C-RAT
A Tool for Cyber-Risk Aggregation

Rodrigo Manuel de Mansilha Flaminio Ribeiro
Master’s Thesis, Spring 2019
Abstract

Today we are living in a digital era characterised by technology which increases the speed and breadth of knowledge turnover within the economy and society. Moreover today we are not allowed to stand still, if we do not move forward we will rapidly be out-performed, out-passed and forgotten. In order to stay relevant organisations needs to be in constant evolution, always in the look for a new opportunity.

However the advances in technologies we made in the past decades also have negative consequences. Cyber threats and cyber crimes, are also in the rise, and the tendency in for them to keep growing, with the latest reports reporting new records in cyber crimes[8].

For this reason, organizations now, more than never, need to pay particular attention to their cyberstate, otherwise they do not only risk getting attack, but also to be fined by governments. Unfortunately, have a good understanding of the cyber landscape of an organisation is very hard task, especially without a proper support tool.

In this thesis we design and create C-RAT, our attempt to mitigate this ongoing problem.

C-RAT, is a stand-alone tool for cyber-risk management created for big organisations. In witch we try help managers in different levels through risk management, risk aggregation and segregation of the risk picture. Allowing at the same time, to abstract from risk, but focus on its origins or consequences.

In order to build this tool we had to focus in three domains:

- Risk Management, once it is the subject of our work;
- Software Development, once we had to build a tool;
- Cybersecurity, once a tool of this nature, needs to have defence mechanism.
I would like to thank everyone who assisted me in various ways and helped me over the duration of the thesis.

First of all, I would like to thank my supervisor, Ketil Stolen for all the patience he has demonstrated to me, and his ability to show me in what areas I should improve.

I am also very grateful to my parents Manuel and Monica, not only for allowing me to come to Oslo and supporting me during my studies but also for being obnoxious concerned with me.

To my sisters Filipa and Patricia for being the best sisters in the world! For always having a smile and motivate their older brother. Filipa, for being one of my biggest supporters and for being always to my side and Patricia for always do humorous stuff that makes me laugh.

I would also like to thanks to my cousins, Ana Teresa, Clara and Margarida. To Ana Teresa, who already did her master, and that showed me then we should enjoy life while writing a thesis, and that serves as a role model. To Margarida and Clara for showing me that life should be lived to the fullest, and for being always happy, laughing, smiling and singing.

In the same note I have to thank my family for always motivate me, and checking up on me from time to time, in this group, I have to thanks especially to my godfather Jorge, once he did not only motivate me but provide me with some helpful tips, during the time I was writing my thesis.

During my time at Oslo, I was blessed to have fantastic flatmates like Sandra, Tian and Damir, who without I would have probably got depressed halfway through.

Lastly but not least, I could not have finished this adventure without the support and motivation from my "putos dos dcc", that always treated me like I was still with them. I have to give a special mention to my good friend Pedro Cunha who heard my struggles and complains during all two years of the master and always kept me motivated.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Abstract</strong></td>
<td>i</td>
</tr>
<tr>
<td><strong>Acknowledgements</strong></td>
<td>iii</td>
</tr>
<tr>
<td><strong>Contents</strong></td>
<td>v</td>
</tr>
<tr>
<td><strong>List of Figures</strong></td>
<td>ix</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Overview</td>
<td>2</td>
</tr>
<tr>
<td><strong>2 Background</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 Risk Management</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 What is Risk?</td>
<td>5</td>
</tr>
<tr>
<td>2.1.2 Risk Management Process</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Cybersecurity</td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Cyberspace &amp; Cyber-Systems</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 What is Cybersecurity?</td>
<td>8</td>
</tr>
<tr>
<td>2.2.3 Information Security &amp; Cybersecurity</td>
<td>9</td>
</tr>
<tr>
<td>2.2.4 Cryptography</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Software Development</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1 Agile</td>
<td>11</td>
</tr>
<tr>
<td><strong>3 Thesis Statement and Success Criteria</strong></td>
<td>13</td>
</tr>
<tr>
<td>3.1 Measurement of cyber-risks</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Aggregation of cyber-risk</td>
<td>14</td>
</tr>
<tr>
<td>3.3 Producing Software</td>
<td>15</td>
</tr>
<tr>
<td>3.3.1 Prototyping</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Success Criteria</td>
<td>16</td>
</tr>
<tr>
<td><strong>4 State of the Art</strong></td>
<td>19</td>
</tr>
<tr>
<td>4.1 Cyber-Risk Assessment</td>
<td>19</td>
</tr>
<tr>
<td>4.1.1 Identifying Risks</td>
<td>19</td>
</tr>
<tr>
<td>4.1.2 Risk Analysis</td>
<td>20</td>
</tr>
<tr>
<td>4.1.3 Risk Evaluation</td>
<td>21</td>
</tr>
<tr>
<td>4.2 Software Development</td>
<td>22</td>
</tr>
</tbody>
</table>
## Contents

4.2.1 Java ................................................. 22
4.2.2 Testing Software ................................. 22
4.2.3 Testing during development .................. 22
4.2.4 Unit Testing ...................................... 23
4.2.5 Software Documentation ...................... 23

4.3 Cybersecurity ..................................... 24
4.3.1 Malicious Threats .............................. 24
4.3.2 Testing within Security ....................... 25

4.4 Human-Computer Interaction ................. 26
4.4.1 Design Concepts ............................... 27
4.4.2 Usability Testing .............................. 27

5 Research Method ............................... 31
5.1 Introduction ..................................... 31
5.2 Technology Research ........................... 31
5.3 Evaluation Strategies ........................... 32
5.4 Selection of appropriate evaluations strategies 34
5.4.1 Artefact Description .......................... 35
5.4.2 Security Risk Assessment ................... 35
5.4.3 Usability Testing ............................. 36

6 Research Based Design ....................... 37
6.1 Artifact Design .................................. 37
6.2 Network Design .................................. 39
6.2.1 Multi-Threaded Server ....................... 40
6.3 Security Design .................................. 41
6.3.1 Security Risk Assessment, Before .......... 42
6.4 Graphical User Interface ...................... 45
6.5 Risk Segregation ............................... 46
6.6 Database Structures ............................ 50
6.6.1 Risk ............................................. 50
6.6.2 User .......................................... 50
6.6.3 Login ......................................... 51
6.7 Summary ......................................... 52

7 Evaluation ..................................... 55
7.1 Artefact Description ............................ 55
7.1.1 Setup ......................................... 56
7.1.2 Start the Client and Authenticate the User 56
7.1.3 UserPage and Risk Insertion and Management 60
7.1.4 Risk Aggregation ............................. 64
7.1.5 Risk Segregation ............................. 66
7.2 Security Risk Assessment ..................... 67
7.3 User Testing Results ............................ 68

8 Discussion .............................. 71
8.1 Success Criteria 1 ............................... 71
8.2 Success Criteria 2 ............................... 72
8.3 Success Criteria 3 ............................... 73
8.4 Success Criteria 4 ............................... 73
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.5 Success Criteria 5</td>
<td>74</td>
</tr>
<tr>
<td>8.6 Success Criteria 6</td>
<td>74</td>
</tr>
<tr>
<td>8.7 Success Criteria 7</td>
<td>74</td>
</tr>
<tr>
<td>8.8 Success Criteria 8</td>
<td>75</td>
</tr>
<tr>
<td>9 Conclusion</td>
<td>77</td>
</tr>
<tr>
<td>9.1 Directions for Future Work</td>
<td>78</td>
</tr>
<tr>
<td>Bibliography</td>
<td>81</td>
</tr>
<tr>
<td>Appendices</td>
<td>85</td>
</tr>
<tr>
<td>A User Testing Questionnaire</td>
<td>87</td>
</tr>
<tr>
<td>B Printings from C-RAT client</td>
<td>91</td>
</tr>
<tr>
<td>C User Testing Results</td>
<td>101</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Risk Assessment Process [44] ............................................. 7
2.2 Symmetric Cryptography [1] ............................................. 10

3.1 Approaches to measuring cyber-risk [45] ............................. 14

4.1 Process for cyber-risk assessment [37] ................................. 20

5.1 Method for technology research – main steps [40] .................... 32
5.2 Evaluation strategies [40] .................................................. 33

6.1 Sequence Diagram: Users Platform Interaction ..................... 38
6.2 Multiple Service Threads Dispatched in One Server Process [47] 40
6.3 Asset Diagram .............................................................. 43
6.4 CAPEC scales [4] .......................................................... 44
6.5 Risk Identification List .................................................... 44
6.6 Risk Matrix ................................................................. 44
6.7 C-RAT Example Organisation Chart ................................. 48
6.8 C-RAT Risks, manager-risk owner example ......................... 49
6.9 Database tables ............................................................ 52
6.10 Threat Diagram .......................................................... 54

7.1 Sequence Diagram Establishing Connection ......................... 57
7.2 Sequence Diagram: Secure Protocol Sequence Diagram ........... 58
7.3 Sequence Diagram: Login Sequence Diagram ....................... 59
7.4 Sequence Diagram: User Page Sequence Diagram ................. 61
7.5 Sequence Diagram: New Risk Sequence Diagram .................. 62
7.6 Sequence Diagram: Update Risk Sequence Diagram ............... 63
7.7 Risk Identification List After ............................................ 68
7.8 Risk Matrix After ........................................................ 68
7.9 Before-After Threat Diagram ......................................... 70

B.1 Login Page ................................................................. 91
B.2 Login Page invalid password ............................................ 91
B.3 Risk Owner UserPage .................................................... 92
B.4 High-Level Manager UserPage ........................................ 93
B.5 Aggregate By Scenario .................................................. 94
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.6</td>
<td>Aggregate By Asset</td>
<td>95</td>
</tr>
<tr>
<td>B.7</td>
<td>Risk Segregation on Manager</td>
<td>96</td>
</tr>
<tr>
<td>B.8</td>
<td>Risk Segregation on Risk Owner</td>
<td>97</td>
</tr>
<tr>
<td>B.9</td>
<td>Low-level Manager UserPage</td>
<td>98</td>
</tr>
<tr>
<td>B.10</td>
<td>Simultaneous Aggregation and Segregation</td>
<td>99</td>
</tr>
<tr>
<td>B.11</td>
<td>Manager Confirmation Box on update</td>
<td>100</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

In the first chapter, we present the motivation for our thesis, what is the problem, and a general idea of what we want to achieve. In the end, we provide an overview of the thesis.

1.1 Motivation

In the last years, there has been a significant increase in cyber-crime for example in 2016 there were more than 4000 ransomware attacks per day, which in between other cyber-crimes lead to an increase of security incidents across all industries of 38%, which was the most significant increase in the past 12 years[8].

Together with the augment of cyber crimes, there are a number of new laws and regulations regarding this subject, with more to come in the next years[7]. Furthermore, governments are fining heavily regarding loss of customer data, either by data-breaches[17], or intentional selling of the information[49][52].

For these reasons companies, now more than never, need to pay special attention to their current cyber-state. Hence if they do not, they are not only exposing themselves to cyber-attacks but also to be fined by governments.

While not all industries have regulatory obligations with respect to cyber-risk management, in practice, they all need it. Therefore, managers in this area need to have a good understanding of the cyber-state in their domain, once they want to implement the best defences mechanisms to mitigate the inevitable damages that will appear.

However, having a good understanding of the current cyber-state is a very hard task. Managers not only need to consider a vast variety of elements but also, the field is in constant change and evolution, which makes the task even harder.

Furthermore, managers in different positions, want to look into different aspects of the cyberspace, while one may be concerned with potential security breaches due to cyber-attacks, others may be concerned whether opening specific ports in a firewall would imply unacceptable risk, or even with the implications of a poor judgement tweet.

In addition, managing directors, cannot afford to be concerned with specific or low-level problems. What they need is an aggregated view that accurately represents the overall cyber-state of the company, in a complete but simple
1. Introduction

What they need a thorough understanding of the current cyber-state of the organisation, in order to make the best decisions.

Unfortunately, having such a view without the help of a proper support tool is close to impossible. To tackle this problem, the goal of this thesis is to create a tool that allows managers of big organisations, to have an overall but complete view of the cyber-state in their area of responsibility.

1.2 Thesis Overview

This thesis is organised in nine chapters as follows:

Introduction The Introduction chapter is divided into two parts, first the motivation for the thesis, and then the thesis overview that gives an overview of all the chapter that constitute the thesis.

Background The background chapter is divided into three sections, one for each of the main topics of the thesis, namely risk management, cybersecurity and software development. In all three of the sections, we cover the basic knowledge required to understand the thesis.

Thesis Statement and Success Criteria In the third chapter of this thesis, we start by specifying the thesis problem, following we address the three main questions we needed to sort of before we started implementing the tool, and finally, we present the success criteria.

State of the Art In the state of the art chapter, we narrow and combine the topics we presented in the background. In the first section, we start by intersecting the topics of risk management and cybersecurity. Then we turn our attention to how to develop software, in this section we start by explaining why Java is a good fit for our project, following we present some techniques to test and develop software. In the section that follows we shift our attention to cybersecurity, in there we start by understanding against whom we have to defend, later we present some good practices, and techniques developers should follow when developing or designing software, in order to mitigate the harm from future cyber-attacks. Lastly, we look at how to design user interfaces. In here, we start by present some designing principles. Moreover, we explain what user testing is and why it is relevant in this field.

Research Method In the fifth chapter we describe how the research in this thesis was conducted. The chapter is divided into 4 sections: the first one is an introduction to research, where we explain the main topics and present the main philosophies. The second chapter is dedicated to technologies research, in here we explain Finally in the section 5.4 and his subsection we discuss the evaluation the evaluation strategies we consider appropriate for this thesis.

Research Based Design In this chapter, we transform the abstract goals we defined in the third chapter of the thesis, into specific requirements,
1.2. Thesis Overview

to which C-RAT must comply. In order to achieve these requirements, we looked at the tool through multiple point-of-views, besides, for each different way we look to the tool we start by presenting the goals we want to achieve in that particular domain, and then processed to find requirements that would allow them.

**Evaluation** As the name suggests, in the evaluation chapter we will evaluate our tool, through the methods we defined in the research method chapter.

**Discussion** In the eighth chapter, we will look to the success criteria we defined in the third chapter, and argue to what point what it fulfilled.

**Conclusion** In the last chapter, we conclude the thesis and present what we believe to be the majors’ differences from C-RAT when compared to other risk management tools. Lastly, we provide direction for future work.
In order to create and evaluate this tool, one needs to have a basic understanding of several topics. In this chapter, we will provide a brief explanation of risk management, cybersecurity and software development.

2.1 Risk Management

In this section will give a brief explanation of risk management. However, before understanding risk management we need to clarify some terminology and basic concepts.

2.1.1 What is Risk?

Most people have a reasonable understanding of what risk is. Something that can cause harm or loss and its gravity is often more tied to its consequence. While this notion is not incorrect, we will define risk as:

Definition 2.1.1. A risk is the likelihood of an incident and its consequence for an asset. [37]

If we look closely to the definition we can easily see four concepts that might not be so clear, namely likelihood, incident, consequence and asset. In order to have a more precise understanding of this definition we will define these four concepts.

Let us start with the incident concept. When someone speaks about risk, there is a need to make a distinction between what causes an incident, and the potential consequences of it, if it would happen. While understanding how these events unfold is critical to understand how risks happen, however, the incident itself is the only event that causes harm or loss.

Definition 2.1.2. An incident is an event that harms or reduces the value of an asset. [37]

This definition highlights that risk is about the occurrence of harmful events. Now we must ask the question to whom are these events harmful? Well, the answer is to whoever values the asset. If we include the definition of asset together with the incident one, we are forced to specify which events can be understood as incidents.
2. Background

**Definition 2.1.3.** An **asset** is anything of value to a party.\[^{37}\]

A party is an entity or person that values the asset. If no one valued an asset, then it could not be harmed, hence it had no value. Therefore an asset cannot exist without a party.

**Definition 2.1.4.** A **party** is an organisation, company, person, group or other bodies on whose behalf a risk assessment is conducted.\[^{37}\]

Before someone discusses risk assessment, there is a necessity to determine the party, and by extensions the assets that concern them. Only after these two elements are determined the discussion about risk assessment can be productive, once it now can occur in a precise and meaningful manner.

The remaining concepts from the definition of risk are likelihood and consequence.

**Definition 2.1.5.** A **likelihood** is the chance of something occur.\[^{37}\]

This notion is only concerned with the chance of something happen, disregarding its representation.

**Definition 2.1.6.** A **consequence** is the impact of an incident on an asset in terms of harm or reduced value.\[^{37}\]

It is important to note that consequence only accounts for negative impacts on an asset. As mentioned before both consequence and likelihood are used to measure the gravity of a risk. Additionally we can also measure the gravity in terms of risk level.

**Definition 2.1.7.** Risk **level** is the magnitude of a risk as derived from its consequence.\[^{37}\]

### 2.1.2 Risk Management Process

Everything is exposed to risk, everything and everyone has something of worth to them, and if something is worth to anyone it can also be harmed. So in order to minimize the exposition of risk from our assets, risk management was created.

**Definition 2.1.8.** Risk Management comprises coordinated activities to direct and control an organization with regards to risk.\[^{37}\]

Organisations are the main users of Risk Management. Their goal when using this set of activities is to minimize risk to their assets. This is mostly accomplished by reducing their exposure to **threats**.

**Definition 2.1.9.** A **threat** is an action or event that is caused by a threat source and that may lead to an incident.\[^{37}\]

A risk management process, must be based on a risk management framework; such framework should comply with basic principles for risk management like
ISO31000[12]; furthermore, the framework should be monitored and improved on a constant basis.

The risk management process is further divided into three components. Namely *communication and consultation, monitor and review, and risk assessment.* From these three components, the first two are endless activities, while risk assessment is a finite process that organisations should conduct regularly.

**Risk Assessment**

Risk assessment is composed of activities aiming to understand and document the risk picture for specific parts or aspects of a system or an organisation.\[37\]

As you can see in fig.2.1 there are five steps concerning risk assessment: Establishing Context, Risk Identification, Risk Analysis, Risks Evaluation, and Risk Treatment.

![Figure 2.1: Risk Assessment Process](image)

The first step is establishing the context. This is a preparatory step for the subsequent activities. The purpose of this step is to document internal and external contexts of relevance for the given risk assessment, as well as define its goals and objectives.

The risk identification step, as suggested by its name, is comprised of activities with the goals of identifying, describing and documenting risks, and their possible causes.

After we identify risk, we go to the risk analysis step. In this step, one is supposed to conduct activities aiming to estimate and determine the risk level in the identified risks.

The fourth step is risk evaluation, in this step, one will conduct activities with the goal of comparing the risk analysis results with the risk evaluation criteria in order to determine what risks should go to the treatment step.
2. Background

The final step is the risk treatment; in this step, one is supposed to conduct abilities to identify and select means for risk mitigation and reduction.

2.2 Cybersecurity

In this section, we will cover the basics of cybersecurity. We will start by explaining in what context does make sense to talk about cybersecurity and what characteristics are tied to it. Additionally, we also explain the difference between information security and security once the terms are sometimes confused. Before we conclude the section, we will still give an introduction to cryptography.

2.2.1 Cyberspace & Cyber-Systems

The way organisations conduct risk management is mostly dependent on the kind and nature of their systems. Additionally, we are concerned about their applications to cybersecurity and therefore cyber-systems.

Definition 2.2.1. A system is a set of related entities that forms an integrated whole and has a boundary to its surroundings.  

While the internet is the most famous example of cyberspace, we must not confuse the terms since they are not interchangeable. Hence we define cyberspace as:

Definition 2.2.2. A cyberspace is a collection of interconnected computerized networks, including services, computer systems, embedded processors, and controllers, as well as information in storage or transit.

While for most entities cyberspace is the same as the Internet (which is a global cyberspace in the public domain), our definition is a little broader since every collection of interconnected networks are a cyberspace. The more common example of our definition are LANs, other examples to which our definition also applies are the military and emergency communications networks and systems.

To have a better understanding of the relation between risks and cyberspace is the scope of the subject matter. If we take into account risks that are driven or that are derived from the cyberspace, such as the Internet, have clear implications that extend the cyberspace alone. A system that is in some form dependent on the cyberspace may also be vulnerable to its dependency. To take this into account the concept of cyber-system was introduced:

Definition 2.2.3. A cyber-system is a system that makes use of a cyber space.

2.2.2 What is Cybersecurity?

"In today’s connected world, everyone benefits from advanced cyberdefense programs. At an individual level, a cybersecurity attack can result in everything from identity theft, to extortion attempts, to the loss of important data like family photos. Everyone relies on critical infrastructure like power plants, hospitals, and
financial service companies. Securing these and other organizations is essential to keeping our society functioning.\textsuperscript{[6]}

As we can see from the quote above cybersecurity is crucial in today’s world. And while the quote did not explain what cybersecurity is, one can deduct that it is concerned with the protection of the cyberspace. On the other hand organisations are more concerned with protection of their own cyber-systems than with the protection of the cyberspace itself. Both of these optics are within our definition of cyber-security.

**Definition 2.2.4.** Cybersecurity is the protection of cyber-systems against cyber-threats.\textsuperscript{[37]}

A cyber-threat is a threat that arrives via the cyberspace and by consequence a threat that any cyber-system is exposed to.

**Definition 2.2.5.** A cyber-threat is a threat that exploits the cyberspace.\textsuperscript{[37]}

It is important to note that there are two very different kinds of cyber-threat, malicious and non-malicious. Example of malicious threats is SQL injection attacks or dictionary attacks on passwords, in short attacks that are caused by intention. On the other hand, examples of non-malicious threats are crashes due to loss of communication with the internet or due to hardware malfunction.

It is of the most importance to remember that cybersecurity is not defined by what we seek to protect, but rather what we seek to protect it from. In other words, it is not defined by the assets that we want to protect, but instead of the threats to those assets.

### 2.2.3 Information Security & Cybersecurity

Information security is concerned with the preservation of confidentiality, integrity and availability of information.\textsuperscript{[16]}

- **Confidentiality** Only authorised actors can see the information;
- **Integrity** Only authorised actors can change the information;
- **Availability** Actors must have access to information whenever they need it.

Furthermore, information security is not concerned, with the state where the information is, as it is concerned with its protection from all threats, may it be physical, human or technology related. Cybersecurity, on the other hand, is concerned with the protection from targets that make use of the cyberspace. Therefore we can conclude, that there is an overlap of these concepts only when it comes to the security of information from threats that makes use of the cyberspace. A good number of standards and guidelines on cybersecurity relate these two concepts since there is a significant overlap. However, in order to have a proper understanding of cybersecurity one must not confuse the concepts.
2. Background

2.2.4 Cryptography

Since ancient times, people have the need to keep secrets. People have the need to transmit information hidden from others. However, since the being of the 20 century, this new science started evolving at a much rapid pace.

*Cryptography is the science of secret writing with the goal of hiding the meaning of a message* [34]

In today’s modern world there are 2 methods of encrypting a message: symmetric cryptography and public key cryptography.

The base for symmetric cryptography is that both parties share a common key, and this key isn’t known by anyone else, Fig. 2.2. Today’s symmetric cryptography is mainly composed by stream cipher and block cipher. The most used algorithms that derive from this theory are in the same order 3DES and AES.

![Figure 2.2: Symmetric Cryptography](image)

However there are 3 main problems with the symmetric cryptography:

1. The key must be transported securely;
2. In a network each pair of users require an individual key, for example in network of 1000 people you would need 499500 pairs of keys for everybody communicate securely with everybody;
3. Alice and Bob can cheat each other: Alice can claim that she never ordered a TV online from Bob, causing a non-repudiation issue;

In order to solve these problems the public key cryptography is born.

The public key cryptography is based, on 2 keys: the public key and the private key. Both of them are generated by you, and then you share the public. Its security is based on one-way functions:
2.3. Software Development

**Definition 2.2.6.** A function \( f() \) is a *one way function* if:

1) \( y = f(x) \) is computationally easy, and
2) \( x = f^{-1}(y) \) is computational infeasible.

Two candidates one way functions are the **modular power function** and **modular exponentiation**. This theory gave birth to 2 of most known and used algorithms for encrypting, respectively RSA, and El Gamal.

### 2.3 Software Development

"Software engineering is the profession that applies scientific knowledge in the construction of software products needed by customers."

In the early stages of development, the developer team should be concerned about the context, the goals and the reason behind the project. After having a reasonable understanding of this one can start extracting the requirements.

After the first gather of requirements, the developer team should follow a software development methodology like agile or scrum. A proper software development methodology should address not only the technological aspect but also be concerned with the work environment and the professional framework.

When developing software, a developer should follow some good coding practices in order to improve software quality. According to Sommerville[41], software quality should be concerned with maintainability, dependability and security, efficiency and acceptability.

**Maintainability** : Software should be written in such a way so that it can evolve to meet charging of customers. This is a critical attribute since software change is an inevitable requirement of a changing business environment.

**Dependability and Security** : Software dependability requires a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of a system failure. Malicious users should not be able to access the damaged system.

**Efficiency** Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc...

**Acceptability** Software must be accepted to the type of users for which it is designed. This mean that it must be understandable, usable, and compatible with other systems that they use.

#### 2.3.1 Agile

The Agile Manifesto was formulated in the Wasatch Mountains of Utah in February of 2001 by seventeen software practitioners.
2. Background

It states four main principles when developing software: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan.

The first principle states that we should focus on the individuals involved in the development process rather than on the process or the tools, making sure everyone understands the development process and is able to help on the development decisions.

The next principle states that the primary goal of a software project is to produce quality software. Moreover, in order to achieve it, we should: make the necessary documentation available to everyone involved at any time; start the software development process as early as possible, and finally, produce a bug-less high-quality product that meets the customer requirements.

The third principle means to change the perception of the customer role in software development processes, asking for the development team to be in contact with the customer on a daily basis.

The last principle was born from the understanding that it is almost impossible to predict all requirements before the development starts, and so, it demands instead of being the focus on a rigid schedule, the software development should be prepared for change while not damaging the software quality.

Finally, the agile process also recommends that the software should be based on short releases (about two months), divided into iterations of 2 weeks. At the end of each iteration, the progress should be shown to the customer. Next, the customer provides feedback to the developer team and they agree on should be the concern(s) of the next iteration.
In this chapter, we will further explore the needs and goals of this thesis.

As we stated in the first chapter of this thesis its focus was to research and then develop a tool with the goal to help managers of big companies to have a better grasp of the cyber-state in their domain of responsibility through aggregation of cyber-risks. This tool was developed to help with the third and fourth steps of a cyber-risk management process, respectively risk analysis and risk evaluation.

The central question to this thesis is: *How can risk aggregation help managers to have a better understanding of the current cyber state of their domain of responsibility?*

Before we can answer this question we must understand:

- How can we measure the risk of a cyber incident;
- How can we aggregate cyber-risks;
- How should the information be presented to managers in different levels of the organisations;
- How should we produce our artefact;
- What characteristics should our software have;

### 3.1 Measurement of cyber-risks

As stated in *OWASP Testing Guide v4*\(^4\) when identifying a security threat we must rate it or measure it; however there are different approaches to rate or measure risk.

As illustrated in fig.3.1 we may use any number of factors, depending on the situation the recommended approach may vary. For example: in cases, where the likelihood is extremely low it may be better to disregard it and focus only on the harm the risk can cause. In the other hand in the security field, the three factors approach is one of the most popular\(^4\).
3. Thesis Statement and Success Criteria

There are many other factors we can consider when measuring cyber-risk like the ease of exploit of vulnerability or awareness of the vulnerability by the attackers. However as we stated in the introduction of this chapter, our primary concern is with automatic aggregation, and because of this, we chose a more practical approach. We will be using a 2 factors approach: consequence and likelihood. By consequence during this thesis we will define risk as:

Definition 3.1.1. A risk is a quadruple \((s, a, f, c)\), where \(s\) is a potential scenario, \(a\) is an asset, \(f\) is the number of times \(s\) occurs per year, and \(c\) is the average loss or damage for the asset \(a\) when \(s\) occurs\[45\].

![Figure 3.1: Approaches to measuring cyber-risk](image)

3.2 Aggregation of cyber-risk

In order to support automated aggregation we need to define rules or algorithms that specify how the computer should combine two or more risks in one. Furthermore as we explained in the previous section one of our measure factors is likelihood, therefore we need to be careful to not count one occurrence more than once, this being said we need to state the concept of overlap:

Definition 3.2.1. Two scenarios \(s_1\) and \(s_2\) overlap if one occurrence of \(s_1\) may count as an occurrence of \(s_2\), or the other way around.\[45\]

According to chapter 9 of the book *Cyber-Risk Management*\[37\], there are two main reasons for combining risk incidents. Either when two or more incidents harm an asset or when an incident harms two or more assets. Additionally, the report *Guidelines for aggregation of cyber-risk in large institutions*\[45\], provided us with three main rules for risk aggregation (frequency aggregation, consequence aggregation and asset aggregation).

- Rule 1 (frequency aggregation) is to be applied when two scenarios \((s_1\) and \(s_2)\) and frequency \((f_1\) and \(f_2)\) do not overlap, in that case, the new scenario will have a frequency \(f_1 + f_2\).
- Rule 2 (consequence aggregation) is to be applied when two scenarios \((s_1\) and \(s_2)\) with frequency \((f_1\) and \(f_2)\) and consequence \((c_1\) and \(c_2)\) target the same asset \((a_1)\). In this case the new scenario will have a consequence of \(\frac{f_1 \times c_1 + f_2 \times c_2}{f_1 + f_2}\).
• Rule 3 (asset aggregation) is to be applied when one scenario harms two disjoint assets \((a_1\) and \(a_2\)) with different consequence \((c_1 + c_2)\). In this case, the scenario will occur with consequence \(c_1 + c_2\) for asset \(a_1\) and \(a_2\).

We can conclude that the frequency aggregation and consequence aggregation rules should be used when we want to combine the potential scenarios. Following these rules we can quickly identify the potential loss of each asset. In opposition, the asset aggregation rule should be used when we want to know which scenario is expected to cause more harm.

### 3.3 Producing Software

There are diverse techniques and methodologies to produce software. Being the most popular methodologies waterfall, agile and spiral.

The waterfall development is characterised by being a sequential approach to software development, in which the development can be compared to levels in a game, in a sense that you can only start the next phase once you clear the previous one. Furthermore this methodology normally follows the following sequence: Requirements, Design, Implementation, Testing and Deployment.

Contrary to the strict waterfall methodology agile supports an iteration model, in which the artefact can go multiple types through the requirements, design and implantation phase, before the end product be deployed.

Finally, the spiral methodology believes that the software development should adopt elements of multiple development methodologies like waterfall or rapid prototyping. Additionally this methodology supports the idea of mitigating risk, through dividing a project into multiple smaller cycles, where at the end we can "pull the plug" if necessary; moreover each of the smaller cycles should go through the same sequence of steps, in each at the end of each cycle an analysis should be performed before advancing to the next iteration.

After we know the basic principles of the methodologies, we as the developers should opt for one of them. The chosen methodology should fit our experience and resources since the methodology should be in function of the developers and not the project. Because this is a master thesis, our resources are quite limited. In addition, only having one young developer, with close to no professional experience. Hence we can assume that some errors will be made along the line, and hopefully, these errors will be caught on time, in order to provide us with enough time to rectify them. Therefore we should aim for a flexible methodology that would allow us to go back, change what we have done wrong, and rectify our errors. Therefore a strict methodology like waterfall, is not the best for our project, since it leaves close to no room, to change decisions we took in the past. Additionally, agile when compared to spiral requires less planning, and provides us with quicker feedback. On the other hand, spiral, when compared with agile requires more documentation and is less suited for low-risk projects, as the case of a master thesis.

We then decided to follow an Agile methodology. Also, we will be using a software development technique called prototyping together with unit testing to produce our artefact.
3. Thesis Statement and Success Criteria

3.3.1 Prototyping

Prototyping is a software development technique that can be used in conjunction with agile software development to put an initial version of the completed product in front of a customer for feedback before committing to the complete development of the final product. This method is especially useful to figure out what are the strengths and weaknesses of our tool early in development and discover new requirements or success criteria.

As stated by Balzer et al.:

Given a proposed solution to a problem, prototyping is used to answer three types of question: Is this a method for achieving the solution; does the proposed implementation have acceptable performance, production cost, and reliability; and is it a good solution?[3]

3.4 Success Criteria

In the first two sections of this chapter, we decided on how we will measure risk, and how it should be aggregated. It is only natural that our first success criteria reflect our approach to how to measure risk, and how to aggregate it.

**SC1** The tool must comply with the principles specified in the report Guidelines for aggregation of cyber-risk in large institutions [45].

Next we must think about the potential users for our tool. The main target of this tool are managers in large companies. Therefore, the tool must be appropriate and comprehensible to them.

**SC2** The tool must be appropriate and comprehensible for managers.

Managers in organisations are generally busy, and therefore they want to save as much time as possible for what the companies need. In order to facilitate their job, we should aim for allowing the managers to have a good understanding of the cyber risk picture as quickly as possible. Therefore we should aim to build a tool that does as much as possible for the manager. Therefore after the tool is installed appropriately, the necessary information provided to the tool, C-RAT should be fully automatic. Automatic in a way that allows the manager to see a complete picture of their domain of responsibly.

**SC3** After the initial setup the tool should be fully automatic.
While the tool must show the entire domain to the manager, it must also be flexible. It should be possible to view only specific parts of their domain if the user so chooses. With this feature the manager, besides seeing an overall picture, should also be able to see a problem in higher detail.

**SC4** Individual managers should be able to select the relevant aspects of their domain.

Platforms like GitHub are to developers what Instagram is for photographers. With this phrase I mean to emphasise that it is like a place where developers/organisations can /should show their projects.

As a student going to the IT market shortly, it is essential to publish some projects that were developed. Publishing projects help us to showcase our skill to potential employees.

As far as companies go, publishing projects on the public domain can help in different aspects, from increasing the trust with customers, to making the software more robust.

Customers like being in control of their data. With open source, a customer knows that their data is secure and is not at the mercy of a third-party. For companies, creating open source software not only builds trust with their clients, but it also provides them with the opportunity to develop an extra feature in private and sell them as plugins or premium versions. Furthermore, if the project is exciting, it can attract more developers, which leads to more people looking at the code, which generally translate into higher quality and more robust code.

**SC5** The tool should be implemented in an open-source software.

In order to build software in open-source, we should follow a standard software convention. Following one convention improves the readability of the source code and makes the software maintenance more manageable. Any other way a project would be a big mess. For these reasons we define our sixth success criteria:

**SC6** The software should be built following the java code convention [46].

As we stated before this tool was built for organisations with considerable size, furthermore the managers on these companies generally are very busy, and the tool must be able whenever they want. The chances that two or more managers eventually desire to use the tool at the same time are pretty high. In order to achieve this, the tool must be able to support multiple users at the same time.
3. Thesis Statement and Success Criteria

**SC7** The tool must support multiple users at the same time.

This tool will handle employee data and cyber-risk related to the organisations. It goes without saying that this is confidential data; therefore it must not be exposed to the public. In order to achieve this, the data must always be secure, whether it is moving from one machine to another or just stored in memory.

**SC8** The data must be always secure.
As argued before this master thesis has a large amount of topics to cover. In this chapter, we will expand the base knowledge provided in the background, by covering the current state within cyber-risk assessment, software development, Human-machine interaction, and how to test in these different domains.

4.1 Cyber-Risk Assessment

In the background chapter we had an entire section dedicated to understanding the basics of risk management, in this section, we will have a closer look into how risk assessment should work when the area of study are cyber-risks.

A cyber-risk assessment is a specialised form of risk assessment focusing on cyber-risks. There are two main differences from a cyber-risk assessment to a general risk assessment. The first one being the vast number of threats, both malicious and non-malicious that have to be considered. The second difference being the origin of threats, given by the far-reaching potential of the cyberspace.

Definition 4.1.1. A cyber-risk is a risk caused by a cyber-threat.

As we can see in fig. 4.1, the first task of this process is the context establishment, followed by risk identification, in this case, we should divide this task into two sub-tasks: malicious cyber-risk identification and non-malicious cyber-risk identification. This separation is necessary given the different threat natures, threat sources and vulnerabilities exploited.

During the analysis phase it can be hard to estimate the likelihood of occurrence due to the human nature in the malicious cyber-risk; however, there are platforms like MITRE and OWASP that can help with this task. The next step is divided into four subtasks: risk consolidation, risk evaluation, risk aggregation and risk grouping. Finally, in the treatment phase, we should remember that the treatment for malicious and non-malicious are entirely different.

4.1.1 Identifying Risks

As you saw in fig. 4.1 after the context establishment phase, we must identify the risks that may cause harm to the assets of our party. This is extremely hard, especially when regarding cyber-security.
Starting with the malicious cyber-risks, it is recommended to start by identifying what threat sources may want to harm us. After the threat source identification, we must think on how would this threat source attack, identifying which vulnerabilities may be exploited, what incidents may result if the attack is successful, and what assets are harmed if the incident occurs. The end goal of this first step is to build a table containing threat sources, vulnerabilities, threats scenarios, incidents and harmed asset.

Next focusing on non-malicious cyber-risks, contrary to the malicious cyber-risks it is recommended the opposite approach, starting by the accidental incidents that may cause harm to our assets. Next we want to access what vulnerabilities and threats can lead to these incidents. Finally, we want to identify the source of the threats. In the end, just like in the malicious cyber threats, we are expected to end up with a table containing threat sources, vulnerabilities, attack scenarios and incidents.

While the process of identifying these risks is different for malicious or non-malicious cyber risks. In both cases, we want to end up with a table containing risks. It is important to remember that each pair (incident, damaged asset) is considered a different risk.

4.1.2 Risk Analysis

After the risk identification, we want to analyse the risks that we just identified. Hence we start the risk analysis phase.

---

1 A threat source is a person or group.
2 A threat scenario is an potential attack launched by a threat source.
In this phase, our task is to assess the likelihood of each incident, and the consequence that each incident does to each asset. To accomplish this task and determine the risk level of every risk, it is usually necessary to start by analysing the threats and vulnerabilities that lead to the incidents. Consequently, by analysing the threats and vulnerabilities, the designer will have a better understanding of what contributes to the risks, which will be useful later, when the designer want to identify treatments.

The information sources used in this phase, should be similar to the ones we used in the previous phase; however this time, we must take note of the severity of vulnerabilities, the likelihood of threats and incidents, and the consequences of the last one.

4.1.3 Risk Evaluation

Before we arrived here, we have identified risks, and assign a likelihood and a consequence to each of them. The next logical step is to evaluate them, ensuring that the correct risk level is assigned to each risk.

The traditional approach consists in creating a risk matrix. Such matrix should have the likelihood represented in one of the axis, and the consequence in the other. When doing this step, the one in charge should use the same scale used in the previous section, since this method works with both qualitative and quantitative scaling.

The next step is to assign different risk levels to different places of the matrix. The outcome of this operation is customarily represented in three colours: red, yellow and green. As most people would guess, red serves to identify the risks that represent the most harm, and that should get treatment. Opposing to the red colour, we have the green, the risks who place in this colour, do not constitute a significant threat. Finally, the yellow is for the risks that are somewhere in between the red and green colours.

Having the matrix designed and the risk analysis complete, assigning the correct risk level to each risk, is achieved by plotting each risk in the matrix. The risk level is then given by the colour where the risk was plotted.

Risk Aggregation

During our risk identification and analysis, we had different tables for malicious and non-malicious risks. However, now in the evaluation phase, all of the risks are put together in the matrix. Furthermore, while risks can have different origins, they may end in the same result.

This is one of the reasons why aggregating 2 or more risk into one may be useful. The other one is when 2 or more incidents harm more than one asset of the same party.

Risk aggregation as the name suggests consists of aggregating risks. When before we had two or more risks, in the end, we will only end up with one. Furthermore aggregated risks, typically result in having either a higher severity or a higher likelihood than the originals ones, depending on the method to aggregate the chosen risks. (See section 3.2 for further details).
4. State of the Art

4.2 Software Development

In this section, we will start by understanding why Java is still one of the most relevant programming languages of today, and after that we will look to software testing.

4.2.1 Java

While Java has been around for a long time (since 1995), as of today, it is still one of the more relevant programming languages in the world[33].

One of the main advantages Java had in the beginning, and it is relevant today, is the Java Virtual Machine (JVM). The JVM allows a program written in Java to run in any computer with the Java Runtime Environment (JRE) installed.

Other of the advantages of Java and the main why we chose Java instead of other programming languages is the Java mature ecosystem, having an enormous number of libraries, modules and frameworks. In particular the libraries concerning network, security and graphical interfaces.

Next java being around for so long, makes it one of the languages with huge online support, from tutorials, to question asked, and a massive number of developers that can help you with problems.

Additionally, Java is considered a secure language[31]. Java has automatic array bounds checking and does not allow the user to manage memory manually. Therefore some programming mistakes that often cause security holes (such as buffer overruns) are impossible to occur.

Finally, Java is in constant evolution, as of 25 September 2018 a new version of Java was launched (Java 11), and while Java 11 is the first long term support (LTS) version since Java 8 launched in 2014. Java 12 will be launched as early as March 2019, just half a year after Java 11.

4.2.2 Testing Software

Testing software is a critical part of software development, however most people today do not test software until it has already been created and is in the deployment phase of its life cycle[24]. This leads to errors only being found out at the end, which normally translates to more time locating them, and then more time to fix them. Therefore this is an ineffective strategy.

Besides finding errors, software testing is also important to verify if the artefact meets the requirements that guided its design and development, if the inputs are working as intended, if the response time is acceptable, if it is usable, if it is secure, between other requirements developers want to test.

4.2.3 Testing during development

As we just stated, only testing after finishing the product is costly, takes more time, and it is not efficient. Therefore we needed to test software while it was in the development process.
4.2. Software Development

Testing during development is very useful in between other things to verify that the code being produced is in compliance with the standards requirements. Furthermore it should help developers to make sure that their functions, classes, methods APIs and libraries are working as intended.

This process should follow both static and dynamic analysis, for example, using unit testing and then debugging, is a great way to test the produced code, and helps to track down, any incorrect behaviour that might be happening.

4.2.4 Unit Testing

Unit testing is a software testing method that consist in testing individual parts (units) of the software.

The primary objective of this test is to validate the code that is being developed. By doing this the developers can validate the security of the individual components, and look for errors in each components before merging it with the others. Furthermore, with the code being tested on a constant basis, it helps the developer to build confidence in their code, and enable him to comfortable refactor his code\(^\text{3}\) which in its turn should lead to more improved code readability and reduce its complexity.

4.2.5 Software Documentation

Software Documentation is written text or illustrations that explains the software for the reader. In addition, the software documentation is typically sent together with the software or is embedded in the source code.

There are 5 main types of documents that document the software\(^\text{11}\):

- Requirements;
- Architecture design;
- Technical;
- End User;
- Marketing;

The first type of documentation in this list is requirements; this kind of documentation focuses on the behaviour of the tool. Its main job consists of describing what a particular software does or will do.

Next, the architecture design documentation is an overview of the produced artefact, in a way that it should lay out the general requirements that motivate the existence of a particular behaviour. Furthermore, a proper document of this type, should not be overly concerned with the details, but instead, its focus should be on the reasoning and explanations.

---

\(^{3}\) Code refactoring is the process of restructuring existing computer code, without changing its external behaviour.
4. State of the Art

Following we have the technical documentation. Contrary to the previous items, this item should present information more close to what is implemented or what will be implemented; furthermore, it is often viewed as a more user-friendly approach to the code targeted to both proficient users of the software, or to developers who will maintain it after its release. However, while most IT professionals, consider that writing this type of document, is necessary, it is also regarded as not as enjoyable or not as rewarding as writing code[36].

The fourth item in our list is targeted to end users. In this type of documentation, the writer usually is more concerned to describe how the program should be used. There are typically three sub-types of writing a document for the end user: tutorial, thematic and list of references.

The last item in our list is often necessary for a corporate environment, this type of document has the goal to encourage the average user to spend more time learning the product or to inform the user how the product can help him.

4.3 Cybersecurity

In this section we start by defining and explaining notions of cybersecurity. What are the threats we are up against and how are we suppose to defend ourselves from them.

4.3.1 Malicious Threats

“One who knows the enemy and knows himself will not be endangered in a hundred engagements. One who does not know the enemy but knows himself will sometimes be victorious. Sometimes meet with defeat. One who knows neither the enemy nor himself will invariably be defeated in every engagement.”[50]

As Sun Tzu pointed out 25 centuries ago, one has better chances to win a battle if he knows his enemy. Cybersecurity is also a battle between the defenders and the attackers; therefore in order to defend ourselves properly against attacks, we must know who is performing the attacks. In this subsection, we will go through the main groups and attacks, their skills and their motivation.

Black hat hacker

**Motivation:** This group is motivated by personal gain. Examples of this can be an user reducing their hospital bill or blackmail a company.

**Skill Level:** The skill level in this group varies considerably; however, the best ones are world-leading experts in cybersecurity. Furthermore, if they are part of a larger criminal organisation, they may command a significant number of resources.

Cyber-terrorist
4.3. Cybersecurity

**Motivation:** This group goal is to cause a disrupt in society through cyber-attacks. This group normally has a strong political ideology or religious motives.

**Skill Level:** This group commands typically significant resources, furthermore, in some cases they may be supported by nation states. This group can perform long-term planning, preparations and carry out the attacks.

**Insider**

**Motivation:** An insider is a disloyal employee or a consultant for the organisations. Furthermore, the attacker is typically motivated by personal gain or desire to harm the employee.

**Skill Level:** Due to the attacker position in the company he usually possesses access to most systems, besides, he has detailed information and knowledge about the system architecture, functionality and security features.

**Script kiddie**

**Motivation:** This attacker aim is to prove his ability to cause harm on others, with the believe that this action will bring him some sort of status. Normally the attacker quits after facing initial adversities.

**Skill Level:** This attacker is relatively unskilled and has limited monetary and computational resources. Therefore this attacker is unable to perform complicated attacks, and most of the times resorts to tools developed by others.

4.3.2 Testing within Security

**Input Validation**

One of the first rules a developer learns is to always validate input from the users. This action is not only necessary to make sure that the user introduces the type of data the developer expects, but it is also crucial to prevent cyber attacks.

Two simple examples of why input validation is necessary are:

- A calculator receives non-numeric characters as input. If there is no input validation, the calculator would (depending on implementation and the chosen language) crash or give an incorrect answer.

- If a website has a connection to a database, and does not validate all the input from the users, especially the ones that are used to making requests to the database, it is likely to suffer an SQL injection in the near future.

The two examples above are good illustrations of why input validation is a must in any program and should be extensively tested during development. Additionally validating input can also undercover gaps in security controls, such as lack of basic authentication, authorisation, or encryption controls.
4. State of the Art

Threat Modelling

Threat modelling is a process by which potential threats can be identified. This process helps system designers to think about security threats that they may face in the future.

Threat modelling is usually seen during risk assessment for applications\cite{24}. Hence this process forces the designer to have a complete understanding of the project allowing the designer to not only find the potential vulnerabilities but also helping his designer focus their resources and attention on the parts of the system that most require it.

Threat Diagrams

One of the most complete threat modelling processes is threat diagrams. The CORAS methodology\cite{18} recommends these diagrams, and as the name indicates it allows the designer to represent threats to subject of analysis, since it has elements representing assets, incidents, threat scenarios, vulnerabilities and sources of attack.

Threat Diagrams serve not only the purpose of identifying risk, as they should also be used to do risk estimation, risk evaluation and risk treatments. (Further explanation of these processes in section 1 of this chapter).

After the first set of iterations in the threat diagrams, the subject should be analysed again after the changes, and compared with the previous diagram in a before-after threat diagram. This set of operations allows the designer to see the changes provided by the chosen treatments.

4.4 Human-Computer Interaction

In the beginning there were Humans. Much later, in 1940 computers were invented. Then in the 1980s came interaction. In this subsection we will start by exploring designing concepts. Then we will explain what is usability and usability testing.

Humans are almost always in contact with computers. The interface between humans and computers is of the most importance in order to facilitate this interaction. Desktop application, internet browsers and mobile applications are just some examples that use graphical user interfaces (GUI) that people use daily and that are always making advances in order to improve the interaction.

The Association for Computer Machinery (ACM) defines human-computer interaction as:

A discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.\cite{15}

As we can read from the quote, the primary element of human-computer interaction (HCI) is the humans, and not the design. The design and other components goals is to help the human. We can then conclude the HCI aims to
improve the users life, in other words, HCI aims to makes user more satisfied. We can now conclude that HCI primary goal is to improve user satisfaction.

On the other hand, poorly designed HCIs can lead to unexpected problems. One of the more classic examples is the Three Mile Island accident [51]. The Three Island accident, was a partial meltdown of a reactor in Pennsylvania, 1979, in which the investigators concluded that the interface was at least partly responsible for the disaster.

4.4.1 Design Concepts

Design concepts at a very high level are open to interpretation; however they are the starting point.

According to Don Norman [30], there are six main principles on design: affordance, mapping, feedback, visibility, consistency, constraints.

The term affordance is concerned with the perceived and actual properties of the artefact, primarily the fundamental proprieties that determine how the artefact could be used. Example: A chair affords support, and, therefore affords sitting.

Next mapping refers to the design of controls such that they reflect the way they behave or the results achieved by using them. Example: In a television remote what each button does.

In third comes feedback, and it is concerned with sending information back to the user about what action has actually been done and what result was achieved. An example of this would be turning the lights of the car on, while the driver can not see the outside of the car to verify if they turned on or not, the user panel, shows it to you.

The visibility principle defends that the more visible an element is, the more likely it is for the users to know about it. In contrast, when an element is out of sight, it is harder to know about it and what it does.

In fifth comes the consistency principle, this principle states that we should use similar elements for similar tasks. One good example of this principle is the icons for mute, upper volume and lower volume. Following this principle does more than showing coherence on the creator’s side, it enables the users to transfer knowledge to a new context.

Last but not least come the constraints principle. Constraints are the limits to an interaction or an interface, restricting the range of action from the user. An example of an obvious physical constraint is, for example, the size of the screen on a smartphone.

In short, the visual structure indicates how an artefact should be used and while complex artefacts may need some explanation, simple ones should not. When simple artefacts need pictures, labels or instructions the design has failed.

4.4.2 Usability Testing

In this subsection we will start by exploring the term usability, then we will explain the main characteristics of a usability test.
4. State of the Art

As stated by Jared Spool:

"A design is intuitive when the gap between a user’s actual knowledge about how to use a product and the knowledge required to use it efficiently is small or non-existence"

Usability

Just like functionality, usability is an attribute of every product. While the first refers to what a product can do, usability refers to how people work with the product. This can be extended to testing. While testing functionality means making sure the product works according to the specifications, testing usability means making sure that people can find and work with the functions to meet their needs.

According to Joseph S. Dumas[10], we define usability as:

**Definition 4.4.1.** Usability means that people who use the product can do so quickly and easily to accomplish their own tasks.

With this definition we mean to emphasise 4 points:

- Usability must focus on users;
- People use products to be productive;
- Users are busy people trying to accomplish tasks;
- The users are the ones that decide if a product is easy to use.

Usability Testing

There are five main goals characteristic that define a usability test[10]:

- The primary goal of a usability test is to improve the usability of the product being tested. In addition for each test the developer should have a specific goal and concerns to take in account when planning the test;
- The participants represent real users;
- The participants do real tasks;
- The developer records information about the users’ results and opinions;
- The developer must analyse the data, and if needed fix the problems.

The primary goal of a usability test is to improve the usability of the product being tested. While it is a reasonably obvious characteristic, it is meant to distinguish the usability testing from other tests, in a way that reinforces that the primary goal of this test is to find a problem within the interaction between the user and the artefact, and not the existence of some unexpected phenomenon or if the product works according to its specifications.
The participants in the usability test must represent the group of people for which the artefact is being developed. A test that uses programmers when the artefact is intended for lawyers is not a usability test. Besides Nielsen defended that 5 is the optimal number of participants for a usability test.\[27\] \[28\].

The tasks realised by the users must be the task that will be performed when using the product. This means that the developer must understand the end user job and the tasks for which the artefact is relevant.

In a usability test, there are usually several people are coming, generally one at a time. The designer besides keep track of the user performance, they should ask feedback from the participants not only at the end of the test but also during the tasks.

While collecting data is necessary, it is not sufficient. For a usability study, the developer needs to analyse the data, and then consider the qualitative and quantitative data from the participants together with his observations and users comments. After, the data should be analysed and documented. Any problems found should be reported and, if possible quickly fixed.

In short usability testing is an evaluation technique that consists of showing the artefact to real users in order to allow them to test it. With this technique, the developers get direct input on how real users interact with the artefact, allowing them to find problems and fix them more quickly.
CHAPTER 5

Research Method

In chapter 4, we described the main topic of this thesis and the goals we want to achieve with this thesis and C-RAT. Based on this, we defined a set of success criteria that this thesis intends to fulfil in order to accomplish our objective. In this section, we discuss the steps required in order to conduct our research, namely the research method to be applied in the proposed thesis.

5.1 Introduction

The word research comes from the French recercher, or recercher in old French and means 'seek out, search closely,' from re intensive prefix + cercher 'to seek for'.

Research has been defined in many different ways. For example, Merriam Webster [23] defines it as:

Investigation or experimentation aimed at the discovery and interpretation of facts, revision of accepted theories or laws in the light of new facts, or practical application of such new or revised theories or laws.

In short, the researcher aims for either adding knowledge or modifying the already existing knowledge. When conducting research, the researcher must first ask a question; then, in order to solve it, he will formulate a hypothesis. Through observations and/or investigation the researcher will prove the hypothesis is true or refute it. This process is called hypothesis testing or evaluation. This approach is known as Scientific Method or Classical Research Method honouring the philosopher and scientist from the classic era; the term classic not only refers to the Greek and Roman culture but also means standard or traditional.

5.2 Technology Research

Technology Research is a research type that aims to the creation or improvements of artefacts. An artefact is an object manufactured by human beings[40].
5. Research Method

When conducting technology research, the research starts by carrying a problem analysis. The *problem analysis* consists of collecting requirements/needs for the artefact. In this phase, we answer the question: *What is /are the need(s) and how can we solve/improve it/ them?*

When the problem analysis is complete, the researcher passes for the second phase, the *innovation* stage. In this stage, the researcher tries to build an artefact that satisfies the requirements collected in the previous stage.

The third and possibly final phase of the technology research is the *evaluation*. After the artefact is complete, the researcher formulates predictions (based on the requirements) about the artefact and checks whether these predictions come true. If the researcher succeeds, he can argue that the artefact fulfils the needs and therefore is free to work on his next project. On the other hand, if the evaluation results are not expected, he might have the need to adjust the requirements and go back to the problem analysis. The overall hypothesis of the technology researcher is: *The artefact satisfies the need(s).*

### 5.3 Evaluation Strategies

As stated earlier the evaluation phase goal is to find out if the predictions are true. However, there are different ways to do this, regardless of the object of study being it a new artefact or a part of the real world.

According to McGrath [22], there are three factors we want to maximise:

- **Generality:** The validity of results across populations;
- **Precision:** The precision of measurement of the acquired results, and control of external variables that are not part of the study.
- **Realism:** To what degree the evaluation reflects realism (if it was performed in a realistic context).
5.3. Evaluation Strategies

In order to achieve the best possible result, we should try to maximise these three factors; however, McGrath tells us this is not possible. Furthermore, he states that when increasing one of the factors we are simultaneously reducing the other two. The nature of this dilemma is more evident in fig. 5.2.

![Evaluation Strategies Diagram](image)

Figure 5.2: Evaluation strategies

According to McGrath, the most common strategies are:

- **Field studies** are direct observations of the "natural" ongoing system while disturbing those systems as little as possible. Field studies have a high degree of realism, but lack precision and usually are hard to replicate;

- **Laboratory experiment** attempts to recreate systems giving the researcher a significant degree of control and the possibility to isolate the variables to be examined; Laboratory experiments have a high degree of precision but lack realism and generality. They usually are easy to replicate;

- **Field experiments** are observations carried out in the natural environment like the field of studies. However, certain factors are deliberately manipulated.

- **Experimental simulation** are laboratory studies in which the researcher tries to replicate a process that occurs in the real world.

- **Surveys** are efforts to gain information from a broad group of selected informants. The information is typically gathered through a small set of questions either as questioners or quick interviews. Surveys have a high degree of generality but lack precision and realism.

- **Quality interview** like a survey is a method to gain information, however, this strategy focuses on a few selected individuals. The answers are more precise than those of a survey, but cannot be generalised to the same degree.
5. Research Method

Non-empirical evidence is a theoretical approach where we use logic as the base for our argumentation. Like the survey, this method has a high degree of generality but lacks in precision and realism.

Computer simulation is an operating on a model of a given system. Usually has a higher degree of realism than generality given the fact that it is system specific.

These eight strategies are further divided into four groups:

1. The evaluation is performed in a natural environment.
2. The evaluation is performed in an artificial environment.
3. The evaluation is independent of the environment.
4. The evaluation is independent of empirical measurements.

5.4 Selection of appropriate evaluations strategies

In order to choose the best evaluation criteria for our thesis and application (C-RAT), we will look back to the success criteria defined in the previous chapter.

1. The tool must comply with the principles specified in the report Guidelines for aggregation of cyber-risk in large institutions.
2. The tool must be appropriate and comprehensible for the managers.
3. After the initial tree setup the tool should be fully automatic.
4. Individual managers should be able to select the relevant aspects of their domain.
5. The tool should be implemented in an open-source software.
6. The software should be build following the java code convention.
7. The tool must support multiple users at the same time.
8. The data must be always secure.

In order to fulfil all of these success criteria, we will need to evaluate the artefact through multiple points of view. First success criteria numbers 1, 3, 4 and 6 are concerned with the development process or behaviour of the artefact and should be looked through a software development perspective. Following that, success criteria number 2 is concerned with the usability of the tool for the users. Next success criteria number 8 is concerned with the security aspect of the tool. Last success criteria number 5 is not concerned with the artefact.
5.4. Selection of appropriate evaluations strategies

itself, but rather with its publication once it is finished, and therefore it does not require evaluation.

Additionally, as we pointed out in chapter 3, we will develop a prototype for C-RAT, and this is will be the artefact to be evaluated.

To evaluate our prototype from a software development point-of-view, we will create an approximation of technical documentation, in which we will describe not only the vital functions of C-RAT but also, justify why the options we took are required.

Next, we must consider the security aspect of C-RAT. As we have previously mentioned C-RAT will be an application to help managers manage their risks. In addition, the information regarding the risks is sensitive information, since organisations do not want their weaknesses exposed. Therefore we must think about security while we are building the tool. Since we are building a tool for cyber-risk management, it is fitting that we also subject C-RAT to a risk assessment.

Finally, we must also think on the user that will use the tool, and as referred in the second success criteria, the tool should be appropriate for them. The most reliable way to evaluate the usability of a tool is through experimental testing. Therefore we will a usability testing experiment.

5.4.1 Artefact Description

To evaluate the software from a software development point-of-view, we will describe our artefact. However this description will not be a typically technical or architectural documentation, but something in the between.

Instead, as in a standard technical documentation, we will go through all the major processes that make C-RAT; however, we will not go as detailed as in those documents. Instead, we will provide an explanation on how the components work; some times we will support these explanations with sequence diagrams\[25\].

In addition, we intend to organise the document in smaller sections, that would emulate the different components within C-RAT. Moreover, to each of these smaller sections we pretend to give an analysis on why we decided to implement those components in that way and not in another.

5.4.2 Security Risk Assessment

Regarding the security aspect, we must remember that we are dealing with confidential data. Therefore we want that data to be secure. Moreover, while there is no silver bullet when we are talking about security, there are several decisions we can make in order to "upgrade" the security of C-RAT. For this reason, before the implementation, we design our security system. We will make use of threat diagrams\[20\], in order to find potential weaknesses in our tool.

However, contrary to what is typically done, we will divide this evaluation per two chapters. First, in the research-based design chapter, where we design the application, we will look for weaknesses before implementing the tool, and try
5. Research Method

to find solutions to mitigate the potential risk. Then in the evaluation chapter, we will perform the same evaluation, but this time with the defence mechanism implemented. Finally, we will be able to easily compare the improvement or lack of it, using before-after threat diagrams.

5.4.3 Usability Testing

As we explained in the state of the art chapter, usability testing is an evaluation technique that consists of showing an artefact to real users in order to allow them to test it. In this way, developers have direct feedback from the users, which will lead to faster changes when necessary.

After C-RAT was implemented and all functions were tested, we wanted to make sure that the tool was intuitive. In order to accomplish this goal, we decided that a usability test would be the best solution, since its main focus is the usability of the tool.

When planning this test, we decided that we wanted to test how intuitive the interface of the application is. For this reason, we decided that the test would mainly consist of completing five tasks, that symbolise the core of what users will do with the tool:

- Login;
- Adding risk;
- Add values to the risks;
- Aggregate your risks either by asset or scenario;
- Logout;

In addition, we only wanted participants within the informatics area since this application is being built to be used by professionals with a background in informatics or cybersecurity.

Additionally, during the duration of the test, the developer will not provide any further assistance, since the goal is to test the tool intuitiveness. Moreover, the test should also serve as an indication of the tool’s viability without user documentation.

The test consisted of having five volunteers within the area of informatics (bachelors and masters), coming one at the time to test the application and fill a questionnaire. When a volunteer arrived, we put the questionnaire (see Appendix A), and two instances of C-RAT open in front of him/her. Then we asked the volunteer first to read the questionnaire and then complete the tasks at the same time as evaluating them on the questionnaire. Finally, when the volunteer finishes the tasks and evaluates them, he/she was asked to give an overall feedback on the tool. During this phase, the developer did not help them; just watch them struggle or succeed in their tasks.
CHAPTER 6

Research Based Design

In this chapter, we start by remembering our success criteria, based on that we will plan and design our tool, achieving a set of requirements to be used in its implementation. In section 1, we analysed our requirements and based on them we defined the design and explained the overall behaviour of the tool.

Next, in sections 2, 3 and 4 of the chapter and we look closely to aspects of Network, Security, and Graphic Interface respectively, justifying the more critical decisions in these domains and providing some explanations.

After in section 5, we take a closer look at the risk segregation feature.

Succeeding in section 6, we look closer to some structures and the database of C-RAT

Finally, in the last section, we do a summary of everything we discussed in the previous sections.

6.1 Artifact Design

In chapter 3 (Thesis Statement and Success Criteria) we defined eight success criteria that must be met to evaluate this project as a success.

SC1 The tool must comply with the principles specified in the report Guidelines for Aggregation of Cyber-Risk in large institutions.

SC2 The tool must be appropriate and comprehensible for the managers.

SC3 After the initial tree setup the tool should be fully automatic.

SC4 Individual managers should be able to select the relevant aspects of their domain.

SC5 The tool should be implemented in an open-source software.

SC6 The software should be built following the java code convention.

SC7 The tool must support multiple users at the same time.

SC8 The data must be always secure.
With the success criteria in mind, we start to think on how to implement a tool that can satisfy all of these requirements. Furthermore, we needed to ask ourselves what is the best way to insert the data, that concerns the managers.

In order to ask answer this question, we had to take a step back and remember that this tool’s role is to help the managers while they are conducting a cyber-risk assessment. With this in mind we decided that the manager would be the responsible for identifying the risk; however, it would be an employee working under him, that is working more closely with the risk, the responsibility to assign the values to the risk.

**Definition 6.1.1.** A **Risk Owner** is an underling of a manager responsible for introducing and updating the values of one or more risks.

As we can see in the diagram, our proposal starts with the manager inserting the risks in our tool, in addition, him/ her must attach to every risk a risk owner responsible for giving both the values for the cost of one occurrence and number of occurrences per year.

Now that we decided on how the tool should behave regarding the data insertion, we must decide on an environment and an architectural model for our application, regarding the network and security.
6.2 Network Design

As we explained in the previous section, C-RAT will be composed of three main components: a server, managers, and risk owners. Besides, we must provide ways to:

1. Allow managers to create risks;
2. Allow managers assign risk owners to the risks they created;
3. Risk owners can only see the risks assigned to them;
4. Risk owners can assign likelihood and consequences to the given risks;
5. Managers must see the values updated on their risks by the risk owners;
6. Managers must be able to aggregate the risks of their domain;
7. Managers, with underling managers should be able to see both their risks, and their underlings risks;

Furthermore, we want to validate and authenticate every action, before sending it to the other users.

If we look back to our Introduction chapter, we stated that the primary target of this tool is big organisations\(^1\), furthermore, more often than not, these organisations employees are often structured in a tree form.

Furthermore, if we could have a user tree inside our tool, that represented the company structure, and if to each node, where a node represents a user, we added a list of risks that concern that user, it would immensely facilitate to know which risks we have to show to each user. Hence if the user were a risk owner or a low-level manager\(^1\), C-RAT just needs to check the risks within the node before sending them to the user; and in the case of high-level managers\(^2\), check the risks in his node, in the node sub-tree, and then send them to the user. Furthermore, we could easily distinguish between managers and risk owners, since risk owners do not have underlings, and managers must have underlings to fulfil their risks. Therefore if a node is a leaf, it must be a risk owners, on the other hand, if the node is not a leaf, then the user is a manager.

Now that we discovered that having a tree with users and risks is an excellent idea, we must decide on how to store the information. One of the more simple and free of charge solutions is a MySQL Community Server\(^3\), because not only our developer already had experience working with it, but also, if a company wants it can be upgraded to the enterprise version.

Moreover, if we have some element like a server to where the users communicate, and the only communication with the database is through this server, we only have to store the database password in one place. Furthermore having a

\(^1\)Managers without managers working for him.
\(^2\)Managers with managers as underlings.
6. Research Based Design

Centralised server allows us to validate and authenticate every request to the database. We can expand this feature and easily create a log file with all the request from all users. This extension allows us to achieve non-repudiation.

After we decided that we wanted a central server in order, we must choose how the managers and risk owners should communicate with the server. One of the most known architecture patterns within software engineering is the client-server architecture. This architecture, besides the advantages of having a central server, is very cost effective, with this we mean that the staff required to maintain the network and maintain access to network resources is very low.

If we look back to our success criteria, success criteria 7, states that our tool must support multiple users at the same time. In our current solution, with a simple client-server architecture, if two users make a request to the server at the same time, user two request will only get taken care after we finish the first user request. If by any reason this first request takes a long time, user two will be waiting for an answer for a long time. This is not acceptable in a real-time server! One way to fix this problem is through a multi-threaded server.

6.2.1 Multi-Threaded Server

The main advantage of a multi-threaded server is to enable multiple simultaneous calls from the same process. This in between other things allows the program to continue running even if a part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the users. While on a non-multi-threaded server if a user asked a lengthy request, all other users requests would be waiting for this request to finish before their requests start being handled. This would be far from ideal in a real-time multi-user server like the one we want to provide. On the other hand, a multi-threaded server can handle multiple requests from multiple users at the same time. Making it a far better server for our application.

![Figure 6.2: Multiple Service Threads Dispatched in One Server Process](image-url)
6.3 Security Design

As we stated before, this application is being built to help managers with the company cyber risk assessment. During this process, the managers are required to introduce the vulnerabilities they found, in the C-RAT software.

If some malicious person or group could get hold of this data, they could cause severe damages to the party. For this reason, we had to think about security when building the software. Hence we wanted to make the application and the data it holds as secure as we could with practically no resources. This meant that all our defence mechanisms would be thought and built by us or use free available software.

If we look back to the definition of security, there are four concepts we must meet:

**Confidentiality** Only authorised actors have access to information;

**Integrity** Only authorised actors can change, create or delete information;

**Availability** Authorised actors have access to information they need when they need it;

**Accountability** It is possible to audit the sequence of events in the system;

These four concepts can be translated into five goals, that our application must satisfy:

- Only authorised users can see the information;
  - Users can only see the information regarding them;
- The users can only change and create information concerning their domain;
- The server must always be ready to answer the users requests;
- All changes to the data must be traceable;

The more obvious solution to target the first requirement is a login page. A login page, would allow our server to identify the user in the user tree, which leads to the server knowing, what privileges that user has[3], and as already referred, by identifying the user in the user tree, we know the node where the user is represented, and that holds all the information that he should have access to, which allows us to quickly and effectively only send this information to him, after authenticated. Therefore a login page would accomplish the firsts three items of our list.

The central role of a login page is to identify the user. Identifying the user allows the server to verify that the user has the authorisation to see the data, and what kind of privileges does the user has. Furthermore, in our case, it also allows the server to identify the user in the user tree, which leads to the user

[3]User privileges are based on the user position within the user tree
only receiving information concerning him or his employees. Therefore a login page process would accomplish the first three items of our list.

In addition, if with base on user’s position in the user tree, users receive some amount of permissions, allowing only someone of them to create, change or delete information, and restrict the changes only to their domain.

The fourth one (the server must always be ready to answer to users requests), does not need any particular change at this time. However we will look more closely into it, once we start to analyse malicious attack.

Finally, the last one can be achieved through a traditional logs file. Furthermore, if we include the username or userID together with the entry, we can achieve non-repudiation.

Non-repudiation is an extension of accountability that besides recording the chain of events in the system, also registers who did it. This makes that if someone working for the company intentionally wants to cause damage it is a lot easier to find the culprit. For this reason, this concept has a huge legal implication once it makes very hard of someone to deny that they realised the operation(s).

6.3.1 Security Risk Assessment, Before

While in a perfect world this would be enough. In reality, this is not the case once there are human threats that want to breach our network. Hence we need to take them into account when we design our application.

Asset Identification

The first step to protection is to know what we want to protect. For this reason, we identify what assets does C-RAT have.

First, we put ourselves in the place of a company using C-RAT, and asked ourselves, "What is important to us?". We rapidly reached the conclusion that we do not want other entities to know what risks we are most vulnerable to. In other words, we value the privacy of the information within C-RAT. The information of C-RAT is either stored in the database, or moving through the network, or in use. In the latter case, we are forced to consider the integrity of the machine where the client is being used.

Furthermore, if any of these primary targets are attacked, not only the confidentiality of the information is at stake but also its integrity.

Finally, we must also consider, the availability of the server, once it is required in order to have C-RAT working.

This is more easily understandable through the diagram [6.3] as you can see, we have 4 primary assets, the client machine, data travelling between the client and the server, the database, and the server availability. The first three lead to loss of integrity or confidentiality.
6.3. Security Design

Vulnerabilities

Now that we know what can be attacked, contrary to suggested, we started brainstorming about what vulnerabilities our application has at a design level. We rapidly conclude that although we have a multi-threaded server, any modern DDoS attack would still crash our server.

Next, we focus on the client machine, from observing our environment, we notice that a vast number of professionals work with the office laptop at home. This behaviour is not recommended, once the defence mechanisms like firewalls are usually weaker at home than at the office. Which leads to more exposure to cyber-threats, and if the laptop contracts a malicious software that spread through the network, it can also compromise the company network. Furthermore, this practice would not only make the machine have a higher exposure to malicious software but also increase the distance between the client and the server. Having a bigger distance between the client and the server, usually means that the information goes through more routers, if it goes through more router there is a higher chance of at least one of them be bugged or contain spyware, this would mean that there is a more significant threat to the confidentiality and integrity of the information.

Our last primary asset is the database, that can only be accessed through the server, once that is the only machine with the database password. This reinforces that the option of having a server-client architecture was correct.

In order to have a better understanding of risks we found we opt for representing the risks on a threat diagram. Represent the risks on a threat diagram will not only allow us to have a better visualisation of the risks we face but can also be used to later compare with an after version.

Analysis & Treatment

As you can note, in the fig[6.10] our diagram, is filled not only with the risks and elements we previous indicated, but the threat scenarios and incidents, also carry a likelihood, and the connection between the incidents and the assets, is also filled with an average consequence for that asset due to that incident.
Assigning likelihoods and consequences to risks, and its elements will help us to have a better grasp of the risk level each incident carry. However, have a good prediction on the risk level is hard, especially when we are not specialists in cybersecurity. To mitigate our lack of experience, we used the CAPEC [4] table, provided by MITRE, to assign both likelihoods and consequences. The reader can quickly note that these scales (see fig. 6.4), are not the same that we will be using in the C-RAT software. The reason for this is that the CAPEC scales are from a theoretical standpoint for organisations of all sizes, which is not what we want to provide with C-RAT.

<table>
<thead>
<tr>
<th>CAPEC Likelihood Scale</th>
<th>CAPEC Typical_Severity Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>Term</td>
</tr>
<tr>
<td>Very Low</td>
<td>Very Low</td>
</tr>
<tr>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Very High</td>
<td>Very High</td>
</tr>
<tr>
<td>Example</td>
<td>C-582: Route Disabling</td>
</tr>
<tr>
<td></td>
<td>C-169 Footprinting</td>
</tr>
<tr>
<td></td>
<td>C-192: Protocol Analysis</td>
</tr>
<tr>
<td></td>
<td>C-151: Identity Spoofing</td>
</tr>
<tr>
<td></td>
<td>C-554: Functionality Bypass</td>
</tr>
<tr>
<td></td>
<td>C-100: Buffer Overflow</td>
</tr>
</tbody>
</table>

Figure 6.4: CAPEC scales [4]

<table>
<thead>
<tr>
<th>Risk Incident</th>
<th>Asset</th>
<th>Likelihood</th>
<th>Consequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Unauthorized people have access to sensitive information</td>
<td>Data in Transit</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>2 Unauthorized actor has access to credentials</td>
<td>Data in Transit</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>3 Unauthorized actor has access to credentials</td>
<td>Client Machine</td>
<td>High</td>
<td>Very High</td>
</tr>
<tr>
<td>4 Data in the Client is read by unauthorized actors</td>
<td>Client Machine</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>5 Server shutdown</td>
<td>Availability of C-RAT</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>6 Server cannot response to user requests</td>
<td>Availability of C-RAT</td>
<td>High</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Figure 6.5: Risk Identification List

<table>
<thead>
<tr>
<th>L/C</th>
<th>Very Low</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td>R6</td>
<td>R1</td>
<td></td>
<td>R2, R3</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>R5</td>
<td>R4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.6: Risk Matrix

As the reader can note in our risk identification list fig. 6.5, all of our risks,
have medium or higher likelihood and consequences. This combinations of likelihood and consequence lead to unacceptable risks levels, which can be easily noticed in our risk matrix, fig. 6.6, once all our risk stand in either red or orange zones. Hence we were forced to find suitable treatments for our vulnerabilities, and incidents. Intending to mitigate these risks, we re-started brainstorming, starting to focus on treatments for our vulnerabilities.

Our first good idea came from the realisation that there was no real need for the C-RAT be available to people outside the company network. Once it is only to be used by employees of the company, even in the off chance of someone be working on C-RAT from home, the user should use a VPN to the company network, and from that point, the user is able to use C-RAT. Therefore we could only have server locally available. This change together with a company policy of not bring the work laptops to home. Would significantly decrease the chances of someone breaching our network, once it is required that they go physically to the organisation.

With the environmental change to C-RAT usage, we reduced the likelihood of the found vulnerabilities significantly. Hence the attacker must first infiltrate in the company network, which should not be easy if the company has good security policies. However, in this unlikely scenario, where someone successfully infiltrates in the company network, all the communication would be exposed.

To tackle this last problem, we started by recognising that an attacker capable of infiltrate in the company network, must be quite skilful, therefore we needed a countermeasure that could not be broken, and that was able to maintain the data in transit hidden or unreadable to this attacker. The only solution we were able to come out with was to encrypt the communication between the server and the clients. In this way even if an attacker is able to infiltrate in the company network, he should not be able to read the data in transport, and his only solution is to either attack the server or the clients, which now should be significantly harder.

6.4 Graphical User Interface

Finally, we have to decide on how the client will look like for the users. When designing the client we start by remembering what the client must let the managers and risk owners do:

- Allow managers and risk owners to see the risks of their domain;
- Allow managers to introduce risks;
- Allow managers and risk owners to select risk, they want to update;
- Allow for an easy automatic risk aggregation in the managers’ page;

Additionally, besides allowing all of this task, we also aim for an intuitive design that allows the managers and risk owners, a clear interface.

\textsuperscript{4}Unless some idiot decides to breach company policies.
6. Research Based Design

The first decision we took, was that the first page a user sees should be the login page, once it allows the user to be authenticated and identified in the server, which is required in order to accomplish the next phase. After the identification was successful, the server will send only the relevant information to the client. The information sent is different for each of our user types (risk owner, low-level managers and high-level manager).

In the first group, the information sent is the risk owner user details and the risks where the user is the risk owner.

Next, the low-level managers receive the user details, a list with the risk owners working for him, and a list with risks where he is the manager. It is important to note that if one of his risk owner, his responsibility for a risk of a high-level manager, a low-level manager will not have access to it.

In our last group, a high-level manager, as the previous group, receives his user details, a list with the managers and risk owners in his subtree, the manager risk and the risks of the managers in his subtree. It is important to note that the risk owner risks are not pulled, but only the risks where the managers belong in this subtree. This behaviour is necessary to ensure that if an even higher manager, created a risk to one of the risk owners in this subtree, the high-level manager of this subtree will not be able to see it.

At this point, he has decided which information we want to show, but not its layout. The more significant piece of information is the risks, and therefore they should occupy a big block in the centre. Next, we have the user details, the details of underlings in the case of managers, and we still have to decide on how to input the data. The user details can be a more or less big section depending on how much information the company wants to show. For this reason, we will put it on the top, as it is the more traditional approach. Next, the input, both the managers and risk owners will need to input values. This operation should be done in a horizontal bar, due to the western way of writing horizontally. Once the top is already taken the bar should go to the bottom. Finally, regarding the underlings' list, it should show who is working under who, and for that reason, we aim to construct a tree menu to represent them, like the left menu with folders on windows operative system.

6.5 Risk Segregation

As we discussed before in this thesis, the information to be held within C-RAT is sensitive information, and we want it to be secure, one way to help with this goal is to only show the necessary information to those who need to see it. One of the steps we took to ensure this in the company is to only show the risks to the managers with access to it. Furthermore, our fourth success criteria state 'Individual managers should be able to select the relevant aspects of their domain'.

While these two tasks might seem opposed at first glance, somehow they must be combined. The solution we found goes through limiting the range of access for the managers. Instead of giving managers access to every other user, we restrict his access to the users in the manager sub-tree. On the other hand, complying with the fourth success criteria, we allow managers to segregate their
6.5. Risk Segregation

risk, and the risks for the users in his sub-tree, through his multiple users, as he pleases. In order to make this feature feasible we created two rules:

1. Risks only travel upwards;
2. Risk can only be filtered downwards;

The first rule job is to ensure that the risks, can only be seen by its owner and the owner superiors, ensuring that users with fewer privileges are not able to see or modify them.

The second rule job is to allow our forth success criteria while not-disregarding the risk security. Our solution is that instead of giving the user all that we possibly would want, we will only let him have access to the risks of his domain. However, it is still possible for the manager to select only relevant aspect, as long as they are inside the manager sub-tree.

As we have already explained before, C-RAT works with a system of managers and risk owners. This means that to each risk we have assigned both a manager and a risk-owner.

In order to define our solution we will need:

- $M_X$ as the list of risks with manager $X$;
- $R_X$ as the list of risks with risk owner $X$;
- $X$ as the list of risks user $X$ can see;
- $X_Y$ as the list of risks manager $X$ can see, when we only want to see the risks in $Y$ department, where $Y$ is either a manager or a risk owner;
- $T_X$ as the list children of manager $X$;
- $T_X$, refers to manager $X$ sub-tree, and not only his direct children;

We can now define our solution in 2 parts. First, we will define all the risks the users can see. Following we will define how the filter feature work.

Let us start by defining what risk owners can see:

$$X = R_X$$

On the other hand, defining what a manager can see is a little more complex:

$$X = M_X + \sum_{Y \in T_X} M_Y$$

We can simplify this equation for the low-level manager, once they do not possess any children that are managers in risks.
Next, we will look at how the engine should filter the risks when a manager only wants to see a part of his domain. It is important to remember that this option is only allowed for managers. Furthermore, a manager can only do this operation in user of the manager sub-tree.

\[ X_Y = M_Y + (R_Y \cap X) \]

We can divide the last equation in two smaller equations; the first part for when \( Y \) is a manager:

\[ X_Y = M_Y \]

or the second part for when \( Y \) is a risk owner:

\[ X_Y = R_Y \cap X \]

In practice, this concept as much simple to understand, let us have a demonstration. Let us have an organisational list as in fig.6.7 and a small list of risk fig.6.8.
6.5. Risk Segregation

<table>
<thead>
<tr>
<th>Risk</th>
<th>Manager</th>
<th>Risk Owner</th>
<th>See it</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>J</td>
<td>A,J</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>I</td>
<td>A,I</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>H</td>
<td>A,B,H</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>J</td>
<td>A,B,J</td>
</tr>
<tr>
<td>5</td>
<td>D</td>
<td>G</td>
<td>A,B,D,G</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>K</td>
<td>A,C,F,K</td>
</tr>
</tbody>
</table>

Figure 6.8: C-RAT Risks, manager-risk owner example

E does not have access to Risk1. Finally, user B, looking at the definition of what managers can see we have two components, the first one being the risks that are owned by the manager, and the second one the risks, where his children, and his children’s children (and so on recursively), are the managers. B is not the manager; therefore we must look at his children: the risks do not belong to either child D and F nor their sub-trees; therefore both of these options are excluded. We are now left with the users E, and J. E is not the manager of the risk. User J is the risk owner of the risk, but he is not its manager. Therefore user B does not have access to Risk1.

Next, we will look at Risk3 with manager B and risk owner H. As in the previous risk, we will exclude the other risk owners, and user E, C and F once neither user B or H belong to their sub-tree. This leaves us with users A and D to analyse. User D is similar to user E in the last example. User A is not the manager of the risks; then we must look into user A children; the first children in this list is user B, once user B is the manager of Risk3, user A can see the risk.

In order to demonstrate how the risk filtering should work, we will put ourselves in user B place. User B has access to risks 3, 4 and 5; furthermore, we can filter for user D, E, G, H, I and J.

Let us start by demonstrating user filtering from B to D. If we look at our definition:

\[ X_Y = M_Y \]

we now replace X as B and Y as D, following we look at \( M_D \), in order to achieve our final result:

\[ B_D = M_D \]
\[ M_D = R5 \]
\[ B_D = R5 \]

Finally, we will demonstrate risk filtering to a risk owner. As in the previous example, we start with the definition and replace X and Y with B and J respectively.

\[ B_J = B \cap R_J \]
6. Research Based Design

The next step is find both $B$ and $R_J$, after find both of the list, all left to do is to verify what risks belong to both of them:

$$B = R_3, R_5, R_6$$
$$R_J = R_1, R_5$$
$$B_J = R_3, R_5, R_6 \cap R_1, R_5 = R_5$$

6.6 Database Structures

Before we can end this chapter, we must decide on how to represent users and risks, once our application is built around these two elements. Furthermore, both of them need to be stored in the database, once they are needed multiple times through times. Besides, we will also look, at what we need in order to implement the login successfully.

6.6.1 Risk

As it is obvious, we do not expect a manager to introduce all the information in a single sitting, and that somehow the chosen risk owners, where all available at that exact time to insert values to the risks. Not only that but also the manager needed to do their analysis before leaving. Moreover, the next time they sited they had to do everything again. This behaviour is obnoxious! For this reason, we need to store risks.

Let us start by the risk representation. First, we start by looking at what defines a risk. We need a description of the risk, the asset that is damaged by the risk, a likelihood and a consequence.

In this thesis as referred before, C-RAT measure likelihood in the number of expected incidents during a year, and the consequence in the cost of one occurrence. Furthermore, we can combine these two, achieving a cost of year field. Such field will provide the manager with an easy form of comparing the different risks.

Next we, in order to take advantage of C-RAT, we also need a manager and a risk owner responsible by it. Finally, in order to quickly identifying them in the database, and to serve as the primary key in the search, we will also need an ID field.

6.6.2 User

Let us start by justifying why we need to store information about the users. As we explained before we want to mount a user tree, such tree is constructed through users nodes, if we do not have the information about the users there are no nodes, and therefore no tree.

The question we want to answer at this point is: What does user nodes need. Previous in this chapter we stated that the users should have a list of risks within them. If we followed this approach, we would need two copies of each risk. This solution is not efficient and would probably lead to problems during
synchronisation. We needed a solution that would achieve the same result but without having multiple copies. At that moment, we had a radical idea. Instead of having the risk itself, we would have something that pointed to them, like a pointer. We could not find strong argumentation against this idea, and therefore we went with it. Instead of having a list of risk, our solution is a list of pointers to risks.

Now our attention will shift to how we could build the three. The user tree must allow the nodes at the top to see the entire tree; however, the reverse is not true. In addition, no matter where we start, we should not have two paths to the same node, once this would mean, that one user as more than one direct boss, which usually is not the case. Remembering from our undergrads' algorithms and data structures class, we recalled binary search trees, and how we could evolve them to instead of having a left and right children, we could have N nodes, once it is not defined how many underlings a manager can have. The simple answer is through a list of nodes instead of two variables. At this point, our tree was the root as our base, and from there we can reach every branch. However, if we only wanted to see a risk owner, we would have to lead the entire tree, once we are required to start in the root, and we do not know the path to the risk owner. To target this problem, we did the same than in the last subsection, we constructed an userId, and the user lists’ instead of having user would contain pointers to the users. This way if we had the risk owner Id, we would just retrieve that node from the database, and this solution still allows us to construct the entire tree starting from the root if we want to do so.

At this point, we know how to mount the tree, how to navigate in it, and we can quickly fetch the risks. However, we still do not have any information about the users. For the sake of our program, we will have two fields, namely the user name and position. Nevertheless, these fields are optional, once we could put any number of variables and types of variables that represent the users.

6.6.3 Login

Finally, we will look at the login, while we have defined how to represent the users, we did not specify how they can log in, especially without a password on their node.

To start have the password in the node is a bad idea. For example, in that way, a manager machine would know the risk owner password, once the manager will receive his subtree, in addition, if a manager in this position, wanted that a given risk owner was fired, what would stop him to login with the risk owner account, and introduce false values or delete the ones already introduced. In conclusion, store the plain password together with the user information is an awful idea. No one should ever do that!

For this reason, we needed a good way to store passwords, the first decision we took regarding this subject was to create a separate table for storing the user passwords, once there is no need for them outside the server, and we do not want the manager to have access to it.

At this moment we wonder what would happen if we left the problem as it is. We quickly realised that the reasoning we had for the manager in our last
paragraph could be applied to a system administrator. Hence we needed to improve our security. In conclusion, leaving the password in plaintext is a bad idea. We needed to hide it, the more obvious solution to us was to hash it. Once hashes are deterministic and make use of one-way functions, it means that a string always hash to the same result, and while it is easy to hash, it is close to impossible to revert the operation, especially if we use the current standard. Unfortunately, this was not enough, once the same system administrator, could identify if two or more users have the same password, if he one of those users wanted to harm the other, both of them (the system administrator and the user), could login in the other user account and get him fired. Therefore we needed to salt passwords.

In conclusion, fig. 6.9 represent our database and all the fields that contain them. In this examples we did not explore the options of a system administrator alter the risks from his account, once the log file would show that he was the culprit.

Figure 6.9: Database tables

6.7 Summary

In this chapter we went through the design of our tool, in order to do this we looked to what want we want to achieve, and we looked for solutions that would let us achieve our goals.

In the end, we decided to implement both a client and a server. Moreover, the server should multi-threaded. In addition, we decided to create a database using the software from MySQL, with three tables: one for risks, one for user details, and one for the user passwords.

On the client side, we will have a login page and a user page. The program should start by showing the login page, which should authenticate the user with the help of the server. Follow, the client will show a user page. The user page, however, has different layouts and functions considering the user position.

---

5Hashing is a deterministic process. This means that a string will always hash into the same result, assuming we use the same algorithm.
During this chapter, we also start to perform a security assessment of C-RAT, that will be continued and concluded in the next chapter.

Additionally, we also dived into one of the main features of C-RAT; the risk segregation feature; in which we decided that the managers should only have access to their risks, but then the risk, can be spread by owner if the manager so decided.

Finally, in order to keep the application secure, it should only be available in a (W)LAN environment. Furthermore, we decided to implement a protocol that would encrypt the data travelling between the server and the clients.
CHAPTER 7

Evaluation

In the last chapter, we went through the process of designing the artefact C-RAT. In this chapter, we start by pointing out the requirement we found in the last chapter, and therefore we should follow when we implemented C-RAT. Next, we start with the evaluation that we defined in the fifth chapter of this thesis. Beginning by describing the tool, following we conclude the security risk assessment and finally we report the conclusion we took with the user testing.

In our last chapter we decided that:

1. There will be two types of users: managers and risk owners;
   a) Managers are responsible for creating risks, and perform the analyse;
   b) Risk owners are responsible for giving values to the risks;

2. We should follow a client-server architecture pattern;

3. All the request to the database should go through the server;

4. All users requests must be validated before carried out;

5. We should implement a login process that authenticates users in the user tree;

6. The graphical interface should be intuitive and provide feedback;

7. The risk segregation feature should comply with the rules defined in section 6.5.

8. We should have a MySQL database;

7.1 Artefact Description

In this section, we will describe and evaluate all the major processes that constitute C-RAT. We have assumed that the tool setup was successfully complete and the server running, in order to successfully complete the task.
7. Evaluation

7.1.1 Setup

The setup of C-RAT is reasonably straightforward. Starting with the server, the system administrator has to download the files into the server machine, configure the database (further details of the database in subsection X) and finally upload the users' information. In this last process, upload the users' information, is strongly recommended that the system administrator start from the top and then build his way downwards, once a user should have a father node.

In the Client side, is just download the program to the machine or machines the user will use, change the target IP and ports of the server, re-compile and it is ready to use.

7.1.2 Start the Client and Authenticate the User

In this subsection, we will start by looking in detail from the moment a user wants to start working with C-RAT until the moment he is authenticated in the server.

Establishing a connection

In a standard client-server architecture, the server is running, and the clients will send a request to their IP and port. This is also the case of CRAT as you can see in diagram 7.1, in addition, the server upon receiving a request from a new client the server will launch a new thread with a ClientHandler object. It is important to note that a ClientHandler will only handle one user, and it will be destroyed at the end of the communication with the client.

The ClientHandler will be responsible and therefore will handle all communications with this user until a logout message is sent, or the communication is interrupted. At this point, the server will close the object and the thread. Moreover, in its turn, the client will close itself.

Secure Protocol

At this moment we have established a connection between the server and a user (or multiple users through multiple clients); however, the communications are not secure. Because of this, as we discussed in the last chapter, we decided to implement a secure protocol.

As we can see in the Secure Protocol Sequence Diagram (dia. 7.2), this protocol starts with the client sending a message to the server informing him that the Secure Protocol will start. Next, the client creates a PubKey object; this object will generate a public key and a private key pair using the RSA algorithm. Succeeding, the creation of the keys, the client will create a SecureProtocol object, containing the public key generated.

Upon receiving both messages, the ClientHandler will create a Encryptor object; this object will not only create an AES key but is also responsible

1The root node is an exception, which should be the system administrator.
7.1. Artefact Description

Figure 7.1: Sequence Diagram Establishing Connection

for encrypting and decrypting all the objects that arrive and leave this ClientHandler.

After the creation of the AES key, the ClientHandler will encrypt it using the client public key provided by the client; additionally, the client will append the hashed key on the SecureProtocol object and return it to its client.

Back on the client side, after receiving the updated SecureProtocol object, it will extract the encrypted key. Next, it will pass the encrypted key to the PubKey object created before. This object on its turn will decipher the AES key, using the private key previously generated and return the unencrypted key to the client and finally delete itself.

Finally, the client creates an Encryptor using the AES key created in the server, in this way both Encryptor objects use the same key to both encrypted and decipher the messages, allowing the communication between the client and the ClientHandler to be encrypted and making it safe versus the man in the middle attack.

Login

After the secure connection is established the client will launch the loginPage. As you can see in the figure this is a simple login page, with two fields
7. Evaluation

(Username and Password) for the user to fill, and then press login 'sing in' button.

This actions as you can see in the diagram 7.3 will trigger the client to send two messages to the server, the first one will let the user know, that the second message is of the type LoginMessage. The second message as the name suggests carries both the Username and the Password values.

Upon the arrival of the second message, the ClientHandler will verify if a user with that name exists. In the positive case, the database will return an object with user type. This new object will now be used to create a Authenticator object.

The Authenticator object is responsible for identifying the user. In order to do this, it starts by request the user password salt and password hash to the database, then using the SHA-256 hashing algorithm will hash the given password with the salt found in the database. If the resulted hash is equal to the password hash fetched from the database, the user gets authenticated.

This will lead to the self-destruction of the Authenticator and the creation of a Session Hash in the ClientHandler. A sessionHash is a byte string that will serve to authenticate the user in future messages, and for that reason, it needs to be copied to the client.

In the case of either the username do not exist in the database or if the password is not correct, the server will inform the client that there was an error. After,
as you can see in fig. 7.3, a message stating the combination was incorrect is presented to the user.

**Evaluation**

In this first phase of C-RAT, we provide a secure way for a user to authenticate in the server. In this small section, we will look to the choices we took when implementing the communication between the client and the server, the secure protocol and the login process.

Let us start with the communication between the client and the server. In our diagrams, you can note that we have written "Messages", but we never explained what they are on an implementation level.

The communication is carried out by java sockets. Sockets can transport vast amounts of data, and cause very low traffic in the network, especially when compared to web pages. Its downside is the need for interpreters in both clients and server, which forced us to spend more time on the implementation.

Furthermore instead of having a single data structure, carried all the information, which would lead to a lot of unused variables every time a message is sent. We felt the need to implement a specialised data structure for every process. This solution makes the program more efficient but cost us extra time during the implementation phase. In addition, the client always starts with a message
identifying what process will start, which lead to the server have the correct interpreter ready to listen to the new message. This feature has a significant relevance regarding maintainability, once if the developers want to change or add a new process, it is only required to change or create a new function and a data structure, and having no risk of compatibility issues, with the already implemented structure. In conclusion, we use sockets to communicate between the clients and the server. Sockets are more efficient but required spent more time during implementation.

Now we will look into our secure protocol. The main question is "Does it works?". In order to answer this question, we will simulate the protocol, with an outsider (Oscar) listening in the middle. A client starts by producing a public key and a private key. Then the client will send the public key to the server. Oscar being in the middle will also have access to it. Then the server will create an AES Key. It will encrypt the key with the public key and send it back. However, Oscar will also get the encrypted AES Key. Then the client is able to decode it because he has the private key, where Oscar has not. Therefore both the client and server share an AES key, while Oscar does not know it. After this step, the client and the server can encode and decode all the communications between themselves using the AES key, while Oscar will not be able to read them.

Finally, our login process is a standard login. As we explained in the last chapter, we decided that we must use salt in order to protect our information correctly.

7.1.3 UserPage and Risk Insertion and Management

After a user is successfully authenticated in the server, he gets a certain amount of privileges and information according to his position in the tree. In this subsection, we will cover, how the information is received after the user is authenticated, and how are risks created and changed within C-RAT.

User Page

In the last chapter, we decided that we want to show different information to different users.

As we can see in the diagram 7.3, the client starts to ask the server for the users’ details and risks. The server after verification that it is a valid session through the SessionHash, will mount a RiskMap and a UserMap for the user.

The riskMap will contain the risk that concern the user. While the UserMap is highly depended on what type of role the user has.

In the case of the user being a risk owner, the UserMap will only contain information about himself. In the other hand, if the user is a manager, it will contain information about him and his sub-tree, this will later be required for both the risk segregation and risk aggregation features.

After the ClientHandler build both maps, it will send them to the client. The last one, upon receiving the objects, will identify if the user is either a risk owner or a manager, and construct the appropriated layout.
In the design as you can in appendix B, all of the user pages have the user details on top, and a table with the respectively risks on the middle. Then the risk owner, that you can see in figure B.3 has a lower bar with two input fields the first one for cost and the second one for the number of occurrences estimated in one year for a given risk. After the input fields, the bar still has a button for updating. Once the only task of a risk owner is to insert values for both of these fields, his options are also very limited in the client.

When regarding the managers of both level, there are two differences when compared to the risk owner page, the first one being a bottom bar with a different set of fields and options. The second being a tree menu left of the panel. The left menu, has the name of the underlings of the given managers, and works by selecting one of them, and then update the table. This action will trigger the client, to only show the risks relevant to that user, removing all that ones that do not belong to his domain. The bar at the bottom contains a drop-down menu, where all are listed the risk owners working under the manager; the menu is followed by two input fields namely scenario and asset. After the input field there are five buttons: ‘Create’, ‘Update’, ‘Delete’, ‘Aggregate By Scenario’ and ‘Aggregate By Asset’. The first three are concerned with the management of risks; and the last two with the aggregation of them.

**New Risk**

Only managers can create new risk, in order to do it using the horizontal bar at the end of the UserPage.

They start by choosing the correct risk owner on the drop-down menu, then they write a description of the scenario and of the asset, finally they click on the add button.

The client receiving this request from the user will first validate the input
7. Evaluation

Figure 7.5: Sequence Diagram: New Risk Sequence Diagram

if the validation is successful it will create a \textit{NewRisk} object containing the information passed by the user. After the creation of the new object, the client will send the object to the ClientHandler.

The server, upon message receiving, will authenticate the user, next when receiving the NewRisk object, it will stamp the current date time, and upload the new risk to the database. Once the riskId variable is auto-generated in the database, the server will fetch it to the database, and then mount a risk object containing all the information. After the risk creation, the client handler, will append it to its riskmap, and send a copy of the risk to the client.

In the end, the client receiving the risk will update both his riskMap object and then the table.

\textbf{Updating Risk}

Different from the previous section, updating risk can be performed by both risk owners and managers.

In the first case, it is expected that they introduce the values of both the number of occurrences and the cost of one occurrence. While in the second case, we allow for a manager to redefine the asset and scenario description, but not the risk owner.

\footnote{All the information available, once the fields for cost, number of occurrences year and cost year are still empty.}

\footnote{In a manager decides to change the risk owner, he as to delete the risk and create a new one, with the new risk owner.}
7.1. Artefact Description

In the diagram 7.6, we cover the path of the risk owner. While it is just equal to the updating on the managers, the only differences being the fields, and on the managers’ side, there’s also a confirmation box, before sending it to the ClientHandler.

As expected, this process starts with the users introducing the values on the field and click on the update button. This will trigger the client to validate the input, which in the manager case would be followed by a confirmation box. After the fields being validated, the values are changed on the Risk object. Following this, the altered Risk will be sent to the ClientHandler. This one will update the database, and then send a message, informing the client that the operation was successful. Finally, the client knowing the operation was successfully on the server side, will update his riskMap and the table for the user.

---

**Deletion of Risks**

As in the creation of risks, only managers are allowed to delete. The way this process works, is similar to the risk update. The manager selects the targeted risk in the table, and clicks on the button "delete", this will trigger the client to show the user a confirmation box, after the user confirms the operation, the client will inform the server that the following risk is meant to be deleted, following this message with the Risk object. Upon receiving of the second message, the ClientHandler will confirm that the manager of the risk, is either the manager responsible for the risks or if this last one belongs to the subtree of the user requesting the operation. After everything is properly validated, the ClientHandler deletes the risk from the database, and sends a message informing the operation was successful to the client, this one upon the receiving the message will finally update the table.
7. Evaluation

Logout

There are two ways a user can log out of the C-RAT software. The first one, also known as the correct one, is by pressing the logout button on the top of the page. This operation will trigger the client to send a message to the ClientHandler informing him the connection as reached an end, and therefore the ClientHandler self destructs, freeing in that way space for other ClientHandlers', after sending the message the client will close itself, terminating the program.

The "wrong" way to close the program, is by clicking in the X of the window. This will lead to the termination of the program. Upon the termination of the program in the client, the ClientHandler will catch the exception, informing him the connection was broken. It will then, as in the previous case, self destruct and free space for other threads or processes.

Evaluation

In this phase of C-RAT, we saw how the client after being authenticated ask the information to the user and then display it to the user.

From a security point of view, we always validate the input from the user, before approving it. This step is not only essential to prevent bugs, but also prevent injection attacks on the database.

Regarding the features, we construct the basic structures, and when necessary created the support data structure, as in the case of the NewRisk object.

Next, the logout button always forces the client to shut down. This behaviour is intentional. Hence we want to break the connection between the client and the ClientHandler, causing the self-destruction of the later one

The last point worthy of note in this segment is the existence of a confirm box regarding the manager update and delete risks. This box is necessary. Hence if this feature did not exist, we could lose valuable data due to a miss click.

7.1.4 Risk Aggregation

As you can see in our user page, we have two buttons regarding risk Aggregation. Aggregate by Scenario and Aggregate by Asset respectively. Furthermore, we are in a situation where multiple risks harm multiple scenarios. To successfully, aggregate the risks, either by aggregating scenarios or asset, we will base ourselves in the rules defined in the thesis statement scenario.

Aggregate by Scenario

In the first case, Aggregate by Scenario, we follow the rules 1 and 3, stated in the thesis statement chapter. On a practical approach, C-RAT "grabs" all the risks shown in the table and create a new list with them, it is important to note that this operation work with the risks shown in the table, and not all the risks that the manager has access too. Therefore a high-level manager can perform these operations for the risks of a single manager of his sub-tree.
7.1. Artefact Description

Now we have a new risk list; we can perform our analysis in it. To start the list will be divided using the string Scenario value, this operation should create several small lists of risks.

Let us take, one if this small list, with two or more elements. In order to calculate the aggregated risk of this small list, we will calculate its frequency using the first rule. To accomplish this task we will call N, to the list, c for the cost of one occurrence of the risk, f to the number of occurrences expected in one year for the risk, and finally, we will create a new variable v1 initialised at 0. Now we can calculate:

\[ v1 = v1 + N_i f, \forall i \in N \]

Next, in order to find its consequence, we will use the second rule. Let us take v2 as a new variable initialised at 0.

\[ v2 = v2 + (N_i f \times N_i c), \forall i \in N \]

Finally, in order to calculate the aggregated cost year we do:

\[ Aggregated = v2/v1 \]

Allowing the managers to compare the aggregated risks, with the other ones easily. This process allows the manager to see the scenarios that cause the most harm, independently to where this harm is caused.

**Aggregate by Asset**

On the other hand, the other button "Aggregate by Asset", will show what asset suffer more or less harm, independently from where the harm originated.

In order to achieve this result, we use the second rule for aggregation of risks (consequence aggregation). We start performing the same operations that in the previous options until we end up with a set of small lists divided by the value on the string asset.

Let us have c and f defined as before, v3 and v4 two variables initialised at 0, we will use v3 for represent the top part of the equation and v4 for the bottom, once we only want to iterate once for every element in the list, and we can only perform the division after we sum all the frequencies at the bottom.

Finally for each list (N) with two or more elements we calculate:

\[ v3 = v3 + N_i f \times N_i c, \forall i \in N \]

\[ v4 = v4 + N_i f \times i, \forall i \in N \]

\[ Aggregated = v3/v4 \]
7. Evaluation

7.1.5 Risk Segregation

As the reader could read in the user page section, if the user is a manager, the user page will present the user with a tree menu; is through this menu that managers are able to use the risk segregation feature.

In the previous chapter, we defined the rules by which this feature would have to comply. In practice what we achieved our goal by restricting the information the client receives, and by implementing new behaviours.

Let us start with the UsersMap object that the client receives upon having a successful login. This object only carries information about users in the manager sub-tree. With this decision, we took away the possibility to the user try to select users outside of his sub-tree once the client does not know any information of the users that do not belong to the sub-tree.

In the same note, the RiskMap object only contains information about risks with managers in the sub-tree as the reader can note in the following code snippet. This snippet is a function in the server called when the server is building the userMap. The part we want to highlight is that the client handler, before constructing the object, verifies if the user is either a manager or a risk handler, and then follow the rules we defined in the last chapter.

```java
//creating an HashMap with users
public HashMap<Integer,Risk> getRiskByUser(User root,HashMap<Integer,User> userMap)
    throws InterruptedException {
    String query="";
    if(root.getChildren().isEmpty()) { // is a riskOwner
        query = "SELECT * from risks WHERE riskOwner = " + root.getUserId();
    } else{
        String searchIntegers = getChildrenArray(root,userMap).substring(1);
        query = "SELECT * from risks WHERE manager in (" + searchIntegers + ")";
    }
    return getRisk(query,userMap);
}
```

Until this point, we demonstrated that the managers only have access to the information they are supposed to have. The other point to which we must comply to is the risk filtering.

The risk filtering happens in the client side, as we had already pointed out, in the managers’ user tree, we implemented a left panel containing a tree menu. This menu contains the users that work in this sub-tree, this can be easily checked by comparing figures B.9 and B.7 in which we can see the entire tree in one of the pictures, and the other picture, being a lower manager, only the user in the manager sub-tree. The operation is started once the manager clicks in the users we want to see (in the tree menu), and after press the update table button. Upon pressing the button, all the risks where the chosen user is not a manager or risk owner, are removed from the table.
7.2 Security Risk Assessment

In our last chapter, we had a section about security design. In there we found some critical vulnerabilities that were unacceptable. We then find solutions and implemented them; however, we never verified if the alterations we made actually worked. In this section, we will verify if our solutions worked, or if they did not work. If the result is the last one, we will discuss why they did not work and look for another solution.

As you could see in the previous chapter, we found six risks in our previous diagram; furthermore, it is easy to note, that most of the risks are in the red part of the matrix, this was an unacceptable situation, and therefore we developed solutions that we hoped mitigate the damage.

Let us start by re-analyse the graph we developed in the last chapter fig. 6.10. With the change to the environment from open to the public to only being available in a (W)LAN environment, all the risks that went through the public router threat scenario become nonexistent; once the tool no longer works in the public domain. Next, all the risks that originate from the script kiddie, also disappear, once we consider that this type of attacker does not possess the necessary skill to infiltrate in our network. This leaves us to only worry about insiders and black hat hackers. Furthermore, we will not consider phishing or social engineering, once this would increase the number of risks immensely and we did not consider it in the last analysis.

To attack the network, the attacker needs or to infect one machine, or to have a new machine by physical form. Without social engineering is almost impossible for the black hat hacker to place a new device inside the company. On the other hand, this would be reasonably easy for the insider. Additionally, with this device, the insider could cause a DDoS attack to the server, on the other hand, with the data being encrypted he is not capable of reading or change the data in transit.

This leads us to our last point of attack, a company computer. Same as before without social engineering is almost impossible for the black hat hacker to infiltrate in the company unless some employee wrongly downloaded the wrong file, which we will not consider. This takes the black hat hacker outside of our primary concern, leaving only the insider. Considering that if he infects is own computer, the company should track him down very easily removes this as being one of the options. The insider goes to another person login either by having their password or if the employee forgets to logout falls into the social engineering category, therefore we will not look at it.

The analyse we conduct in this section lead us to construct the diagram fig. 7.9 from which we can identify two risks, (fig. 7.7). As before, after identifying the risk, and have a likelihood and a consequence associated with each risk, we should construct their risk matrix, fig. 7.8. We can now observe that not only the number of risks decrease significantly, but also the risks in this table have a lower risk level. We can also observe the decreased in the risk level of these risks when compared to their previous versions is due to his lower likelihood. In addition, if we wanted we could aggregate both of this risk into only one once the assets damaged by the incidents are the same.
7. Evaluation

Figure 7.7: Risk Identification List After

Figure 7.8: Risk Matrix After

7.3 User Testing Results

In the research method chapter, we have proposed ourselves to conduct a user testing experiment, the main target of our test was the intuitiveness of the tool, or with other words, how well could our volunteers complete the task without any assistance.

As you can see in our questionnaire (Appendix A), was built in order to allow the volunteers to perform necessary, but essential tasks that a user of C-RAT would have to perform in a daily basis.

The questionnaire is composed of 8 tasks; 5 of them were related to the managers’ job and 3 to the risk owners. At the end of each task, the volunteers rate the task from 1 to 5. Where 1 means that was very hard to perform, and 5 very easy to perform. At the end of all tasks, we asked the volunteers to give us an overall feedback of the tool.

Next, in appendix C you are able to see the results of the questionnaire. The overall sum of the tasks scores was:

1. 0 times;
2. 0 times;
3. 1 time;
4. 4 times;
5. 35 times;

As the reader can see we had 35/40 (87.5%) answers with the highest possible score. Furthermore, the answer with the "worse" result, is the second one, when
we asked for the first time, for the users to introduce data in the tool. They were in an environment that they never saw before, and because of that, we believe that they were a bit confused at first. Next, in the second time that we asked them to introduce data into C-RAT, we achieved a perfect score.

Furthermore, from the feedback that the participants gave us, most of them said that the tool was simple and/or easy to use. This result is what we were hoping for.

In conclusion, we believe that we exceeded our expectations regarding the intuitiveness of the tool. In retrospective, going for a straightforward design, with a simple placement of elements and procedures, probably led to volunteers to remember similar programs with similar elements, which lead them to quickly identify what they had to do if it was not evident at first, that led us to this fantastic results.
7. Evaluation

Figure 7.9: Before-After Threat Diagram
In chapter 3, we presented our thesis statement and our success criteria. In addition in chapter 5, we explained our research method, and in that chapter, we explained that when an artefact is being developed, the research must discuss, until which point does the artefact satisfy the given requirements. Hence now we will look back to all the success criteria we proposed ourselves to achieve, and discuss to what end they were fulfilled.

8.1 Success Criteria 1

**SC1** The tool must comply with the principles specified in the report Guidelines for aggregation of cyber-risk in large institutions [45].

As described in sections 3.1 and 3.2, our application should follow the principles stated in the book. There are three fundamental principles:

- Risks, should be represented through a potential scenario, an asset, a number of occurrences per year and the average loss of damage for the asset in case the incident occurs.

- The risks should not overlap;

- The three rules to risk aggregation;

Let us start by addressing the first of this principles. As we explained in the second paragraph of section 6.6.1, in our data structure we have 4 fields concerning the definition of risk, namely description (of the risk/ scenario), asset (a description of the asset), a likelihood and a consequence. Further, in the next paragraph we expand on how the likelihood and consequence should be measured. We state that the likelihood will be measure in the number of expected incidents during a year of the given risk, and the cost will be measure on the average loss of damage for the given asset in the given scenario.

The second point in our list, need to be taken into account when performing the second step of risk assessment, namely identifying risks. This tool was designed to help with the third and fourth steps of risk assessment. Therefore
8. Discussion

this principle is outside of the scope of C-RAT. Then Therefore the manager conducting the assessment is the one responsible for upholding the principle.

Finally the last point in our list, is concerned with the three rules for risk aggregation, namely frequency aggregation, consequence aggregation and asset aggregation. We address this principle in section 7.1.4. Our solution, was to either aggregate the risk by asset or scenario. Once we have a situations where multiple scenarios affect multiple asset, we cannot aggregate all at once, which lead to our solution, where we aggregate based on what the manager want to see.

8.2 Success Criteria 2

SC2 The tool must be appropriate and comprehensible for the managers.

The goal of this tool was to provide a way to help the managers have a complete picture of their cyber state, in a quick, complete and efficient form, furthermore, such service would need to be available whenever they request it. Therefore we aimed to build a simple tool, simple in a way that is easy to use, but at the same time, that would provide the main features in order to create, manage and aggregate risks.

The results we achieved with the user testing, strongly suggest that this tool is easy to use. In addition, on this iteration of the tool, we had implemented quick ways to create risks, add value to them, alter them to some extent, delete them, and aggregate them.

The question we must ask now is if our projects work in practice while there is no way to know for sure until it is tested on the field. We did our best to prove it from a theoretical standpoint. Staring with the user testing, all our volunteers were bachelor or masters in computer science or a close field. With this, we tried to emulate the users that would use the tool in real life.

Next, we must look for our process of risk management. Our approach was mostly based on the book Cyber-Risk Management[37], that in its turn, is based on principles stated on ISO31000[12] regarding risk management and ISO27005 for information security risk management.

On the downside, we believe that the aggregation as we implemented it is not ideal. Instead, a system where assets have a structure underneath might be better perceived. Such system should allow assets to be aggregated by category.

As we told before, C-RAT was constructed to help with cyber-risk assessment, especially regarding the evaluation phase. In here we aimed for mostly updating the traditional risk matrix, once it did not provide enough feedback. C-RAT instead of a risk matrix divided into risk levels show a risk list, in each to each risk there is a column for the predicted cost for a given risk in a year. C-RAT instead of saying "this is dangerous", "you should worry about this", says "if you do nothing, expect to pay this until the final of the year". Now instead of a theoretical risk level, managers have a prediction in euros, norwegian kroner or other currencies.
8.3 Success Criteria 3

SC3 After the initial tree setup the tool should be fully automatic.

As we explained before our goal is to make the life of the managers easier. For them the tool should be "plug and play", with this expression, we mean that they as soon as the tool is properly installed, it should be ready to use.

There are two stages that can be considered a setup for the tool. The first one is the installation of the server and the clients. In this phase, the system administrator has to configure the server IP, build the user tree, and create passwords for all the users.

At this moment the managers can use C-RAT, once they are able to introduce risks. However, the aggregation feature brings close to no value, once the risks are still without values.

The second phase is when the tool starts to being used by the managers and the risk owners. The managers creating risks, and the risk owners assigning values to them. This phase will bring out the full potential of C-RAT. Once is only after it has data in its database, that the aggregation feature is most useful, and should be taken advantage of.

8.4 Success Criteria 4

SC4 Individual managers should be able to select the relevant aspects of their domain.

As we explained before managers in high position should be able to see the risks from their underlings. Furthermore, managers in the lower positions are able to separate their risks, through their underlings.

The risk segregation feature was implemented in the left panel of the managers’ user page\(^1\). As the reader can see in see in appendix B, the left panel is filled with a tree list structure, containing the names of the manager underlings.

This structure was built in a way that allows for a manager to select one of the tree elements, and then upon updating the table, the table will only show the risks that concerning the selected user. Note that this feature does not allow a manager to see risks that he should not, but instead filter the manager already have access.

This feature is especially useful when high-level managers or managing directors want to see, the individual aspect of departments\(^2\).

---

\(^1\)Not available in the risk owners user page.
\(^2\)The departments have one main managers, to whom all the other managers in the department are underlings.
8. Discussion

8.5 Success Criteria 5

SC5 The tool should be implemented in an open-source software.

As we stated before, one of the conditions for the project was to implement it in open software. In chapter 3, we explained why this action is still relevant, for both developer and organisation.

Unfortunately, this is our less accomplished success criteria. While we disclosed the source code for the public in late February, until now we got close to no feedback on the project. However, the developer should still benefit from the project, once it serves as an advertisement for his skill when he will eventually look for a job.

8.6 Success Criteria 6

SC6 The software should be build following the java code convention \[46\].

As we stated before, when developing a project, the programmers should follow a convention. Following a convention has a significant impact on the maintenance of software.

For example, 80% of the lifetime cost of a piece of software goes to maintenance,\[convention\], furthermore is very hard for the same author to maintain a piece of software during its whole lifetime. This means that we need to make the software clear and easy to understand.

A simple practice like start class names by an uppercase letter, and variable by a lowercase, goes a long way. Not only that but to also give spaces between lines and give names to the variable that explain what they are holding are simple tricks that improve by a large margin the readability of the code.

Following a convention has also obviously implication when it comes to peer code review,\[3\] once if the reviewer know what are the guidelines for writing, he can understand more thoroughly the code, and from that give a better and quicker review.

In C-RAT we tried to follow the java code convention as closely as possible, once it is likely that this code will be reused, or evaluated by others.

8.7 Success Criteria 7

SC7 The tool must support multiple users at the same time.

After we discuss the goal of the tool, it becomes more apparent that we would need to satisfy more than one user in the company; in addition, when we

\[3\]Peer code review, is a software quality activity, in each one or more developers look to the code of another programmer.
chose how we wanted to input data in our application, we realised we will need different groups of users.

In section 6.2 we explain our solution, and why was it necessary. In there we explained why the client-server architecture pattern was the best fit for the project, and why a traditional server did not meet our standards. Our solution, using a multi-thread server, while still weak to DDoS attacks, is able to handle a considerable number of connections, enough for day to day use.

8.8 Success Criteria 8

SC8 The data must be always secure.

After we started to research about risk management and start making plans on how to construct the tool, we rapidly reached the conclusion that C-RAT would need to handle sensitive data, and therefore it must be secure.

To ensure that C-RAT was secure to handle sensitive data, we saw as necessary to conduct a risk assessment in our tool. In sections 6.3 and 7.2 we conduct a risk assessment to our risk assessment tool. As you were able to see, in the table (fig. 6.5) if we did not have any measure to mitigate the damage, it would be unthinkable for any organisation to use C-RAT. However, in the end, as you can read in section 7.2 and in the table (fig. 7.7), we only needed to worry about attacks to our server, and social engineering.

In the other hand, if the company does not have proper security policies, the machines can be easily breached, and from there C-RAT would be exposed, mainly because the data is not encrypted in the operative systems, but only in transit.

Moreover, while the server verifies the client identity, the reverse is not through. While these may seem odd, we have to remember that the client is the one responsible to initiate the communication, and this operation is started, by sending a request to the server IP address. Which means, that for the user to connect to an attacker, the IP table in the router must be already compromised. Even in the supper unlikely scenario that this happens, the attacker would need to provide the user with his information, that he should not have, once they are only stored in the database. If the attacker has already access to the database, he does not need the user. We can now conclude, that the client does not require to authenticate the server.

We now are able to say that we are reasonably happy with the result, we achieved regarding the security of the data. Hence it is very difficult for an attacker to get his hands on hit without inside help, assuming the company have good security policies.
At the beginning of our thesis, we established that we are living in an era where organisations must move forward; however, take on new opportunities always bring some risks associated. Further, we also showed that the cyberwar is real, and the number of cyber-crimes is breaching new records every year. At the same time, nations and supra-nation organisations are creating new, stricter policies regarding cybersecurity, with a primary focus in user privacy.

For these reasons organisations now more than never, need to pay special attention to their cyber-state, and properly manage their cyber-risk. Otherwise, they risk not only getting significantly harmed by the inevitable cyber-attacks but also to breach the recent rules imposed by the governments.

However, managing cyber-risks is easier said than done. Organisations need to take into consideration not only possible attacks but also accidents that might happen. In addition, big organisations typically have different managers looking into different aspects of the organisation. There was a need to summarise all of this information in a clean way.

C-RAT is an attempt to solve this problem through risk-aggregation. While risk aggregation is not a new concept, we felt like it was one of the best options to try to solve this problem. And this was the starting point for this thesis.

In addition to this, we also tried to give as much flexibility as we could when defining risks, once organisations might want to tweak the system a little for their preferences.

In our eyes there are four major differences from C-RAT to other risk management tools:

- The first one, and probably the more significant difference, is the lack of risk level and risk matrixes in our evaluation. Instead, we opt for a system where we showed the managers the consequence or severity of the incidents in money, either by dollars, euros or norwegian krones, etc. With this change, we hope to give managers a better understanding of how much is at stake and not only theoretical advice.

- The other big difference is that instead of having one user having all the work, we separate the risk assessment phases, through multiple kinds of users. In C-RAT managers are responsible for identifying risks, risk
owners are in the ones in charge when it comes to assigning estimates for both likelihood and consequences to each risk. In the end, managers are still the ones responsible for analysing risk picture after the evaluation.

- Next, we must talk about the risk aggregation and segregation features. While the risk aggregation was a given at the beginning of the thesis, the segregation feature, was something we realise beginning to mid-way of the project. Moreover, while the names seem opposite, both of the features can be used simultaneously if the managers find it useful. By having these two features, we hoped we provided a successful way to at the same type abstract from the specific risks and look more closely into details that might be missed by other applications.

- Finally, C-RAT is highly customizable. Given that we wanted to give as much liberty as possible for managers to describe their risks. With close to no changes, C-RAT is capable of supporting multiple types of risks and not only cyber-risks. This characteristic can be extensively explored allowing C-RAT to hold risks of multiple fields at the same time.

Both the client and the server can be found at:

https://github.com/ThePoog/C-RAT

### 9.1 Directions for Future Work

There are a few routes we could take in order to improve C-RAT:

- Create more features in the client. In order to help managers to have different point-of-views to look at risks. For example, creating a new view of risks, in which we show the sum of the departments in the table; or create graphs based on what is displayed in the table.

- Another route that could take place is to deeper the ways by which risks can be aggregated. Our solution was a very basic one, where we just aggregated by name. One of the possible ways to improve this feature is through an asset tree, just like we did for users. Such system should show the "lower-level" of an asset to low-level managers; and, as the users go up in the chain, the assets should be aggregated based on its category.

- Empirical evaluation, in order to access the scalability of the tool. Carrying out a performance testing should be able to return the throughput and response time of the application. While the application does response time do not need to be instantaneous, the application still needs to have decent response time, in order to be used appropriately.

- Implementing other kinds of risks, and not only cyber-risks. Given the broad ample we allow for risks, the only necessary items for risks are a description, likelihood or consequence. Therefore the tool can easily be extended to support any type of risk.
9.1. Directions for Future Work

- Improve security, while C-RAT is already quite secure, there are always improvements that can be made, one example would be sand boxing the application within the operative systems of both clients and in the server.

- Build a more scalable server. While the tool is not particularly concerned with the network, if an organisation decided that it would want to make a version of C-RAT available on the public domain, we would strongly building a more robust server.

- Improve usability, by creating a notification system. Such system should allow users to know when an update as reached their domain. Another example of improvements in this area would be allowing risk owners to divide risks into smaller ones.

- Create a new page for admin work, such page should allow for managers creating and manage users, in between other admin work. In the same note, the IP and database credentials should not be hardcoded, but instead should be in a supporting document.

- Improve the aesthetic of the client. While our client had very positive feedback, it is still an amateur work, clearly made by a developer and not a designer.
Bibliography


<table>
<thead>
<tr>
<th>Citation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[27] Nielsen, J.</td>
<td>‘Usability Testing with 5 Users’. In: ().</td>
</tr>
</tbody>
</table>
[34] Paar, C. *Understanding cryptography : a textbook for students and practitioners*. 2010.


[49] Toplensky, R. ‘EU fines Alphabet’s Google €1.5bn for antitrust violations’. In: *Financial Times*. 20th March 2019 ( ). [Online; accessed April 07, 2019]. url: [https://www.ft.com/content/e26b4ae0-4b00-11e9-8b7f-d49067e0f50d](https://www.ft.com/content/e26b4ae0-4b00-11e9-8b7f-d49067e0f50d).


APPENDIX A

User Testing Questionnaire
C-RAT Usability Testing

This form will be used to evaluate the user interface of the application C-RAT being developed by Rodrigo Flaminio Ribeiro during his Master Thesis.

The tool is concerned with the aggregation of cyber-risk. And its main goal is to help managers to have a more easily understanding of the current cyber landscape of their domain.

The Tool is to be used by managers and risk owners. A manager is responsible for creating risks and associate them with a risk owner. On the other hand the risk owner is responsible to attribute a value to each risk that was assign to him.

In this 1st section you will be playing the role of a manager when he/she wants to add a new risk.

* Required

1. Login (User: BigBoss, Pass: super) *
   
   Mark only one oval.

   1 2 3 4 5
   Very hard   Very easy

2. Add a new Risk with description and asset. *
   
   Mark only one oval.

   1 2 3 4 5
   Very hard   Very easy

Optional

Feel free to create update or delete new risk. Be careful to leave at least 1 new risk with risk owner Rodrigo

Risk Owner User interface

In this second section you will be playing the role of a risk owner filling the risks you previously created.

3. Login as a risk owner (user: Rodrigo, pass: example) *
   
   Mark only one oval.

   1 2 3 4 5
   Very hard   Very easy

4. Add a cost and nr of occurrence expected , to the risk(s) you created. *
   
   Mark only one oval.

   1 2 3 4 5
   Very hard   Very easy
Optional

Feel free to explore the user interface.

5. Logout *
   *Mark only one oval.*
   
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Hard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Manager Page 2

Now, we are concerned with the readability of the information on the Managers Page. Therefore you will be playing the role of an Manager with complete risks.

6. See only the risk were Rodrigo is the owner. (Tip click on his name on the left tree and then refresh the page) *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Hard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7. Aggregate the risk by Scenario or Asset *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Hard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optional

Feel free to explore the user interface and try any test you want!

8. Logout *
   *Mark only one oval.*

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Hard</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Feedback

9. What did you think of the User Interface *

   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
   __________________________________________________________
APPENDIX B

Printings from C-RAT client

Figure B.1: Login Page

Figure B.2: Login Page invalid password
### Figure B.3: Risk Owner UserPage

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Email</th>
<th>Contact</th>
<th>Status</th>
<th>Description</th>
<th>Team</th>
<th>Role</th>
<th>Class</th>
<th>Code</th>
<th>Type</th>
<th>Code Field</th>
<th>Code Value</th>
<th>Code Field</th>
<th>Code Value</th>
<th>Code Field</th>
<th>Code Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td><a href="mailto:john@example.com">john@example.com</a></td>
<td>123-456-7890</td>
<td>Active</td>
<td>Description1</td>
<td>Team1</td>
<td>Role1</td>
<td>Class1</td>
<td>Code1</td>
<td>Type1</td>
<td>Code Field 1</td>
<td>Code Value 1</td>
<td>Code Field 2</td>
<td>Code Value 2</td>
<td>Code Field 3</td>
<td>Code Value 3</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td><a href="mailto:jane@example.com">jane@example.com</a></td>
<td>987-654-3210</td>
<td>Active</td>
<td>Description2</td>
<td>Team2</td>
<td>Role2</td>
<td>Class2</td>
<td>Code2</td>
<td>Type2</td>
<td>Code Field 2</td>
<td>Code Value 2</td>
<td>Code Field 3</td>
<td>Code Value 3</td>
<td>Code Field 4</td>
<td>Code Value 4</td>
</tr>
</tbody>
</table>

Note: The table continues with more entries and fields.
Figure B.4: High-Level Manager User Page

<table>
<thead>
<tr>
<th>ID</th>
<th>Manager</th>
<th>RiskOwner</th>
<th>Description</th>
<th>Asset</th>
<th>Date</th>
<th>Nr Occurrences</th>
<th>Cost</th>
<th>Cost per Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1023</td>
<td>2</td>
<td>1</td>
<td>Scenario6</td>
<td>Asset4</td>
<td>2019-01-04</td>
<td>6.0</td>
<td>100.0</td>
<td>600.0</td>
</tr>
<tr>
<td>1029</td>
<td>9</td>
<td>11</td>
<td>Scenario5</td>
<td>Asset7</td>
<td>2019-01-04</td>
<td>3.0</td>
<td>6.0</td>
<td>18.0</td>
</tr>
<tr>
<td>1030</td>
<td>8</td>
<td>1</td>
<td>Scenario4</td>
<td>Asset3</td>
<td>2019-01-04</td>
<td>1.0</td>
<td>23.0</td>
<td>23.0</td>
</tr>
<tr>
<td>1031</td>
<td>8</td>
<td>10</td>
<td>Scenario4</td>
<td>Asset9</td>
<td>2019-01-04</td>
<td>9.0</td>
<td>89.0</td>
<td>792.1</td>
</tr>
<tr>
<td>1032</td>
<td>1</td>
<td>1</td>
<td>Scenario1</td>
<td>Asset1</td>
<td>2018-12-27</td>
<td>4.0</td>
<td>400.0</td>
<td>1600.0</td>
</tr>
<tr>
<td>1017</td>
<td>2</td>
<td>10</td>
<td>Scenario2</td>
<td>Asset1</td>
<td>2019-01-04</td>
<td>3.0</td>
<td>23.0</td>
<td>1230.0</td>
</tr>
<tr>
<td>1018</td>
<td>8</td>
<td>11</td>
<td>Scenario2</td>
<td>Asset2</td>
<td>2019-01-04</td>
<td>5.300000199734563</td>
<td>112.75</td>
<td>597.5750122070312</td>
</tr>
<tr>
<td>1019</td>
<td>9</td>
<td>11</td>
<td>Scenario1</td>
<td>Asset2</td>
<td>2019-01-04</td>
<td>2.0</td>
<td>150.0</td>
<td>450.0</td>
</tr>
<tr>
<td>1020</td>
<td>9</td>
<td>10</td>
<td>Scenario3</td>
<td>Asset5</td>
<td>2019-01-04</td>
<td>4.0</td>
<td>100.0</td>
<td>400.0</td>
</tr>
<tr>
<td>1021</td>
<td>2</td>
<td>1</td>
<td>Scenario5</td>
<td>Asset10</td>
<td>2019-01-04</td>
<td>3.5</td>
<td>299.25</td>
<td>1047.375</td>
</tr>
<tr>
<td>1022</td>
<td>8</td>
<td>1</td>
<td>Scenario1</td>
<td>Asset3</td>
<td>2019-01-04</td>
<td>2.0</td>
<td>75.0</td>
<td>150.0</td>
</tr>
<tr>
<td>1023</td>
<td>2</td>
<td>1</td>
<td>Scenario6</td>
<td>Asset6</td>
<td>2019-01-04</td>
<td>70.0</td>
<td>500.0</td>
<td>3500.0</td>
</tr>
</tbody>
</table>
Figure B.5: Aggregate by Scenario

<table>
<thead>
<tr>
<th>ID</th>
<th>Manager</th>
<th>Accountant</th>
<th>Actual</th>
<th>Date</th>
<th>In Circulation</th>
<th>Cost</th>
<th>Case</th>
<th>Case per Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. Printings from C-RAT client
Figure B.6: Aggregate By Asset
Figure B.7: Risk Segregation on Manager

<table>
<thead>
<tr>
<th>ID</th>
<th>Manager</th>
<th>Rights</th>
<th>Description</th>
<th>Base</th>
<th>Assessed</th>
<th>Value</th>
<th>Date</th>
<th>Cause</th>
<th>Issue</th>
<th>Case</th>
<th>Case</th>
<th>Case</th>
<th>Case</th>
<th>Case</th>
<th>Case</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>2</td>
<td>2</td>
<td></td>
<td>4.0</td>
<td>1250.00</td>
<td>3500.00</td>
<td>2010-10-30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure B.8: Risk Segregation on Risk Owner
Figure B.10: Simultaneous Aggregation and Segregation
Figure B.11: Manager Confirmation Box on update
APPENDIX C

User Testing Results

Login (User: BigBoss, Pass: super)

5 responses
C. User Testing Results

Add a new Risk with description and asset.
5 responses

Login as a risk owner (user: Rodrigo, pass: example)
5 responses
Add a cost and nr of occurrence expected, to the risk(s) you created.

5 responses

Logout

5 responses
See only the risk where Rodrigo is the owner. (Tip click on his name on the left tree and then refresh the page)

Aggregate the risk by Scenario or Asset
Logout
5 responses

What did you think of the User Interface
5 responses

- simple
- simple and clear, double click should enable to switch the user on the user tree
- nice
- It's a very simplistic user interface, a person without a good previous knowledge can complete the task easily
- Very easy to use