Internship at Be One Solutions

Master degree in Cybersecurity and Digital Forensics

Tiago Alexandre Pinheiro Martins

Leiria, November of 2021

Internship at Be One Solutions

Master degree in Cybersecurity and Digital Forensics

Tiago Alexandre Pinheiro Martins

Internship Report under the supervision of Doctor Paulo Costa

Leiria, November of 2021

Originality and Copyright

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

Dedication

To my grandmother Laurinda, and to my parents Lina and Vitor.

Acknowledgments

I would like to thank everyone who helped me during the research, planning and writing of this document, especially my family who supported and motivated me not only during the period but also throughout the master's course.

I am very thankful for the support offered by my supervisor, Professor Paulo Costa, for all the feedback to improve this document.

I am very thankful for the opportunity offered by the Be One Solutions company and for all the support offered by my supervisor, Dr. Filipe Felisberto, he has helped me integrate into the company in a manner where I could feel welcome. I am also thankful towards my colleges at the company for all of their support and motivation.

Finally, I would like to thank all my fellow colleges and professors of the master's course for helping me grow academically, by providing me with all their expertise regarding cybersecurity, but also as a person throughout our personal relations.

Abstract

Included in this document is the report of my internship undertaken in the fulfilment of my Master of Cybersecurity and Informatic Forensics degree from the Polytechnic Institute of Leiria, at Be One Solutions. During the internship, I identified several issues regarding security protocols and procedures at the company, more specifically in regards to credential management.

After identifying the issues, I started researching enterprise level solutions for credential management, for which the requirements had been established beforehand with the IT manager. After comparing a set of solutions based on the features they provided and the price quoted, it was possible to conclude that all solutions were unsuitable due to either unreasonable pricing or previous security issues.

Since the solutions analysed were deemed unsuitable, I started working on a Proof of Concept (PoC) for a custom solution that would be able to integrate with the project structure already present in the company's in house project management solution. It started with defining the concept of the solutions in regards to how the encryption process would be performed, then the designing of the data structure in order to integrate with the project management solution, an afterwards came the development process of said solution.

Keywords: password manager, asymmetric encryption, symmetric encryption

Resumo

Este documento representa o relatório de estágio efetuado na empresa Be One solutions no âmbito do Mestrado de Cibersegurança e Informática Forense da Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria (IPL). Durante o estágio, foram identificados alguns problemas relacionados com protocolos e procedimentos de segurança, mais especificamente no processo de gestão de credenciais de acesso.

Após a identificação destes problemas, foram efetuadas pesquisas no âmbito de soluções de gestão de credenciais para ambiente empresarial, de acordo com requisitos estabelecidos em conjunto com o administrador de TI. Após a comparação de um conjunto de soluções encontradas, determinou-se que nenhuma era apropriada devido a um preço demasiado elevado ou por ter tido problemas de segurança no passado.

Devido a todas as soluções analisadas terem sido consideradas como inapropriadas, definiu-se a construção e desenvolvimento de uma solução para Prova de Conceito (PoC) que fosse capaz de ser integrada com a solução de gestão de projeto atualmente existente na empresa. Começou-se pela definição do conceito relativamente ao processo de encriptação, de seguida procedeu-se ao desenho da estrutura de dados, e por fim iniciou-se o processo de desenvolvimento da solução.

Palavras-chave: gestor de passwords, criptografia simétrica, criptografia assimétrica

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

AD	Active Directory
AES	Advanced Encryption Standard
APAC	Asia Pacific
API	Application Programming Interface
B1if	Integration Framework for SAP Business One
CA	Certification Authority
CERN	Conseil Européen pour la Recherche Nucléaire
CI/CD	Continuous Integration and Continuous Delivery
CRUD	Create Read Update and Delete
CSS	Cascading Style Sheets
DES	Data Encryption Standard
DLL	Dynamic-Link Library
DMS	Dealership Management System
ECC	Elliptic-Curve Cryptography
EMEA	Europe, the Middle East and Africa
ERP	Enterprise resource planning
ESTG	School of Technology and Management
FIPS	Federal Information Processing Standards
GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IP	Internet Protocol
IT	Information Technology
JOSE	Javascript Object Signing and Encryption
JSON	JavaScript Object Notation
JWS	JSON Web Signature
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol

MSSQL	Microsoft SQL Server
MVC	Mode-View-Controller
NDA	Non-Disclosure Agreement
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PHP	PHP: Hypertext Preprocessor
PKCS	Public Key Cryptography Standards
PoC	Proof of Concept
RC4	Rivest Cipher 4
RC5	Rivest Cipher 5
REST	Representational State Transfer
RFC	Request for Comments
RSA	Rivest–Shamir–Adleman
SAP	System Analysis and Programming
SQL	Structured Query Language
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UAT	User Acceptance Testing
UI	User Interface
URI	Uniform Resource Identifier
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
XOR	Exclusive OR
XSS	Cross-Site Scripting

1. Introduction

This document serves as record for the activities performed by the author of this document in the host company by the name of Be One Solutions.

Nowadays IT security is one of the most important aspects of a company that works in the IT workspace. With the magnitude of different applications, operating systems, and other software solutions that we deal with every day, it is quite possible that one of them had security issues in the past, and some issues might be up to be discovered.

Good security practices and procedures are a must to properly secure the data used inside the company, or to enforce certain behaviours amongst the employees. If no such practices are in place, the employees will often prefer convenience over security, leading to potential data compromises that could cost the company's image or even millions of dollars [1].

The opportunity at Be One Solutions allowed me to analyse this type of issues and propose solutions at a closer level. The internship had the duration of approximately one year, and during this time I gave my input in security-oriented topics and worked on solutions in attempt to mitigate issues present in the company. The internship officially started on the 14th of October 2019, and it ended on the 31st of January 2021. During this period and until the 30th of November of 2021, I was also writing this report with all my discoveries, hardships, and accomplishments.

This internship was taken in the fulfilment of my Master of Cybersecurity and Informatic Forensics degree from the Polytechnic Institute of Leiria, which started on September of 2018.

The main goal of this internship was for me to grow as security oriented professional and to learn how to integrate in the workforce of an organization. After my discoveries regarding the concerning state of credential management in the company, my objective shifted into providing solutions to properly solve this problem.

This chapter covers the global scope of the document and presents in a short and concrete manner the contents of the report by going over each chapter.

Chapter II consists in the characterization of the host company. A brief description of the company's history is made, we glance over the core services provided by the company, and we look at a very rough diagram of the company's internal structure. The internship programme is also described in this chapter. It touches briefly on the main topic of this document which is a security-oriented project that was carried on throughout the internship.

On chapter III a critical analysis is made in regards to information security and procedures and some improvement measures are proposed. Research on the ideal solution is conducted and we reach the conclusion that there isn't a suitable solution on the market and so the development of a Proof of Concept (PoC) custom solution is proposed.

Chapter IV serves as an encyclopaedia for the concepts and technologies used during the planning and development of the custom solution.

On Chapter V we have the detailed process of the construction of the custom solution, it goes from the concept into an overview of the data structure used along with the description of the cryptographic process used. It also goes over the software solution itself in terms of software architecture and an infrastructure overview is performed. Finally, a brief analysis of the client application is made and the challenges during the development are brought to light.

Chapter VI holds the conclusion to this document where the results are presented, and the future work is proposed.

2. Characterization of the Host Institution

Be One Solutions is a company based on Switzerland that specializes in SAP related products and services. The company is an SAP partner, and its business focus lies on providing services to worldwide companies that want to centralize or unify their operations. The services go from SAP products implementation, upgrade, support, training, consulting, and development for more specific requirements.

The company has over 18 legal entities worldwide and most of its team members work remotely. In Portugal, there is an office in *Marinha Grande* which is usually frequented two to three times a week with the team working remotely from home during the remaining days.

The company was founded on 2008 with the office in Portugal being establish in January 2018 after more than two people from Leiria started working for the company. Initially the market was focused on the Europe, the Middle East and Africa region, EMEA, but soon it would expand to the Americas region and Asia – Pacific region, APAC.

While I'm not able to show the organization chart, I'm allowed to describe it in a general manner according to the company's public website [2]. At the highest level of the organization, we have Uzi Halfon the Founder & CEO accompanied by Rainer Vischer the Vice President. For each region there is a Regional Director, Mario Candido for AMERICAS, Olli Kylänpää for APAC, and Arnaud Viviant for EMEA. Under each region there are multiple legal entities established and each legal entity has a manager with the title of Country Manager. The rest of the team is split up between each legal entity and there are smaller hierarchies inside of them.

Most of the team is composed by SAP Business One consultants, SAP S/4HANA consultants, and developers with other smaller percentages being taken by internal roles such as marketing, human resources, administration, and finance.

SAP Business One is an Enterprise Resource Planning (ERP) solution designed for small and medium sized enterprises, which provides a single solution for financial management, sales and customer management, and purchasing and inventory control. It's a software solution that offers localized versions for 39 different countries, and some countries without a specific localized version are also supported [3]. Under the SAP Business One suite of products we have the Integration Framework for SAP Business One (B1if) which provides

an official way of interfacing with the core of the ERP solution, allowing the creation, updating and deletion of Business One entities such as Business Partners, Purchase Orders, Sales Orders, Invoices, Credit Notes and many more [4].

Originally the company product focus was on the SAP Business One line of products but recently, it has been establishing itself on the SAP S/4HANA product line.

Be One Solutions is an SAP Gold partner that is very experienced in SAP rollouts across multiple countries and even continents. In most rollouts the customer usually has special requirements, such as an existing workflow, dependency on another solution or might want something that needs to be integrated with SAP Business One. These scenarios are what mainly keeps the development team busy, along with the development of products and internal solutions.

2.1.Internship Programme

Initially I was brought into the company to support a senior developer on an in-house time and expenses reporting software solution, however the senior developer decided to leave the company soon after I joined and so I became the only developer supporting the project. After some months of working alone, two more junior developers have joined, and I was assigned as lead developer of the project. During the first few months, I was mainly focused on understanding the project itself and providing support and bug fixing. Occasionally, I would also perform some activities related to the degree, with them being mostly *pen testing* of other in-house developed solutions.

In the internship contract, there was a clause which granted me a day per week that I could fully dedicate to activities related to my master's degree. In the first couple of months, this was not utilized for the intended purpose due to the circumstances explained above. After a few months of idea gathering and research, some security issues regarding the handling and storing of customer's credentials were detected. Due to the nature of the services the company provides, it is often necessary to access and work in the customer's infrastructure. Usually, the customer sends the necessary credentials via email to the required personnel. The problem arises when the team members need a convenient way of storing said credentials, which is usually notepad, one note, or other non-encrypted media. In many

cases there is also the need to share those credentials as the customer may only provide a generic access and may not want to create individual ones. This sharing is often done via non-encrypted messaging mediums such as email, Skype, and Microsoft Teams.

By shining light into this problem, I was assigned to conduct a market analysis regarding enterprise level credential storage solutions which proved to them all being very expensive. After gathering information regarding necessary and "nice to have" features together with the help of the company's IT manager, the development of a PoC of a custom solution is proposed, which could also become a concrete piece of work that would be representative of what I leant throughout the master course.

During the internship there were also some occasions where security advice was requested, and I was assigned to give my thoughts on the issues presented. On a few instances, there was the need to perform security analysis on internal or company partner's software solutions and I was assigned to perform said analysis. Due to the confidentiality I'm unable to directly include the pen testing reports in this document, but I can share some of the background regarding one of those instances.

A Be One Solutions partner company had requested assistance during development of one of their solutions. The solution in question was a bank transfer service focused in providing the best exchange rates for a particular transfer. Due to the high importance of the data being handled by the service and with affiliation with Be One Solutions, a pen test report was requested. The report was executed from an outsider's perspective without privileged access in any way. A regular user account was created through their registration page and access was granted.

During the exploring of all the available endpoints, it was possible to identify that the transfer id and user id was part of the URL, so, multiple requests were performed for transfer ids and user ids other than the ones performed by our user account. These requests successfully returned data which posed a major flaw in the core of the service.

Thankfully, after delivering the first report to the partner entity, they immediately applied a fix that solved the issue, and after another round of attempts, no further issues were discovered. The response to the audits performed was overwhelmingly positive.

My other functions at the company were focused on the development process of the company's time and expense tracking, project management, and billing solution, Billingo.

The solution built in PHP, follows and Model-View-Controller (MVC) pattern through the usage of the Yii2 PHP framework.

The development planning of all the applications on the company is aided by the usage of Jira, an Issue and Project tracking software provided by Atlassian, while all of the code is managed through Git by using Atlassian's Bitbucket platform. Throughout the stages of a project, the solution being developed can be in three different types of environment: Development, User Acceptance Testing (UAT), and Production. The deployment to the different environments can be aided by tools such as a Jenkins server, which provides a simple way to perform specific tasks to automate deployments. This type of tool is a key component of a Continuous Integration and Continuous Delivery (CI/CD) workflow [5].

By working as a team in a project of such a large scale, my Git understanding has greatly improved, and I became aware of the importance of proper project documentation and code readability.

In summary, Be One Solutions is a medium size company with around 120 workers, that operates globally. There is an office in Portugal which is the one where I was delegated to. Despite not being familiar with the SAP line of products, I was still welcome and been provided with the required training and knowledge transfer. I was assigned to assist a senior developer in development of the internal time and expense tracking, project management and billing tool but soon after joining the company he left leaving me with the responsibility of keeping the project up and running. Some months afterwards I finally had a team supporting me in development and I was put in charge of leading the project. During the internship period I worked on many security related tasks, one of which was the development of an enterprise grade password manager which I will be discussing in the following chapters.

3. Critical analysis and improvement measures proposal

When learning about the company's work conduct, internal procedures, and the usual workload, I've encountered some concerns regarding the way some team members managed their access credentials.

3.1. Problem

Due to the nature of the company's work, many times the team must handle customer credentials and other sensitive information. This information should be properly secured, unfortunately, since the way of storing it is up to the team member, most of the time leads to poor security practices for sake of convenience [6]. As the company operates in a mostly remote work manner, this type of behaviour is amplified by the lack of visibility over the team member's workspace.

As a company that operates in the IT business space, and with the confidentiality of the information is handled, the company and its employees should strive to maintain the best practices in order to keep up to the upmost security standards. Especially when considering that the customers we mostly interact with are multinational companies whose confidential information is very valuable and desired by third parties [7]. As such, we should always be concerned with possible corporate espionage, spear phishing and many other threats.

In the following subchapters, some common and fictional scenarios will be described in order to illustrate how the company operates, as well as how the carelessness of a team member could impact negatively the company's image and reputation.

3.1.1. Scenario 1 – Credential sharing

A team member named Paul is assigned to a project in which he is given access to the customer's development environment. The customer's IT department sends an email containing the credentials for their company's VPN, along with the credentials for RDP into the development Windows machine, as well as some generic credentials for accessing a database.

Paul is going on vacations, but another team member, Rui, is assigned in order to keep development going. The customer's IT department sends him an email containing the VPN

and RDP credentials but the credentials for the database access are missing. In order to avoid having to exchange emails back and forth with the customer's IT department, which could slowdown development and impact the project's timeline, Rui sends a message on an instant messaging platform, such as Microsoft Teams or Skype, to his colleague who is going on vacations asking for the database credentials. The credentials are sent either through the same platform or by forwarding the original email also containing the VPN and RDP credentials. In this case Paul chose the first option because he is aware of the implications of the second option, but there could be another team member unaware of the security implications of proceeding with the second option.

Since it's not convenient to have to search for emails containing the credentials and connection information, the Rui saves this data in his personal OneNote in a folder specifically for this type of information.

3.1.2. Scenario 2 – Leaving the company

Continuing from the previous scenario, Paul returns from vacations and finds out that Rui has not made any progress since, and after some internal discussions, management decided to let go Rui.

Rui was unhappy with the decision and after finding out that he still has access to the customer's system, he decides to cause as much damage as he can, hindering his previous colleagues work and damaging the company's image.

3.1.3. Scenario 3 – Stolen laptop

A different team member, Jean, has been working and managing another customer project and was time to deploy it onto production, or as we call it 'go-live'. For the go-live, the customer has requested Jean to be on site, and so he packs everything and travels from France to Germany (customer's headquarters). After arriving he waits for a taxi when suddenly a thief steals his backpack which carried his laptop. While the laptop is password protected, the hard drive was not hardware encrypted leading to all the data in it being compromised.

3.1.4. Scenario 4 – End of services

Continuing from the previous chapter, after the go-live, the customer no longer wants to be affiliated with the company and requests a report containing all the users who had access

to their sensitive information while also requesting deletion of all confidential information other than contractual data, such as passwords, access tokens, IP addresses domain names, and other documents. While this is not under the GDPR's right of erasure since both parties are companies [8], the NDA agreement conceived during the initial contract can allow for such requests.

3.1.5. Conclusions

Regarding scenario 1, the major concerns were the sharing of confidential information through unsecure channels of communication. While there isn't much that can be done to the way the customer shares the information with us, the way we share it internally is our responsibility, thus a secure way of sharing this information is a must. In this situation, an enterprise level password manager would be well suited. Also on scenario 1, Rui should have at least saved this information under the company account's OneNote instead of his personal account.

The second scenario although unlikely, it is a possibility, thus it is important to communicate to the customer whenever one of our team members stops working with us so that their IT department can revoke access to said team member. It would also be beneficial for the company to have a centralized place to store customer and internal credentials as it would be easy to revoke access if necessary.

There is a lack of means to allow the team to properly handle this information, and that's why research on company security policies and enterprise grade password managers was performed.

3.2.Solution

With the main issues well identified, the next step is to identify a solution that can cover most of the issues. To this effect, a meeting with the company's head of IT was scheduled to present the issues and to identify the requirements for the ideal solution.

The head of IT was aware of most of the issues, however lately the company has been oversaturated with work leading to little time to be dedicated to the research of proper solutions. So, after some brainstorming, we've managed to compile a list of requirements and "nice to have's". The list was as follows:

- Ability to secure identifiers, passwords, and access tokens.
- Ability to relate documents/instructions to those credentials.
- Hierarchy for credentials based on Customer > Project > Team member assignment.
- The IT Manager would have a full access account.
- Centralized on premise system.
- Brute force protection and IP block mechanism.
- Access log for login, login attempts and credential view/decrypt.
- Ability to remove all private information regarding a customer upon request as well as providing a log of every access to those customer's credentials.
- Automatically grab hierarchy of Customer > Project > Team member assignment from our in-house time control and project management solution.
- Full encryption of the database with periodical key rotation would be a plus.

3.3.Market Analysis

With a clear idea of what the solution to this issue would look like, a market analysis was conducted to search for an of the shelf solution. There are many password managers that are focused on password sharing, below follows a compilation of solutions that followed most of the requirements established in the previous chapter.

- Thycotic Secret Server.
- IT Glue.
- Clickstudio Passwordstate.

3.3.1. Thycotic Secret Server

This solution was enterprise focused, it had AD and SAP integration which for the company could be very valuable since we deal with both technologies, it also possessed file attachment support which would be very useful to provide context to the credentials. One other feature was RDP and PuTTY support which could lead to better usability. It also provided auditing and reports based on many statistics related with sharing and viewability of credentials.

While pricing is not displayed in their website, it is possible to submit a form contacting them for an estimate. Upon checking the feature comparison chart in their website [9], it was

possible to exclude the ‘Free’ and the ‘Vault’ options since we would need more than 25 users, leaving the ‘Professional’ and ‘Platinum’ options. After some quick research there was some instances reporting prices starting from 5.000€ for a basic cloud configuration to 63.519,00 € a year for unlimited users and credentials for an on-premise setup which was a preference. This solution was used previously by the IT manager in his previous company, and he also confirmed that he remembered it being quite expensive and the UI was also very clunky and not user friendly so it could drive some people to not using it.

While it posed many positive points, the negative points were too overpowering which led to this option being discarded.

3.3.2. IT Glue

Like the previous solution, IT Glue is also focused on the enterprise market and so it offers robust access control, ability to attach documents, auditing and reports, and many other features that may or may not be beneficial to us. Pricing for basic package starts at 29€ per user per month and with an estimate of 150 users, it would lead to 52.200€ a year [10]. With this option also being very expensive and without an on-premise option, it was also discarded.

3.3.3. Clickstudio Passwordstate

This solution is also enterprise focused, it possesses good role-based access control, it has the ability for a user to request access to a certain resource, and it also has auditing capabilities.

This solution seemed to be the most cost effective at 33$ per named user a year, up to 199 users on a regular license, or unlimited users on an enterprise license. Both the regular and the enterprise licenses allow for a production installation. For unlimited production installations under the same company there is the ‘global’ license option. Most of the cost is a once off cost which can go up to 6,270$ on a regular license, always 6,270$ on an enterprise license and 16,610$ for a global license. Software updates and support are part of an annual cost which on a regular license can go up to 1,254$, always 1,254$ for an enterprise license and 3,322$ for a global license [11].

While in contrast to all the previous options this seems to be the least expensive, upon some research on the quality, news regarding a security breach showed up. According to a news article published by Ars Technica on the 23rd of April 2021, 29,000 users of

Passwordstate password manager downloaded an update that included malware which extracted data from the app and sent it to an attacker-controlled server. The attacker exploited the update mechanism of the app and by injecting malicious code into a DLL, which made the app send back runtime data to the attackers [12]. The severity of the issue was more impactful than the lower price leading to this solution being discarded.

3.4. Conclusions

In the company, there is the need to operate in the customers' servers which leads to the usage of multiple credentials. Team members are expected to properly secure these credentials, however due to most of the work being performed remotely instead of in an office environment, there is a lack of visibility over their behaviour which leads some of them to seek convenience over proper secure solutions. Some scenarios are presented in which the company's image is always impacted negatively.

After a meeting with the IT manager, it was possible to identify the core problems, and a list of proposed solutions and requirements was made. With this list it was possible to proceed to the next step, a market analysis.

Most of the mentioned solutions had a very steep price to adoption and a significant price for continuous support. The ones that had a more reasonable price, had been affected by a severe vulnerability that led to leaks their customers' data. It was clear that a different approach had to be taken thus the idea of a custom solution started to emerge.

4. Literature Review

This chapter provides the definitions and context necessary to better understand the terminology and concepts used throughout the document.

4.1. Access control

In computer security, when referring to the broad topic of 'access control', it usually refers to other two terms, authentication, and authorization.

Authentication is, in simple terms, validating the identity of someone. Applying this concept in a server is to have the server or a 3rd party, validate that the client behind a request is who it claims to be. However, a client can be authenticated and still be rejected by the server from being able to access specific resources.

Authorization is what determines if an actor is able to perform a certain action on a determined resource, whether it being accessing a certain page on a web server, creating a new resource or deleting / modifying an existing one.

4.2. Cryptography

There are two main types of cryptographic functions: hash functions and ciphers. Hash functions are one-way functions meaning any data that goes through it is impossible to recover its original state, while ciphers can encrypt data and decrypt cipher data.

4.2.1. Hash functions

Hash functions are used to map data of arbitrary size into a fixed size value called hash. Hash functions are one-way functions and there is no way of recovering the original data other than breaking the algorithm or by using rainbow tables. Rainbow tables are tables created by running dictionary of words or other stings through a hash function and saving the input and the output into it [13]. While hash functions have predictable outputs, it is possible to add a random string as an additional input called salt to further hinder the generation of rainbow tables for attackers. This salt must be safely stored so that it is possible to run the hash function with the same input parameters plus the salt in order to reach the same hash. There is also another type of salt that is secret, usually referred to as pepper, that

is a fixed string, unlike salt, and it is not saved in the database in an attempt to mitigate hash regeneration through a database leak [14].

The norm for password storing is to run the password through a hashing algorithm and storing the result in the database. This way the service providers have no way of reading user passwords while still being able to authenticate them. However, in the context of this project, a hashing algorithm would only by applicable for user authentication. For storing passwords that can be read, it is necessary to apply an algorithm with that employs two-way functions.

4.2.2. Ciphers

Ciphers are two-way functions that allow the cipher data to return to its original state. To recover the data's original state, it is necessary to also have key or keys used during the encryption process. Ciphers can be characterized by two different criteria, by type of key used, and by the type of input data. When characterizing by type of key, ciphers are divided into symmetric key algorithms and asymmetric key algorithms. When characterizing by the type of input data, ciphers are divided into block ciphers and stream ciphers.

4.2.3. Symmetric key algorithms

Symmetric-key algorithms are ones that use the same key for both encryption of text and decryption of cipher text. If the same key is used to encrypt multiple instances of data, and we want to provide access to one of those instances to someone, they would also be able to access other instances due to them sharing the same key. Symmetric-key encryption can use either stream ciphers or block ciphers.

Stream ciphers are used to transform a stream of plain text combined with a pseudorandom cipher digit into a stream of cipher text. Each text digit is encrypted one at a time, usually as a XOR operation applied to bits. Stream ciphers were primarily used for ciphered data transmission, a notable example was the usage of Rivest Cipher 4 (RC4), for the WEP protocol used in Wi-Fi connections, and TLS/SSL used in HTTPS. However, RC4 has been proven to be insecure, with it being outright forbidden by the IETF [15], and the most prominent stream cipher used nowadays is ChaCha20 a variant of Salsa20 [16].

Block ciphers on the other hand, work by ciphering fixed blocks of data, and whenever the data to be ciphered is smaller than the cipher block, padding is applied in order to meet the block size. Padding can be obvious to detect unless random padding is introduced, a technique that introduces random characters and at the end of the block specifies the length

of the padding [17]. Block ciphers can also be used in other ways depending on their mode of operation. A block cipher mode of operation is an algorithm that features the use of a symmetric key block cipher to provide an information service, such as confidentiality or authentication [18]. Notable examples of symmetric block cipher algorithms are: Data Encryption Standard (DES), Advance Encryption Standard (AES), and RC5.

Stream ciphers are typically faster than block ciphers, however they are much harder to implement correctly. Tampering with the cipher text in stream ciphered data is much less destructive compared to tempering with block ciphered data. Since stream ciphered data is encrypted digit by digit, randomly changing some bytes or characters in the cipher text would only result in part of the text being unrecoverable, while in block ciphered data it would render the whole block unrecoverable [19].

4.2.4. Advanced Encryption Standard (AES)

The AES algorithm, initially named Rijndael algorithm, is the core algorithm used in the Advanced Encryption Standard specification introduced by the National Institute of Standards and Technology (NIST), in the Federal Information Processing Standards Publication 197 (FIPS 197) [20]. AES is an iterative round based symmetric key block cipher that supports a key size of 128, 192 and 256 bits, and a block size of 128 bits. The cryptographic strength is directly proportional to the key's length, and the number of iterations to be performed increase along with the size [21].

The byte is the basic unit for processing in the AES algorithm so a sequence of eight bits is treated as a single entity. Operations inside the algorithm are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing *Nb* bytes, where *Nb* represents the block length divided by 32, in the AES specification *Nb* is always 4 as 128 bits divided by 32 bits equals 4 bytes.

Table 1 - AES State array representation [20]

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

As represented in Table 1, each cell represents one byte, and four bytes in each column form 32-bit words. Each word can be represented in the following way:

Table 2 - AES State as an array of columns [20]

$W_0 = S_{0,0}\ S_{1,0}\ S_{2,0}\ S_{3,0}$	$W_2 = S_{0,2}\ S_{1,2}\ S_{2,2}\ S_{3,2}$
$W_1 = S_{0,1}\ S_{1,1}\ S_{2,1}\ S_{3,1}$	$W_3 = S_{0,3}\ S_{1,3}\ S_{2,3}\ S_{3,3}$

Using the notation from Table 2, the State can also be represented as a one-dimensional array of 32-bit words, $S_t = W_0\ W_1\ W_2\ W_3$.

As previously mentioned, the AES algorithm supports keys of length 128, 192 and 256 bits. This key length is represented by $Nk = 4$, 6 or 8 respectively, and it reflects the number of 32-bit words (number of columns) in the Cipher Key. The number of rounds to be performed during the execution of the algorithm is dependent of the key size and is represented by Nr, where $Nr = 10$ when $Nk = 4$, $Nr = 12$ when $Nk = 6$, and $Nr = 14$ when $Nk = 8$.

Table 3 - Key-Block-Round Combinations [20]

	Key Length (*Nk words*)	Block Size (*Nb words*)	Number of Rounds (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

The algorithm has two main operations, the Cipher, and the Inverse Cipher.

```
0 Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
1 begin
2    byte  state[4,Nb]
3
4    state = in
5
6    AddRoundKey(state, w[0, Nb-1])
7
8    for round = 1 step 1 to Nr-1
9       SubBytes(state)
10      ShiftRows(state)
11      MixColumns(state)
12      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
13   end for
14
15   SubBytes(state)
16   ShiftRows(state)
17   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
18
19   out = state
20 end
```

Figure 1 - Pseudo Code for the Cipher [20]

Using Figure 1 as reference, the Cipher operation starts by copying the input into the State array (line 4). After an initial Round Key Addition (line 6), the State is transformed by implementing a round function that runs Nr-1 times from round = 1 to Nr-1 (line 8 - 13) plus a slight variation of the round function one last time. The round function is composed by four other functions all of them with the State as an input, with them being Byte Substitution (line 9), Row Shifting (line 10), Column Mixing (line 11), and another Round Key Addition (line 12). Finally, after all the iterations, a final subset of operations from the round function are performed (line 15 - 17) and we end up with the output (line 19).

```
0  InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
1  begin
2     byte  state[4,Nb]
3
4     state = in
5
6     AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
7
8     for round = Nr-1 step -1 downto 1
9        InvShiftRows(state)
10       InvSubBytes(state)
11       AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
12       InvMixColumns(state)
13    end for
14
15    InvShiftRows(state)
16    InvSubBytes(state)
17    AddRoundKey(state, w[0, Nb-1])
18
19    out = state
20 end
```

Figure 2 - Pseudo Code for the Inverse Cipher [20]

Using Figure 2 as reference, the Inverse Cipher operation starts by copying the input into the State array (line 4). After an initial Round Key Addition (line 6), the State is

transformed by implementing a round function that runs Nr-1 times from round = Nr-1 to 1 (line 8 - 13) plus a slight variation of the round function one last time. The round function is composed by four other functions all of them with the State as an input, with them being an inverse Row shifting (line 9), inverse Byte Substitution (line 10), Round Key Addition (line 11), and an inverse Column Mixing (line 12). Finally, after all the iterations, a final subset of operations from the round function are performed (line 15 - 17) and we end up with the output (line 19).

AES is a very strong cipher especially when used with 256-bit long keys, and it is used by a multitude of software solutions nowadays. One good example of this is the BitLocker feature of Windows 10 [22].

4.2.5. Asymmetric key algorithms

Public key encryption or Asymmetric Encryption is a cryptographic system that uses a pair of keys. Generally, this pair contains a public key known to every relevant party, and a private key which is only known by the owner. The encryption process in this type of algorithms works by using the recipient's public key to encrypt the desired message, then only the recipient themselves can decrypt the message by recurring to their own private key. The sender is also able to digitally sign the message by using his private key, then the recipient can validate the signature by using the sender's public key. A popular example of this type of algorithms is the Rivest–Shamir–Adleman, more commonly known as RSA, introduced in the Public-Key Cryptography Standards (PKCS) #1 specification [23].

4.2.6. Elliptic-Curve Cryptography

Elliptic-Curve Cryptography, ECC, is an encryption model that uses the characteristics of special mathematical curves to generate keys. There are many elliptic curve functions however only some are suited for application in ECC. There was an initial preposition by NIST in the FIPS Publication 186 [24], however many security experts raised concerns regarding potential exploits devised by the United States National Security Agency, NSA, and so only some of the curves were accepted and published under the Request For Comments (RFC) issue 7748 titled 'Elliptic Curves for Security' [25].

4.3.HTTP

The Hypertext Protocol more commonly known as HTTP, is an application layer protocol, that is the standard of the modern web. It was initially developed by Timothy Bern-Lee at CERN in 1989 [26], and it was later standardized with the official release being HTTP/1.1 in 1997. This protocol became widespread and is the foundation of the internet as we know it today. Later in 2015 version HTTP/2 was released and introduced many improvements over its predecessor.

With the popularity of the internet and its exponential growth it became clear that it was necessary to have control of who access certain information. Fortunately, in the original release of the protocol, a special header was introduced to this effect, the 'Authorization' header. There are many ways to utilize this header nowadays however at the time options were more limited, with one of them being 'Basic Authentication'.

4.3.1. Basic Authentication

Basic Authentication was one of the earliest ways of web services authentication, it leveraged the "Authorization" HTTP header, and it was sent encoded in base 64 [27]. This is method of authentication is considered unsecure as this data is sent in every request. This method should not even be considered using if not under HTTPS [28].

4.3.1. Cookies

The 'Cookie' and 'Set-Cookie' headers were defined as part of the HTTP State Management Mechanism and using the 'Set-Cookie' header field, an HTTP server can pass a name – value pair and associated metadata that can be used to employ authorization, client identification, or used to store session settings.

Cookies have many attributes, the 'Expires' and 'Max-Age' attributes both refer to the validity of the cookie. The 'Domain' attribute specifies the host to which the cookie will be sent to, this header depends heavily on the user agent's implementation. The 'Path' attribute limits the cookie to a set of paths in the HTTP server. The 'Secure' attribute is a flag that signals the user agent that the cookie can only be sent to the specified domain if the connection is over HTTPS. The 'HttpOnly' attribute is a flag that signals the user agent to limit the scope of the cookie to HTTP requests only, meaning that JavaScript would not be able to access it [29].

4.3.2. Bearer Token

The Bearer Token is part of the OAuth 2.0 Authorization Framework, and it is a way telling the server that a client has access over a certain resource without the client having to send the username and password every request like in the Basic Authentication protocol. It uses the same 'Authorization' header as the Basic Authentication protocol and the usage of HTTPS with this protocol is also mandatory [30].

4.3.1. JSON Web Tokens

JSON Web Tokens, JWT, is a compact format used for authentication in space constrained environment such as HTTP Authorization headers and URI requests. A well formed JWT is composed of three concatenated Base64url-encoded strings separated by dots. The first section, JOSE Header, contains metadata that describes the type of token and the cryptographic operations and algorithms applied to it to secure its contents. The second section is the JWS payload, a set of claims that contain security statements such as the identity of the user and the permissions they possess, this set of claims is verifiable through the last section which is a JWS signature. This signature is a way of verifying if the token is trustworthy and has not been tampered with [31].

4.3.2. X.509 certificates

The purpose of a X.509 certificate is to validate the identity of a party, the most common usage is to validate a server's identity so that the client can know that the server is trustworthy and then establish a HTTP connection over TLS, HTTPS [32].

This type of certificate contains a public key that is certified by a Certification Authority (CA), this authority can also be certified under other authorities creating an authority chain. The certificate also contains its expiration date as well as a signature of the CA validating the certificate itself. There are other fields such as the version field, used to describe the version of the certificate, and the serial number field [33].

4.4.Proxy and Reverse proxy

A proxy server is a server application that acts as an intermediary between a client performing a request and a server replying with the response. There are many types of proxies, the most common being anonymous proxies which are generally used to bypass country restriction measures or to conceal an IP address. Another common type of proxy is

a reverse proxy, this type of proxy appears as a normal server to the client, but it forwards all the requests to another internal web server. This type of proxy can provide many benefits face having the web server directly exposed such as, providing encryption in the form of X.509 certificates and providing a secure communication channel between the client and the web server; it can also be used to perform load balancing by serving as the central point to the application while surrogating a request to a number of available web servers depending on their load. A reverse proxy can also be used for caching [34]. NGINX, pronounced engine x, is a popular software solution to act as a reverse proxy.

5. Custom Solution

After investigating the most prominent solutions and failing to select one that would be valuable while being cost effective, it was determined that a Proof of Concept for a custom solution should be developed for the duration of the internship. Amidst this there were some unforeseen circumstances that severely limited my allocation at internship related tasks. Just one month after joining the company, the lead developer of the project I was helping on, left the company and all the support and feature development fell on me. With a still unfamiliar codebase on my hands, it was very difficult to dedicate any time to any tasks related to my master's degree. Fortunately, about four to five months afterwards I finally had a team supporting me on the project leaving me with a more open schedule to start working on my master's degree again.

5.1. Concept

The base concept for the solution consisted in having a centralized storage system where credentials would be stored while encrypted. Only users with the clearance to access a specific credential would be able to decrypt them. The system consists in two encryption steps which are respectively symmetric and asymmetric encryption.

Generating a random encryption key (secret), and encrypting the credential symmetrically makes it so that only users with access to the secret can decrypt it. This secret could not be stored in the database without also being encrypted. In order to specify which users had access over a certain secret, it was necessary to introduce asymmetric encryption in the manner of private and public keys. Each user would have its own key pair, and so would the service itself. This would allow for the service to grab the user's public key and use it along with its own private key to encrypt the secret which only the user would be able to decrypt using their own private key together with the service's public key.

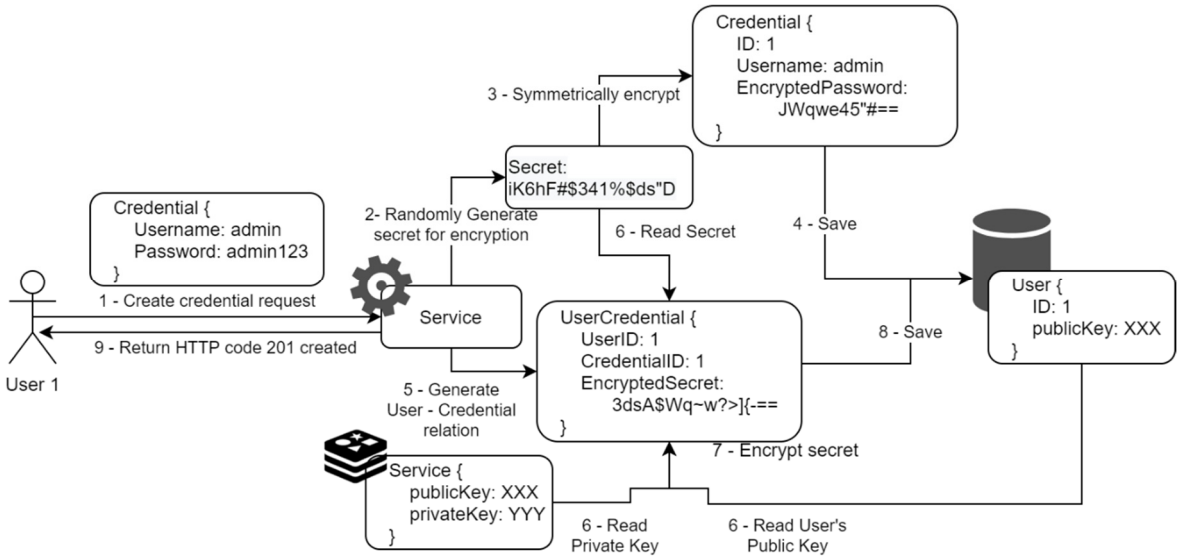

Figure 3 – EPMS Encryption Process

Figure 3 demonstrates the process of adding a new credential in the system and all of the steps involved.

It starts with the user submitting a new credential to the service. The service creates a new 'Credential' object and encrypts symmetrically the password using a randomly generated secret. The credential object which only contains an encrypted version of the password is then saved into the database. The secret is not saved into the database, but it is kept in memory to be used in the next encryption step.

After the credential is saved into the database, the service starts creating the user – credential relation objects which is always one per user, meaning that if three users were given access to a specific credential, the credential itself would be saved once in the database, but three user – credential relations would exist. This relation object contains the user ID, the credential ID and an asymmetrically encrypted version of the secret from the previous step. The asymmetric encryption uses the user's public key, which is stored in the database, and the service's private key, which is kept in cache, more specifically in a Redis instance. This relation object is then saved into the database.

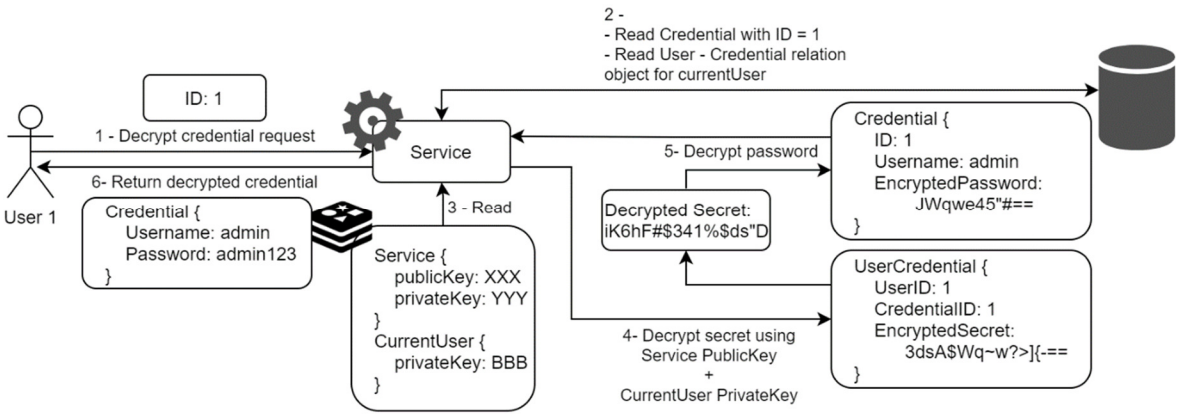

Figure 4 – EPMS Decryption Process

Figure 4 demonstrates the inverse process of Figure 3, which occurs when a user requests decryption. The service will start by reading the requested credential from the database along with its user relation object. Once the service fetches the data, it will read from the cache, the user's private key plus its own public key, and it will use them to decrypt the secret stored in the user – credential relation object. Once the secret is decrypted, the service will then proceed to use it to decrypt the credential itself and it ends with the decrypted credential being sent to the user.

As described in Figure 4, the private keys are kept in cache, more specifically in a Redis instance. The service's public / private key pair always exists in cache, while a user's key pair will only exist while a session for that user is active. The key pairs are differentiated from a fixed seed, this way it is possible for the keys to only exist in memory. This seed could be the user's password, or it could come from another source such as a private variable in the Active Directory under each user.

5.2.Implementation planning

With the main concept planned out, the next step would be implementing it. But before writing any code, it was necessary to determine what technologies to use in order to build a fast and secure version of the concept.

The database would be running Microsoft SQL Server 2017 due to being a very powerful, fast and fully featured database engine. It also supports full database encryption which would provide a second layer of encryption for all our data. It is also one of the primary database engines used for SAP Business One so it's widely used in the company meaning that I would be able to easily deploy a new database in one of many instances we have running for fast prototyping.

For the frontend I chose React.js as it's a JavaScript framework that I'm very familiar with together with SAP Ui5 web components in order to provide a familiar look and feel for our team members. The HTTP server serving the static HTML, CSS and JavaScript files would be an NGINX instance with TLS v1.3 enabled. NGINX also allows fine control over the SSL cyphers to use which could be useful to mitigate newer exploits.

The backend was the least straightforward decision as many options were on the table, with them being PHP (Yii2), JavaScript (Node.js), C# (ServiceStack), and Go (Golang). Initially PHP using the Yii2 framework seemed a faster way to start working on the PoC since this framework is the same used in the main internal project I've been working on while in the company thus being quite proficient in it, however the limited selection of cryptography functions led me to discarding it. Node.js would be the next option as I also had some exposure to it and it's excellent to write fast prototype code. While Node.js seemed a good candidate, I was worried with performance more specifically in the encryption / decryption process. Internally I also had been working with ServiceStack, a C# framework for developing HTTP APIs and with C# being a very powerful language, it was a strong contender. Golang was suggested by a work college, and it is also used internally on some projects. Despite being a newer language, it's been gaining ground for high traffic websites [35], and it's considered one of the fastest languages for web services [36].

After analysing the options, it seemed clear that a strongly typed language would be preferable since a compiler would help us prevent potential bugs increasing code stability. Another benefit of C# and Go would be having garbage collection which would contribute

to preventing memory leaks and add to memory safety [37]. Since PHP and JavaScript are weakly typed languages, they were discarded from the options. Between C# and Go, it was a matter of analysing the available cryptography libraries and, in this scenario, Go was superior as it had a very promising library that would allow the usage of the differentiable keys. Another big influence that made me decide to implement Go, was the fact that the public reception of a newly developed product in Go, the Dealership Management System (DMS) was being very positive, thus influencing the company to shift towards the Go programming language. After speaking with the lead developer of the product, I was convinced to make the decision as he displayed an overwhelmingly positive opinion towards the technology.

For the fast in memory database, Redis seems to be the go-to solution for Go development and now as of version 6, SSL/TLS encrypted communication is supported, though it is still not part of the main distributable and needs to be compiled from source with a special flag enabled [38].

5.3.Data structure overview

One of the main functionalities to implement would be the integration with the company's internal time tracking and project management tool, Billingo. The ability to access all of the already defined project data would provide a quick and user-friendly experience when adding a new credential. This integration could be built in a modular manner so that it would be relatively easy to perform similar integrations with other project management solutions.

In order to accomplish compatibility not only with Billingo's internal structure but also with other project management solutions, a generic model structure had to be defined. Then, a connector could be built for each type of data source which would transform the data from the source into the service's own data structure.

To be able to design such a data structure, it is important to visualise the possibilities of the source data structures. One of the simplest solutions would be a Project, User and Assignment table layout. The Project table stores the project data, the User table stores the user data, and the Assignment table stores the relation between the User and the Project, from here the authorizations can be calculated. However, most project management solutions

have more complex structures to accomplish a more granular control over the resource planning and access control.

The following image illustrates a more complex data structure, which is the one currently being used in Billingo.

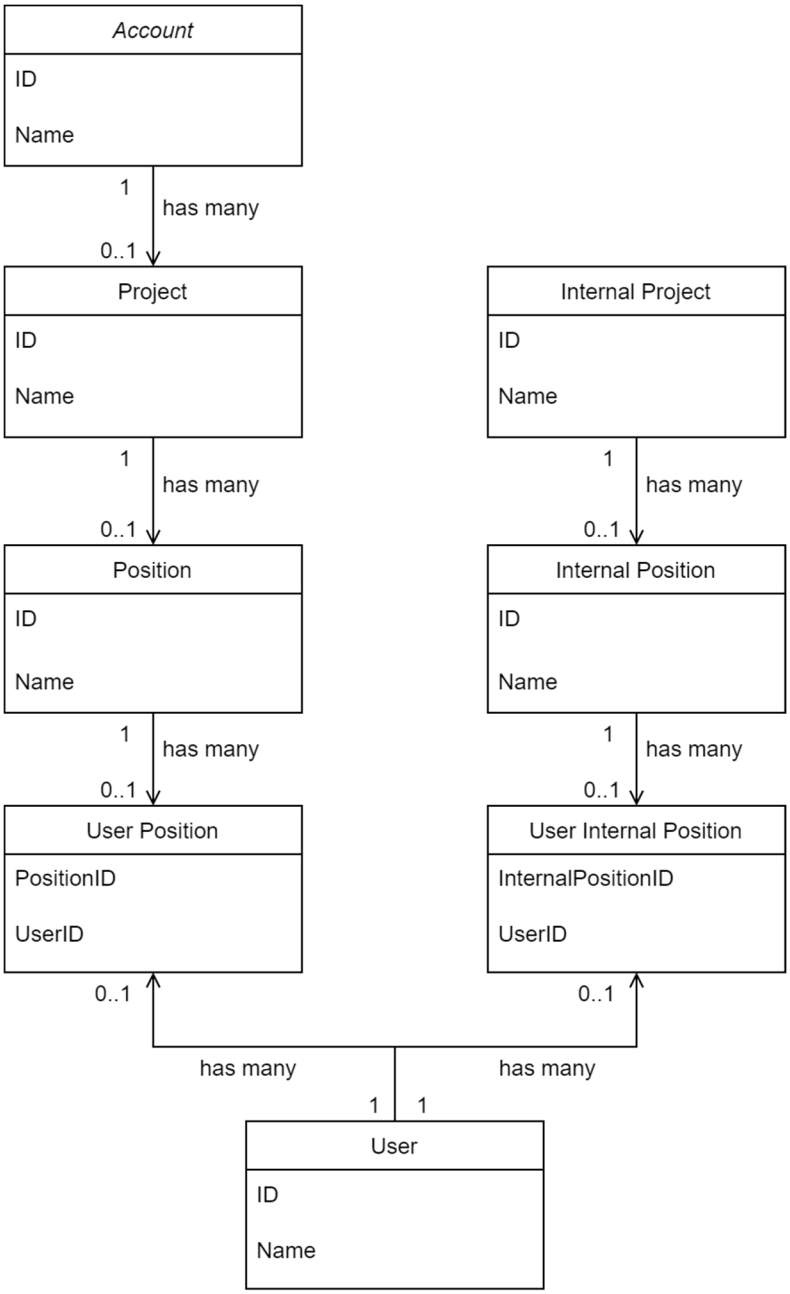

Figure 5 - Billingo's internal data structure

As illustrated in Figure 5, starting from the top left, the ‘Account’ table represents the customer. An account can have multiple projects which are stored in the ‘Project’ table, while a project can only belong to one account. To be able to charge different rates depending on the work being done in a specific project, we have the ‘Position’ table, where a project

can have multiple positions, but a position can only belong to one project. Team members are represented by the ‘User’ table. When a team member is working on a project, they are assigned to one or more positions, and a position can also have multiple team members assigned to it. This team member – position relationship is represented by the ‘User Position’ table. When the context of a project is for the company itself, then it’s considered an internal project. Internal projects are represented by the ‘Internal Project’ table and since their context is well defined, there’s no need to define an account. While internal projects could have been represented as a regular project associated to an account named after the company, special requirements for internal projects made them different enough to require a separate table. The rest of the structure is very similar to the previously described, with the ‘Internal Position’ table being for positions for internal projects and the subsequent ‘User Internal Position’ table used for assignments.

The only possible way to keep a simple data structure while still retaining the ability to accommodate data from data sources such as Billingo, was to undo its normalization. In Billingo’s data structure, from a hierarchical standpoint, we can identify three different types of objects. The ‘User’, the ‘Assignment’ represented by the ‘User Position’ and ‘User Internal Position’ tables, and the ‘Business Object’ represented by the ‘Account’, ‘Project’, ‘Position’, ‘Internal Project’ and ‘Internal Position’ tables. The ‘User’ and the ‘Assignment’ objects were very similar to their source counterparts; however, the ‘Business Object’ was an agglomerate of many fundamentally different tables.

Figure 6 illustrates the final version of the service’s data structure layout. It is composed of four main objects, ‘Source’, ‘Business Object’, ‘User’, and ‘Credential’; three relationship tables, ‘Assignment’, ‘User Credential’, and ‘Credential Business Object’; three log tables, ‘Access Log’, ‘Action Log’, and ‘Decryption Log’; an auxiliary table, ‘Attachment’; and two general service tables, ‘Blocked IP’ and ‘System Public Keys’.

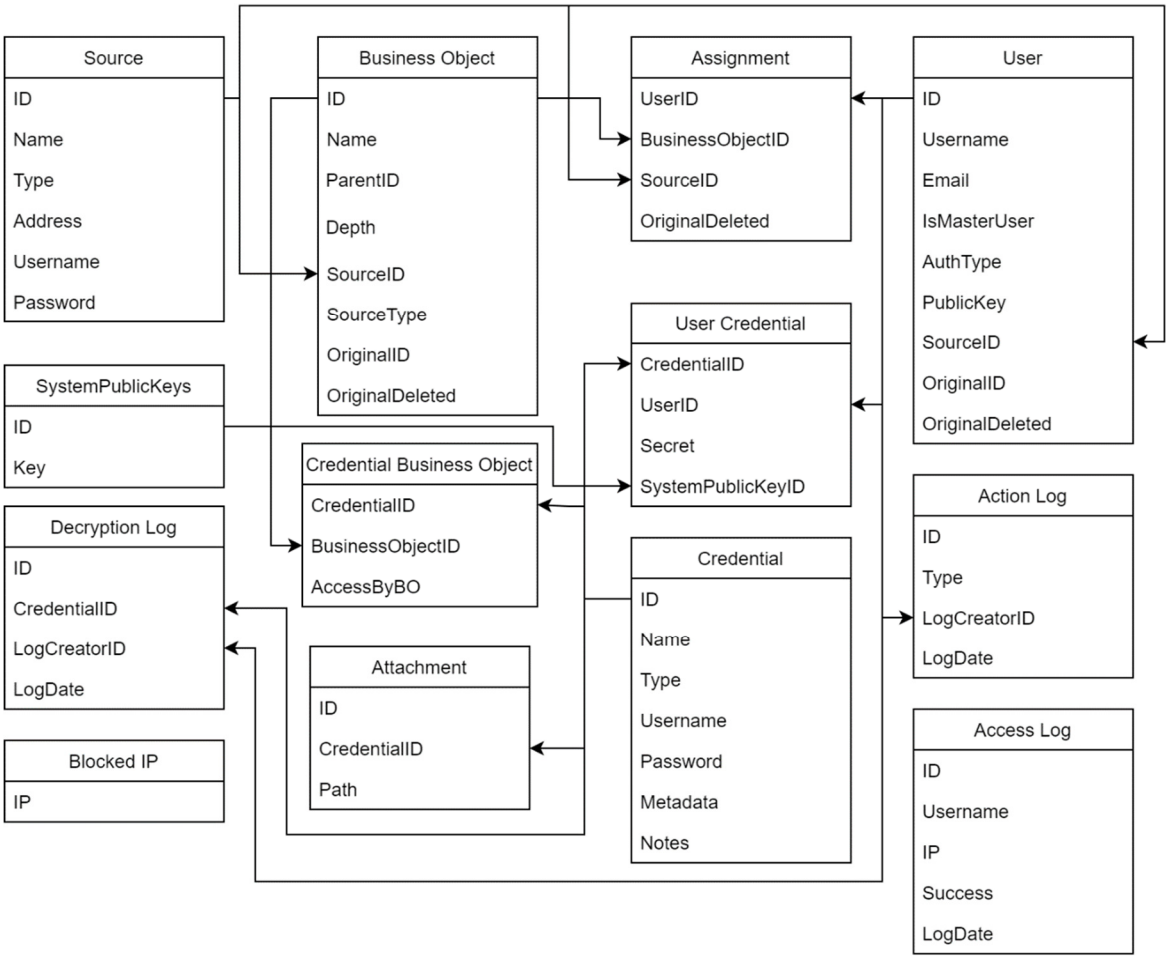

Figure 6 - EPMS data structure

The ‘Source’ object represents a possible data source where information can be extracted, converted, and merged into the service’s database. It is also where the connection details are saved to allow the service to open a connection and perform the synchronisation of data. There are three objects which support data synchronisation with them being, the ‘Business Object’ object, the ‘User’ object and ‘Assignment’ object. The tables for these objects have special fields that are fundamental for the synchronization to successfully detect existing entries and update them in case changes are detected. These fields are the ‘Original ID’, the ‘Original Deleted’, and the ‘Source Type’. The ‘Original ID’ reflects the ID of the object in the source system, this ID is not present in the ‘Assignment’ table as it is a relationship table, and its primary key is composed of two foreign keys. As such, it has a different and more complex process of synchronization. The ‘Original Deleted’ column as the name suggests, represents if the object in the source database was deleted and it can be populated with either a 0 (not deleted) or a 1 (deleted). It would be very problematic to automatically delete records while a synchronization was being performed as it could lead to cascade deletion of other

data or the halt of the synchronization process all together. This way, using the ‘Original Deleted’ field, it is possible to create a queue of deleted objects that would be processed manually by the administrator, leaving the decision of keeping or deleting the object to him. The ‘Source Type’ column is only present in the ‘Business Object’ table, and it is used to identify the type of data which the record previously was on its source database counterpart.

The ‘Business Object’ object is the single block used to build the hierarchy of customers, projects, and positions. The objects in this table reference themselves using the ‘Parent ID’ and the ‘Depth’ column is used to quickly referencing the nesting of a selected business object. Users have certain authorizations to business objects through the ‘Assignment’ table.

The ‘Credential’ object is the centrepiece of the application, as the name suggests it stores all information related to a credential saved in the system. In the ‘Password’ field, the data is encrypted using symmetric encryption. The access that a user can have over a certain credential is defined by the relationship table ‘User Credential’. This table contains the ‘User ID’, the ‘Credential ID’ and the ‘Secret’ which is the key used that was used by the symmetric encryption step during the encryption of the credential’s password. This key is encrypted using the System’s private key and the user’s public key and, to decrypt it, the System’s public key and the user’s private key are used.

During the creation process for a credential, it is also possible to select which business objects are connected to it through the relationship table ‘Credential Business Object’. This table contains the ‘Credential ID’, the ‘Business Object ID’ and a special option ‘AccessByBO’. This option is used to create extra ‘User Credential’ records based on the assignments that exist for the selected business object. This means that if this option is selected, and the user is not specified by the creator of the credential to have access over the permission, he would still have access to it.

The ‘User’ table contains all the users in the system, these users can be from multiple sources which are specified by the ‘Source ID’ column and the traceability with the source’s id preserved through the ‘Original ID’ column. At the time of this report the only authorization level for a specific user is the role of ‘Master User’ which has full control over all the Credentials in the system. Due to the nature of the application, if the database connection were to be compromised, and an attacker had set a user which he had access to as a ‘Master User’, while he would gain visibility over all the Credentials in the system, he would only be able to decrypt the password which the compromised user had access to.

The ‘Attachment’ object is used to add files that are related to a certain credential. The ‘System Public Keys’ table is used to save all of the System’s past and current public keys, this is to allow for key rotation. The remaining tables are used for logging, ‘Access Log’, ‘Action Log’ and ‘Decryption Log’ and another is an auxiliary table ‘Blocked IP’.

5.4.Cryptographic solution

With the backend service solution being written in Golang, a multitude of options are available when it comes to cryptographic algorithms [39]. The main challenge was finding an already implemented algorithm that allowed the output keys to be constant whenever the input parameters remained the same. At the time of implementation, given the unfamiliarity with the language, it was difficult to identify an official Go library that would suit these needs, and a third-party library ended up being chosen. More recently after conducting some more research and already being quite familiar with the language, some official libraries were determined to possibly being compatible but due to time constraints they are yet to be tested, the libraries in question are the ‘crypto/eliptic’ [40] and the ‘crypto/curve25519’ [41].

The library used to implement the asymmetric cryptography was the ‘nacl/box’ library [42]. Three functions were used with them being, ‘GenerateKey’, ‘Seal’, and ‘Open’. These 3 functions utilize two cryptographic algorithms with them being the ‘curve25519’ and the ‘XSalsa20’. The ‘Curve25519’ algorithm is based on the Curve25519 curve and the X25519 function referenced in the RFC 7748, Elliptic Curves for Security [43], and the official Go library, ‘crypto/curve25519’, is used for this effect.

The ‘GenerateKey’ function utilizes the ‘curve25519’ algorithm to generate a public/private key pair suitable for use with the functions ‘Seal’ and ‘Open’. This function receives as the sole parameter, a ‘Reader’ interface which is an interface that wraps a ‘Read’ method that is used to read byte slices. It is possible to have a string as an input for this function if used in conjuncture with the ‘strings.NewReader()’ function. For the purpose of the project, this input variable was named ‘seed’ and a private variable in the Active Directory which is unique to every single user was used, however the user’s password would also be a good option to be used here. The function outputs three different parameters as more than one variable as output is commonplace in Go, which are the public key, private key, and an error. As the seed remains the same, the output of this function also remains constant.

The ‘Seal’ function takes in a byte slice for the nonce, another byte slice for the secret we want to encrypt, the public key of the recipient (user), and the private key of the sender (service). The nonce is a random slice of bytes used to further increase the strength of the cypher text. The output of this function is the cypher text in byte slice form which will be converted into string and stored in the database under the ‘User – Credential’ table.

The ‘Open’ function is the opposite of the ‘Seal’ function and takes in the cypher text as a byte slice, the private key of the recipient (user), the public key of the sender (service) and the nonce. This function will output a byte slice with the clear text, or it can also output an error if anything goes wrong during the process.

The AES cipher suite was chosen for the symmetric encryption step as it was initially presented by NIST and is still recommended to use today, and by using 32-byte length keys it matches the AES-256 specification which is the strongest amongst AES.

5.5. Backend service

For the backend service a RESTful approach was taken as it cuts the ties between the backend and the frontend altogether, which leads to a more modular approach that can often translate into better stability and security. The backend service takes a stateful approach as it is necessary to keep in-memory data for a specific user, in this case his private key that was differentiated during the login process.

5.5.1. Routes

The routes are separated into two main categories, protected and public. Public routes, as the name suggests, are routes that are available to anyone without requiring any authorization and at the date of this document the only route under this category is the ‘/login’. All other routes require authorization which is carried through a cookie in each request to the server. The routes are defined using a popular and lightweight Go web framework called Chi. This framework eases the creation of routes and any middleware that we might want to apply to them. In this case, for every protected route, there is a middleware that check for the existence of a cookie with the ‘session-token’ key value pair, and after finding it, checks if this is a valid and active session. If it fails any of these two steps, the request is immediately replied with an HTTP status 401 Unauthorized. In the case of any

route, there is also a middleware that check if the requester's IP is blocked, and if it were, it would reply immediately with an HTTP status 403 forbidden. The '/login' route also applies a special middleware where if the requester fails the correct username – password combination more than five times, his IP will be put on the block list. This IP can be managed by users with admin permissions.

Since a RESTful approach was taken, the routes rely on the correct HTTP method to be considered a valid route, else an HTTP 404 is thrown. For CRUD operations, the respective HTTP methods are used accordingly, that is, Create uses POST, Read uses GET, Update uses PATCH, and Delete uses DELETE. CRUD operations are available for the main objects of the applications, Sources, Business Objects, Users and Credentials.

5.5.1. Logging

As previously mentioned, there is the existence of log tables, which used in conjunction with the middleware capabilities of Chi, allow us to quickly log certain actions performed by a user.

In the event of a source database sync, adding and removing users to access a certain credential, and re-encrypting a credential, these actions would be logged in the 'Action Log' table with the respective tags. Whenever someone attempts to login regardless of success, a record is created in the 'Access Log' table with the username entered and the IP address of the requester. If the requester attempts 5 times without success, his IP address will automatically be listed under the 'Blocked IP' table. When a credential's password is decrypted, an entry is created in the 'Decryption Log' table which contain the credential id, and the user's id. All of the log tables mentioned have a 'Log Date' column which is the timestamp of the action.

From these tables it would be possible to create a dashboard with relevant statistics and information for a certain user, such as, top 10 most accessed credentials, top 5 usernames with most login attempts, etc. I would also be possible to trace which user's had accessed which credentials, allowing for the creation of a report that could be delivered to a customer that had requested this type of information over their credentials.

5.5.2. Data management

As previously mentioned, the backend service is stateful and as such it keeps session data. Some web development oriented languages provide their own session management

solution, in PHP a session is easily manageable through the provided functions and the session related data is stored in the filesystem [44]. In Go there is no provided solution for this and the Chi framework used also doesn't provide this functionality. We have access over the request structure, and it is possible to read the cookies sent in the request and write cookies in the response, but the management of the session data is up to the developer. The most prominent solution for session data storage in Go is the usage of the Redis data structure store. While Redis is not focused on security, when compiled from source with specific flags it is possible to use TLS 1.3 in the communication channel.

All other data that is not session related is kept in a regular database under a Microsoft SQL Server instance.

5.6. Infrastructure overview

The entire solution is composed of multiple components, while most of the components run in the same environment and machine, other such as the MSSQL databases and the AD do not. As such, the connection between environments must be carefully secured to not make the entire system compromised.

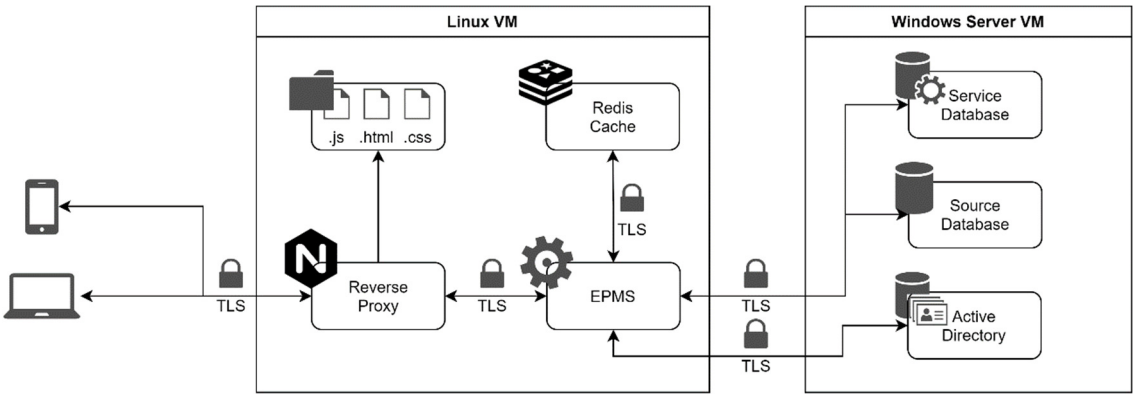

Figure 7 – EPMS Infrastructure Overview

Figure 7 illustrates the infrastructure of the entire solution. The client is an external system that resides outside Be One Solutions' network. The client, usually a browser, directly interfaces with an NGINX instance that is operating as an HTTP server and as a reverse proxy simultaneously. Connections between the NGINX instance and the client are secured through TLS 1.3 through the usage of an X.509 certificate defined in the settings of

the NGINX instance. The NGINX instance is configured to act as a reverse proxy that also secures traffic to upstream servers, this means that the connection between the client and the backend server is forced to be in TLS mode with the certificates defined in the NGINX instance [45].

The NGINX instance, the backend service, the Redis instance and the compiled client application files all reside in a Linux environment, while the MSSQL instances and the AD instance are running in other Windows Server environments inside the same network. Although they share the same network, it is important to keep all connections between these environments secure, therefore all connections are using the available encryption standards. In the case of the AD connection, it is using the Go library 'ldap/v3' and it supports TLS connections through the 'DialURL' function which supports ldaps:// schemas [46]. As for the MSSQL connection is being performed by the SQL driver for Go 'denisenkom/go-mssqldb' and it through it is possible to specify parameters to perform TLS secure connections to the database instance [47].

5.7. Client application

The clients will be consuming a production build of a React.js application. The authentication to the backend service is session based and handled through secure HTTP only cookies. While HTTP only cookies are resistant to XSS attacks [48], they also become inaccessible through JavaScript, which complicates maintaining a session on a React.js application. For this purpose, an axios instance was setup with interceptors to intercept all responses that come with an HTTP status code 401 'Unauthorized'. This allowed the application to know that the user was not authenticated and that it should present the login page.

Upon a successful login, the application keeps in memory the state of 'logged in', along with a timer that reflects the cookie's validity. Whenever this timer reaches zero, or when a request response comes with a status code 401, the application will automatically destroy the session both in memory and on the server through a logout request, prompting the user to login again. For the application to only keep the session status in memory it meant that if the user closed the tab and opened it again, the application would prompt the user to login again as the action of closing all tabs related to a domain flushes the memory thus destroying the session [49]. To overcome this issue, whenever the application is opened, it will

automatically perform a request to the server to check if the user is currently authenticated. The server can determine this without receiving any input from the user because if a session was previously opened from the user's browser, even if it was previously destroyed by closing the tab, the cookie would persist on the user's browser until the session's validity expired and thus being sent in the request [50]. If no cookie was sent, it meant that the user had no active session, leading him to the login page. This way it was possible to keep a 'logged in' state in a safer way than if it was saved on the local storage or if the cookies were JavaScript accessible, without compromising the user experience.

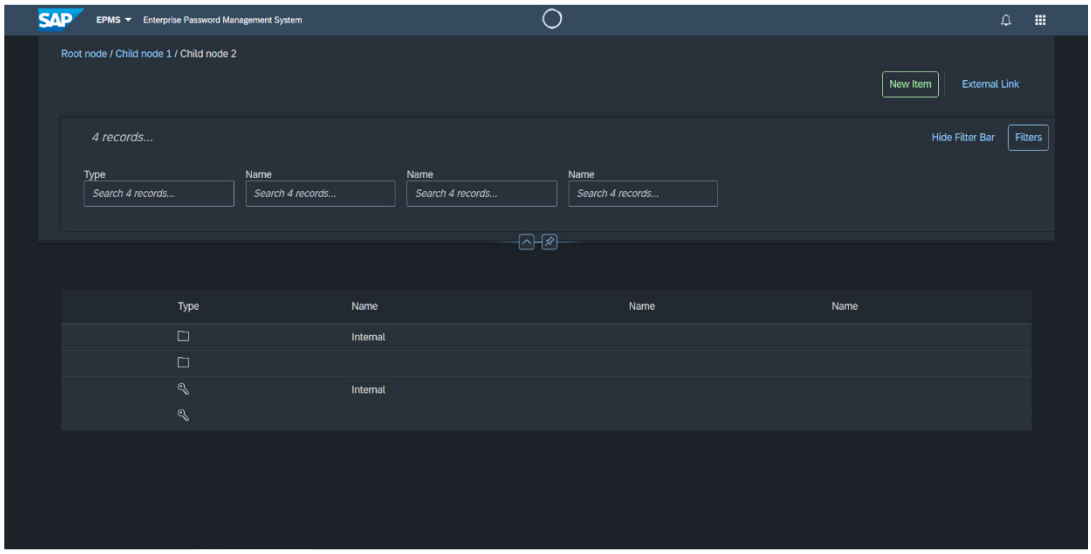

Figure 8 - EPMS Client Web Interface

Figure 8 is a screenshot of the web interface in its current state, this page is the list of resources, which combines business objects and credentials based on the depth level and Credential – Business Object relations. The framework used is SAP Ui5 react, and it provides a modern and familiar look for users that are experienced in using SAP products.

5.8. Challenges

The development of a solution of this kind is very time consuming, and if not developed under a strict planning, it can extend into a long period of little development. Initially it was planned to have a day per week dedicated to my master's activities and this day would be used mainly for developing this solution, however, due to the circumstances explained at the beginning of this chapter, this ended up not being the case and the development speed was heavily influenced.

After implementing the new data structure and after performing data synchronization form Billingo source database, it became clear that having a table with self-referencing relations would impact the hierarchy tree generation's performance quite significantly. One way to overcome this was to have a stored procedure that would take as an input the business object for the starting point, and then depending on the needs it would generate the list of ancestors or the list of children.

Another problem had to do with the permissions a user would have over a certain business object. It became clear that by having the permissions defined at the 'Position' level, would mean that it would be very expensive to compute the permissions a user would have over a particular business object's ancestor, and the same would apply to their children. Therefore, during data synchronizations, a special 'Assignment' record would be created for every ancestor or child. This was done using a trigger that would run for every assignment created or updated. Some other issues came up regarding the fact that triggers are set based and not row base [51], which led to rewriting the trigger script.

When it came to key generation, the initial plan was to use the user's password, however this would mean that every time a user changed his password, the key pair would change making it impossible for the user to access any credential which they previously had access to without decrypting and encrypting them all again through a master user. Two solutions came from this, one was to randomly generate a seed that would be encrypted using the user's password and kept in the database, the other one would be to use private properties per user from within the Active Directory to generate the seed. The latter was the chosen solution since client authentication is performed by the Active Directory and as the connection was already being established, it was just a matter of querying additional information and using it to generate the keys.

5.9.Conclusions

During the conceptual phase of the project, the encryption process which is the core of the solution was established. It consisted in two steps of encryption, one symmetric, the other asymmetric, and allowed a specific credential to be encrypted once and still be accessible by different users.

The next step was the selection of the technologies to be used for this effect. Many things were taken into consideration, the technologies I was mostly familiar with, the capabilities

of those technologies face the issue at hand, the technologies being used at the company, and the recommendations from company colleges were the main factors to the decision.

After establishing the concept and defining the technologies to be used, it was a matter of designing a data structure that could accommodate data from the company's project management solution, Billingo. This data structure had to be able to accommodate frequent synchronizations with Billingo and possibly with other project management solutions, without losing the connection to the original data. This was one of the biggest challenges on itself, but in the end, it was possible to deploy a solid solution.

With the data structure well defined, it was finally possible to start the development process for the backend service and the frontend client. For the backend service a RETSful approach was taken, and the routes were carefully established accordingly. Important actions performed by the user were saved into a set of log tables.

For the frontend application, React.js was used due to being the frontend JavaScript framework that is used in the company. By dealing with HTTP Only cookies, some workarounds had to be taken in order to keep session state at the client application.

6. Conclusion

I think my growth as a security focused developer is quite substantial after coming to work at Be One Solutions and by taking on the challenge of this project.

In regards to security, some workers need to be educated about the implications bad security practices may have on the company's image. Since most of the work performed at the company is done in customers' servers and since the company has many different customers, a multitude of credentials are used every day and storing them securely can be challenging as convenience overtakes security practices. By bringing convenience to the team members through the form of an enterprise password manager, the company can increase strengthen its security.

The market for enterprise grade password managers is substantial as well as the average price quoted for such solutions. The ones that offer a reasonable price appeared to have their solution compromised leading to huge data leaks. The conclusions from this market analysis led to the development of a custom solution.

After some research and planning, I decided to start working on a custom solution and present it as a Proof of Concept (PoC). This quickly proved to be more challenging that anticipated and due to time constraints, it is not yet completed. During the research and development, I've learned about a broad range of topics such as software development, infrastructure design and planning, and application security and broadened my knowledge on already familiar web development topics such as reverse proxies, X.509 certificates, JWTs, cookies and session management.

Currently, the PoC while not finished, it is able to perform data synchronization with the company's in house project management solution, and it also allows the creation of credentials in the system with the full encryption – decryption process working. The most time-consuming aspect of the project is developing a client interface that is user friendly and performs all the necessary tasks to fully utilize the solution feature set.

As for future work, once the PoC is fully implemented, a suite of tests would be performed to all the components of the solution. The areas to be covered would be the React.js client, the RESTful API and overall backend stability, the reverse proxy service, the

in-memory data storage (Redis), the source and service's database connection security, and the Active Directory (AD) authentication.

7. Bibliography

[1] K. A. Whitler and P. W. Farris, “The Impact of Cyber Attacks On a Brand’s Image,” *JOURNAL OF ADVERTISING RESEARCH,* 2017.

[2] be one solutions, “Meet our Lead,” be one solutions, 16 April 2021. [Online]. Available: https://www.beonesolutions.com/our-management/. [Accessed 10 November 2021].

[3] SAP, “SAP BUSINESS ONE AVAILABILITY BY COUNTRY,” 19 October 2019. [Online]. Available: https://archive.sap.com/kmuuid2/2083d347-c563-2c10-b7ad-e0796454ea8d/SAP%20Business%20One%20Availability%20By%20Country.pdf. [Accessed 10 November 2021].

[4] M. Rieger, “Integration Framework for SAP Business One (B1if) – Central Blog,” 14 August 2018. [Online]. Available: https://blogs.sap.com/2018/08/14/integration-framework-for-sap-business-one-b1if-central-blog/. [Accessed 20 November 2021].

[5] “Building CI/CD pipelines with Jenkins,” 5 September 2019. [Online]. Available: https://opensource.com/article/19/9/intro-building-cicd-pipelines-jenkins. [Accessed 29 November 2021].

[6] B. Cantafio, “Security vs. Convenience,” GIAC Security Essentials Certification, 2004.

[7] N. Hopkins, “The Guardian,” The Guardian, 25 09 2017. [Online]. Available: https://www.theguardian.com/business/2017/sep/25/deloitte-hit-by-cyber-attack-revealing-clients-secret-emails. [Accessed 17 10 2021].

[8] Intersoft Consulting, “Recital 14 Not Applicable to Legal Persons,” [Online]. Available: https://gdpr-info.eu/recitals/no-14/. [Accessed 29 November 2020].

[9] T. corp., “Secret Server Features Chart,” Thycotic, [Online]. Available: https://thycotic.com/products/secret-server/features/. [Accessed 21 10 2021].

[10] I. Glue, “IT Glue,” IT Glue, [Online]. Available: https://www.itglue.com/pricing/. [Accessed 21 10 2021].

[11] Clickstudios, “Purchase Passwordstate,” 1 01 2021. [Online]. Available: https://www.clickstudios.com.au/buy-now.aspx. [Accessed 11 04 2021].

[12] D. Goodin, “Backdoored password manager stole data from as many as 29K enterprises,” Ars Technica, 23 04 2021. [Online]. Available: https://arstechnica.com/gadgets/2021/04/hackers-backdoor-corporate-password-manager-and-steal-customer-data/. [Accessed 6 06 2021].

[13] P. Oechslin, “Making a faster cryptanalytic time-memory trade-off,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),* vol. 2729, pp. 617-630, 2003.

[14] OWASP, “Password Storage Cheat Sheet,” 28 November 2021. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. [Accessed 28 November 2021].

[15] A. Popov, "Prohibiting RC4 Cipher Suites," 01 02 2015. [Online]. Available: https://tools.ietf.org/html/rfc7465.

[16] Y. Nir. and A. Langley, “ChaCha and Poly1305 for IETF Protocols,” *IETF Request For Comments (RFC),* vol. 7539, no. 2070-1721, 2015.

[17] W. J. Buchanan, "Padding (AES)," 13 November 2020. [Online]. Available: https://asecuritysite.com/encryption/padding.

[18] NIST, "Block Cipher Modes," 06 November 2012. [Online]. Available: https://web.archive.org/web/20121106212417/http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html.

[19] Wikipedia, "Malleability (cryptography)," 13 November 2020. [Online]. Available: https://en.wikipedia.org/wiki/Malleability_(cryptography).

[20] M. J. Dworkin, “FIPS 197, Advanced Encryption Standard (AES),” *Network Security, National Institute of Standards and Technology,* vol. 197, no. 12, 2001.

[21] U. Farooq and M. F. Aslam, “Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA,” *Journal of King Saud University - Computer and Information Sciences,* vol. 29, no. 3, 2017.

[22] Microsoft, “What's new in Windows 10, versions 1507 and 1511 for IT Pros,” 25 March 2021. [Online]. Available: https://docs.microsoft.com/en-us/windows/whats-new/whats-new-windows-10-version-1507-and-1511. [Accessed 26 November 2021].

[23] J. Jonsson, B. Kaliski and R. Laboratories, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,” *IETF Request For Comments (RFC),* no. 3347, 2003.

[24] P. D. Gallagher, “FIPS PUB 186-4, Digital Signature Standard (DSS),” *Network Security, National Institute of Standards and Technology,* 2013.

[25] A. Langley, Google, M. Hamburg, R. C. Research, S. Turner and sn3rd, “Elliptic Curves for Security,” *IETF Request For Comments (RFC),* no. 7748, 2016.

[26] T. Berners-Lee and M. Fischetti, Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its inventor, Harper, 1999.

[27] J. Franks, N. University, P. Hallam-Baker, I. Verisign, J. Hostetler, I. AbiSource, S. Lawrence, I. Agranat Systems, P. Leach, M. Corporation, A. Luotonen, N. C. Corporation, L. Stewart and I. Open Market, “HTTP Authentication: Basic and Digest Access Authentication,” *IETF Request For Comments (RFC),* no. 2617, 1999.

[28] J. Reschke and greenbytes, “The 'Basic' HTTP Authentication Scheme,” *IETF Request For Comments (RFC),* no. 7617, 2015.

[29] A. Barth and U. Berkeley, “HTTP State Management Mechanism,” *IETF Request For Comments (RFC),* no. 6265, 2011.

[30] M. Jones, Microsoft, D. Hardt and Independent, “The OAuth 2.0 Authorization Framework: Bearer Token Usage,” *IETF Request For Comments (RFC),* no. 6750, 2012.

[31] M. Jones, Microsoft, J. Bradley, P. Identity, N. Sakimura and NRI, “JSON Web Token (JWT),” *IETF Request For Comments (RFC),* no. 7519, 2015.

[32] E. Rescorla and I. RTFM, “HTTP Over TLS,” *IETF Request For Comments (RFC),* no. 2818, 2000.

[33] D. Cooper, NIST, S. Santesson, Microsoft, S. Farrell, T. C. Dublin, S. Boeyen, Entrust, R. Housley, V. Security, W. Polk and NIST, “Internet X.509 Public Key Infrastructure Certificate,” *IETF Request For Comments (RFC),* no. 5280, 2008.

[34] The Apache Software Foundation, “Apache Module mod_proxy,” 2013. [Online]. Available: http://httpd.apache.org/docs/2.0/mod/mod_proxy.html#forwardreverse. [Accessed 28 November 2021].

[35] W3Techs, “Usage statistics of Go for websites,” W3Techs, 14 November 2021. [Online]. Available: https://w3techs.com/technologies/details/pl-golang. [Accessed 14 November 2021].

[36] B. Peabody, “Server-side I/O Performance: Node vs. PHP vs. Java vs. Go,” Toptal, 28 May 2016. [Online]. Available: https://www.toptal.com/back-end/server-side-io-performance-node-php-java-go. [Accessed 14 November 2021].

[37] Microsoft, “Fundamentals of garbage collection,” Microsoft, 15 September 2021. [Online]. Available: https://docs.microsoft.com/en-

us/dotnet/standard/garbage-collection/fundamentals. [Accessed 21 October 2021].

[38] Redis Ltd., "TLS Support," Redis Ltd., [Online]. Available: https://redis.io/topics/encryption. [Accessed 30 October 2021].

[39] Go, "golang/crypto," Google, 8 December 2020. [Online]. Available: https://github.com/golang/crypto. [Accessed 25 November 2021].

[40] Go, "eliptic," Google, 4 November 2021. [Online]. Available: https://pkg.go.dev/crypto/elliptic?utm_source=godoc. [Accessed 25 November 2021].

[41] Go, "curve25519," Google, 17 November 2021. [Online]. Available: https://pkg.go.dev/golang.org/x/crypto/curve25519?utm_source=godoc. [Accessed 25 November 2021].

[42] Libsodium, "Sealed boxes," Libsodium, 25 October 2021. [Online]. Available: https://libsodium.gitbook.io/doc/public-key_cryptography/sealed_boxes. [Accessed 25 November 2021].

[43] A. Langley, Google, M. Hamburg, Rambus Cryptography Research and S. Turner, "RFC7748 - Elliptic Curves for Security," January 2016. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7748. [Accessed 25 November 2021].

[44] The PHP Group, "session_save_path," The PHP Group, 2021. [Online]. Available: https://www.php.net/manual/en/function.session-save-path.php. [Accessed 28 November 2021].

[45] F5 NGINX, "Securing HTTP Traffic to Upstream Servers," [Online]. Available: https://docs.nginx.com/nginx/admin-guide/security-controls/securing-http-traffic-upstream/. [Accessed 28 November 2021].

[46] J. Weldon, "ldap," 17 July 2020. [Online]. Available: https://pkg.go.dev/github.com/go-

ldap/ldap/v3@v3.2.3?utm_source=gopls#DialURL. [Accessed 28 November 2020].

[47] denisekom, “denisenkom/go-mssqldb,” 8 July 2021. [Online]. Available: https://github.com/denisenkom/go-mssqldb. [Accessed 28 November 2021].

[48] OWASP, “HttpOnly,” [Online]. Available: https://owasp.org/www-community/HttpOnly. [Accessed 28 November 2021].

[49] Mozilla, “Window.sessionStorage,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Window/sessionStorage. [Accessed 28 November 2021].

[50] D. Kristol, L. T. Bell Laboratories, L. Montulli and I. Epinions.com, “HTTP State Management Mechanism,” October 2000. [Online]. Available: https://www.ietf.org/rfc/rfc2965.txt. [Accessed 28 November 2021].

[51] J. McLeod, “Triggers – Set Based, not Row-based,” 05 June 2008. [Online]. Available: http://www.jimmcleod.net/blog/index.php/2008/06/05/triggers-set-based-not-row-based/. [Accessed 25 April 2021].

[52] Google, "Package aes," 13 11 2020. [Online]. Available: https://golang.org/pkg/crypto/aes/.

Appendices

A. API Routes

```
Public routes:
POST    /login                          # Login/create session

Private routes:
POST    /logout                         # Logout/destroy session
POST    /active-session                 # Validate session

GET     /credentials                    # Credential list
POST    /credentials                    # Create new credential
GET     /credentials/<id>               # Fetch credential
PATCH   /credentials/<id>               # Update credential
DELETE  /credentials/<id>               # Delete credential
GET     /credentials/<id>/decrypt       # Fetch decrypted fields

GET     /business-objects               # Business Object list
POST    /business-objects               # Create Business Object
GET     /business-objects/<id>          # Fetch Business Object
PATCH   /business-objects/<id>          # Update Business Object
DELETE  /business-objects/<id>          # Delete Business Object
```

```
GET     /business-objects/<id>/children # Business Object list


GET     /users                          # User list
POST    /users                          # Create user
GET     /users/<id>                     # Fetch user
PATCH   /users/<id>                     # Update user
DELETE  /users/<id>                     # Delete user
POST    /users/<id>/generate-keys       # Generate user’s keys


GET     /sources                    # Source list
POST    /sources                    # Create new source
GET     /sources/<id>               # Fetch source
PATCH   /sources/<id>               # Update source
DELETE  /sources/<id>               # Delete source
GET     /sources/<id>/sync          # Sync source data
```