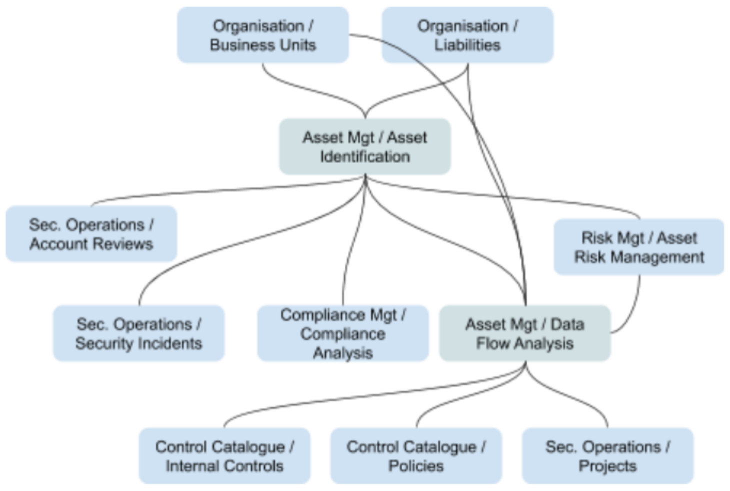

Organisation / Business Units
Organisation / Liabilities
Asset Mgt / Asset Identification
Sec. Operations / Account Reviews
Risk Mgt / Asset Risk Management
Sec. Operations / Security Incidents
Compliance Mgt / Compliance Analysis
Asset Mgt / Data Flow Analysis
Control Catalogue / Internal Controls
Control Catalogue / Policies
Sec. Operations / Projects

Figure A.3: Asset Management Module [19].

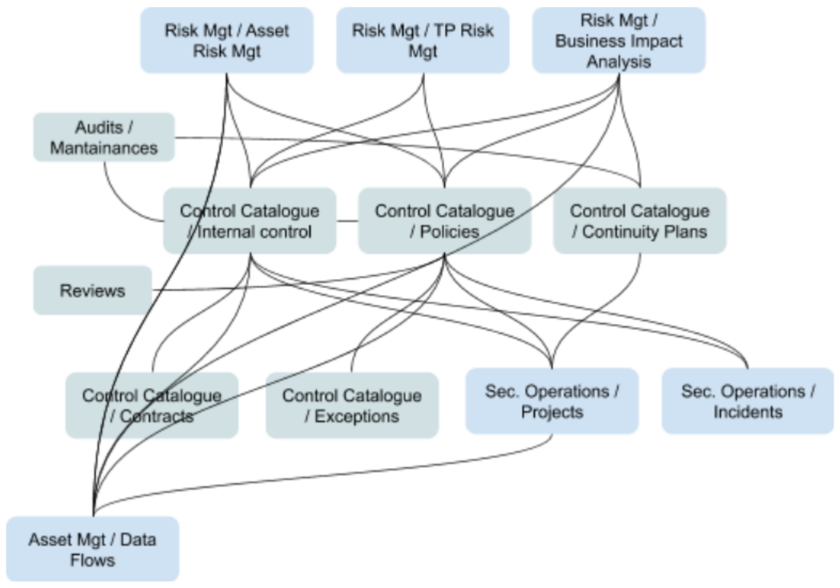

Risk Mgt / Asset Risk Mgt
Risk Mgt / TP Risk Mgt
Risk Mgt / Business Impact Analysis
Audits / Mantainances
Control Catalogue / Internal control
Control Catalogue / Policies
Control Catalogue / Continuity Plans
Reviews
Control Catalogue / Contracts
Control Catalogue / Exceptions
Sec. Operations / Projects
Sec. Operations / Incidents
Asset Mgt / Data Flows

Figure A.4: Control Catalogue Module [19].

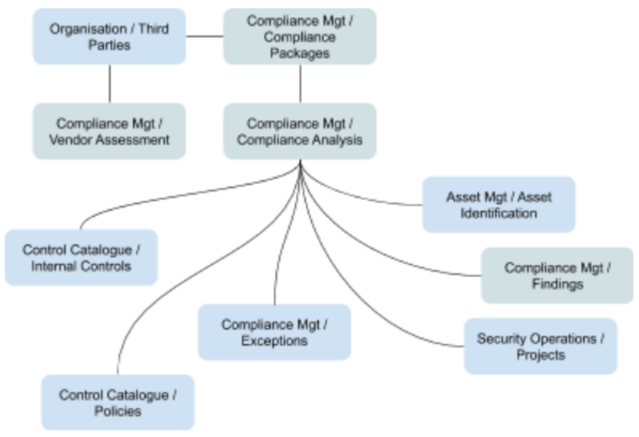

Organisation / Third Parties
Compliance Mgt / Compliance Packages
Compliance Mgt / Vendor Assessment
Compliance Mgt / Compliance Analysis
Asset Mgt / Asset Identification
Control Catalogue / Internal Controls
Compliance Mgt / Findings
Compliance Mgt / Exceptions
Security Operations / Projects
Control Catalogue / Policies

Figure A.5: Compliance Management Module [19].

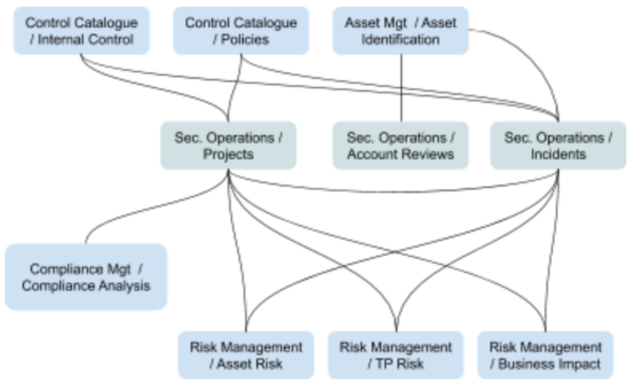

Control Catalogue / Internal Control
Control Catalogue / Policies
Asset Mgt / Asset Identification
Sec. Operations / Projects
Sec. Operations / Account Reviews
Sec. Operations / Incidents
Compliance Mgt / Compliance Analysis
Risk Management / Asset Risk
Risk Management / TP Risk
Risk Management / Business Impact

Figure A.6: Security Operation Module [19].

Section	Description
Dashboard	Shows a summary for all major KPIs on the system, these KPIs are pre-defined key metrics from the core modules.
Program	Describes the scope of your GRC program. Describes internal or external program Describes the program goals, objectives and issues.
	Describes the members of the GRC team, its members, and their competences.
Organization	Describes the organization’s business units and business processes.
	Describes third-party affiliations and obligations.
	Describes organizational liabilities.
Asset Mgt	Identifies and classifies assets.
	Describes the flow a data asset goes through throughout its lifecycle in the organization.
Control Catalogue	Describes the organization’s internal controls, their testing, audits, issues and maintenance. Describes contracts their expiration date, value and these can be linked to controls.
	Describes continuity plans and their tasks. Describes policies and track their reviews. Manages policy exception requests to policies described in the Policy module.
Risk Mgt	Manages asset-based risk and their reviews. Manages third-party risk and their reviews. Manages business process risk, their impact on the organization, and their reviews. Manages exception requests to risks and links them to risks defined in the Risk module.
Compliance Mgt	Manages compliance exceptions and links them to Compliance Analysis module. Manages the list of regulatory, standards and frameworks the organization needs to be compliant with. Maps compliance requirements to controls, policies, risks, etc.
	Documents compliance findings and keeps track of their progress, deadlines, etc..
Security Operations	Manage custom questionnaires and submit them to different audiences to gather remote feedback through Eramba’s web portal. Describes security incidents and their lifecycle. Manages custom training and awareness activities through the use of videos, multiple choice and disclaimer texts. Activities are assignable using Active Directory groups and can be scheduled regularly to meet compliance needs. Collects account information from different systems and ensures they are reviewed regularly. Describes projects and their tasks, this module links to all other modules in Eramba.

Table A.1: Eramba specifications extracted from Basic GRC relationships [19].

Appendix B

Eramba Dockerfile

```
version: "3.7"
services:
  db:
    image: markz0r/eramba-db
    environment:
        MYSQL_DATABASE: erambadb
        MYSQL_USER: eramba
        MYSQL_PASSWORD: root
        MYSQL_ROOT_PASSWORD: root
  app:
    build: .
    depends_on:
      - db
    environment:
      ERAMBA_HOSTNAME: app
      MYSQL_HOSTNAME: db
      MYSQL_USER: eramba
      MYSQL_DATABASE: erambadb
      MYSQL_PASSWORD: root
      DATABASE_PREFIX: ""
      DB_SCHEMA_SCRIPT: /c2.8.1.sql

    ports:
     - "8080:8080"
    links:
     - db
```

Listing B.1: Eramba dockerfile.

Appendix C

Eramba Kubernetes Solution

```
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    kompose.cmd: kompose -f docker-compose-copy.yml convert
    kompose.version: 1.21.0 (992df58d8)
  creationTimestamp: null
  labels:
    io.kompose.service: app
  name: app
spec:
  replicas: 1
  selector:
    matchLabels:
      io.kompose.service: app
      app: app
      tier: web
  strategy: {}
  template:
    metadata:
      annotations:
        kompose.cmd: kompose -f docker-compose-copy.yml
          convert
        kompose.version: 1.21.0 (992df58d8)
      creationTimestamp: null
      labels:
        io.kompose.service: app
        app: app
        tier: web
    spec:
      imagePullSecrets:
      - name: regcred
      containers:
      - env:
        - name: MYSQL_DATABASE
          value: erambadb
      - name: MYSQL_PASSWORD
        value: root
      - name: MYSQL_ROOT_PASSWORD
        value: root
      - name: MYSQL_USER
        value: eramba
      image: markz0r/eramba-db
      imagePullPolicy: ""
      name: db
      resources: {}
    - env:
      - name: DATABASE_PREFIX
      - name: DB_SCHEMA_SCRIPT
        value: /c2.8.1.sql
      - name: ERAMBA_HOSTNAME
        value: app
      - name: MYSQL_DATABASE
        value: erambadb
      - name: MYSQL_HOSTNAME
        value: 127.0.0.1
      - name: MYSQL_PASSWORD
        value: root
      - name: MYSQL_USER
        value: eramba
      image: registry.gitlab.com/miguelrchaves/thesis:latest
      livenessProbe:
        httpGet:
          path: /
          port: 8080
        periodSeconds: 5
        timeoutSeconds: 60
        successThreshold: 1
        failureThreshold: 3
        initialDelaySeconds: 70
      readinessProbe:
        httpGet:
          path: /
          port: 8080
        periodSeconds: 5
        timeoutSeconds: 60
        successThreshold: 1
        failureThreshold: 3
        initialDelaySeconds: 70
      imagePullPolicy: ""
      name: app
      ports:
      - containerPort: 8080
      resources: {}
    restartPolicy: Always
```

```
      serviceAccountName: ""
      volumes: null
status: {}
---
apiVersion: v1
kind: Service
metadata:
  annotations:
    kompose.cmd: kompose -f docker-compose-copy.yml convert
    kompose.version: 1.21.0 (992df58d8)
  creationTimestamp: null
  labels:
    io.kompose.service: app
    app: app
  name: app
spec:
  ports:
  - name: "8080"
    port: 80
    targetPort: 8080
  type: NodePort
  selector:
    io.kompose.service: app
    app: app
    tier: web
```

Listing C.1: Eramba Kubernetes solution.

```

apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-ml
  annotations:
    kubernetes.io/ingress.global-static-ip-name: "eramba-ip"
spec:
  rules:
  - host: "eramba.organizationDNSdomain.com"
    http:
      paths:
      - backend:
          serviceName: app
          servicePort:


so


\begin{lstlisting}[language=yaml,caption={Eramba Managed
   Certificate.},label={erambassl},captionpos=b]
```

```yaml

apiVersion: networking.gke.io/v1beta2
kind: ManagedCertificate
metadata:
  name: erambacertificate
spec:
  domains:
    - "eramba.organizationDNSdomain.com"
```

Listing C.2: Ingress to create a load balancer.