Consulting in computer systems and software

Master degree in Computer Engineering – Mobile Computing

François Brunet

Leiria, June 2023

Consulting in computer systems and software

Master degree in Computer Engineering – Mobile Computing

François BRUNET

Project Report under the supervision of Professor Leonel Filipe Simões Santos
Leiria, June 2023

Originality and Copyright

This project report is original, made only for this purpose, and all authors whose studies and publications were used to complete it are duly acknowledged.

Partial reproduction of this document is authorized, provided that the Author is explicitly mentioned, as well as the study cycle, i.e., Master degree in Computer Engineering – Mobile Computing, 2022/2023 academic year, of the School of Technology and Management of the Polytechnic Institute of Leiria, and the date of the public presentation of this work.

Acknowledgments

I would like to express my thanks to all the people in the Infrastructure Security and Production team from Tehtris for the welcoming working environment and the caring atmosphere.

I would like particularly to express my gratitude to Logan Scartoni, my supervisor from Tehtris who helped and guided me throughout this experience, Leonel Filipe Simões Santos, my supervisor from School of Technology and Management of the Polytechnic Institute of Leiria, who also guided and helped me to write this report.

I would like to sincerely thank the professors from the Master of Computer Engineering – Mobile Computing of the School of Technology and Management of the Polytechnic Institute of Leiria who made sure I was well integrated and put in the best conditions to succeed.

Finally, I would like to thank Ms Cecile Soldati from EIGSI, La Rochelle, without whom this academic experience at Polytechnic Institute of Leiria would never have been possible.

Abstract

This report aims to describe the work I have done during my project in company. It is part of my second year in the Master of Computer Engineering – Mobile Computing of the School of Technology and Management of the Polytechnic Institute of Leiria. During this experience, I was assigned two missions. The first one is about the study of a S/MIME solution for email security and the second one is about the improvement of Microsoft Office 365 security score.

For both missions I had material at my disposal and some instructions were given to me. I began by analysing the situation, and then established a state of the art in terms of technologies used. Then, thanks to my knowledge, I simulated virtual computer networks, tested encryption solutions, determined what were the best security practices, automated my work by scripting, reported the difficulties, and provided a detailed documentation about my work.

The solutions produced respond to the problems, and they are functional. For the first mission, client machines can send S/MIME emails in a virtual network. For the second mission, the scripts and the tool provided allow to improve Microsoft Office 365 security score.

Keywords: state of the art, virtual networks, encryption, S/MIME, emails, security, scripts, Microsoft Office 365, documentation.

Contents

Originality and Copyright ..... **i**

Acknowledgments ..... **ii**

Abstract ..... **iii**

List of Figures ..... **vi**

List of Tables ..... **viii**

List of Abbreviations and Acronyms ..... **ix**

1. Introduction **1**

2. TEHTRIS **3**

2.1. About TEHTRIS **3**

2.2. Tehtris XDR **6**

2.3. Working environment **9**

3. Background **12**

3.1. Email security **12**

3.1.1. Mail architecture 12
3.1.2. Aliases 13
3.1.3. Public Key Infrastructure for S/MIME certificates 14
3.1.4. Email encryption 14
3.1.5. S/MIME principles 15
3.1.6. Differences between PGP and S/MIME 16
3.1.7. PKI principles 17
3.1.8. How does the PKI operates? 18
3.1.9. Segmentation levels 18
3.1.10. Certificate path 21
3.1.11. PKI solutions 22
3.1.12. OCSP server 23

3.2. Microsoft Office 365 security **25**

3.2.1. Creating an Azure and Office 365 environment 25
3.2.2. Security score 25
3.2.3. Security actions 26
3.2.4. Environment 31
3.2.5. Concept of PowerShell modules 32
3.2.6. Microsoft Graph Application Programming Interface 33
3.2.7. OWASP top 10 API 34

4. Implementation 35

4.1. Email security 35
4.1.1. Network architecture 36
4.1.2. Domain Name Services 37
4.1.3. MDA 38
4.1.4. MTA 40
4.1.5. Nmap script 42
4.1.6. Users and email addresses management 43
4.1.7. Aliases 44
4.1.8. PKI implementation 44
4.1.9. Certificate creation 45
4.1.10. Automating the process by scripting 48
4.1.11. OCSP server 49
4.1.12. Debugging Logs 50
4.1.13. Challenges and enhancements 51
4.1.14. Documentation 52

4.2. Microsoft Office 365 security 54
4.2.1. Sort data 56
4.2.2. PowerShell script 57
4.2.3. PowerShell modules 57
4.2.4. Use least privileged administrative roles 58
4.2.5. Creation of a Python tool to assign administrator roles to users in the Azure AD 60
4.2.6. Creation of the API on Azure 61
4.2.7. OWASP Top 10 API 62
4.2.8. Token acquisition 64
4.2.9. Get user 65
4.2.10. Get role 65
4.2.11. Post role 65
4.2.12. More details 66
4.2.13. Containerise the application 67
4.2.14. Documentation 69

5. Conclusion 70

6. Bibliography 73

Glossary 76

Appendices 78

List of Figures

Figure 2.1 – Tehtris XDR platform. 6

Figure 2.2 – XDR products. 7

Figure 3.1 - Email architecture 13

Figure 3.2 - Alias principles 13

Figure 3.3 - PGP encryption process 15

Figure 3.4 - 3-tier hierarchy 19

Figure 3.5 - 2-tier hierarchy. 20

Figure 3.6 - 1-tier hierarchy. 21

Figure 3.7 - Certificate creation process 21

Figure 3.8 - Azure AD security console 29

Figure 3.9 - Azure AD guest and member users. 30

Figure 3.10 - Concept of PowerShell modules. 33

Figure 4.1 - Network infrastructure 36

Figure 4.2 – Ping the direction.org domain. 38

Figure 4.3 - Dig command for direction.org domain. 38

Figure 4.4 - Telnet command output. 39

Figure 4.5 - Netcat command output. 40

Figure 4.6 - Maildir's structure. [1] 41

Figure 4.7 - Telnet command output. 41

Figure 4.8 - Netcat command output. 42

Figure 4.9 - Output of the nmap script. 43

Figure 4.10 - Content of the /etc/passwd file. 43

Figure 4.11 - Email test. 44

Figure 4.12 - Package verification 48

Figure 4.13 - CSR generation 48

Figure 4.14 - Certificate signing 49

Figure 4.15 - PKCS#12 obtention 49

Figure 4.16 - Certificate deletion 49
Figure 4.17 - Command to start the OCSP server. 50
Figure 4.18 - Revoked certificate for a client. 50
Figure 4.19 - Authentication failure logs. 50
Figure 4.20 - Mail logs 51
Figure 4.21 - Azure AD users. 54
Figure 4.22 - Azure AD groups. 55
Figure 4.23 - Azure AD applications. 55
Figure 4.24 – Extract of the Python script output for the CSV file. 56
Figure 4.25 - Import and connect PowerShell 57
Figure 4.26 - User can choose between three options 58
Figure 4.27 - Get a name and an administrative role 59
Figure 4.28 - Verification about the role entered 59
Figure 4.29 - Function to associate multiple users with administrative roles 60
Figure 4.30 - Permissions for the API. 62
Figure 4.31 - User needs to provide its IDs throughout the terminal 64
Figure 4.32 - Output of the Python application 66
Figure 4.33 - Functioning of the Python application. 66
Figure 4.34 - Security condition Python. 67
Figure 4.35 - Help command. 67
Figure 4.36 - Docker file 69

List of Tables

Table 2-1 - Organisation chart of Tehtris 5

Table 2-2 - Tehtris values 5

Table 2-3 - Missions 11

Table 3-1 - Differences between S/MIME and PGP 17

Table 3-2 - PKI solutions 22

Table 3-3 - Differences between CRL and OCSP 24

Table 3-4 - PowerShell modules for the Office 365 products 32

List of Abbreviations and Acronyms

AD	Active Directory
API	Application Programming Interface
APT	Advanced Package Tool
CA	Certification Authority
CLI	Command Line Interface
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CSV	Comma-Separated Values
DNS	Domain Name System
DoS	Denial of Service
GNU	GNU's Not Unix
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
ID	Identification
IMAP	Internet Message Access Protocol
IP	Internet Protocol
JSON	JavaScript Object Notation
MDA	Mail Delivery Agent
MiTM	Man in The Middle
MSAL	Microsoft Authentication Library
MTA	Mail Transfer Agent
MUA	Mail User Agent
MX	Mail eXchange
OAuth	Open Authorization
OS	Operating System
OWASP	Open Web Application Security Project
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
POP3	Post Office Protocol3
RA	Registration Authority

RSA	Rivest-Shamir-Adelman
S/MIME	Secure Multipurpose Internet Mail Extension
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locators
VPN	Virtual Private Network

1. Introduction

As any other company that possesses an internal computer network, the number of cyberattacks faced is important and continuously rising. The threats and attack techniques are evolving, because Tehtris creates and develop cybersecurity software its internal network must be well protected.

As part of my second year in the master's degree "Computer Engineering – Mobile Computing" at the university Instituto Politecnico de Leiria, I have chosen the Project in company as my main curricular unit. I have worked for 5 months at Tehtris in Pessac, France. Tehtris is a company that was founded in 2010, specialised in cybersecurity. I have chosen a project in company in the cybersecurity industry because I have worked on cybersecurity topics during my 1st year in the master and I wanted to develop and enhance my skills in order to start a career in this field. I also wanted to discover what a cybersecurity company is and what are the different jobs linked to it.

I had the privilege to work with the Integration Security and Production team. During my stay in the company, I was assigned two missions in relation with email security and Microsoft Office 365 security. The time allocated for this work experience was split in two parts. two months and a half were dedicated for each mission. I was free to organise my work as I saw fit while of course respecting the 9am to 5pm working hours, as long as the criteria for each mission were met. These criteria involved providing a functional solution for both missions along with documentation that carefully detailed how I was able to perform my missions.

During this project in company, at the beginning of both missions, I had a meeting with my company supervisor with the aim of explaining to me the expectations and the means I have at my disposal, it included both hardware and software. Every Friday afternoon, a thirty minutes meeting was held with my supervisor. The goal was to share my progress with him and the difficulties I encountered, and he would then give me instructions for the next week. In case I had an urgent problem, I would talk with my supervisor through a secure application, and if needed we would hold a meeting. A secure server was used to exchange files.

This project report will be structured in four parts. The first part will describe how Tehtris was founded and by who, how the company evolved since its creation, who are its directors, how the departments are structured, what are Tehtris values, what is the eXtended Detection and Response (XDR), the main solution sold by Tehtris and more importantly, the various products that make up the XDR. This part will also describe what has been the working environment, what equipment were used, and with whom I had the opportunity to work with. The second part is about the theory concepts covered for each mission, it will explain how S/MIME and Microsoft Office 365 security score work. It will be followed by the implementation part which will describe technically how the solutions implemented were conceived. The last part will conclude the thesis and explains the limitations and potential future enhancements.

2. TEHTRIS

2.1. About TEHTRIS

Tehtris is a French company based in Pessac, France. Its acronym means Technological Ethical Hacker Trusted Robust Information Security. It was founded in 2010 by two former DGSE (French intelligence agency for external security) agents: Laurent Oudot and Elena Poincet.

From 2010 until 2012, the main expertise of the company is relying on conducting penetration tests and discovering 0-days vulnerabilities. In 2012, Tehtris begin to develop cyber defence solutions know as e-Gambit, and honeypots. In 2013, the company launches its Endpoint Detection Response (EDR) solution as well as its Security Operation Centre (SOC) service. Since 2015, the XDR solution is able to neutralize cyberattacks automatically without any human intervention. Nowadays, Tehtris's XDR is deployed worldwide in more than a hundred countries and in broad range of activities such as: industry, transport, education, health sector, bank, insurances, public administrations, services providers, and more ...

In 2020 the start-up has raised twenty millions of euros in funding. The main goal of this fund-raising strategy was to expand Tehtris technology abroad. During the 2020 year, most of the employees were working remotely due to the Covid-19 crisis. This way of working has amplified the potential vulnerabilities a company can suffer. This same year, Tehtris's number of employees were around sixty, the following year 2021, they were one hundred and eighty five of them.

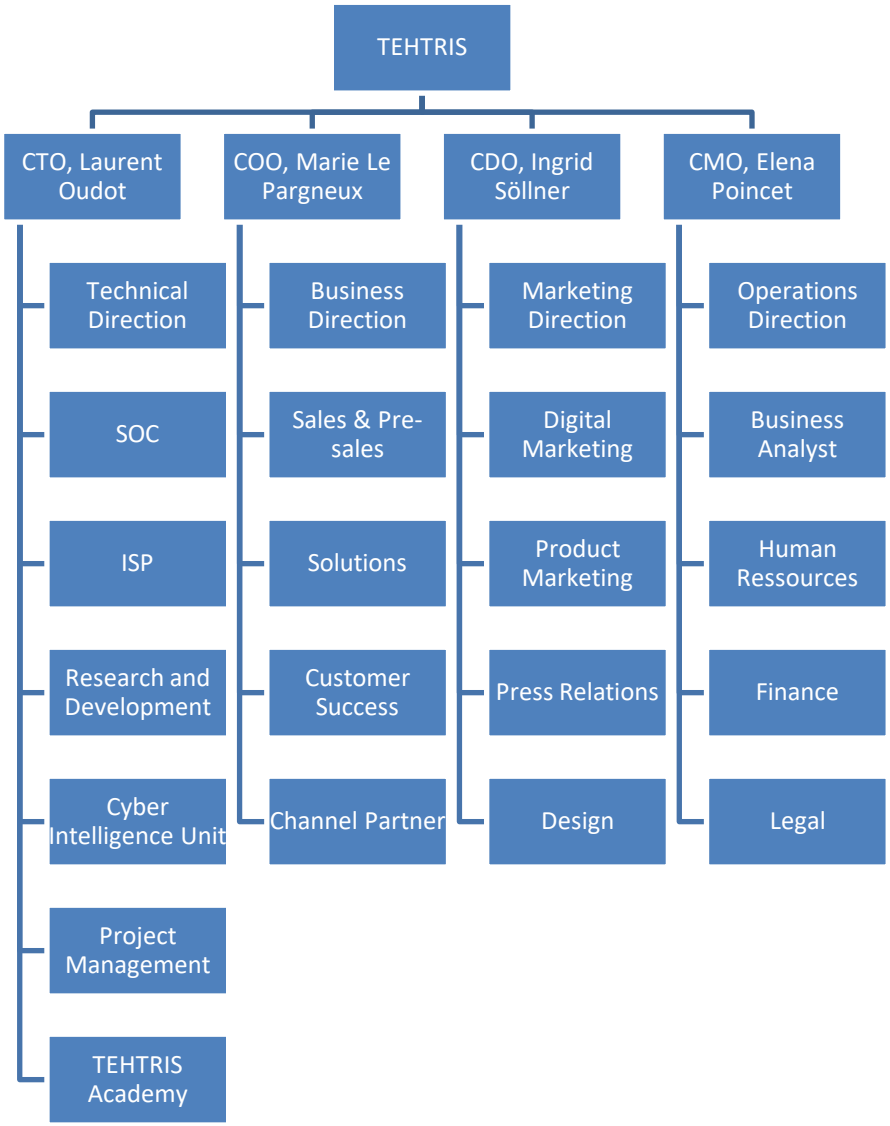
In October 2022, Tehtris has realized a new fundraised. This time the company has raised forty four millions of euros in funding. The reason is to accelerate its international development. Indeed, the company Research & Development (R&D) centre and technical teams have always been in Pessac, France but since 2022 they opened up two new offices for technical teams in Vancouver, Canada and Tokyo, Japan.

Nowadays, Tehtris is a leader in France and Europe in the publishing of software for the automatic neutralisation of cyber-attacks. The company is committed to provide software to protect European country's digital sovereignty and therefore, represents a trusted third

party for the protection of company's information systems. Indeed, it is currently the only French XDR and EDR fully developed and hosted in Europe.

The company's executive committee is made up of four different directions. The first one is the technical direction which is managed by the chief technical officer and co-founder: Laurent Oudot. The technical direction is split up in two units: the R&D and expertise and services teams. The chief technical officer is supervising the various teams that work in those two units. The second one is the business direction. This one is supervised by Marie Le Pargneux who has the chief development officer role. The business team is responsible for developing commercial opportunities in France and abroad. The third one is the marketing direction. This one is supervised by Ingrid Söllner who has the chief marketing officer role. This department works on subjects such as: design, social networks, events organisation, interviews, communication with the medias. The last direction is the operations direction. It is supervised by Elena Poincet who plays the chief operation officer role. This department deals with human resources, financial and legal subjects. The table 2 1 summarizes this paragraph.

Table 2-1 - Organisation chart of Tehtris



Tehtris values are summarized in the table 2-2 below.

Table 2-2 - Tehtris values

Agile mindset	**Personal development**	**Social commitment**	**Professional development**	**Good working environment**	**Benevolence and conviviality**
Tehtris fosters teamwork and proximity between every employee.	The balance between professional and personal life must be taken seriously.	Diversity, inclusion, and equity are key words.	There are many opportunities of development, meritocracy is a key element.	Tehtris pays attention to the well-being of its employees.	Benevolence and conviviality are fostered between employees.

2.2. Tehtris XDR

Since 2012, Tehtris produces its main product that is known as the eXtended Detection & Response platform also known commonly as the XDR platform. This platform enables the automatic collection, processing, and analysis of cybersecurity events from Tehtris technologies or from other editors. Thanks to various technologies, the XDR platform is capable of neutralising immediately cyber-attacks and brings an on-board hyper-automation. The figure 2.1 summarizes the technologies used in the XDR.

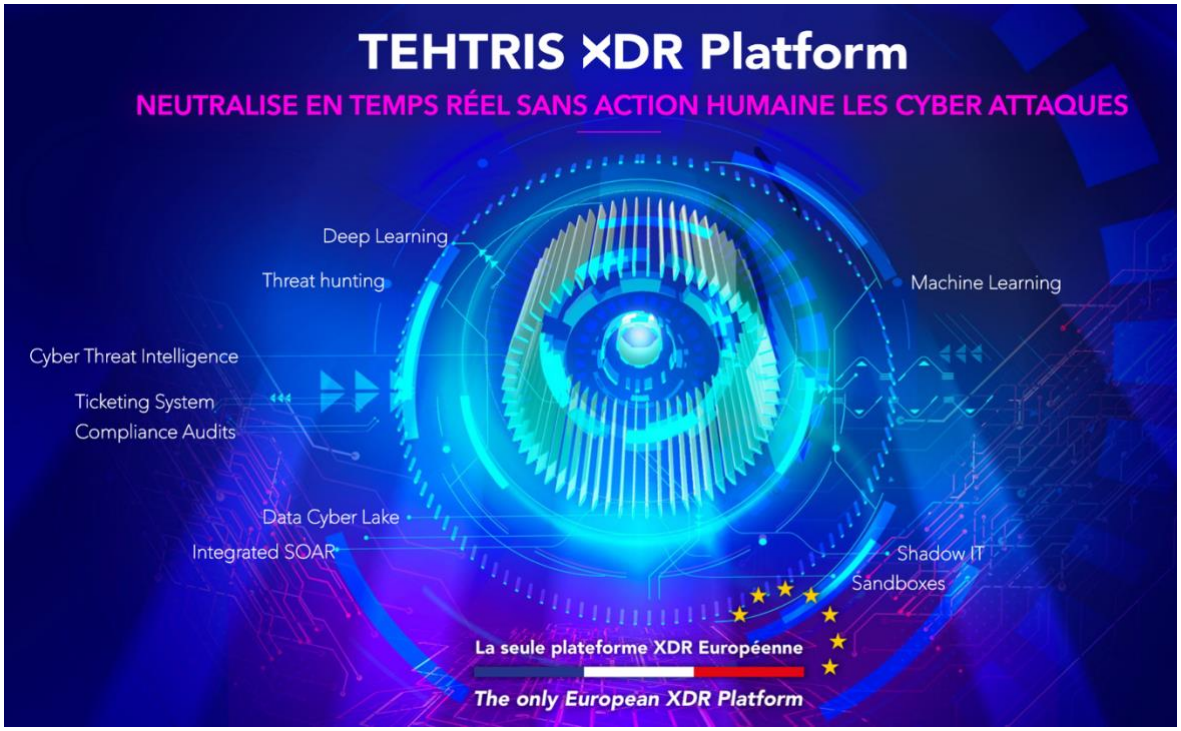


Figure 2.1 – Tehtris XDR platform.

The XDR platform can be deployed on most of the information systems, among them we have cloud infrastructures, endpoint, servers, mobile devices, internet of things, and network equipment such as firewalls, Virtual Private Network (VPN), and more…

Tehtris edits cybersecurity solution that are natively compatibles with the XDR platform. The figure 2.2 below, shows the products integrated in the XDR.

Figure 2.2 – XDR products.

Among those solution there is the Security Information and Event Management (SIEM), it is a system that collects and analyses all kind of cybersecurity events and logs that come from various sources.

The Domain Name Service FireWall (DNS FW) allows to block Domain Name Service (DNS) resolution towards domain names or IP addresses known as dangerous. Moreover, logging DNS queries allows to detect weak signals. Weak signals may correspond to data exfiltration through DNS queries or Command & Control (C&C) towards a malicious server.

The EDR is a technology that allows to respond to know and unknown sophisticated cyber-attacks. It is made up of various tools (audit and hunting tools and more …), which enables an in-depth layered defence. The EDR system, gathers a couples of technologies such as artificial intelligence, automated responses, or behaviour analysis. It monitors Random-Access Memory (RAM) memory flows, Universal Serial Bus (USB) ports, processes. It makes possible to isolate the machine from the network as well as some malicious programs.

The Endpoint Protection Platform (EPP) is a EDR complementary technology and allows to take advantage of the next generation antiviruses. It allows to manage the firewall

rules, protect internet browsing, and 0-day exploit against the operating system. It allows to restrict all physical ports on a computer.

The Mobile Threat Defence (MTD) is a platform conceived to protect mobile devices, smartphones, and tablets. It has many features related to security models of operating systems like Android, iOS or iPadOS.

Zero Trust Response (ZTR) is a technology that protects in real time every network flow between client machines thanks to a monitoring based on data flow capture mechanisms (NetFlow) and other protections mechanisms such as automatic incident response, and DNS protection. It is especially efficient for Internet Of Things (IoT) equipment.

Network Traffic Analysis (NTA) technology is responsible for monitoring networking exchange inside an information system. A passive auditing system embedded allows to scan for vulnerabilities in the network. Its usage can identify data exfiltration and communication with a command-and-control server.

Deceptive Response offers an alarm system made of honeypots. Honeypots is an active defence method which consist of attracting attackers into fake systems in order to identify and neutralise them. When an attacker performs a scan of a network the deceptive response system, logs the information of the scan and unveil the presence of a malicious actor which represents another difficulty for hackers.

Cyber Threat Intelligence (CTI) is a feature that allows the XDR to query a cyber threat database. This database is composed of data and knowledge about emerging viruses and threat as well as former ones.

Cloud Workload Protection Platform (CWPP) was conceived to address the challenges brought by cloud transition. The aim of CWPP is to increase the protection of the SIEM for protecting cloud infrastructures.

Security Orchestration Automation and Response (SOAR) allows the definition, thanks to playbook, of the multi-technological response to be provided in case of pre-established scenarios. When an anomaly is detected by the EDR it triggers an automatic and

coordinated response with the help of other technologies such as: EPP, CTI, MTD, and others…

A unified console, accessible through a web navigator, allows the administration and operation of all Tehtris technologies. Thanks to CYBERIA, which represents all the data science behind the XDR, the user can visualize all sort of data related to Tehtris technologies on various dashboards. The XDR platform also has its own artificial intelligence which is associated to other modules and the CTI data to better qualify threats. The XDR also takes advantage of a cyber data lake to gather a large quantity of data and analyse it in a short time delay. Every alert generated by Tehtris technologies are stocked in a data warehouse which allow to perform accurate research and create personalized dashboards.

2.3. Working environment

During this Project in company, I worked in the Infrastructure Security and Production (ISP) team. The main tasks of the engineers working in this team are to: deploy, integrates remotely, maintain the infrastructures, secure the production, diagnose, and resolve incidents and advise other internal teams. I had the occasion to work on the "secure the production theme". Indeed, the goals of my missions were to enhance security aspects both inside the internal network and outside (cloud security).

For this project in company, I have been supervised by Logan Scartoni. Logan works at Tehtris as a cybersecurity consultant since a few years. His main tasks are related to the maintenance and evolution of Tehtris's internal structure. Because my missions were related to those themes, I worked closely with Logan throughout all the project in company period. If I ran into difficulties, Logan provided me resources that I should investigate to get unstuck, he also gave me advice if I was on the right track to solve the problem or not. During this few months' period, Logan and myself held a meeting every Friday afternoon. The purpose of this weekly meeting was to debate about what have I done during the week, the difficulties that I have encountered, how the project was progressing, and what were the objectives for the incoming week. In other words, the principle of those meetings was to track the evolution of my work.

I was supervised by Julie Aceto who was my People Manager. I had the occasion to regularly exchange with Julie about my situation in the company. Every 8 weeks, we did a 30-minutes meeting to address topics such as if: am I feeling well at work, am I advancing well in my work, I need something else to work more efficiently, my supervisor helping me enough, and other subjects. Julie Aceto was also responsible for organizing the meetings with my Politecnico de Leiria supervisor, Mister Leonel Santos, and supervising my project in company's report. Overall, thanks to Julie Aceto, I was able to be operational quickly and evolve in optimal conditions in order to carry out the missions that I have been assigned.

I worked in one of Tehtris open space in Pessac. Regarding the equipment, I used a computer that has strong enough characteristics to handle multiple virtual machines (VM) running at the same time.

The table 2-3, on the next page, summarize the missions related to this project in company along with the context, the issues and the time allocated.

Table 2-3 - Missions

Missions	Context	Issues	Time allocated
Study of a S/MIME solution for email security.	TEHTRIS tries to enhance its security for emails communications	Provide a functional solution that is able to create and deliver certificates for S/MIME emails communications inside a virtual network. Provide documentation about the theoretical concepts involved in this mission as well as how to implement the solution.	three months
Improve Microsoft Office 365 security score.	TEHTRIS possesses a Microsoft O365 subscription and needs to increase the security score.	Provide valid technical solutions capable of increasing the security score. Provide a documentation that explains the solutions delivered and how to improve the security score.	two months
Master thesis writing.	TEHTRIS required some parts of the Master thesis to be written before the end of the project in company period.	Parts that TEHTRIS require must be written and reviewed before the end of the project in company period.	five months

3. Background

The part number three will focus on explaining theoretical concepts related to the missions.

3.1. Email security

3.1.1. Mail architecture

The mail architecture involves a Mail delivery Agent (MDA) and a Mail Transfer Agent (MTA). A MTA is software that transfers electronic mail messages from one computer to another using the Simple Mail Transfer Protocol (SMTP) [1]. On the other hand a MDA is a software that is responsible for the delivery of e-mail messages to a local recipient's mailbox. [2]. Both must be used in order to send and download an email to a server.

When sending and reading an e-mail two protocols must be used. The first one is responsible of forwarding the email from the client to the server. In case it needs to reach another server, it will also be handled by this protocol: the SMTP protocol. When downloading the e-mail from the server and reading it on the Mail User Agent (MUA), two protocols exist, Internet Messaging Access Protocol (IMAP) and Post Office Protocol (POP3).

POP3 downloads the email from a server to a single computer, and then deletes the email from the server. Whereas IMAP stores the message on the server and synchronizes the message across multiples devices. For a better security POP3 is more suitable. POP3 allows one machine to connect to the server and download the emails related to one account. Once those emails are downloaded, they are deleted from the servers. With POP3 configured, if one attacker manages to take control of the email server, he will only find the emails that were not downloaded yet. Thus, the attacker could find no emails or a slight quantity of emails. In the other case, if the server is configured with IMAP all emails are kept on the server.

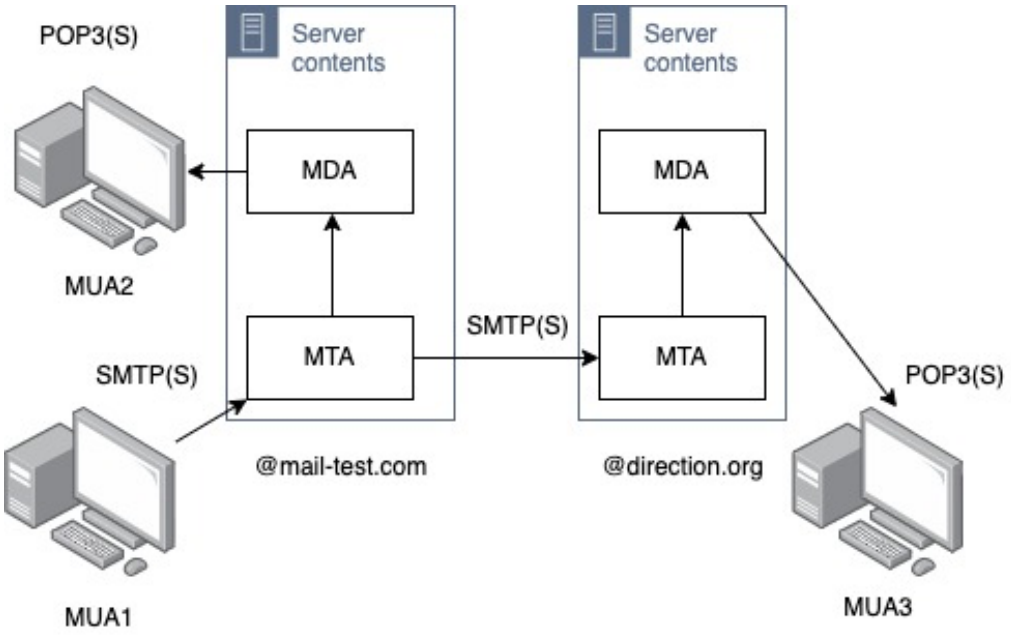


Figure 3.1 - Email architecture

On the figure 3.1, the MUA1 sends an email. If the recipient's domain name is related to MUA2 (@mail-test.com), the mail will be forwarded thanks to the SMTP protocol toward the MTA, the MTA will forward it to the MDA and then the MUA2 will be able to download the email thanks to the POP3 protocol. If the recipient's domain name is @direction.org, the MTA from the @mail-test.com will forward the mail towards the other server's MTA thanks to the SMTP protocol. Then to the MDA and the MUA3 will be able to download the email thanks to the POP3 protocol.

3.1.2. Aliases

The scenario could use aliases. According to Wikipedia, an alias can be defined as "a simple forwarding address" [2]. A company with an important number of employees relies on aliases. Instead of typing the email address of every member in a team, an end user could simply send the email to an alias which then will be responsible for forwarding the email to every member linked to the alias. The figure 3.2 illustrates the concept of an alias.

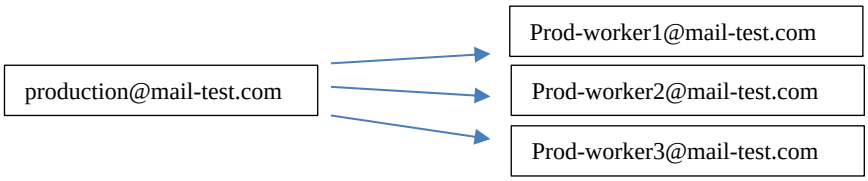


Figure 3.2 - Alias principles

3.1.3. Public Key Infrastructure for S/MIME certificates

The main goal of this project is to study the feasibility of creating a Public Key Infrastructure (PKI) for delivering S/MIME certificates for email encryption. Nonetheless, S/MIME is not the only way of encrypting data for secure email exchange. The advantages brought by S/MIME will be detailed compared to another form of email encryption such as GNU PGP key.

3.1.4. Email encryption

Nowadays, emails may be one of the most popular ways of communication, nevertheless it is subject to different types of attacks. One of the most efficient ways of securing emails communication is to encrypt them. Among the popular type of emails encryption there is the Pretty Good privacy (PGP) encryption system. Even though PGP is not only used for emails encryption, it has become the encryption standard for emails [4]. PGP uses both symmetric encryption and public key encryption. Symmetric keys encryption relies on one key used to both encrypt and decrypt data, whereas public key encryption also known as asymmetric encryption relies on both a public key, that is used to: cipher data, authenticate and validate a signature; and a private key that is used to: authenticate a user and decipher data that was previously ciphered thanks to the public key. PGP implements the following cryptographic principles:

- Authentication
- Integrity
- Privacy
- Data security

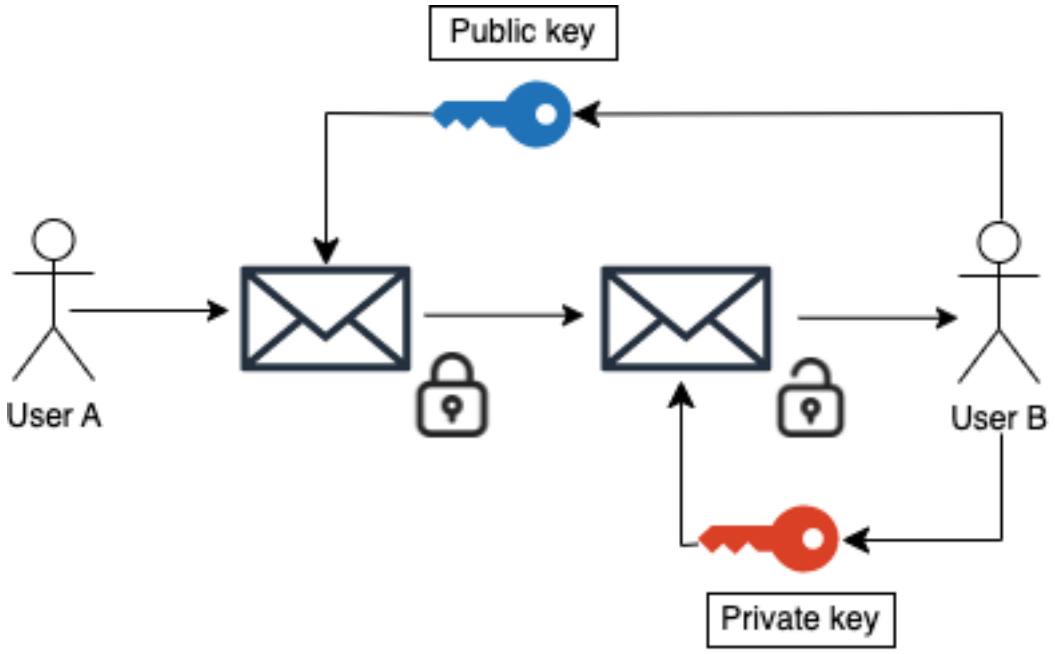


Figure 3.3 - PGP encryption process

The diagram 3.3 simulates the process of encrypting an email thanks to PGP encryption. User A on the left of the diagram wants to send an email to User B which is located on the right side. To do so, user B generates a public and a private key. The public key, colored in blue, is sent to user A, and this one uses the public key to cypher the content of the email. The private key is kept by user B and it is never exchanged. Then, the email is forwarded to user B. When it is received, user B deciphers the content of the email thanks to the private key. Only this private key, which is linked to the public key can decipher the email. Any other private key would be useless.

Encryption brings many benefits from a security point of view. It prevents Man in The Middle (MiTM) attacks. Anyone inside the network with the appropriate software could intercept emails and thus read the content. Encryption prevents this kind of attack, but it goes further, if an email server is compromised it prevents attackers to read the content of the emails stored on this server. Different kind of encryption exist for securing emails, the PGP encryption was detailed previously. The differences between PGP and S/MIME is explained in the upcoming part.

3.1.5. S/MIME principles

S/MIME is a standard for public key encryption and signing of mime data [5]. S/MIME implements the following cryptographic principles:

- Authentication

- Integrity
- Non-repudiation
- Privacy
- Data security

One of the main concepts of S/MIME is that it is based on X.509 certificates. Those certificates are delivered by a PKI. The public key of a user is linked to a certificate which means that it is signed by a Certificate Authority (CA). By doing so, the user must trust the CA. It guarantees that the public key used to cipher data is the right one and no malicious user has tried to steal the identity of someone during the key exchange.

As mentioned earlier S/MIME relies on a digital certificate. The CA confirm the identity of the end entity (or person) who will use this certificate [6]. Then, every time the person will send an email thanks to this certificate it will be digitally signed. By sending digitally signed email thanks to the S/MIME certificate it allows to prove the identity of the sender and the authenticity (the email was created and signed by the person who owns the certificate).

3.1.6. Differences between PGP and S/MIME

A major benefit that S/MIME brings, is the key management. In case of a company relying on PGP encryption for its emails, the key management can be a daunting task for administrators. Every time a new worker joins the company, he needs to generate a pair of public and private key. Then the public key must be sent to every contact to allow encrypted communication. When another user does not keep a record of every user's public key the emails cannot be encrypted. This problem sometimes encourages users to not encrypt their emails and represents a burden for non-technical users.

This problem can be solved by using S/MIME, a certificate with the ".p12" extension used for emails communications, bundles the private key and the public key as well as information about the owner. Because the pair of keys are tied to the certificate, the user never needs to exchange any key or to keep a record of other user's keys.

We could also mention the cost. Indeed, PGP is a less cost-effective solution than S/MIME. Mainly because PGP is easy to set up and S/MIME relies on a PKI. Most of the

businesses nowadays relies on third-party PKI provider that can be free (Let's Encrypt) but other are chargeable. The table 3.1 highlight the differences between PGP and S/MIME.

Table 3-1 - Differences between S/MIME and PGP

PGP	S/MIME
Keys management can be a tedious task	Better keys management
Easy to set up	More difficulties to set up
Depends on user key exchange	Depends on a PKI
Encrypts only text	Encrypts text and multimedia files

3.1.7. PKI principles

The primary goal of a PKI is to confirm identity. It proves the ownership of a private key and also verifies that a sender or receiver are exactly who they claim to be. A private PKI is used to authenticate entities on an internally hosted service. This contrasts with a publicly hosted PKI service which will authenticate end entities on public services. In the case of this project, the solution of a private PKI was retained. Internal or private PKI act as a trust service for a broad range of services and technologies such as VPNs, web applications, internet of things, and end user device authentication.

Depending on the complexity of the PKI, it is made up of different components. In this case it will implement a CA, which issues an end entity certificate and act as a trusted component within a private PKI. When a CA issues a certificate, it is trusted by every entity that trust the CA, a CA is on the top of a PKI hierarchy. The different hierarchy types and architecture will be defined in the part 3.1.9.

A Registration Authority (RA) receives the Certificate Signing Request (CSR) and is responsible for verifying the identity of an end entity. Thus, the RA will verify and approve or not a request before it can be processed by the CA. A CA maintains a database of the certificates that have been issued and the ones that have been revoked. The verification process can be played by a Certificate Revocation List (CRL) or an Online Certificate Status Protocol (OCSP) server. Modern PKI requires the use of an OCSP server. The reasons will be detailed in the "OCSP server" part. Finally, a secure storage for private keys must be implemented on both the CA and end entity. If an attacker steals the private key of an end entity or worse the root CA, he could impersonate the roles of the stolen keys.

3.1.8. How does the PKI operates?

An end user needs a certificate, to do so he needs first to register to the PKI by contacting the RA and providing a level of identification. A CSR is generated, and it contains elements such as: the end user's public keys, and a signature using the private key to prove the ownership and to create a certificate. Second step includes the verification of the CSR. Does it match the specifications that were provided during the creation phase? If it is the case the RA will then approve the request and forward the request and instruct the CA to issue the certificate. If it is not the case the request is abandoned. When a certificate is issued, some key details are stored on the CA server.

3.1.9. Segmentation levels

Different types of hierarchy structures exist for a PKI. Each one is designed according to some requirements. The two most commons are the 2-tier and the 3-tier. In a 3-tier hierarchy, the root CA issues intermediate CA certificates that serve different purposes, the intermediate CA will then issue certificates for Issuing CA that will finally issue end entities (mail certificate) certificates as illustrated on the figure 3.4. In this case an extra layer is added which requires more administration and it is more suitable for larger networks.

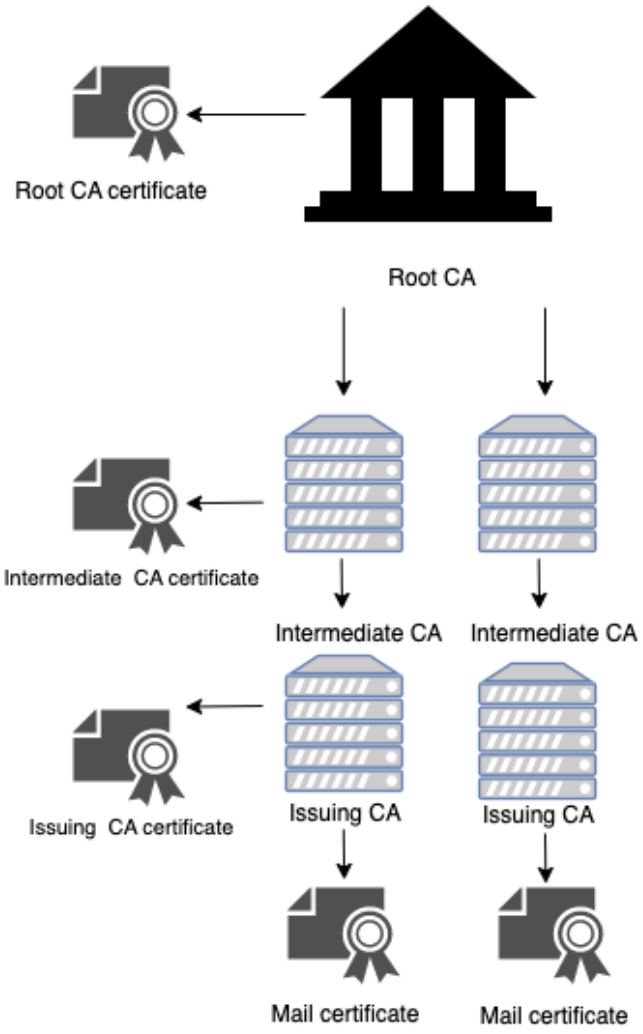


Figure 3.4 - 3-tier hierarchy

In a 2-tier hierarchy, the root CA issues certificates to issuing CA servers, those ones will provide certificates for end entities. The figure 3.5 shows the functioning of a 2-tier hierarchy.

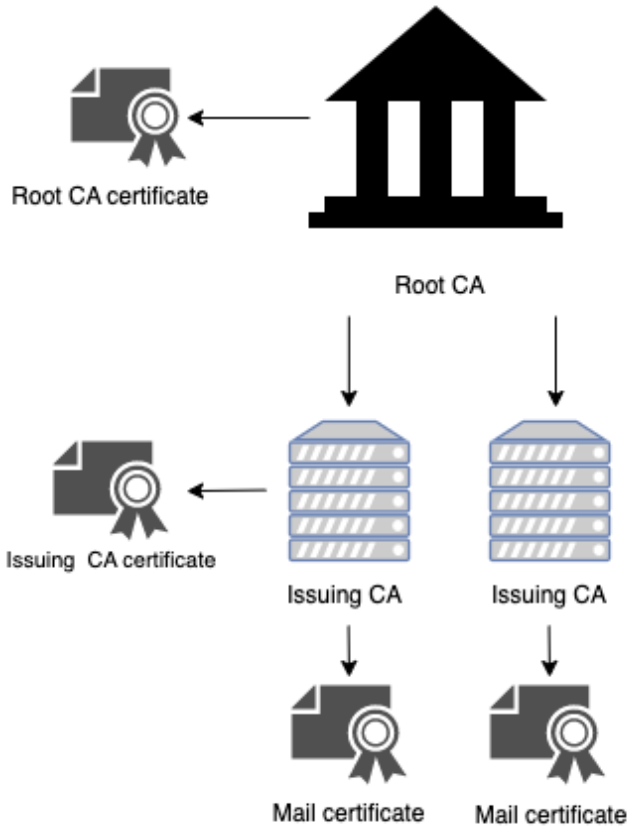


Figure 3.5 - 2-tier hierarchy.

The numbers of layers in a PKI depends on the: separation required, the level of administration that is possible and the complexity of the networks. To keep things simple and focus on a fully operational PKI, the choice was made to create a 1-tier hierarchy PKI, as illustrated on the figure 3.6. It means that the root-CA will be responsible for delivering certificates to end entities.

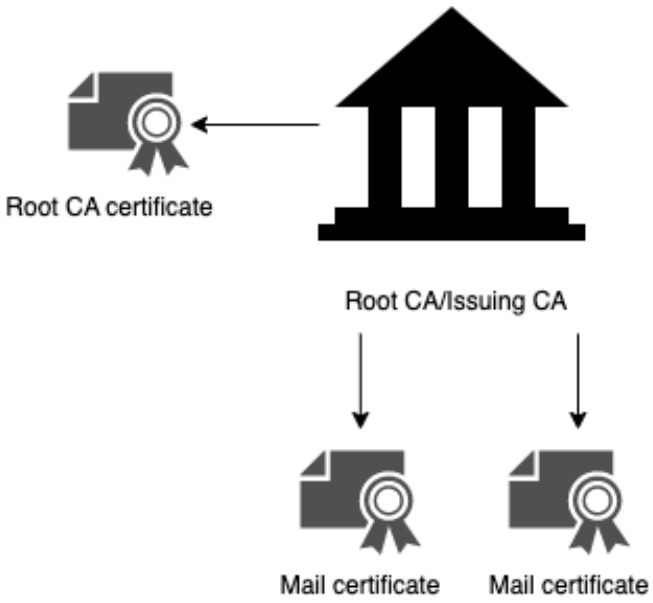


Figure 3.6 - 1-tier hierarchy.

3.1.10. Certificate path

When installing a certificate on an end entity, it will verify the collection of certificates that are trusted until it reaches the certificate on the top of the collection which is the root CA. Though, in the case of a private PKI, the CA certificate is installed on the end entity. The concept is about the end entity checking the certification path. Every time it checks a higher certificate it will also verify the CRL to control its expiry date and its configurations. Any failure in this process will result in an abandonment. If the process is a success, the end entity will have a full trust in the identity of the certificate chain. The picture 3.7 illustrates the process of a certificate creation.

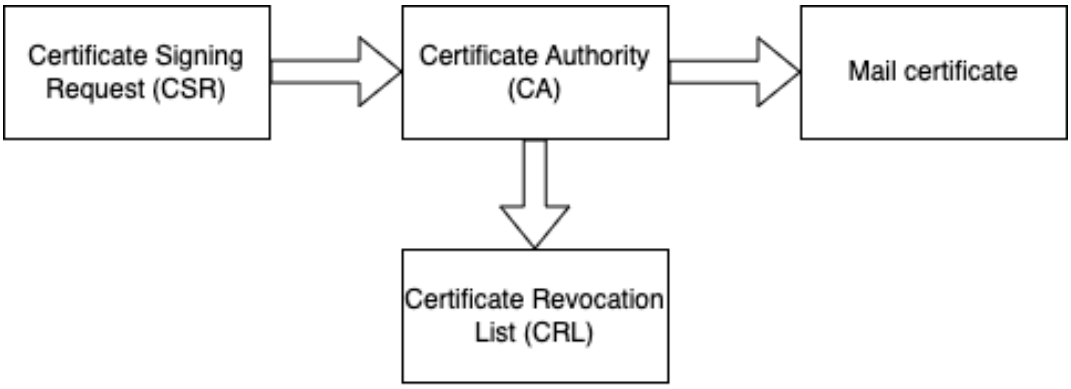


Figure 3.7 - Certificate creation process

3.1.11. PKI solutions

The first step is to choose the PKI technology. There are some solutions that are ready to use with minimal control over the process and low possibility of making modification however there are solutions that can be modified at will. Hence, a study was written. This one, compare the different solution. Indeed, the company needs to create its own CA and thus its own internal PKI: which brings advantages: complete control over the PKI, as well as drawbacks: limitation with external domains. Some Among the different criteria selected to choose which tool will be held, there are:

- The documentation
- The complexity
- The state of the ports 80/443
- The necessity of opening a web server

The table 3-2 illustrates the various technologies to set up a PKI along their criteria.

Table 3-2 - PKI solutions

Technology	Documentation	Web server	Port 80/443
OpenSSL	Good	No	Closed
Let's Encrypt	Good	In some case yes	Open
Stunnel	Average	No	Closed
startSSL	Average	No	Closed
EasyRSA	Weak	No	Closed

The documentation has been the most important aspect when selecting the solution. Indeed, when a problem arise it is always easier to solve it. It means various documents are available publicly explaining the content and principles of each command, but it can also signify that the common problems encountered can be described and solved.

The web server is another critical factor. As the security aspect needs to be always prioritized, opening a web server to receive certificates can be problematic. In the let's encrypt case, a web server may need to be configured and opened to external traffic to receive the certificates from the Let's Encrypt PKI [6]. Even though the communications with this web server are taking place during a very short amount of time, it needs to be configured with security in mind. Also, it can be considered as a doorway for hackers if this server is poorly secured.

Finally, opening the port 80 for Hypertext Transfer Protocol (HTTP) or 443 for Hypertext Transfer Protocol Secure (HTTPS) can present more security flaws for the company. In the case of Let's Encrypt, the machine that receives the certificate needs to open port 80 or 443. Though, it is only for a very short amount of time which considerably reduce the risk.

The chosen technology was OpenSSL. Mainly because it is a well-known solution in the industry therefore there is a wealth of documentation available also because it is open source. It means it is free to use and if a vulnerability is discovered it won't take long before a patch appears. OpenSSL also does not require the use of a web server and thus new ports don't need to be opened which reduce the attack surface.

3.1.12. OCSP server

When issuing a certificate, it needs to be registered somewhere. The administrators need to have a proof that the certificate was issued to someone. Indeed, a certificate can be revoked for different reasons. Among those reasons we have:

- the expiry dated passed
- the issuing CA has been compromised
- the private key linked to the certificate has been compromised
- the certificate owner no longer owns the domain
- the original certificate has been replaced with a new certificate

Thus, it is very important for a PKI to have a list of the current state of every certificate. It is a security feature that is indispensable. The former method employed by PKI to keep track of the certificates and implement security measures is known as CRL [7].

In this project, it is very important to keep track of every issued certificate, for instance if a malicious user had compromised a certificate and its keys, he could impersonate someone by sending email with the stolen certificate and thus obtain precious information.

When a client opens his email software or when he sends an email, the CRL is downloaded on his machine thanks to a HTTP GET request towards the supplied URL. The CRL is parsed and the certificate status for the client is checked. If the status is valid the client can continue otherwise the process is aborted. This method presents some advantages

but also drawbacks. Another method exists, the OCSP server, to replace the drawbacks related to CRL. The table 3-3 shows the differences between the CRL and OCSP methods:

Table 3-3 - Differences between CRL and OCSP

CRL	OCSP
Revocation status verification takes a lot of time for large list	Revocation status verification is faster than CRL
Can consume an important quantity of resource and thus have an impact on the network	Consume less resources, bandwidth than CRL
Does not provide a real time update	Provide a real time update

Nowadays, if security is a major concern for the PKI, CRL is deprecated, and it is replaced by OCSP [8]. Based on the advantages that OCSP brings, it is the technology that was implemented for this project. Because it is just a study in small local environment, the OpenSSL solution is conceivable. For a large enough environment, it should not be implemented. It would require a solution that can handle a bigger number of requests.

3.2. Microsoft Office 365 security

The purpose of this second mission is to improve the Microsoft Office 365 security score. Microsoft Office 365 gathers various products made by Microsoft. To begin the score improvement a typical company Microsoft Office 365 environment must be considered. To enhance the security score, some actions need to be completed thanks to the environment previously mentioned. This part will describe the theoretical concepts that are involved in order to complete the security actions.

3.2.1. Creating an Azure and Office 365 environment

According to Microsoft's official website: "Microsoft Office 365 is a cloud powered productivity platform. With a subscription to Office 365 the client can get the latest productivity applications such as Microsoft Teams, Excel, PowerPoint, Outlook, and more." [12]. Microsoft Office 365 is also linked to an Azure Active Directory (AD).

What is an AD? It is a directory service for Windows network environment. It is a distributed, hierarchical structure that allows for centralized management of an organization's resources, including users, computers, groups, network devices, file shares, group policies, devices, and trusts. AD provides authentication and authorization functions within a Windows domain environment. [3]

Microsoft AD was introduced with Windows 2000, with the rise of cloud technologies, Microsoft released the Azure AD. According to Microsoft official's website: "Azure AD takes this approach to the next level by providing organizations with an Identity as a Service (IDaaS) solution for all their applications across cloud and on-premises." [13]. It enables employees to access thousands of Software as a Service (SaaS) applications with an access management service. Many differences exist between the traditional AD and its modern version, the Azure AD.

3.2.2. Security score

When accessing the administrator console on the Microsoft Office 365 portal there is a section about the security score. Inside this section there is a list about security actions that must be realized in order to improve the security score. To take a look at this section, administrative rights must be granted. As some very sensitive information are displayed on this console (administrative rights, emails, passwords, locations, configurations, and more

…). Therefore, only a Comma-Separated Values (CSV) file from the Microsoft Office 365 console was provided. Inside this file there are various data about security actions. After the infrastructure was created, the security actions that could be completed without purchasing more licenses were selected. Every action does not have the same impact on the score. Inside the file, the actions could increase the score in a range from 0,43% to 4,33%. Nonetheless, it is important to consider the complexity of every security action. Even though some of them can increase the score by 0,43% they can be very complex to set up and require a lot of time whereas some security actions have a high impact on the score but can be configured quickly through the Microsoft web portal.

3.2.3. Security actions

This section is about the various security actions provided in the CSV file. It aims to to clarify the security principles. Different Microsoft products are concerned about those actions. They are all included in the Office 365 subscription. [22]

- Some actions concern the Azure AD, this product is defined in the section 5.11.
- Microsoft Teams is a communication application. It allows to: chat in real time, send files, set up meetings, start and participate to video a meeting.
- Microsoft SharePoint in Office 365 is a cloud-based service that allows employees of a company to create sites to share documents.
- Microsoft Exchange Online is a cloud messaging solution that delivers emails, calendars, and contacts.

Require multifactor authentication for administrators

In the Azure AD, accounts with high administrator privileges are prime targets for attackers. Indeed, if a malicious actor manages to take control of an account with a high administrator privilege, he could perform harmful actions on the infrastructure and also have access to confidential information. To reduce the risk of these accounts being manipulated by an unauthorised person, multi-factor authentication is used. For instance, multi-factor authentication could be the administrator login on the web portal, provides its credentials, then he receives a code on his phone, gives the code to the web application and then gets authenticated.

Set up multifactor authentication for Microsoft 365 (Ensure all users can complete multifactor authentication)

This time the goal is to implement multi-factor authentication for users. When logging into a Microsoft account, they should be able to authenticate at multiple levels.

A legacy authentication signifies that a client or network protocol is incapable or not configured to do modern authentication which provides more security and capabilities. For instance, a legacy authentication could be associated with a login and password authentication. Blocking legacy authentication drastically reduces the risk of exposure to password spray attacks and credential stuffing. Therefore, the goal of this action is to use a modern authentication for every user and for every Office 365 product, by creating a conditional access policy or by using a pre-defined one in Azure AD.

Do not expire passwords

Studies have shown that changing passwords often present more risks than benefits. Indeed, passwords that change frequently are generally weaker in terms of security. In addition, users tend to reuse certain passwords. The goal of this action is then to enforce users to choose a strong password that respect password security best practices and to keep this password as long as it is possible.

Do not allow Exchange Online calendar detail to be shared with external users

Users can exchange their calendars via Outlook to users inside and outside the organisation. The administrator decides whether users can share their calendars or not and with whom they can. If the “online calendars can be shared with external users” policy is active, malicious actors could gather information about the schedule of an employee.

Do not allow users to grant consent to unreliable applications

Before an application can access data on the computer, it must be authorised. There are different levels of permissions for different levels of access. All users can give access rights to an application without having administrator rights. The aim is to reduce the risk of a fraudulent application accessing data about the organisation. Depending on the role of a user, he should only use the applications he needs to work.

Designate more than one global admin

The person who purchases the subscription to Microsoft Office 365 is designated as the global administrator for the Azure AD and possesses all the rights. It means that this administrator can read and modify every setting in the Azure AD. To speed up processes, the global administrator needs to delegate some rights and he is the only one who can do it. When delegating rights to other administrators, some best security principles must be respected. For instance, global administrators and other administrators with high privileges should have strong passwords and multi-factor authentication enabled, principle of least privileges for administrators should be enforced, and logs generated by those accounts should be monitored carefully.

Use least privileged administrative roles

The purpose of this action is to restrict an administrator's rights as much as possible according to their role. Depending on his role as we have seen in the previous action, it is necessary to designate several administrators who have different levels of rights. An administrator should not have too many rights. The goal is that for each AD domain and each task, an administrator with the least amount of rights is designated. In order to minimise the number of administrators with too many rights.

Below, the picture 3.8, illustrates the Azure AD web console regarding the security actions that have been discussed. This only concerns the security actions linked to the AD. There are: the current security score, the actions name, the score impact, user impact, implementation cost displayed.

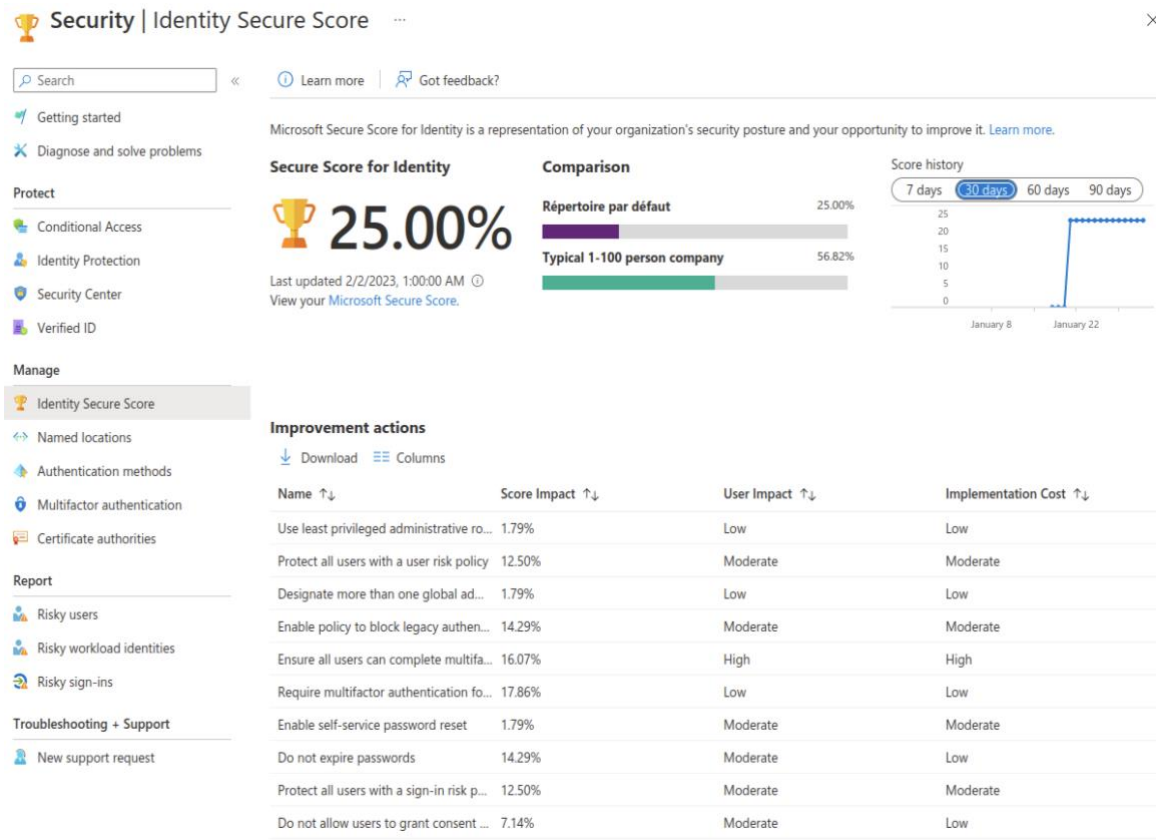


Figure 3.8 - Azure AD security console

Restrict anonymous users from starting teams

This action consists of removing the right of external users, i.e., users who are not authenticated on the Azure AD, to start a Teams meeting. To do this, it is required to modify the "meeting policies". These are used to control the actions that can be performed by participants in a meeting. A malicious user could start a Teams meeting and therefore use the company's subscription and resources.

Only invited users should be automatically admitted to Teams meetings

This action is pretty straightforward, as its name suggest the users invited for a meeting can be automatically admitted to Teams meeting. If a user is not invited to a meeting, he should firstly authenticate to the Azure AD, and then send a request to the meeting creator that must be accepted to let the user access the meeting. Otherwise, a malicious actor could try to attend a Teams meeting he is not invited in and get confidential information.

Limit external participants from having control in a Teams meeting

The concept of this action is to limit the rights of users that are categorized as external, or according to the Azure AD, "Guest". The Azure AD categorizes user type either as

"Member" or "Guest". A "Guest" user possesses an email address that has a different domain name from a typical "Member" email address. Therefore, it is easy to select external users with the web portal or a PowerShell script. Again, a malicious actor could join a Teams meeting and for instance turn on or off a microphone, kick out a user, show his screen, create a private meeting, and more.

Below, on the screenshot 3.9, it is possible to clearly differentiate a "Member" and a "Guest" user thanks to the "User Type" column.

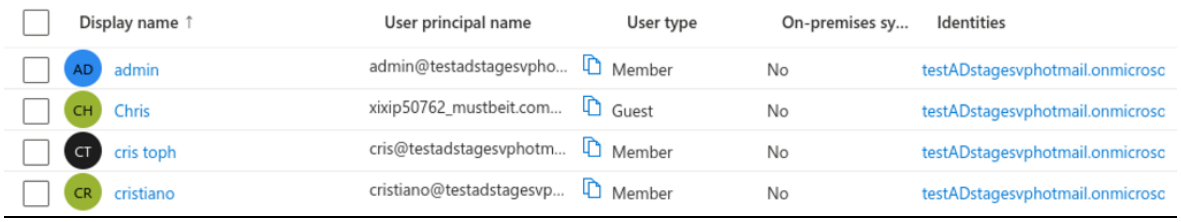


Figure 3.9 - Azure AD guest and member users.

Then the following rights can be limited:

- Who can present in the meeting
- Who can admit users to a meeting
- Who can chat in a meeting
- Who can react in a meeting
- Who can create a private meeting

Configure which users are allowed to present in Teams meetings

Some users attending a meeting should not be able to present. The rights of presentation should be distributed carefully to prevent any user from displaying unauthorized or confidential information to every other people attending the meeting. To do so, the Teams administrator can distribute three types of rights for presentation:

- "Organizers but users can override": only the user who created the meeting can present.
- "Everyone in the organization, but user can override": users that are participating in the meeting and identified can present.
- "Everyone, but user can override everyone in the meeting can present": this is the default value.

Restrict dial-in users from bypassing a meeting lobby

When a certain type of user tries to access a Teams meeting, he has to wait until an organizer or administrator accept him to join the meeting. This way adds an extra layer of security to control who joins the meeting. The organizer or administrator is sure that every participants in the meeting are here for a reason and it prevent unauthorized or malicious user to access a Teams meeting.

Sign out inactive users in SharePoint Online

This action allows to control user access to SharePoint data on unmanaged devices. It allows to keep the SharePoint data safe no matters where the user is located, the kind of device is working with, and the security level of the network connection used. It is possible to specify how long it will take for the users to receive a message to warn them of a future disconnection if they are still idle. Microsoft recommends sending the message after forty five minutes of inactivity, then the user has fifteen minutes to react if nothing is done the user's account is signed out.

3.2.4. Environment

Actions that can be found in the CSV file are related to four different Microsoft products: Azure AD, Exchange Online, Microsoft teams, SharePoint Online. Microsoft possesses both a web portal for Office 365 and Azure. Therefore, some actions can be done easily and quickly with administrator's privileges thanks to the web portals, nonetheless they can also be done thanks to other solution. As Microsoft already makes available a well-furnished documentation, one of the main tasks was to look for alternatives to the web portals. For instance, if the action of enabling multifactor authentication for every administrator in the company is taken, and a hundred administrators exist, this would be a very dauting task. Thus, the goal is to develop tools that will ease and make faster the work for administrators.

Because it concerns Windows technologies, the scripting language that comes by default is PowerShell. According to Windows's website "PowerShell is a cross-platform task automation solution made up of a command-line shell, a scripting language, and a configuration management framework. PowerShell runs on Windows, Linux, and macOS." [14]. Another major reason of choosing PowerShell for automating tasks is the various modules it has. Our scripts need to be executed to configure: an AD in the Azure cloud,

Microsoft Teams, Microsoft SharePoint online, and Microsoft Exchange online in the Office 365 cloud. PowerShell possesses a module for each of those products. The second row of the table 6 1 indicates the name of the module with the latest version number, the third row is about the command needed to install the module. The table 3-4 summarizes the various PowerShell modules and their commands to install them.

Table 3-4 - PowerShell modules for the Office 365 products

Azure Active Directory	Microsoft Teams	Exchange Online	SharePoint Online
Azure AD 2.0.2.140	MicrosoftTeams 4.9.3	ExchangeOnlineManagement 3.1.0	Microsoft.Online.SharePoint .PowerShell 16.023311.12000
Install-Module -Name Azure AD	Install-Module -Name MicrosoftTeams	Install-Module -Name ExchangeOnlineManagement	Install-Module -Name Microsoft.Online. SharePoint.PowerShell

Nonetheless, those modules come with slights disadvantages. They all need a specific PowerShell and Windows version to be executed. To begin the work of developing tools to configure the different Windows products mentioned before, a Windows environment is created thanks to a virtual machine. The OS version is Windows 10 professional. It is worth mentioning the OS version because some PowerShell modules can only be executed on Windows 10. Newer version such as Windows 11 or even prior ones before Windows 7 won't run the modules.

PowerShell is not the only option available to perform the configuration. We could use any other programming language. To do so, we can use Microsoft Graph API. A later part will detail the work done on this solution.

3.2.5. Concept of PowerShell modules

To ease the process of creating a script and to reduce the time of executing it, decision was made to create and run the script from a virtual machine with the help of Microsoft Visual Studio Integrated Development Environment (IDE). Indeed, the scripts need to perform configurations on the Azure cloud and not the local machine, somehow a connection must be made between the IDE used and the cloud. Otherwise, the scripts have to be written on the machine, uploaded on the Azure account and get executed thanks to the cloud shell console. It would take a lot of time just to execute the scripts and verify it works as expected.

This is where the PowerShell modules help. The process is simple, the PowerShell modules are imported, connection is made to the Azure account, the program is executed, and it is verified if it works as expected. The process is summarized on the picture 3.10.

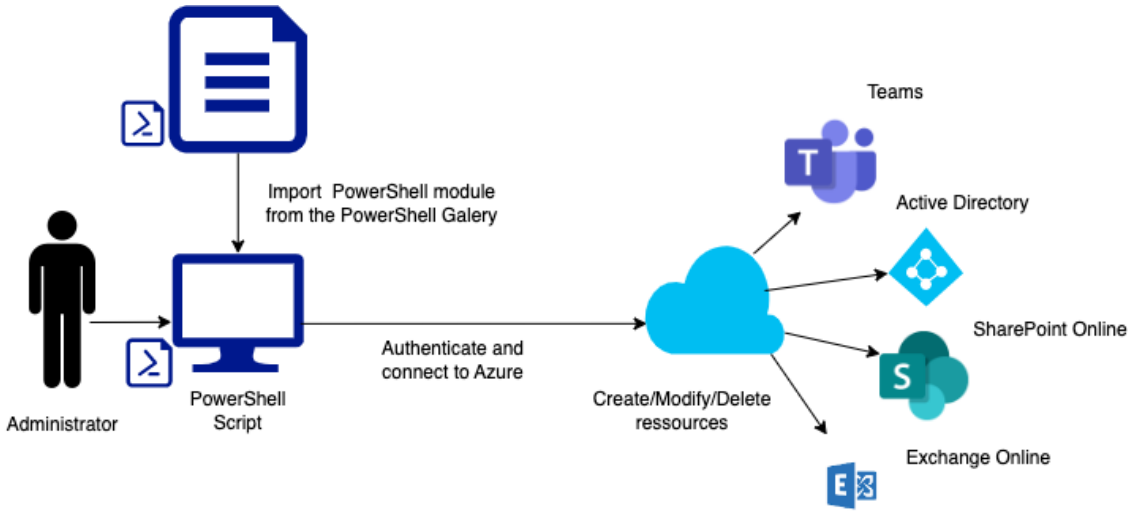


Figure 3.10 - Concept of PowerShell modules.

3.2.6. Microsoft Graph Application Programming Interface

In addition to the PowerShells scripts that allow to complete the security actions, an Application Programing Interface (API) to configure the Azure AD resources can be an alternative. Then an application could associate an administrative role to a user, or performs any other action, through the API. This part will describe the theory behind the Microsoft Graph API.

According to Microsoft's official website the Microsoft Graph API can be described as "a RESTful web API that enables access to Microsoft Cloud service resources" [16]. REST stands for "Representational State Transfer". It defines a set of rules about how applications and devices can connect to and communicate with each other. A REST API conforms to the design principles of the REST architectural style, for this reason, REST APIs are called RESTful APIs sometimes. REST APIs communicate thanks to HTTP requests to execute functions such as: creating, reading, updating, and deleting records within a resource. Microsoft Graph API supports the following HTTP requests: GET, POST, PATCH, PUT, DELETE. To secure as much as possible an API, authorization and authentication principles must be implemented. Authorization means that only a person who possesses the rights to interact with the API can do it, those rights are distributed through the Azure web portal. Authentication means that the person who wishes to interact with the

service must prove its identity. Various authentication mechanisms exist for APIs. Among those one, the application uses the Open Authorization 2.0. (OAuth 2.0) mechanism. This approach relies on a dedicated authentication server communicating with the API server to grant or deny access. To begin the process, the client sends an application-related key and a secret to the authentication server. The latter provides a token in response. Then the application sends the application's token inside a HTTP header. This token contains the authorization granted to the user as well as a time limit validity. The resource server (API) extracts the token and validates it with the authentication server. If the token is valid the client receives a valid response. Microsoft Graph API allows to create an application that relies on those authorization and authentication principles through the Azure web portal.

3.2.7. OWASP top 10 API

When this API was created the "OWASP API security top 10" [17] document has been consulted. This document is about strategies and solutions to understand and mitigates vulnerabilities linked to APIs. Open Web Application Security Project (OWASP) is a non-profit foundation that works to improve the security of software, they release every three or four years a state of the main vulnerabilities about web applications, mobile applications, and APIs. In the "OWASP API security top 10", the vulnerabilities are classified in order of danger. Not all the vulnerabilities were checked because of a lack of time, nonetheless the first one, which represent a greater danger, were highly considered.

4. Implementation

The part number four will explain how the solution provided were developed with a focus on technical concepts.

4.1. Email security

The company tries to make its internal communications as secure as possible. Setting up a PKI is a great solution to enforce emails encryption. We consider the scenario of a company that studies the differences between two encryption solutions for his email's communication, among which we have the S/MIME protocol. This protocol brings more security benefit than PGP, it allows to trust the identity of another user.

The main goal of this first mission is to simulate a PKI implementing the S/MIME protocol for emails security. The scenario will be developed and tested inside a VM network on one computer. Thus, the email server inside the company will remain stable if any problem arises.

The technologies choice for this first mission is free. Only the Operating Systems (OS) were imposed since this work will serve as a reference for a potential implementation. The servers will run under the Linux OS, the distribution is free to choose, the OS for the clients is also free to choose. Concerning the various protocols that will be used throughout the chain of sending and delivering an email, they must use Transport Layer Security/Secure Socket Layer (TLS/SSL) mechanisms. The last criteria that must be respected is the use of at least two different domains. The goal is to simulate an infrastructure that is a minimum realistic and therefore it contains various domains. Consequently, using one domain would underestimate the complexity of the current situation.

Finally, an open-source email software is used to test the solution. It is installed on the client machines. Because it is open source, it does not require any expenses and can be installed on any OS.

4.1.1. Network architecture

To begin this part, a basic email network will be detailed. The concept is to create a server-client infrastructure oriented for e-mail exchange. A virtualization software was used to create a VM network. The goal is to create a virtual network such as defined in the picture 4.1 Network infrastructure. To do so, various VM were downloaded with different OS. During the description of what the company is expecting, they need to rely on various domains. For simplicity purpose, two different mail domains will be created: @mail-test.com and @direction.org. Because the top-level domain is different, two email servers will be set up. Regarding the OS for the servers, Ubuntu was chosen. The reasons are it is one of the most used Linux distributions, thus the documentation is well furnished; it has an important community that keeps alive and regularly update the OS; also Gnome UI is available which makes some tasks easier. The figure 4.1 illustrates the infrastructure.

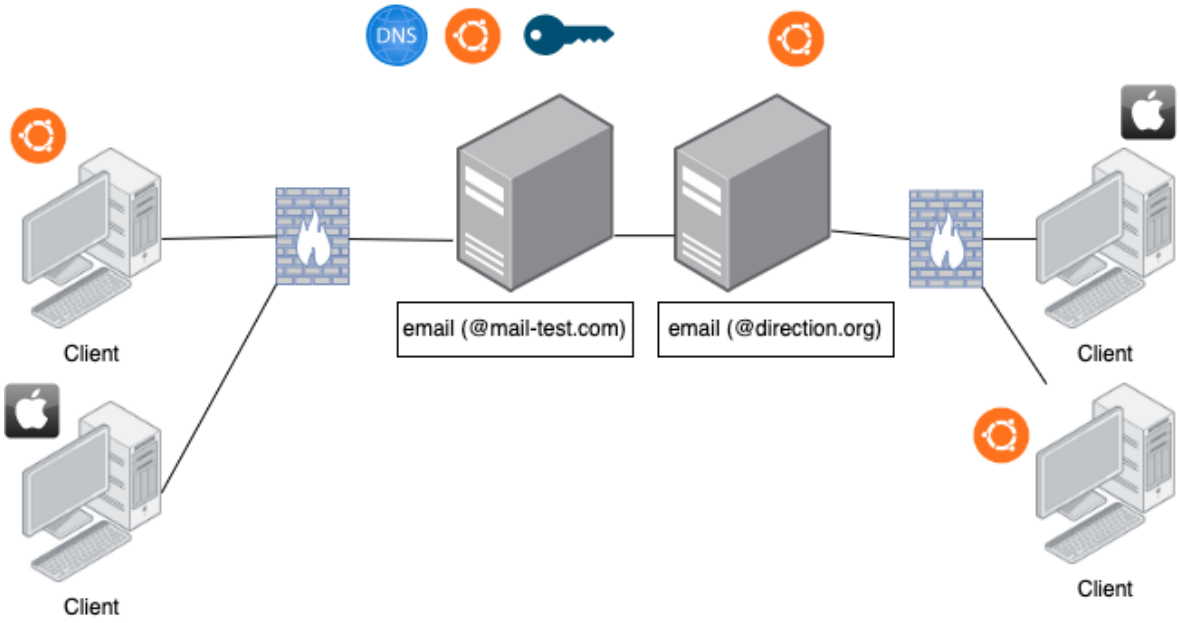


Figure 4.1 - Network infrastructure

On this diagram, there are two Ubuntu servers with different domain name. The first one, with the mail-test.com domain name, holds the PKI symbolized by the blue key, the DNS for this network and an email (SMTP and POP3) server as well. The second server with the direction.org domain name holds only an email (SMTP and POP3) server. At both extremities of those servers, there are firewall icons, which will actually be embedded into the Ubuntu servers. Finally, the clients are symbolized by desktop computers, some are running MacOS and others Ubuntu OS. The next step was to configure the DNS so VM can communicates with a hostname instead of an IP address.

4.1.2. Domain Name Services

The first thing to implement on both servers is the DNS service. Indeed, both servers will have a fixed IP address, which is inevitable. If IPs were changed, clients would not know where to forward the e-mail. To do so, “DNSmasq” was chosen. It is a lightweight and easy to configure DNS service and suits perfectly the need for small to medium networks. DNSmasq accepts DNS queries and either answer them from a small local cache or forward them to a recursive DNS server. It loads the content of the `/etc/hosts` directory, so that local host names which do not appear in the global DNS can be resolved. Moreover, it is available for every Linux distribution as well as macOS but not for Windows, thus meets the requirements for both client and server’s OS. DNSmasq is available in the Advanced Package Tool (APT) repository, therefore can be install with the `sudo apt-get install dnsmasq` command on the Linux servers. In case of a macOS client it can be installed through brew or other package managers. to install DNSmasq the command is: `brew install dnsmasq`.

Once the package is downloaded and ready to use, the first step before configuring the DNS service is to set a hostname which corresponds basically to the name of the machine. It can be done thanks to the command `sudo hostnamectl set-hostname servername`. By default, Ubuntu uses systemd-resolved as a network name resolution service because it comes with systemd which is already installed. Nonetheless this service can be handy to configure and because in this case the network architecture is not complex DNSmasq seems more appropriate.

The first step when configuring the DNS was to stop and disable systemd-resolved thanks to systemctl. Then the dnsmasq.conf file which is located in the `/etc/` folder needs to be edited. This file contains information about interfaces, domains, external server, local domains and more. The file hosts inside `/etc/` also need to be configured. This is the local DNS system which will only be consulted by the machine. It is stored in the cache memory, thus allows to perform fast query for mapping an IP to a domain and save resources instead of forwarding the whole query to the resolver. Finally, in order to use the DNS, we need to start and enable the service throughout systemctl. We can test our solution by pinging a domain as demonstrated on the figure 4.2.

```
--- direction.org ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.366/0.446/0.649/0.118 ms
francois@Mac-de-francois pki-multiple-domains % ping direction.org -t 5
PING direction.org (192.168.228.142): 56 data bytes
64 bytes from 192.168.228.142: icmp_seq=0 ttl=64 time=0.454 ms
64 bytes from 192.168.228.142: icmp_seq=1 ttl=64 time=0.706 ms
64 bytes from 192.168.228.142: icmp_seq=2 ttl=64 time=0.686 ms
64 bytes from 192.168.228.142: icmp_seq=3 ttl=64 time=0.665 ms
64 bytes from 192.168.228.142: icmp_seq=4 ttl=64 time=0.673 ms

--- direction.org ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.454/0.637/0.706/0.092 ms
francois@Mac-de-francois pki-multiple-domains %
```

Figure 4.2 – Ping the direction.org domain.

When configuring DNS services for email servers, it is important to consider that the Mail Exchanger (MX) record needs to be established. This record is linked to email servers. Because two email servers, with two domain names are present in this network, there is no choice but to precise the MX record. Of course, the Address (A) record, which is responsible for linking a domain name to an IP address; must be declared. The dig command allows to show the records for a domain as illustrated on the figure 4.3.

```
francois — francois@mail-test: ~/test2 — -zsh — 80×24

; <<>> DiG 9.10.6 <<>> direction.org -t MX
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 21009
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;direction.org.                 IN      MX

;; ANSWER SECTION:
direction.org.          0       IN      MX      10 direction.org.

;; ADDITIONAL SECTION:
direction.org.          0       IN      A       192.168.228.142

;; Query time: 1 msec
;; SERVER: 192.168.228.138#53(192.168.228.138)
;; WHEN: Tue Nov 22 08:13:57 PST 2022
;; MSG SIZE  rcvd: 87
```

Figure 4.3 - Dig command for direction.org domain.

4.1.3. MDA

Regarding the MDA that implements the SMTP protocol, I chose Postfix for the Ubuntu servers. Postfix is an open-source software built for delivering emails, it was built as faster, easier to administrate, and a more secure alternative than the former solution: Sendmail. Postfix can be installed thanks to the following command: `sudo apt install postfix`. Various configuration files come with the installation of Postfix. During this project, the main.conf file was the only one configured, it is located in the directory

`/etc/postfix/`. Indeed, the goal of this project is not to create a complex mail infrastructure but to set up a basic one with 2 different domain names and later set up a PKI.

The SMTPS protocol was implemented during this phase. It adds a security layer on top of the SMTP protocol, to do so it relies on the TLS protocol. On the picture number 4.4, there is the line "`250-STARTTLS`" which proves that the SMTPS protocol is working.

To test that Postfix was well implemented, I used both Telnet and Netcat tools. Telnet is a network protocol that provides commands for client/server communication, in this case it is used to test and troubleshoot an email server. Netcat is a command line utility that allows to open up Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) connections between two machines from any desired port, it is essential for network debugging or investigation. The figure 4.4 shows the result of the Telnet command.

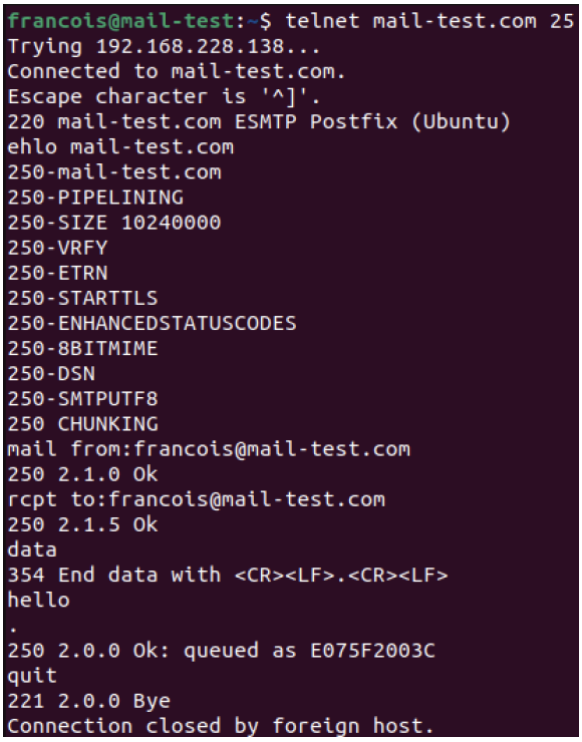


Figure 4.4 - Telnet command output.

We can see that our DNS is working when the Telnet command is used for the domain "mail-test.com", port 25 is the dedicated port for SMTP communications. Then the mail is being queued without any issue. The figure 4.5 shows the output of the command Netcat "nc".

```
francois@direction:~/pratice-csr$ nc -v mail-test.com 25
Connection to mail-test.com (192.168.228.138) 25 port [tcp/smtp] succeeded!
220 mail-test.com ESMTP Postfix (Ubuntu)
ehlo mail-test.com
250-mail-test.com
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250-SMTPUTF8
250 CHUNKING
mail from:francois@mail-test.com
250 2.1.0 Ok
rcpt to:francois@mail-test.com
250 2.1.5 Ok
data
354 End data with <CR><LF>.<CR><LF>
hello
.
250 2.0.0 Ok: queued as 0D34320247
quit
221 2.0.0 Bye
```

Figure 4.5 - Netcat command output.

After this protocol is configured, the mail can be safely sent to the email server. Then, the client needs to retrieve or download the mail to his computer.

4.1.4. MTA

The selected solution to implement the MTA is Dovecot. It can serve both as an IMAP and POP3 server for Unix OS. It was built with security in mind. Both SSL and TLS can be used.

To begin, the package must be installed thanks to apt: `sudo apt-get install dovecot-pop3d`. Dovecot must be edited to suit our needs. Its configuration files are available inside the `/etc/dovecot/` directory as it is a Linux OS. Editing dovecot configuration file is an easy task as only a couple of configurations must be declared: the protocols to downloads emails and the format.

In this case, both POP3 and POP3S are declared. POP3S is a version of POP3 with an additional layer of security. The data flow is processed by SSL protocols before adding the POP3 protocol. This way, it prevents some common attacks such as MiTM and satisfies some requirements like authentication, confidentiality, and integrity. Nonetheless, POP3 protocol will remain during the configuration phase because it makes the troubleshoot process easier. After this phase, it will be removed and only POP3S will be used.

The Maildir format is going to be implemented on the server. It is known for being fast and reliable. Maildir basically creates a directory architecture for every user. Moreover, it is supported by the email software used by clients. The figure 4.6 shows how Maildir is structured.

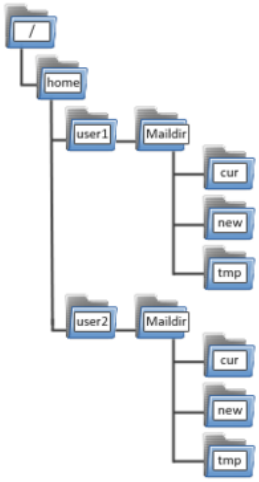


Figure 4.6 - Maildir's structure. [1]

Dovecot is a daemon; it needs to run continuously on our servers. It can be managed throughout the systemctl command. Every time some changes are being performed on the configuration files; the daemon needs to be restarted in order to take into accounts those changes.

To test if the MTA is working as expected, we use the same methods and tools defined earlier, Netcat and Telnet. The figure 4.7 shows the successful output from the Telnet command for the domain: mail-test.com associated with the port number 110.

```
francois@mail-test:~$ telnet mail-test.com 110
Trying 192.168.228.138...
Connected to mail-test.com.
Escape character is '^]'.
+OK Dovecot (Ubuntu) ready.
user market
+OK
pass 123
+OK Logged in.
quit
+OK Logging out.
Connection closed by foreign host.
```

Figure 4.7 - Telnet command output.

The figure 4.8 shows the output for the Netcat “nc” command for the domain: mail-test.com associated with the port number 110.

```
francois@mail-test:~$ nc -v mail-test.com 110
Connection to mail-test.com (192.168.228.138) 110 port [tcp/pop3] succeeded!
+OK Dovecot (Ubuntu) ready.
```

Figure 4.8 - Netcat command output.

4.1.5. Nmap script

Nmap stands for Network Mapper, it is a free and open-source utility for network discovery and security auditing. This tool was used to ensure that the services and their ports were working well. Because the client and the server ports needed to be scanned, a Python script was created. Nmap is easily installed on a Linux machine thanks to APT but on macOS various problems were encountered to install and run Nmap properly. Having a Python script allows to run it easily from any kind of machine as long as Python is available in the binaries.

When setting up the DNS and the email server, Nmap can be a useful tool to troubleshoot any problem related to DNS or email protocols. In this case the services are running on some well-known and defined ports. DNS is running on port number 53, SMTP is running on port number 25, and POP3 on port number 110. Thus, if the state of one those ports is different than “open”, it means something is wrong. Depending on the port number it is possible to know which service needs to be inspected. Python possesses a library named “Python-nmap” and can be installed with the Python package manager: `pip install Python-nmap`. After the package is installed, it can be used inside a Python script.

The concept of the script is simple. When it is launched, it requires an IP address, the one that is going to be scanned. This IP address format must be valid, it means that every number entered must be between 0 and 9 and each port number must be between 1 and 3 figures. Then the IP is checked, if it matches the criteria, it is processed for the scan, if it is not the case the program is stopped. After, comes the range, it is basically a list that will contains a range of port numbers to be scanned which is provided by the user. Same concept, if the range is not appropriate, the program will be stopped. After those parameters are correctly provided a `nmap.portscanner()` object is instantiated from the library. This one will allow to use methods from the nmap object and later perform the scan on the IP address and the port range provided.

On the figure 4.9, we have the proof that the port number 53 is open for the 192.168.228.138 IP address, which indicates that the DNS service is configured to accept UDP packets.

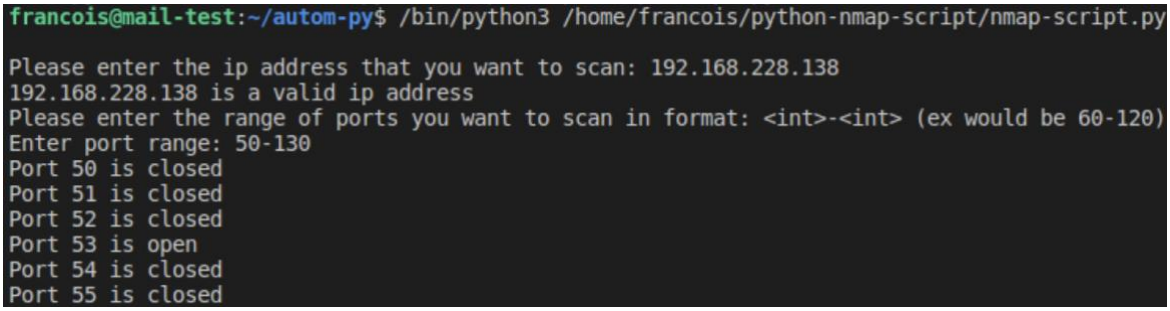


Figure 4.9 - Output of the nmap script.

4.1.6. Users and email addresses management

One of the advantages brought by Postfix and Dovecot is the management of various e-mail addresses. In order to create new email addresses, new users need to be created on the servers. This can be done thanks to simple commands such as `useradd -m` to create a new user along with its own directory (where /Maildir and email data will be stored) and `passwd` to create a password for the newly user created. Finally, clients just need to create an email account on the mail software and fill the username and password. To keep an eye on the users created, it is possible to give a look at the `/etc/passwd` path and check the first column of this files. The figure 4.10 shows the output of the `/etc/passwd` file. We can see our users: market, business, and direction.

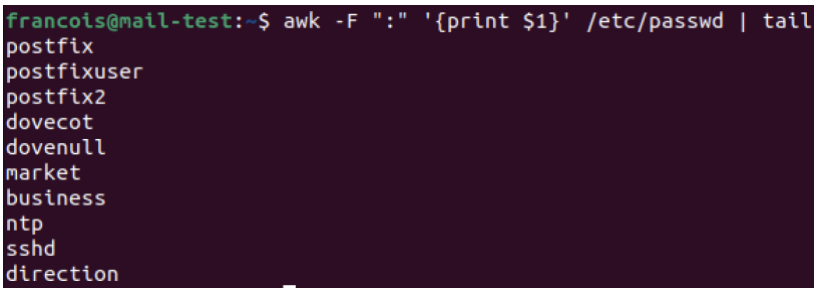


Figure 4.10 - Content of the /etc/passwd file.

Final test, as it is illustrated on the figure 4.11, is to send an email from two different email addresses that belong to different domains thanks to the mail software.

Figure 4.11 - Email test.

4.1.7. Aliases

To implement this concept, the MTA, Postfix, possesses a file under the `/etc/aliases` directory that needs to be completed. An example of how to fill this previous file would be: `production: prod-worker1@mail-test.com, prod-worker2@mail-test.com, prod-worker3@mail-test.com` . Then to update the alias and make it functional, the command `newaliases` must be executed. This command allows to initialize the alias database. The main.cf file under the `/etc/postfix` directory contains the direction to the `/etc/aliases` file and therefore they are connected. After the changes are done, the Postfix daemon needs to be restarted.

4.1.8. PKI implementation

After the email infrastructure is configured and working the PKI implementation can starts. The email infrastructure has two email servers. For simplicity's sake, one of those two servers will also be the PKI server. The first step is to create a CA. Indeed, all clients on the network require a certificate and the number is consequent. Having a CA provide benefits:

- Certificates are verified by a thrusted third-party.
- Centralized source for posting and obtaining information about certificates.

The CA contains a lot of data about the certificates. It maintains a database of every certificate that have been delivered. To setup a CA with OpenSSL, the first phase is to create the directory structure for some files. It is important to differentiate the root certificate from other certificates, but it is also important to keep track of the keys, the serial numbers. To define the following architecture, the recommendations edited by Ubuntu have been taken into account [3].

To begin, one central path needs to be defined, where every file related to the PKI will be stored: `/etc/ssl/`

Then, two directories are created:

- `/etc/ssl/CA/` this one contains the root CA certificate.

- `/etc/ssl/newcerts/` this one contains a record of every created certificates. Every files in this directory are stored with the ".pem" extension which signifies that the certificates are base64 encoded and delimited with the following lines : -----BEGIN CERTIFICATE----; -----END CERTIFICATE-----.

The root-CA needs to keep track of every certificate distributed and shared. To do so, various files, all of them located in the `/etc/ssl` path, are created:

- `/etc/ssl/CA/index.txt` is a text file that contains various information about the certificates issued. Whether they are valid or revoked, their serial numbers, and other information. It behaves as a database of every certificate. Though, it should not be confounded with the `/etc/ssl/newcerts/` path. The index.txt file does not store the content of a certificate; only minimal number of details are stored. It was made to keep a record of the number of series of the certificates that have been created.
- `/etc/ssl/CA/serial` contains the current number of the last certificate issued and is updated every time a new certificate is issued.

Two others important directories contain:

- the CA certificate with a ".pem" extension `/etc/ssl/certs/CAcert.pem`
- the private key of the CA also with a ".pem" extension, `/etc/ssl/private/cakey.pem`. The content of this path should possess strict rights. Only the administrator should be able to read or modify the content of this file.

All those paths constitute the base of the CA, they allow the administrator to store every file in an organized way and if it is needed to access a specific file, it can be done quickly.

4.1.9. Certificate creation

Once the CA structure is finished, the appropriate certificates must be attached to the CA. Because it is a one-tier private PKI, a self-signed root certificate is needed. To create one, the following OpenSSL command is issued: `openssl req -new -x509 -extensions v3_ca -keyout cakey.pem -out cacert.pem -days 3650`

Some key details are in this command. The req flag indicates it is creating a CSR, then the X509 format is specified. X509 is an international standard which defines the format of public key certificates it also enhances the interoperability of digital certificates. It binds an entity to a public key using a digital signature. Then, the other flags specify: the v3 version,

the name of the private key, the name of the certificate, the extension version, and the numbers of days the certificate is valid. After the root CA possesses its own certificate, the generation of certificates can start.

To begin the generation of an end entity certificate, a CSR must be issued first. The process of generating a certificate that is trusted by the root CA involves a few steps. The first step is to create both a private and a public key pair. OpenSSL allows to do so thanks to the following command: `openssl genrsa -out server.key 2048`

If for some reasons the public key needs to be extracted, the following command makes it possible: `openssl rsa -in server.key -pubout -out file_name.pem`

Then, `genrsa` command allows to generate an RSA, public and private key pair, which is an asymmetric encryption algorithm, the out command specifies the format of the output file and 2048 is the size of the private key in bits. To enhance security, a passphrase could be added for the server.key file. Every time this one is opened; the passphrase should be entered. To do so, the last command can be modified such as**:** `openssl genrsa -aes256 -out server.key 2048`. The -aes256 flag specifies with what encryption algorithm the passphrase would be encrypted.

The second step is to create a CSR based on the public key: `openssl req -new -key server.key -out server.csr`

The req command from OpenSSL generates a certificate request in PKCS#10 format. A PKCS#10 format consists basically of three main parts: the certification request information, a signature algorithm identifier, and a digital signature on the certification request information. The key used is the public key from the rootCA server. The output is as ".csr" file which contains information about the server and the company hosting it.

The chain of trust related to this PKI is very simple because it is a 1-tier PKI. The root CA's mission is to validate and sign the end-entity's certificate, the end-entity is responsible for verifying the origin of the certificate, if it comes from a trusted source when verifying the chain of trust the certificate will be issued and trusted. Once the CSR is obtained, it must

be sent to the root CA for processing: `sudo openssl ca -in server.csr -config /etc/ssl/openssl.cnf`

The openssl.cnf file is a configuration file that contains all the key components of the CA. All those components make the "identity" of the CA. The CA will compare the details entered in the CSR step with the configurations inside the openssl.conf file. If the criteria match, the CA will issue the certificate. Among the criteria that are in the openssl.conf file that can be specified, there are: the domain name, the country, the city, the email address, the company's name, and more. This kind of security allows the administrator to create a certificate that follows rules otherwise a fake certificate could be created therefore the openssl.conf file is a crucial file which only the administrator should read and open. Those rules can be changed to meet the administrator's requirements. Among the output from the last command, only the content between the line "-----BEGIN CERTIFICATE-----" and "-----END CERTIFICATE----" must be saved to a file with the ".crt" extension. The final requirement to use the certificate created for email purpose is to transform it into a PKCS#12 format. This format bundles the certificate with its private key in a single cyphered file. It is mandatory for a user to possess his private key as well as the certificate. During an email exchange the private key will decipher the mail. To do so, the following OpenSSL command is used: `openssl pkcs12 -export -name "name" -inkey server.key -in cert.crt -out cert.pk12`

Thanks to this command the private key is by default encrypted to protect from any theft.

Throughout this process, we began by creating and associating a root self-signed certificate for the CA. To summarize, the following steps are performed on the CA server:

- Generate a pair of encryption keys
- Generate a CSR
- Compare the CSR with the configurations inside the `/etc/ssl/openssl.cnf` file and generate the certificate
- Transform the obtained certificate into PKCS#12 format

The result is a valid certificate that can be used for email purpose. Every time a new certificate will be needed, it will follow those steps.

4.1.10. Automating the process by scripting

To ease the task of generating a certificate and speed up the process for an administrator, various Bash scripts were created. For instance, generating a script involves various commands. Thanks to a script, only one command needs to be executed. The administrator will type: `./certificate_generation.sh` and a new certificate will be created.

The first script repeats the steps described above. It generates a certificate request and translate into a mail certificate trusted by the CA. Nonetheless, some enhancements were brought. It starts by checking that the packages required are installed if it is not the case it installs them as illustrated on the figure 4.12 and the appendix A.

```
REQUIRED_PKG1="figlet"
REQUIRED_PKG2="openssl"
PKG_OK1=$(dpkg-query -W --showformat='${Status}\n' $REQUIRED_PKG1|grep "Package already installed")
PKG_OK2=$(dpkg-query -W --showformat='${Status}\n' $REQUIRED_PKG2|grep "Package already installed")

echo Checking for $REQUIRED_PKG1: $PKG_OK1 && echo Checking for $REQUIRED_PKG2: $PKG_OK2
if [ "" = "$PKG_OK1" ] && [ "" = "$PKG_OK2" ] ; then
  echo "No $REQUIRED_PKG1. Setting up $REQUIRED_PKG1."
  echo "No $REQUIRED_PKG2. Setting up $REQUIRED_PKG2."
  sudo apt-get --yes install $REQUIRED_PKG1> /dev/null 2>/dev/null
  sudo apt-get --yes install $REQUIRED_PKG2> /dev/null 2>/dev/null
```

Figure 4.12 - Package verification

Then, the script splits into three different parts the process of generating a certificate. It generates the CSR first as illustrated on the figure 4.13 and the appendix A.

```
generate_csr(){
    openssl genrsa -des3 -out server.key 2048
    echo -e "${RED} RSA keys generated successfully ${NC}"
    openssl rsa -in server.key -out server.key.insecure
    mv server.key server.key.secure
    mv server.key.insecure server.key
    openssl req -new -key server.key -out server.csr
    echo -e "${RED}CSR generated sucessfully ${NC}"
}
generate_csr
```

Figure 4.13 - CSR generation

Then it signs the certificate according to the specifications available inside the configuration file as illustrated on the figure 4.14 and the appendix A.

```
sign_certificate(){
    sudo openssl ca -in server.csr -config /etc/ssl/openssl.cnf> your-cert.pem
    if [ $? -eq 0 ];
    then
       echo -e "${RED} certificate signed sucessfully ${NC}"
    else
      echo -e "${RED} something went wrong. Please check the output above ${NC}"
    fi
}
sign certificate
```

Figure 4.14 - Certificate signing

Finally, it transforms the certificate created into a PKCS#12 file. This script also creates a path to stock the various PKCS#12 certificate created with the command `sudo mkdir -p /etc/ssl/pkcs12` as illustrated on the figure 4.15 and the appendix A.

```
get_certificate(){
    sudo awk '/-----BEGIN CERTIFICATE-----/,/-----END CERTIFICATE-----/ { print $0 }' your-cert.pem > your-cert.crt
    echo -e "${RED} certificate file created ${NC}"
    sudo mkdir -p /etc/ssl/pkcs12
    echo "donner le nom du pkcs12 et pas oublier .p12 a la fin"
    read pkcs12_name
    openssl pkcs12 -export -name "ce nom" -inkey server.key -in your-cert.crt -out $pkcs12_name  # peut être ajouter
```

Figure 4.15 - PKCS#12 obtention

The second script's purpose is to mark a certificate as revoked. Some steps are similar to the previous script. It checks if the required packages are installed and if it is not the case it installs them. Then a simple command makes the chosen certificate unusable and marks it as revoked inside the database as illustrated on the picture 4.16 and the appendix B.

```
delete_certificate(){
    echo "please, enter a certificate name as displayed in the /etc/ssl/newcerts/ path"
    echo "it should be number.pem format. For example : 01.pem"
    read certif_name
    sudo openssl ca -config /etc/ssl/openssl.cnf -revoke /etc/ssl/newcerts/$certif_name -crl_reason superseded
```

Figure 4.16 - Certificate deletion

The last script allows to automatically import a certificate inside the email software. It gets the certificate from the PKI server, transfers it to client machine and pushes it directly inside the software, so it is ready to use.

4.1.11. OCSP server

The following command allows to start an OCSP server that takes into account the index.txt file which contains various information about the certificates, it is described as a database for every certificate. It is linked to the 8080 port, and it is listening for any IP address that will make a request. This command was issued on the same server that contains the PKI. The figure 4.17 illustrates the command to start the OCSP server.

```
francois@test-ocsp:~$ openssl ocsp -index /home/francois/demoCA/index.txt -port 8080
ACCEPT 0.0.0.0:8080 PID=38278
ocsp: waiting for OCSP client connections...
ocsp: Received request, 1st line: POST / HTTP/1.0
ocsp: Received request, 1st line: POST / HTTP/1.0
```

Figure 4.17 - Command to start the OCSP server.

This server only accepts POST request as it is more secure. Indeed, the client make a POST request towards the server with his certificate information then the server performs some operation to verify if the certificate is valid or not and send a reply to the client. This way, it prevents the client from downloading index.txt file that contains sensitive information. The figure 4.18 is an example of a response to a client with a revoked certificate.

```
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: C = FR, ST = gironde, O = the mail company, CN = the mail service
    Produced At: Nov 30 10:49:51 2022 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 3B9E0A1F21FBEBE8A9CFD51B191A89FB33D815CA
      Issuer Key Hash: F40286209E34CF50C26D8411BD4E656C6757C5D5
      Serial Number: 0C
    Cert Status: revoked
    Revocation Time: Nov 30 10:48:59 2022 GMT
    This Update: Nov 30 10:49:51 2022 GMT
```

Figure 4.18 - Revoked certificate for a client.

We can see the Cert Status as “revoked”.

4.1.12. Debugging Logs

Debugging has been a very important and time-consuming task during this project. The logs under the `/var/log` directories were extremely useful. Mainly the “auth.log” and “mail.log” were of interest. The auth.log file contains the authentication logs for both successful and failed logins. The mail.log file contains the logs related to email services running on the server. Example of an authentication failure when a user is trying to connect to the email software with his account is illustrated on picture 4.19.

```
mail-test auth: pam_unix(dovecot:auth): check pass; user unknown
mail-test auth: pam_unix(dovecot:auth): authentication failure; logname= uid=0 euid=0 tty=dovecot ruser=engi@mail-test.com rhost=192.168.228.138
```

Figure 4.19 - Authentication failure logs.

As the software doesn’t return explicit nor useful indicators, the auth.log show the true nature of the problem. The picture 4.19 indicates that we cannot send nor receive an email due to an authentication problem.

With a further investigation in the mail.log we learn that during the creation of a user account, some rights for writing a file were missing. Indeed, only the administrator of the email server must have the right to add a new user. It could also indicate that when creating a new user, the administrator forgot to add the -m flag which allows to create a home directory for a newly created user. This home directory will contain the `/Maildir` directory. The screenshot 4.20 shows a permission problem with the `/home/engi/Maildir` path.

Figure 4.20 - Mail logs.

4.1.13. Challenges and enhancements

Using the TShark command line allowed me to analyze the traffic. According to its man page, TShsark is described as "a network protocol analyser. It allows to capture packet data from a live network or read packets from a previously saved capture file" [4].

```
Source Address: 104.123.50.19
    Destination Address: 192.168.228.144
Transmission Control Protocol, Src Port: 80, Dst Port: 47228, Seq: 1,
Ack: 422, Len: 730
    Source Port: 80
    Destination Port: 47228
```

The frame above gives us hints about the source and the destination IPs and ports. The IP 192.168.228.144 is our VM's IP address and 104.123.50.19 is the OCSP server's address. As the source port number is 80, we know it is a HTTP request.

```
[HTTP response 1/1]
    [Time since request: 0.110834599 seconds]
    [Request in frame: 25]
    [Request URI: http://e1.o.lencr.org/]
    File Data: 344 bytes
Online Certificate Status Protocol
    responseStatus: successful (0)
```

This second frame above from the traffic capture gives us more information. We know that by default the email software is forwarding its OCSP request to the following URI: http://el.o.lencr.org/ which points to Let's Encrypt company. According to Let's Encrypt's website it is described as: "a free, automated, and open certificate authority (CA), that runs

for the public benefit. It is a service provided by the Internet Security research Group (ISRG)" [5]. Their main goal is to provide HTTPS (SSL/TLS) for website for free.

In the picture 4.18 the OCSP response shows that the certificate hashing algorithm is SHA1. The NIST recommends using at least SHA-256 as SHA-1 is no more considered as secure. To do so, the administrators only need to replace the sha1 by sha 256 in this line: `ess_cert_id_alg= sha1.` The part 4.1.14 explains the documentation process.

Inside the architecture we have two Linux servers. An attacker should focus on those servers because they contain emails and other sensitive information. Thus, they present more risks than the client's machines. It is important to secure as much as possible those servers. Various security measures were implemented, the first one is about keeping the OS and packages installed up to date. It can be done with a simple command such as: `sudo apt update && apt dist-upgrade`

The second one is to keep only necessary ports open and restrict the traffic. It can be done by using a Linux incorporated firewall. In this case UFW was used. For example, the Telnet protocol was used for email testing purpose. This protocol must be blocked. Telnet is known for being insecure. It can be done thanks to the following command: `sudo ufw deny 23`

Telnet's port is a well-known port that is linked to the port number 23. The keyword "deny" allows to set a new rule that blocks any connection to the port 23. Of course, if the administrator needs to use Telnet to troubleshoot any problem related it can be done thanks to the following command: `sudo ufw allow 23`

4.1.14. Documentation

Whilst I have been working on this first mission, various documentations have been written. The goals were to write reports about subjects related to this first mission. It aims to ease the understanding of some theoretical concepts. The main purpose of this mission is to create a working solution that later will be implemented in the company therefore if some theoretical concepts are not clear it needs to be documented for the administrators that will work on this solution. The first one is related to a state of the art of the PKI technologies. I have made research about open-source tools that are able to build a private PKI and

concluded why I picked OpenSSL over other solutions. Research was made about the open-sources tools that exist, then every found tools were classified according to criteria such as the ones defined in the part 3.1.11: documentation available, necessity to create a web server and to open the ports 80/443. To write this first report, documentation such as [6], [7], [8] [9] [10], were all carefully taken into account to describe the strengths and the weaknesses of each one.

Another important documentation written is about good security practices when it comes to creating a certificate and a PKI. It includes details such as: how long should be the validity period of a certificate, why the private key of any CA must remain uncompromised, the cryptographic algorithm that should be use and the ones to avoid, the recommended key length, the certificate management strategy to implement and more. This documentation was inspired by the "Securing Web Transaction: TLS Server Certificate Management" edited by the National Institute of Standard and Technologies (NIST)[11]. NIST is a federal organization from the United States of America that develop cybersecurity standards and guidance for security systems. To establish rules for certificate management both the NIST recommendations and the specificities of TEHTRIS were considered. For instance, to determine the time validity of a certificate the NIST recommends one year or less, with the network architecture complexity (number of endpoints, number of servers) in mind it was decided to allocate ten months validity for every certificates. Indeed, from a security point of view the best practice is to have a short validity period for certificates nonetheless if a company needs to renew ten or a thousand certificates then the process is more complex and time consuming. In this document made by NIST, there are also recommendations about the key length and the signing algorithm, in this case NIST recommend using a minimum of 2048 key length, an approved signing algorithm such as RSA and an approved hashing algorithm such as SHA-256. This is why the CSR command uses RSA and a 2048 key length: `openssl genrsa -out server.key 2048`. Then, the rest of other documentations are mainly about how I have created the PKI, the goal is to explain carefully step by step how I made it work. For example, every script and every function or classes inside those ones were explained and detailed as much as it could be. The final objective is to ease the deployment of this project.

4.2.Microsoft Office 365 security

This second mission began by subscribing to a Microsoft Azure and Microsoft Office 365 account. Therefore, the goal was to recreate a company's infrastructure. An AD inside Azure was created with a couple of different users, applications, administrators, groups, and every aspect that it involves. The goal is to recreate as much as possible a typical company's AD. This screenshot of the Azure AD web portal shows the various internal and external users created. To create a user, the process is very simple as it only requires to click on the "New user" button and fill information such as the name, user principal name, user type, and a password. The figure 4.21 shows some users created.

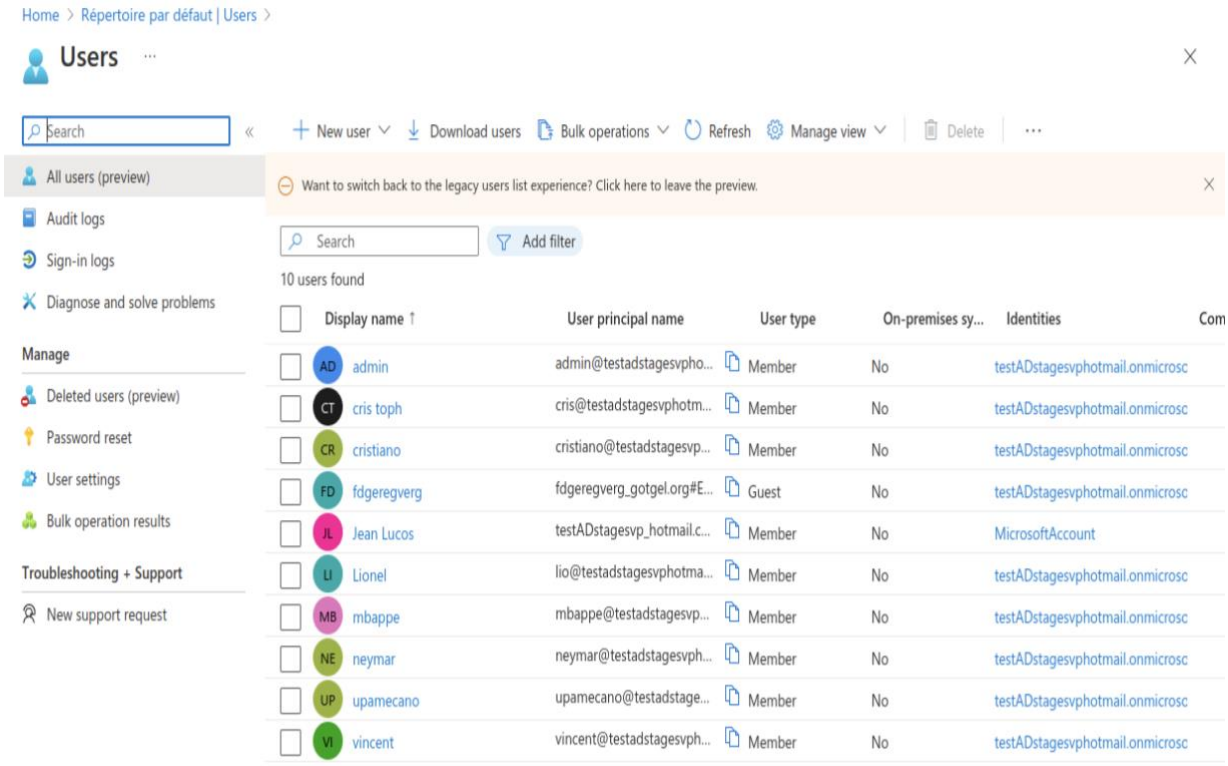


Figure 4.21 - Azure AD users.

Same process for the group. This screenshot 4.22 shows that a few groups were created inside the Azure AD for testing purpose.

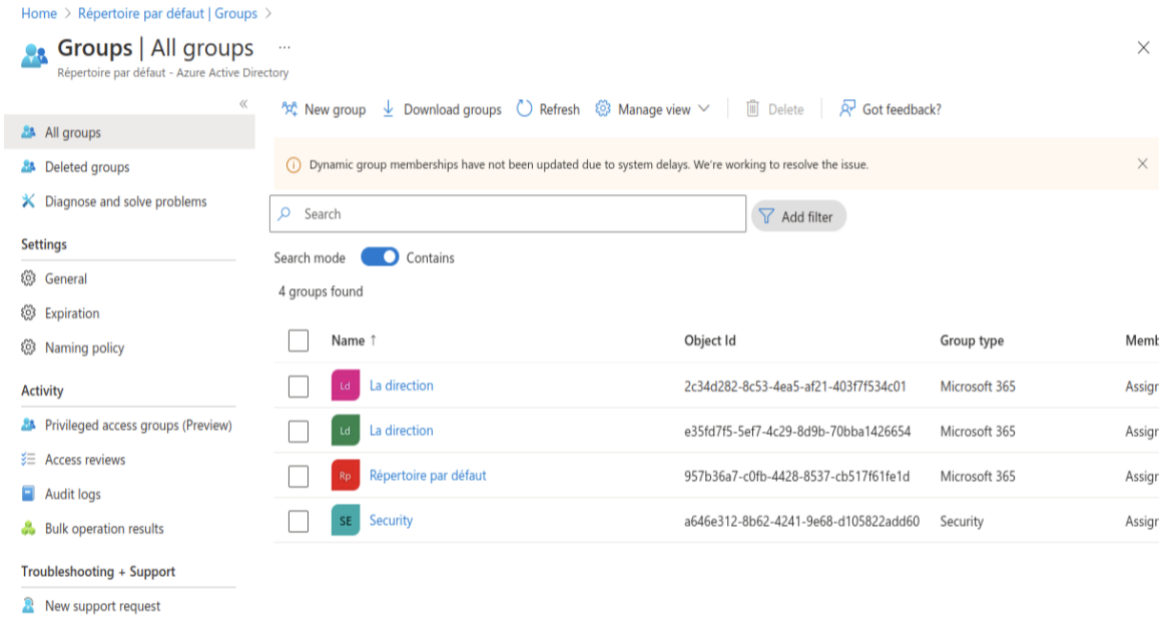


Figure 4.22 - Azure AD groups.

When it comes to the screenshot 4.23, it shows the applications that were deployed for users. Those applications were chosen because they did not require any additional payment, independently of the name and the role of the application, they were only chosen for testing purpose.

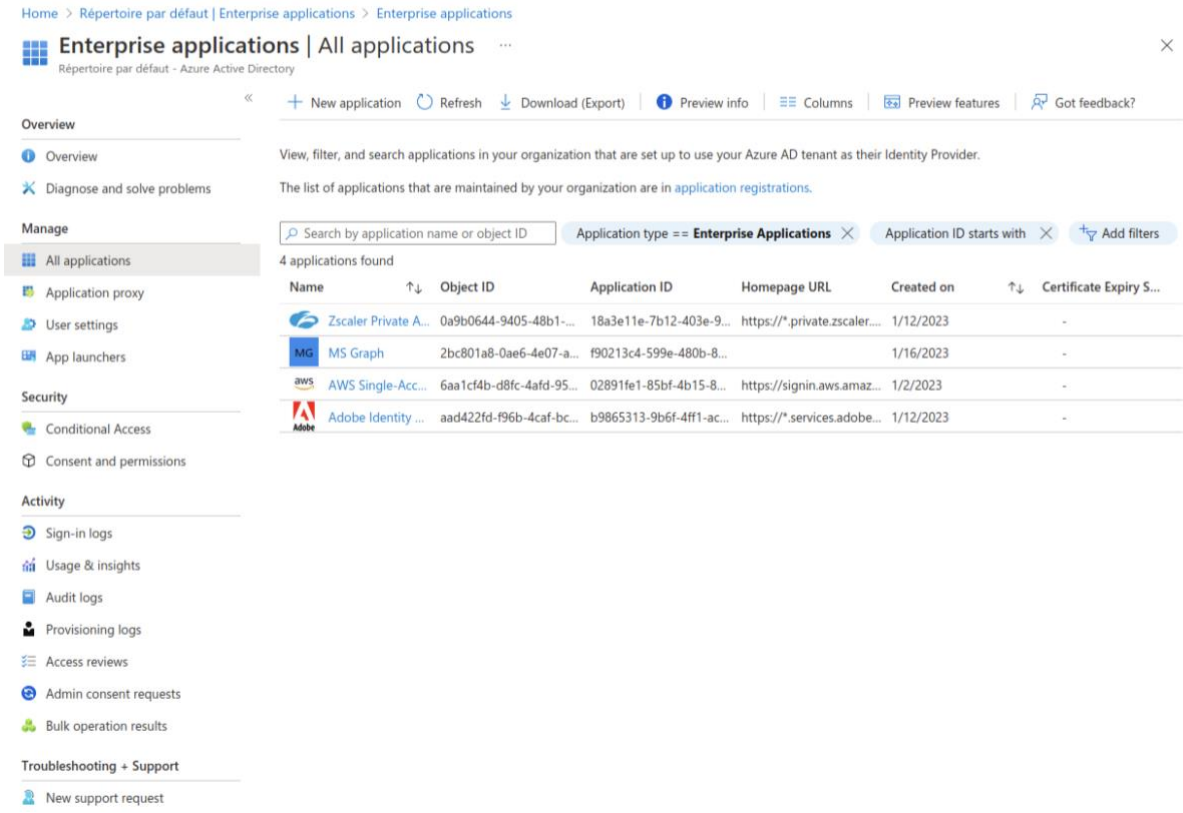


Figure 4.23 - Azure AD applications.

By creating users, group of users and applications that users can access the goal is to recreate a small company infrastructure. The users and the groups are the employees, the applications available are the SaaS and the AD is the access management service. Once this step is done, it is possible to get the security actions in order to improve the security score related to the infrastructure previously created. It can be downloaded from the Azure web console and obtained as a CSV file.

4.2.1. Sort data

Firstly, a CSV file, is messy and needs to be cleaned to have a decent visualization. As its name suggest, every value is separated by a comma and therefore its needs to be rearranged. The data must be separated into columns, they must be separated with a space. Some other features can be added, the actions can be sorted in decreasing order according to the score. We also need to track the date at which every action is completed so we can have an idea of how much time it needs to be done. To complete all the above features, a Python script was realized.

Some of the key features enabled by this script:

- Open the CSV file and cleans data
- Display data by action score in decreasing order
- Track actions completed by asking the user
- Verify if user's input is valid, if not raise an error
- Add the date of the day when the action was completed

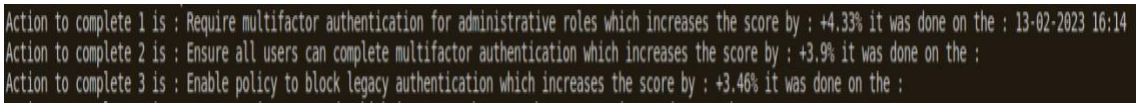


Figure 4.24 – Extract of the Python script output for the CSV file.

The screenshot 4.24 shows the results produced by the script. This Python script takes advantage of the CSV library which is available thanks to Python's package manager library. It contains methods and classes which allow to sort data from a CSV quickly and correctly. This script was not mandatory as any other software (such as excel for instance), could open the CSV file and provide a decent visualization. Though, the main advantages brought by this script is to display the actions by decreasing order, thus the administrator working on this task knows what actions he should give priority on. It can also be enhanced, somebody can bring new features, hence it automates the way of working with CSV file.

4.2.2. PowerShell script

A PowerShell script was delivered for all security actions present inside the CSV file. The concept was the same for every action:

- connect to the service name provided in the module
- print what his is purpose
- ask the administrator to choose between various options
- perform the chosen task and inform the administrator if it succeeded

This way of executing the script presents more advantages from a security point of view but also when running the program, the administrator knows what he is doing. For instance, the process of executing the PowerShell script to fill the "Use least privileged administrative roles" action will be described in the part 4.2.4.

4.2.3. PowerShell modules

Inside the scripts, we will find every time the same first lines, the `Import-Module` which, as its name suggest, import the desired module and the `Connect` keyword which creates the connection with the desired product. In the case of creating a script that will set up configurations for the Azure AD, the two lines on the picture 4.25 are required.

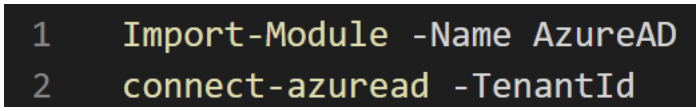


Figure 4.25 - Import and connect PowerShell

This process will be the same every time configurations need to be done. Only the name of the module will change. For instance, if it connects to Microsoft Teams, it will look like: `Import-Module -Name MicrosoftTeams Connect-MicrosoftTeams -TenantId`.

However, some steps must be performed before using a PowerShell module inside a script:

- Must be executed inside a Windows environment (Windows 10 VM)
- The module must be installed thanks to the PowerShell packages management: `Install-Module ModuleName`
- The PowerShell script execution policy must be set to remote signed: `Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser`

Once every of those steps are performed, it is possible to create and execute a script by taking advantage of the benefits of a module. According to Microsoft's website a module is described as "a package that contains PowerShell members, such as cmdlets, providers, functions, workflows, variables, and aliases" [15]. A cmdlet which stands for command-let is a PowerShell command that execute a small script which performs a single specific function. For instance, `Get-Azure ADUser`, is a command-let from the Azure AD module that allows to get every user present in the Azure AD.

4.2.4. Use least privileged administrative roles

An official documentation made by Microsoft, identifies all the administrative roles and the privileges that they should have. To enhance the security, the principle is to restrict a user's administrator permissions by assigning least privileged roles in Azure AD. Depending on the size of a company, but if the number of employees is important, configuring administrative roles in the Azure Active Directory can be a very long task. Therefore, the aim of this script is to simplify the process for an Azure AD global administrator. The script starts by explaining its concept to the user by printing lines in the terminal. Then the user has the choice between 3 options as illustrated on the picture 4.26 and the appendix C.

```
function Show-Menu {
    param (
        [string]$title = "L'option 1 permet de remplir le role d'un utilisateur alors que la 2eme permet
    )
    Clear-Host
    Write-Host " `n  $title "
    Write-Host " `n Entrez 1 pour remplir le role d'un utilisateur"
    Write-Host " `n Entrez 2 pour remplir plusieurs utilisateurs"
    Write-Host " `n Entrez q pour quitter le programme"
}
```

Figure 4.26 - User can choose between three options

The first one asks the user to provide the name of an administrator inside the Azure AD and an administrative role according to Microsoft's least privilege policy. This policy suggests assigning an administrative role with the least privileges to a user. It allows the user to simply enter a name and an administrative role and associate it throughout the terminal. This part of the script contains error checking. In case a wrong name is entered in the terminal, the script prints an error to notify the user that the name entered does not exist in the AD. It works the same way for administrative roles. When the user selects an administrative role to associate to a user the role is verified. If this role indeed exists, the process can go on. Then the program checks that the newly added administrative role to the

user does not already exist. If it is the case, an error is raised in the terminal, indicating that this role is already linked to the user. If the username and the administrative role are correct, the script ask the user if he is sure to associate the username entered with the administrative role and then sends a request to the Azure AD to add this modification. This functionality allows the Azure administrator to make one modification at a time. The picture number 4.27, illustrated by the appendix C, shows the variable `$name` which ask the user to enter a user name that is present in the Azure AD and it selects this user then the process is repeated for the administrative role, it asks the user to input an administrative role and stores it in the `$role` variable.

```
function Set-UserRole {
    param (
    [String]$name
    )
    $name = Read-Host "Entrer le display name d'un utilisateur "
    Get-AzureADUser -SearchString $name
    write-host "Administrators roles :"
    Get-AzureADDirectoryRoleTemplate | Select-Object "DisplayName"
    $role = Read-Host "Entrez le display name d'un role"
```

Figure 4.27 - Get a name and an administrative role

The picture number 4.28, illustrated by the appendix C, shows how the role verification is performed. The first if loop verifies that the role supplied by the administrator in the console indeed exists. The second if loop verifies that the administrative role is activated and if it is not the case it will activates it thanks to the `Enable-AzureADDirectoryRole` command.

```
if ($roleArray -match $role) {
    write-host  "Le role fourni existe"
    $role_final = Get-AzureADDirectoryRole | Where {$_.displayName -eq $role}
    write-host $role.gettype()
    write-host $role
    if ($role_final -eq $null) {
        $roleTemplate = Get-AzureADDirectoryRoleTemplate | Where {$_.displayName -eq $role}
        Enable-AzureADDirectoryRole -RoleTemplateId $roleTemplate.ObjectId
        $role_final = Get-AzureADDirectoryRole | Where {$_.displayName -eq $role}
    }
```

Figure 4.28 - Verification about the role entered

Finally, if both roles and username exist and are not already configured it will associate them in the Azure AD and if there is an error about the username or the role, the program will return an error in the console.

The second functionality of this script is the possibility of configuring multiple administrators at a time. This is possible by using a CSV file. Every line of the CSV is composed of a username and an administrative role separated by a comma. A ForEach loop in the function checks every line of the CSV file and if the username, the administrative role exists and are not already configured, then the modifications are uploaded into the Azure AD. By using this functionality, the Azure global administrator simply needs to write the usernames and administrative roles into the CSV file which at the end, results in a great gain of time as illustrated on the picture number 4.29, illustrated by the appendix C.

```
function Set-MultipleUsers {
    #Display the available roles
    Get-MsolRole | Sort Name | Select Name,Description
    $csv_file = "C:\Users\role\role.csv"
    $role_changes = Import-Csv $csv_file | ForEach {Add-MsolRoleMember -RoleMemberEmailAddress $_.UserPrincipalName -RoleName $_.RoleName }
    write-host $role_changes
```

Figure 4.29 - Function to associate multiple users with administrative roles

When the global administrator decides to launch the script, he has the option of using the 1st functionality by typing "1" in the terminal, the 2nd functionality by typing "2" in the terminal and to quit the program by typing "q". The program is running as a continuous loop, the global administrator can execute the function 1 or 2 as much as he wants, as long as he has not pressed "q" to end the program, it does not quit.

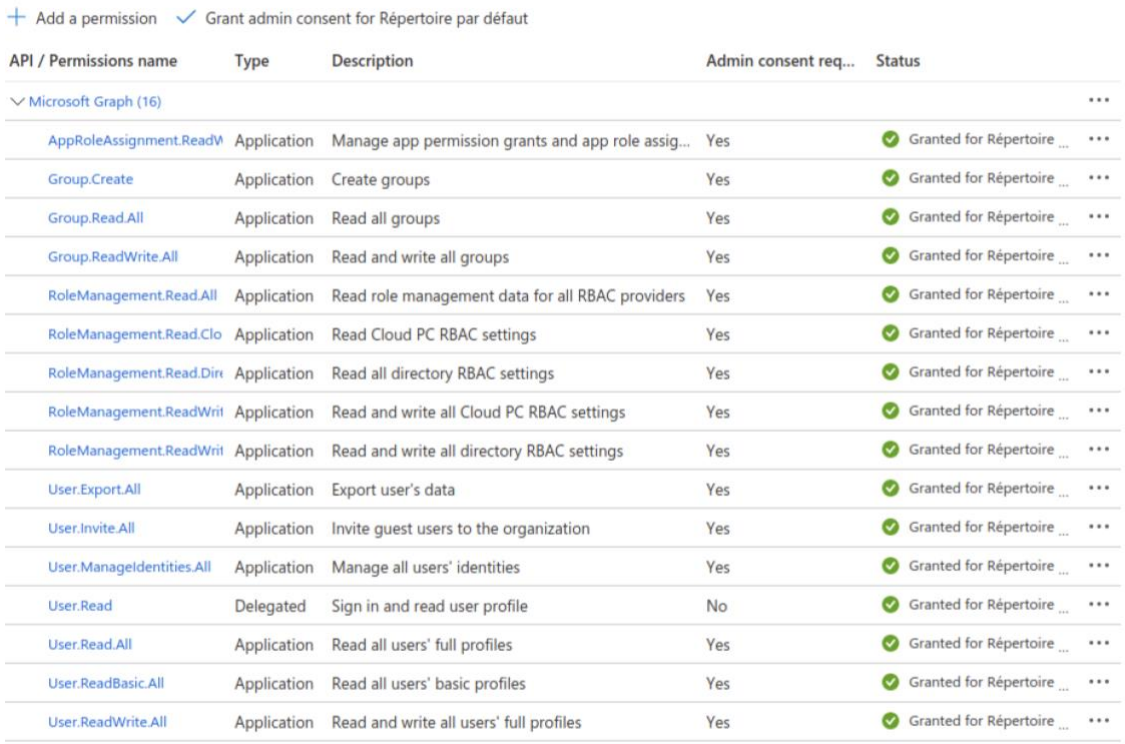
Again, this script respects some basic security principles. To execute it, the global Azure AD administrator role must be granted which is the highest administrator role. When executing the script, the global administrator must authenticate. The script shows least possible details about the Azure AD. The script only allows certain commands which makes impossible for a user to enumerate the different administrative roles and details about the users inside the Azure AD.

4.2.5. Creation of a Python tool to assign administrator roles to users in the Azure AD

The following parts will describe the various steps involved to implement a Python application that allows to assign administrator roles to users in the Azure AD. It will start by describing how the API was created and the security concepts behind it. Then the three main processes of this tool are explained and finally how a Docker container that comes along was created.

4.2.6. Creation of the API on Azure

To begin, the API must be created on Azure to allow the interaction between the client computer and Azure. To do so, an administrator, needs to log into the Azure web portal, then the Azure AD section must be opened, the “application registration” section contains what it is needed to create the API. A name needs to be provided for the API, and which administrators can access the API. This is a very important step; only carefully selected administrators should be able to access this API. To finish the creation of the API, a valid URI must be supplied. This URI will return the authentication response after successfully authenticating a user. Once those steps are done, Azure provides some IDs: the application ID, the object ID, and the directory ID. Those IDs are very important and must be kept carefully by the administrator as they are useful for authenticating to Graph API. To enhance the API security a client secret was created. This client secret consists of a secret string the application will use to prove its identity when requesting for a token. To add even more security when a user is authenticating to a Graph API, a certificate could have been used. The following step is to configure the permissions. Applications are authorized to call APIs when they are granted permissions by administrators as part of the consent process. Various permissions exist, here is a recap of the different permissions related to Microsoft Graph API created as illustrated on the picture 4.30.



API / Permissions name	Type	Description	Admin consent req...	Status
Microsoft Graph (16)				
AppRoleAssignment.ReadW	Application	Manage app permission grants and app role assig...	Yes	Granted for Répertoire ...
Group.Create	Application	Create groups	Yes	Granted for Répertoire ...
Group.Read.All	Application	Read all groups	Yes	Granted for Répertoire ...
Group.ReadWrite.All	Application	Read and write all groups	Yes	Granted for Répertoire ...
RoleManagement.Read.All	Application	Read role management data for all RBAC providers	Yes	Granted for Répertoire ...
RoleManagement.Read.Clo	Application	Read Cloud PC RBAC settings	Yes	Granted for Répertoire ...
RoleManagement.Read.Dir	Application	Read all directory RBAC settings	Yes	Granted for Répertoire ...
RoleManagement.ReadWrit	Application	Read and write all Cloud PC RBAC settings	Yes	Granted for Répertoire ...
RoleManagement.ReadWrit	Application	Read and write all directory RBAC settings	Yes	Granted for Répertoire ...
User.Export.All	Application	Export user's data	Yes	Granted for Répertoire ...
User.Invite.All	Application	Invite guest users to the organization	Yes	Granted for Répertoire ...
User.ManageIdentities.All	Application	Manage all users' identities	Yes	Granted for Répertoire ...
User.Read	Delegated	Sign in and read user profile	No	Granted for Répertoire ...
User.Read.All	Application	Read all users' full profiles	Yes	Granted for Répertoire ...
User.ReadBasic.All	Application	Read all users' basic profiles	Yes	Granted for Répertoire ...
User.ReadWrite.All	Application	Read and write all users' full profiles	Yes	Granted for Répertoire ...

Figure 4.30 - Permissions for the API.

On the picture 4.30 it is shown the permissions selected for an administrator to perform basic operations such as: getting and writing information about a group nor a user, getting and writing information regarding an administrative role, export user's data, invite guest users to the organization. According to what the application must do, the administrator needs to provide the accurate permissions for the application. The least privilege strategy should be implemented for the application, it should access the specific data and resources needed to complete a required task. Hence, the application should have the least permissions so that it can only performs what it was designed for.

4.2.7. OWASP Top 10 API

The first and most dangerous vulnerability "Broken object level authorization" happens when a malicious user replaces the IDs of their resource with a different one, the API then does not check the permissions if not configured well and let the call through. To prevent this vulnerability, permissions are checked when a call is made to the API and random IDs are generated.

The second vulnerability, "Broken authentication", happens for multiple reasons: APIs that use weak authentication, weak IDs, weak passwords, weak access token, and credential

keys exposed in the URL. To prevent this vulnerability, the best authentication practices are implemented. It means, only a strong password is used (at least twenty-two characters, upper case, lower case, figure, special character), the code will be designed to not contain and show credentials or keys, and the access token is ciphered.

The third one, “Excessive data exposure”, is about an API that would expose more data than the client is supposed to see. Various kind of requests were sent by the administrator that possesses the right to execute the program to verify what would be the different outputs. By adjusting the permissions, he should only access the data he needs, and also to run the program an administrative role needs to be assigned. If a user does not have this role the request to the API will be blocked during the authentication phase. The application also enforces response errors, if a malicious user tries to modify the request a response code is returned, the program is stopped automatically, and the process is exited.

The other vulnerabilities need to be taken seriously as they prevent various attacks such as: DoS, command injection, privilege escalation. To avoid those attacks, measures are provided in the OWASP document.

The fourth measure is “Lack of resources and rate limiting”, APIs must impose requests limitations in order to prevent DoS attacks as well as authentication brute force.

The fifth is one is “Broken function level authorization”, clear policies must be defined between administrative and regular functions to prevent authorization flows.

The sixth vulnerability is called “Mass assignment”, The API design needs to hide the underlying implementation of the application along with the property’s names.

The seventh is called “Security misconfiguration”, it is a result of unsecure configurations such as error messages printed containing sensitive information, unnecessary HTTP methods, open cloud storage and more.

The eighth one is about “Injection”, measures need to be set in order to avoir SQL, NoSQL, and any kind of command injection.

The ninth is about “Improper assets management”, a documentation about the different endpoints linked to the API and the API version must maintained.

The tenth vulnerability is about “Insufficient logging and monitoring”, the logs produced by the API need to be monitored. An incidence response strategy needs to be implemented in case of an attack.

4.2.8. Token acquisition

When developing this Python tool, security was the main concern. The main goal of this tool is to assign administrative roles to users inside the Azure AD, but because it is doing a potential risky operation it needs to respect several security criteria. The script begins by obtaining the token. To do so, it needs information such as: the client ID, the client secret, the authority URL, and the scope URL. This information should not be displayed in the script otherwise someone that could read the script would be able to authenticate to the API and potentially performs malicious actions. To prevent those identifiers from being stored inside a file, the click library is used. Click, according to its website, is a “Python package for creating beautiful command line interfaces” [18]. Therefore, it makes possible to write in the terminal the different IDs needed and avoid storing them in a separate file or in the script. When the script is launched, it asks the user to provide those IDs directly in the terminal as illustrated in the picture 4.31.

Figure 4.31 - User needs to provide its IDs throughout the terminal

To continue the process of obtaining a token, the application takes advantage of the Microsoft Authentication Library for Python (MSAL). MSAL can be used in various scenarios including this application: a web application that authenticates users and calls web API. In order to use MSAL, we need to install the package thanks to the command: `pip3 install MSAL`, inside the script it is imported with the following command: `import MSAL`, then a MSAL object is instantiated, and this object uses the IDs to authenticate defined previously. Finally, the token is either already in cache memory and can be loaded directly or it is not in the cache and the application acquires a new one. To store the token in cache memory allows to reduce the amount of request towards the Graph API and therefore reduce the cost. When then script is launched for the first time, it will acquire the token and stored the token in the cache memory for a certain amount of time.

4.2.9. Get user

The second part of the script gets the user's information related to the user's name provided in the terminal. A HTTP GET request is sent towards the API. Again, because the tool needs to be as secure as possible, the URL that is processed by this request must not be visible in the script. This one contains the user's principal name that is being processed. To do so, the click library was used once again. The administrator executing the script, needs to supply the user's principal name he wants in the terminal. The API returns in JSON format the information related to the user such as: the date and time it was created, the user type, if the account is enabled, the group to which the user belongs, the assigned administrative roles, the application he can access and more. Because those data should be confidential, the script only retrieves the user's ID which is stored inside a variable and will be used later. Another feature of this, is the error handling. Thanks to Python library "requests" it is possible to manipulate HTTP request, but this library also brings more features. When a request is done, the script gets the HTTP status code of the request and if it is different from the 200 code, which corresponds to a successful GET HTTP request, the cause of the failure is printed in the terminal. Hence, it helps the user or administrator of the tools to understand why it failed and it helps to provide the right response. Finally, if the request fails, the "sys" library will exit the process linked to the Python script and the program will be stopped.

4.2.10. Get role

The process to get the administrative role we want is pretty much the same as the one to get a specific user. The person using the program needs to supply the appropriate URL pointing to a specific administrative role. Then a HTTP GET request is sent to the API and the result, which is the role's ID, is returned in JSON format and will be stored in a variable. The error handling process is exactly the same as the previous GET request because it deals only with GET request.

4.2.11. Post role

The main purpose of this tool is to associate a user to an administrative role inside the Azure AD. Our previous GET requests got the user ID and administrative role ID we want. Now it needs to link those two, in order to do so a POST request is made to a specific URL : https://graph.microsoft.com/v1.0/roleManagement/directory/roleAssignments. Inside this URL there is the service accessed "graph.microsoft.com", the version number "v1.0", and

then the “roleAssignments” that is inside the “roleManagement” and “directory” path. By associating the user ID and the administrative role ID, the POST request will associate the new administrative role to the user. The data are transmitted to the API in JSON format. Here again, if an error occurs during the process a mechanism is presents to handle the error. This time the HTTP error code must be different than 201, which corresponds to “the request has been fulfilled, resulting in the creation of a new resource » [19]. So, any other code than 201 signifies the POST request didn’t manage to associate the administrator role to the user. As it is displayed on this screenshot below, various information needs to be filled in the terminal. Responses code with the number 200 indicate the requests were successful and changes were done in the Azure AD. The picture 4.32 demonstrates a successful execution of the Python application.

```
francois@direction:~/Desktop/API$ /usr/bin/python3 /home/francois/Desktop/API/Script_API/graph-api.py
Client ID: f90213c4-599e-480b-8c28-e384d6073ffa
Client Secret: 2KN8Q~QQhe52c0Ze4YGHIUyVHQymFPVbKCJfDarm
URL user: https://graph.microsoft.com/v1.0/users?$filter=userPrincipalName eq 'vincent@testadstagesvphotmail.onmicrosoft.com'
URL role definitions: https://graph.microsoft.com/v1.0/rolemanagement/directory/roleDefinitions?$filter=displayName eq 'Application Developer'
URL role assignements: https://graph.microsoft.com/v1.0/roleManagement/directory/roleAssignments
Token successfully obtained
Response code is : 200
Request successfully executed
Response code is  : 200
Request successfully executed
POST request was successfully executed and data have been modified in the Azure AD
```

Figure 4.32 - Output of the Python application

The diagram 4.33 summarizes the functioning of this tool.

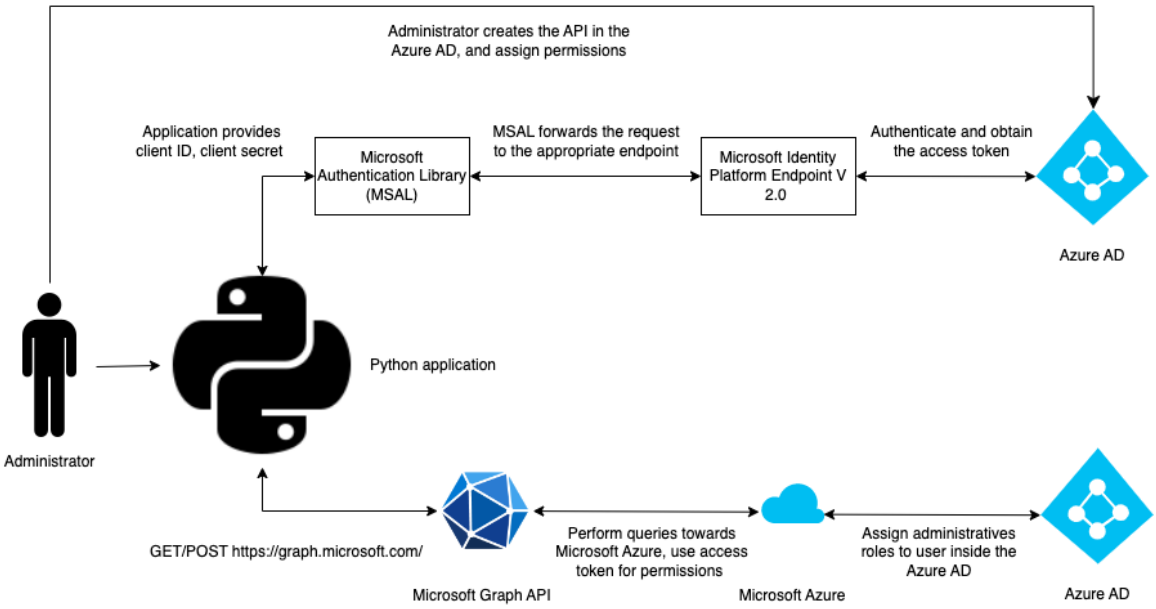


Figure 4.33 - Functioning of the Python application.

4.2.12. More details

It is worth to mention that some other functions inside this script were essential. Among them, the function `outputJSONFormat` allows to convert a Python object into a

JSON string as well as to convert JSON string into a dictionary. This function is essential to have a better visualization of the response returned by the API. The second functionality is useful when needing to access specific data inside the JSON response. Particularly, the IDs that are needed to fulfill the POST request. It allows to transform the JSON string into a dictionary, so they are already Python methods and techniques that can be used to access those data.

The last functions are basically to organize how the script should execute the other functions, in which order. However, this condition, on the screenshot 4.34 below, is a security best practice and it is worth to specify its presence. When somebody import a Python script the code would run as soon as imported. Thanks to this statement, the script gets executed only when it is run and not imported.

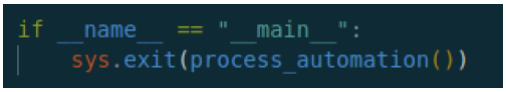


Figure 4.34 - Security condition Python.

Thanks to the click library, the user can get help when he launches the script. If he has no idea what to supply as parameters, simply add `--help` after the command typed to execute the script and some more information are displayed as it is illustrated on the screenshot 4.35.

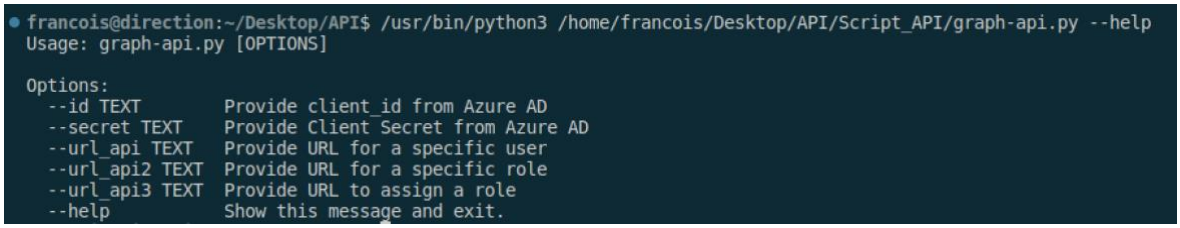


Figure 4.35 - Help command.

4.2.13. Containerise the application

Once the Python application is up and it runs as expected, choice was made to provide a container for it. By keeping the idea of simplifying the work for an administrator, running the application through a container is part of the process. Indeed, the application was built on an Ubuntu VM, and contains dependencies. If another administrator tries to run the application from a different OS and if Python and the required dependencies are not installed on the machine, the application would not run correctly. Therefore, in this case the container provides a package mechanism for the application and its dependencies and an OS level virtualization which allows another administrator to run the application on a different OS.

Thus, problems related to conflicting libraries versions and replicability are solved, everything should be the same in any environment.

To build this container, Docker was used for multiple reasons. This tool was studied and mastered last year during the cloud computing course. Docker creates the industry standards for containers, therefore is it a well-known technology and documentation is available. Docker containers are lightweight in term of resources, and they are also secure.

To begin the construction of this container, the first step is to create and edit a Docker file. As described by Docker's official website, a Docker image is: "a text document that contains all the commands a user could call on the command line to assemble an image. Docker can build images automatically by reading the instructions from a Docker file" [20]. This file start with the `FROM` command which initializes a new build stage and sets the base image. The image in the file is Python version 3.9. Then, the Python application is added inside the Docker file thanks to the `ADD` command, then the dependencies must be installed. To do so, a Python virtual environment is set up. A virtual environment can be described as a" self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages» [21]. Adding a virtual environment allows to tackle the requirement's version problems. The command: `python3 -m venv venv` creates a virtual environment and it needs to be activated with the following command: `. venv/bin/activate`. Next, with Python's package manager pip, a list named requirements.txt is created thanks to the following command: `pip freeze > requirements.txt`, it contains every package the application needs in requirements format. Then this file, requirements.txt, is added inside the Docker file. The Docker file now contains the list of the packages needed for the application and with the command: `RUN pip install requirements.txt`, the packages are installed in the container. The flag: --no-cache-dir is added, it allows to minimize the impact on the memory and keep the Docker image as small as possible when installing the packages. Finally, the last command launches the Python application inside the container. The picture 4.36 below is the complete Docker file, also represented by the appendix D.

```
FROM python:3.9

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

COPY ./app /code/app

CMD [ "python", "./app/python-api.py" ]
```

Figure 4.36 - Docker file

Once the Docker file is completed, the image needs to be built. The Docker CLI allows to build the image with the following command: `docker -t build image.name.` Then the image needs to be converted into a container, again Docker's CLI allows to do it by typing: `docker run image.name`.

4.2.14. Documentation

In addition to the various tools and scripts that have been delivered to help administrators, a documentation about every action to enhance Microsoft Office 365 security score has been written. This document includes every action that were in the CSV file and for each one of them, three paragraphs have been written. The 1st one is about how to solve the action, the 2nd is about the potential impacts on the users and the 3rd one is about the possible complementary actions.

Regarding the actions, they are about generic security principles for Microsoft products. We find themes such as: passwords management, multifactor authentication, management of administrator roles, management of external users, and more. Each one of those action is going to be detailed in the upcoming part.

5. Conclusion

Overall, my experience of as a consultant in computer systems and software at Tehtris was very enriching. I had a great collaboration with my supervisor and with all the people whit whom I worked. It was a great opportunity to work with another computer engineer, my supervisor, I managed to communicate my results both in writing and orally, which required me to be organized.

The missions I was assigned were challenging, indeed I had to learn new concepts sometimes by myself, however both missions were carried out successfully and my supervisor was satisfied with the results I provided. I also consider that the new concepts I have learned during this experience will be assets for my future career. Indeed, I had the responsibility of choosing technologies to carry out my missions which was rewarding because I had to inform myself about the state of current technologies, choose one, and use it to build a solution.

At the end of the first mission, a functional solution was provided, in other words a PKI that was able to create and deliver S/MIME certificates for endpoints to encrypt email communication was delivered. To test it, email addresses from different domains sent an email to each other along with the S/MIME certificates, the software which is the same for every email address indicated that the identity for every email address was verified as well as the content of the emails was encrypted. Later, this solution was implemented in the company's infrastructure. Along with the functional solution, documentations about the theoretical concepts related to a PKI, the choice of OpenSSL and as well as the process of creating and maintaining the PKI were written. Regarding the results for the second mission, a general documentation was written, this latter explains the theory behind every security action, to help the administrator to understand the context, how to solve an action thanks to Microsoft O365 web portal. This documentation also explain that a functional PowerShell script is available for every action, how to use it and what is the content of those scripts. Finally, a RESTful API along with a Python application that connects to it was delivered to complete a specific action. To ease the deployment of this application a Docker container was provided. This has made possible to increase the Microsoft O365 security score by twenty five percent.

The courses I have attended during my master's degree at Instituto Politecnico de Leiria were very helpful, I had the opportunity to work on subjects such as: software, cloud technologies, APIs, computer network, cybersecurity, encryption, and all of those were covered last year. During this experience I also learnt more about the XDR technology and about all the products included. I have discovered the various cybersecurity jobs in the company, which helped me to associate the names given to a job with the task they perform as well as the technologies they use. I consider that this experience has reinforced my desire to work in the cybersecurity industry.

The main difficulty in the first mission was to make the email software used in the company works with the OCSP server. After a lot of research concerning this issue, no simple solution was found to redirect the OCSP request from the software towards the server. By default, the email software is sending an OCSP request to a server that is outside of the networks. To tackle this problem, some solutions were considered such as setting up a proxy server or even creating one inside a Docker container which could be an easier and lighter solution to deploy.

The difficulty encountered during the second mission was to conceive a Microsoft (Azure, AD and every Microsoft product related) infrastructure that was as close as possible to TEHTRIS infrastructure. Indeed, the data contained in the TEHTRIS Azure AD were confidential. Because both Azure AD were not exactly the same, some actions score were different. For instance, a precise action on TEHTRIS infrastructure would improve the score by five percent whereas the same action on the Azure AD, created especially for this second mission, would improve the score by fifteen percent. Nonetheless, the concept behind the action remains the same and all scripts and tools delivered to complete an action on the Azure AD created for this mission will also allow to complete the action on TEHRIS Azure AD.

Regarding the potential improvements that could be done, they are mainly about the Python application. The PowerShell scripts are functional and commented, the code could be improved to run faster but the scripts allow to complete every action. Two potential improvements could be added for the Python application. The first one would enhance the security. A certificate could be used to connect to the API. The authentication is made thanks to a secret string and some IDs that the user needs to provide. By using a certificate, it allows

to get rid of the secret string, therefore it prevents attack such as brute force. Multi-factor authentication could also be enforced to enhance the authentication process. Finally, a graphic interface could be added to simplify the process for a person that does not have knowledge about Python.

.

6. Bibliography

[1] Wikipedia, «Message transfer agent,» 2023 05 2. [En ligne]. Available: https://en.wikipedia.org/wiki/Message_transfer_agent. [Accès le 2023 05 27].

[2] Wikipedia, «Message delivery agent,» 16 12 2022. [En ligne]. Available: https://en.wikipedia.org/wiki/Message_delivery_agent. [Accès le 2023 05 27].

[3] "Wikipedia Email Alias," 2022. [Online]. Available: https://en.wikipedia.org/wiki/Email_alias. [Accessed 13 November 2022].

[4] Wikipedia, «Pretty Good Privacy,» [En ligne]. Available: https://en.wikipedia.org/wiki/Pretty_Good_Privacy. [Accès le 2023 04 04].

[5] "Wikipedia S/MIME," 2023. [Online]. Available: https://en.wikipedia.org/wiki/S/MIME. [Accessed 2 December 2022].

[6] Wikipedia, «Digital signature,» [En ligne]. Available: https://en.wikipedia.org/wiki/Digital_signature. [Accès le 31 05 2023].

[7] Let's Encrypt, «Documentation,» [En ligne]. Available: https://letsencrypt.org/fr/docs/. [Accès le 2022 12 16].

[8] NIST, «Certificate Revocation List,» [En ligne]. Available: https://csrc.nist.gov/glossary/term/certificate_revocation_list. [Accès le 2023 05 10].

[9] Wikipedia, «Online Certificate Status Protocol,» [En ligne]. Available: https://en.wikipedia.org/wiki/Online_Certificate_Status_Protocol. [Accès le 2023 04 07].

[10] Microsoft, "What is Microsoft 365," 2022. [Online]. Available: https://support.microsoft.com/en-us/office/what-is-microsoft-365-847caf12-2589-452c-8aca-1c009797678b. [Accessed 2 January 2023].

[11] Hack The Box, «What is Active Directory?,» 17 05 2022. [En ligne]. Available: https://www.hackthebox.com/blog/introduction-to-active-directory. [Accès le 26 05 2023].

[12] Microsoft, "Compare Active Directory to Azure Active Directory," 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-compare-azure-ad-to-ad. [Accessed 11 January 2023].

[13] Microsoft, «products apps and services,» [En ligne]. Available: https://www.microsoft.com/en-us/microsoft-365/products-apps-services. [Accès le 2023 05 23].

[14] Microsoft, "What is PowerShell ?," 2022. [Online]. Available: https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3. [Accessed 20 January 2023].

[15] Microsoft, "Use The Microsoft Graph API," 2023. [Online]. Available: https://learn.microsoft.com/en-us/graph/use-the-api. [Accessed 24 January 2023].

[16] OWASP, «OWASP API Security Project,» 2019. [En ligne]. Available: https://owasp.org/www-project-api-security/. [Accès le 6 February 2023].

[17] Wikipedia, "Maildir," 2022. [Online]. Available: https://en.wikipedia.org/wiki/Maildir. [Accessed 4 April 2023].

[18] Canonical Ltd, "Security - Certificates," 2023. [Online]. Available: https://ubuntu.com/server/docs/security-certificates. [Accessed 18 Octobre 2022].

[19] Wireshark, "Tshark Manual Page," [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html. [Accessed 8 December 2022].

[20] Internet Security Research Group, "About Let's Encrypt," [Online]. Available: https://letsencrypt.org/about/. [Accessed 19 December 2022].

[21] OpenSSL Project Authors, "OpenSSL," 2023. [Online]. Available: https://www.openssl.org/docs/. [Accessed 8 December 2022].

[22] E. RSA, «Easy RSA Home,» [En ligne]. Available: https://easy-rsa.readthedocs.io/en/latest/. [Accès le 2022 12 16].

[23] stunnel, «Documentation,» [En ligne]. Available: https://www.stunnel.org/docs.html. [Accès le 2022 12 16].

[24] Q. Drouet, «Documentation Technique,» [En ligne]. Available: https://technique.arscenic.org/ssl-securisation-des-communications-serveur-client/article/startssl-utiliser-un-certificat. [Accès le 2022 12 16].

[25] NIST, «Securing Web Transactions TLS Server Certificate Management,» June 2020. [En ligne]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-16.pdf. [Accès le 2022 12 16].

[26] Microsoft, "about_Modules," 2022. [Online]. Available: https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_modules?view=powershell-7.3. [Accessed 20 January 2023].

[27] K. Das, "Building command line application with Click," 2022. [Online]. Available: https://pymbook.readthedocs.io/en/latest/click.html. [Accessed 15 February 2023].

[28] "Status Code Definitions," [Online]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html. [Accessed 5 April 2023].

[29] Docker Inc., "Dockerfile reference," 2023. [Online]. Available: https://docs.docker.com/engine/reference/builder/. [Accessed 23 February 2023].

[30] Python Software Foundation, "Virtual Environments and Packages," 2023. [Online]. Available: https://docs.python.org/3/tutorial/venv.html. [Accessed 5 April 2023].

[31] S. Kelley, "Dnsmasq," [Online]. Available: https://thekelleys.org.uk/dnsmasq/doc.html. [Accessed 18 October 2022].

[32] "Dovecot manual," [Online]. Available: https://www.dovecot.org. [Accessed 24 October 2022].

[33] "Postfix Documentation," [Online]. Available: https://www.postfix.org/documentation.html . [Accessed 24 October 2022].

[34] A. Norman, "python-nmap," 2021. [Online]. Available: https://pypi.org/project/python-nmap/. [Accessed 2022 November 2022].

[35] "Wikipedia Let's Encrypt," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Let%27s_Encrypt. [Accessed 7 December 2022].

[36] I. Ristic, "openssl cookbook," [Online]. Available: https://www.feistyduck.com/library/openssl-cookbook/online/. [Accessed 8 December 2022].

[37] Microsoft, "Implementing Least-Privilege Administrative Models," 2021. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/implementing-least-privilege-administrative-models. [Accessed 23 January 2023].

[38] S. Peyrott, "The JWT Handbook," 2018. [Online]. Available: https://jwt.io/introduction. [Accessed 25 January 2023].

[39] Microsoft, "Common Conditional Access Policy," 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/conditional-access/howto-conditional-access-policy-admin-mfa. [Accessed 26 January 2023].

[40] Microsoft, "Set up multifactor authentication for Microsoft 365," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/admin/security-and-compliance/set-up-multi-factor-authentication?view=o365-worldwide. [Accessed 26 January 2023].

[41] Microsoft, "Set the password expiration policy for your organization," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/admin/manage/set-password-expiration-policy?view=o365-worldwide. [Accessed 30 January 2023].

[42] Microsoft, "Share Microsoft 365 calendars with external users," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/admin/manage/share-calendars-with-external-users?view=o365-worldwide. [Accessed 31 January 2023].

[43] Microsoft, "Configure how users consent to application," 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/manage-apps/configure-user-consent?pivots=portal. [Accessed 2 February 2023].

[44] Microsoft, "Assign admin roles in the Microsoft 365 admin center," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoft-365/admin/add-users/assign-admin-roles?view=o365-worldwide. [Accessed 2 February 2023].

[45] Microsoft, "Least privileged roles by task in Azure Active Directory," 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/roles/delegate-by-task. [Accessed 3 February 2023].

[46] Microsoft, "Manage meeting settigns in Microsoft Teams," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoftteams/meeting-settings-in-teams. [Accessed 6 February 2023].

[47] Microsoft, "Teams policies reference," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoftteams/settings-policies-reference. [Accessed 7 February 2023].

[48] Microsoft, "Control who can bypass the meeting lobby in Microsoft Teams," 2023. [Online]. Available: https://learn.microsoft.com/en-us/microsoftteams/who-can-bypass-meeting-lobby. [Accessed 8 February 2023].

[49] Microsoft, "Sign out inactive users," 2023. [Online]. Available: https://learn.microsoft.com/en-us/sharepoint/sign-out-inactive-users. [Accessed 9 February 2023].

[50] K. Reitz, "Request: HTTP for Humans," [Online]. Available: https://requests.readthedocs.io/en/latest/. [Accessed 13 February 2023].

[51] Microsoft, "Quickstart: Add sign-in with Microsoft to a web app," 2023. [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/develop/web-app-quickstart?pivots=devlang-python&tabs=windows. [Accessed 16 February 2023].

Glossary

Authentication: verify the identity of a user

Cache memory: small-size and fast type of volatile data in a computer

Class: concept of the object-oriented programming

Cloud shell console: an integrated console in Azure web portal, allowing to execute Shell commands

Command injection: a cyber-attack which aims to execute a command via a vulnerable application

Conditional access policy: if a user wants to access a resource, then they must complete an action

Credentials stuffing: a cyber-attack that consists in massive authentication attempts on web sites and services using login password pairs.

Daemon: a process that runs continuously in the background

Data security: protect the access of data, from access or nor corruption

Dependency: a code package or code library that is reused in software

Domain name: an easy name associated with an IP address

DoS attack: a cyber-attack which makes a service unavailable

End entity: a person or an end user

ForEach: a loop that allows the processing of all entries in a table

Function: a named piece of computer code that performs a specific task

Integrity: data are not modified or erased during the transport phase which ensures information non-repudiation as well as authenticity

Methods: concept of the object-oriented programming

MIME data: multipurpose Internet Mail Extensions data or in other words audio, video, photos, and all kind of files.

Non-repudiation: the sender of information receives a proof of delivery, and the recipient possesses a proof of the sender's identity

Object: concept of the object-oriented programming

OpenSSL: a command line cryptography library

Outlook: a Microsoft application for emails

Package: a set of files and directories required by a software product.

Password spraying: a cyber-attack that uses common passwords to try to access multiple accounts

Port: a virtual port that receives incoming and outgoing network connections

Privacy: confidentiality and access to certain information about an entity is protected

Privilege escalation: a cyber-attack which aims to gain more privileges

Python : a programming language

State of the art report: a report that indicates the technologies that exist and their status

Systemctl: command line utility to manipulate system

Systemd: software suite that provides an array of system components for Linux OS

Token: a form of authentication that allows a user to access an online service

Variable: code variables are elements that associate a name with a value

Appendices

Appendix A
Bash script to create a certificate

```
#!/bin/bash
RED='\033[0;31m'
NC='\033[0m'
REQUIRED_PKG1="figlet"
REQUIRED_PKG2="openssl"
PKG_OK1=$(dpkg-query -W --showformat='${Status}\n' $REQUIRED_PKG1|grep "Package already
installed")
PKG_OK2=$(dpkg-query -W --showformat='${Status}\n' $REQUIRED_PKG2|grep "Package already
installed")

  echo Checking for $REQUIRED_PKG1: $PKG_OK1 && echo Checking for $REQUIRED_PKG2: $PKG_OK2
  if [ "" = "$PKG_OK1" ] && [ "" = "$PKG_OK2" ] ; then
   echo "No $REQUIRED_PKG1. Setting up $REQUIRED_PKG1."
   echo "No $REQUIRED_PKG2. Setting up $REQUIRED_PKG2."
   sudo apt-get --yes install $REQUIRED_PKG1> /dev/null 2>/dev/null
   sudo apt-get --yes install $REQUIRED_PKG2> /dev/null 2>/dev/null
  fi
  figlet !!! CA root must be configured !!! -c -f smslant

  sleep 2
  echo "--------------------------------------------------------------------------------"

  figlet generate a new certificate -c -f smslant

  echo "--------------------------------------------------------------------------------"

  #Create a certificate
  generate_csr(){
     openssl genrsa -des3 -out server.key 2048
     echo -e "${RED} RSA keys generated successfully ${NC}"
     openssl rsa -in server.key -out server.key.insecure
     mv server.key server.key.secure
     mv server.key.insecure server.key
     openssl req -new -key server.key -out server.csr
     echo -e "${RED}CSR generated sucessfully ${NC}"
  }
  generate_csr

  #Generate the certificate based on the CSR
  sign_certificate(){
     sudo openssl ca -in server.csr -config /etc/ssl/openssl.cnf> your-cert.pem
# sign certificate according to configurations inside openssl.cnf
     if [ $? -eq 0 ];
     then
       echo -e "${RED} certificate signed sucessfully ${NC}"
     else
      echo -e "${RED} something went wrong. Please check the output above ${NC}"
     fi
  }
  sign_certificate

  get_certificate(){
     sudo awk '/-----BEGIN CERTIFICATE-----/,/-----END CERTIFICATE-----/ { print $0 }' your-cert.pem >  your-
     cert.crt
     echo -e "${RED} certificate file created ${NC}"
     sudo mkdir -p /etc/ssl/pkcs12
     echo "donner le nom du pkcs12 et pas oublier .p12 a la fin"
     read pkcs12_name
```

```
    openssl pkcs12 -export -name "ce nom" -inkey server.key -in your-cert.crt -out $pkcs12_name
    if [ $? -eq 0 ];
    then
      echo -e "${RED} pkcs12 file created sucessfully ${NC}"
    else
      echo -e "${RED} something went wrong. Please check the output above ${NC}"
    fi
    sudo mv ~/generating-certif/$pkcs12_name /etc/ssl/pkcs12/
}
get_certificate
```

Appendix B

Bash script to delete a certificate

```
#!/bin/bash
RED='\033[0;31m'
NC='\033[0m'
REQUIRED_PKG1="figlet"
REQUIRED_PKG2="openssl"
PKG_OK1=$(dpkg-query -W --showformat='${Status}\n' $REQUIRED_PKG1|grep "Package already installed")
PKG_OK2=$(dpkg-query -W --showformat='${Status}\n' $REQUIRED_PKG2|grep "Package already installed")

echo Checking for $REQUIRED_PKG1: $PKG_OK1 && echo Checking for $REQUIRED_PKG2: $PKG_OK2
if [ "" = "$PKG_OK1" ] && [ "" = "$PKG_OK2" ] ; then
 echo "No $REQUIRED_PKG1. Setting up $REQUIRED_PKG1."
 echo "No $REQUIRED_PKG2. Setting up $REQUIRED_PKG2."
 sudo apt-get --yes install $REQUIRED_PKG1 > /dev/null 2>/dev/null
 sudo apt-get --yes install $REQUIRED_PKG2 > /dev/null 2>/dev/null
fi
figlet !!! CA root must be configured !!! -c -f smslant

sleep 2

echo "--------------------------------------------------------------------------------"

figlet You are about to delete a certificate -c -f smslant

echo "--------------------------------------------------------------------------------"

delete_certificate(){
   echo "please, enter a certificate name as displayed in the /etc/ssl/newcerts/ path"
   echo "it should be number.pem format. For example : 01.pem"
   read certif_name
   sudo openssl ca -config /etc/ssl/openssl.cnf -revoke /etc/ssl/newcerts/$certif_name -crl_reason
   superseded
   if [ $? -eq 0 ];
   then
     echo -e "${RED} certificate deleted sucessfully ${NC}"
   else
    echo -e "${RED} something went wrong. Please check the output above ${NC}"
   fi
}
delete_certificate
```

Appendix C

PowerShell script to assign an administrative role to a user

```
Import-Module AzureAD
connect-azuread -TenantId #insérer le numéro TenantId


Import-Module MSOnline
connect-msolservice


function Show-Menu {
    param (
        [string]$title = "L'option 1 permet de remplir le role d'un utilisateur alors que la 2eme permet d'en
        remplir plusieurs simultanement"
    )
    Clear-Host
    Write-Host " `n  $title "
    Write-Host " `n Entrez 1 pour remplir le role d'un utilisateur"
    Write-Host " `n Entrez 2 pour remplir plusieurs utilisateurs"
    Write-Host " `n Entrez q pour quitter le programme"
}

#Import du module Azure AD et connexion à l'AD via le tenant ID
function Set-UserRole {
    param (
    [String]$name
    )
    $name = Read-Host "Entrer le display name d'un utilisateur "
    Get-AzureADUser -SearchString $name
    write-host "Administrators roles :"
    Get-AzureADDirectoryRoleTemplate | Select-Object "DisplayName"
    $role = Read-Host "Entrez le display name d'un role"
    $selectRole =  Get-AzureADDirectoryRoleTemplate | Select-Object "DisplayName"
    $roleArray = ($selectRole | Foreach {"$($_.DisplayName)"})
    if ($roleArray -match $role) {
        write-host  "Le role fourni existe"
        $role_final = Get-AzureADDirectoryRole | Where {$_.displayName -eq $role}
        write-host $role.gettype()
        write-host $role
        if ($role_final -eq $null) {
            $roleTemplate = Get-AzureADDirectoryRoleTemplate | Where {$_.displayName -eq $role}
            Enable-AzureADDirectoryRole -RoleTemplateId $roleTemplate.ObjectId
            $role_final = Get-AzureADDirectoryRole | Where {$_.displayName -eq $role}
        }
        Add-AzureADDirectoryRoleMember -ObjectId $role_final.ObjectId -RefObjectId (Get-AzureADUser |
        Where {$_.DisplayName -eq $name}).ObjectID
    }
    else {
        write-host "Le role fourni n'existe pas"
    }
    Start-Sleep -Seconds 5
}
function Set-MultipleUsers {
    #Display the available roles
    Get-MsolRole | Sort Name | Select Name,Description
    $csv_file = "C:\Users\role\role.csv"
    $role_changes = Import-Csv $csv_file | ForEach {Add-MsolRoleMember -RoleMemberEmailAddress
    $_.UserPrincipalName -RoleName $_.RoleName }
    write-host $role_changes
}
#Main class
do {
    Show-Menu
    $choosenOption = Read-Host "Entrez l'option choisie :"
    Clear-Host
```

```
    switch($choosenOption){
      '1'{Set-UserRole;break}
      '2'{Set-MultipleUsers;break}
      'q'{break}
      default{
       Write-Host "L'option entrée est '$choosenOption'" -ForegroundColor Red
       Write-Host "Sélectionner une des options possibles" -ForegroundColor Red
      }
    }
    Pause
} until($choosenOption -eq 'q')
```

Appendix D
Docker file for the python application

```
FROM python:3.9

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

COPY ./app /code/app

CMD [ "python", "./app/python-api.py" ]
```