



# **Projeto SIO**

## **Entrega 3: Features e decisões**

### **Segurança Informática nas Organizações**

Ano Letivo: 2024/2025

Prof. João Paulo Barraca

#### **Trabalho realizado por:**

Rodrigo Abreu, N°113626

Eduardo Lopes, N°103070

Raquel Vinagre, N°113736

# Índice

<b>Introdução.....</b>	<b>3</b>
<b>Features e decisões.....</b>	<b>4</b>
Local Commands.....	4
rep_subject_credentials <password> <credentials file>.....	4
rep_decrypt_file <encrypted file> <encryption metadata>.....	4
Anonymous API.....	5
Verificações.....	5
Comandos.....	5
rep_create_org <organization> <username> <name> <email> <public key file>.....	5
rep_list_orgs.....	5
rep_create_session <organization> <username> <password> <credentials file> <session file>.....	5
rep_get_file <file handle> [file].....	6
Authenticated API.....	7
Verificações.....	7
Roles.....	8
Comandos.....	8
rep_assume_role <session file> <role>.....	8
rep_drop_role <session file> <role>.....	8
rep_list_roles <session file>.....	9
rep_list_subjects <session file> [username].....	9
rep_list_role_subjects <session file> <role>.....	9
rep_list_subject_roles <session file> <username>.....	9
rep_list_role_permissions <session file> <role>.....	9
rep_list_permission_roles <session file> <permission>.....	9
rep_list_docs <session file> [-s username] [-d nt/ot/et date].....	10
Authorized API.....	10
Verificações.....	10
Comandos.....	10
rep_add_subject <session file> <username> <name> <email> <credentials file>....	10
rep_suspend_subject <session file> <username>.....	11
rep_activate_subject <session file> <username>.....	11
rep_add_role <session file> <role>.....	11
rep_suspend_role <session file> <role>.....	12
rep_reactivate_role <session file> <role>.....	12
rep_add_permission <session file> <role> <username>.....	12
rep_remove_permission <session file> <role> <username>.....	13
rep_add_permission <session file> <role> <permission>.....	13
rep_remove_permission <session file> <role> <permission>.....	13
re_add_doc <session file> <document name> <file>.....	14
rep_get_doc_metadata <session file> <document name>.....	14

rep_get_doc_file <session file> <document name> [file].....	14
rep_delete_doc <session file> <document name>.....	15
rep_acl_doc <session file> <document name> [+/-] <role> <permission>.....	15
Medidas de Segurança.....	16
<b>Features a melhorar.....</b>	<b>16</b>
<b>Segurança (OWASP).....</b>	<b>17</b>
V2.1 Password Security.....	17
V2.2 General Authenticator Security.....	21
V2.3 Authenticator Lifecycle.....	25
V2.4 Credential Storage.....	25
V2.5 Credential Recovery.....	25
V2.6 Look-up Secret Verifier.....	25
V2.7 Out of Band Verifier.....	25
V2.8 One Time Verifier.....	26
V2.9 Cryptographic Verifier.....	27
V2.10 Service Authentication.....	27
<b>Bibliografia.....</b>	<b>29</b>

# Introdução

O objetivo deste projeto foi o desenvolvimento de um repositório para documentos de organizações, que possam ser partilhados de forma segura entre várias pessoas. Para o efeito, foi criado um conjunto de comandos que permitem realizar operações sobre o repositório.

O código foi desenvolvido em Python e, de modo a organizar todas as entidades e relações do projeto, foi criada uma Base de Dados relacional em SQLite3.

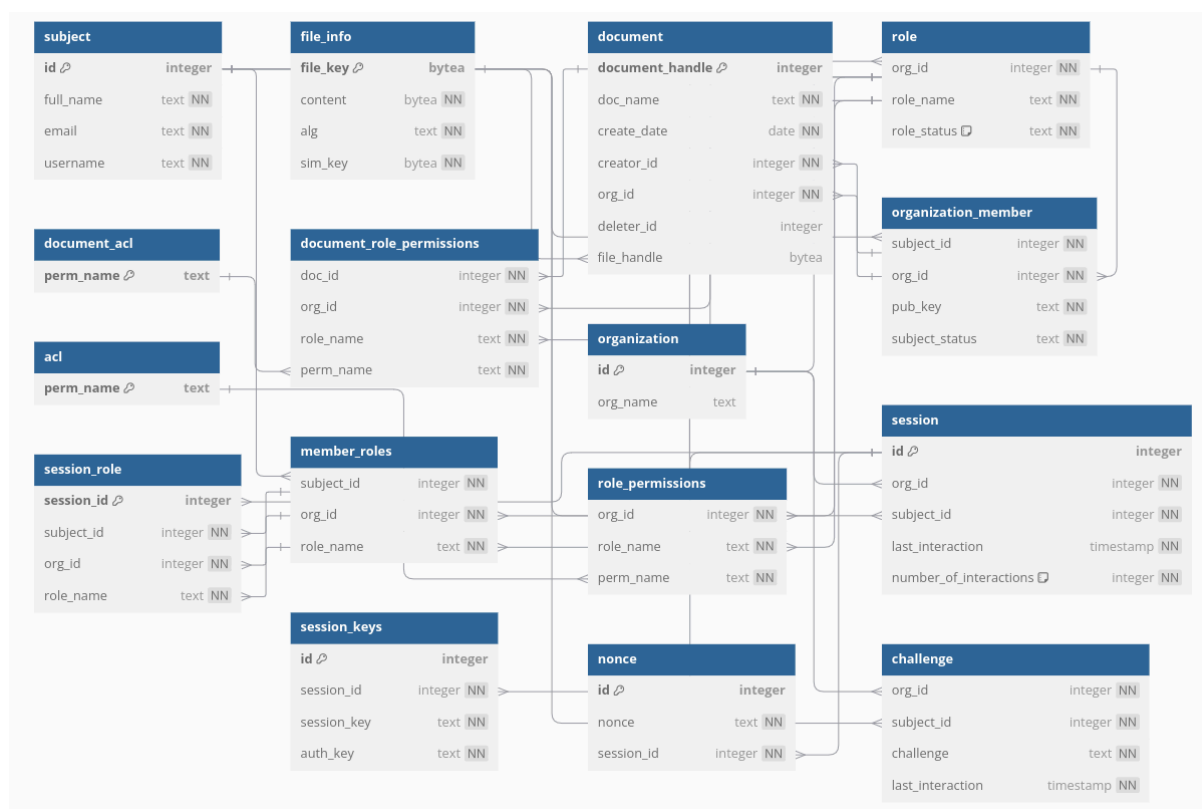


Imagem 1: Base de Dados

De entre as várias entidades, podemos salientar as seguintes:

- **Document**: contém um ficheiro (informação fundamental) e metadata (informação auxiliar). Identifica-se pelo seu *document\_handle*.
- **Organization**: tem uma lista própria de documentos, e podem ser listadas. Distingue-se pelo seu id.
- **Subject**: pessoas ou aplicações que interagem com o repositório, associadas a uma ou mais organizações. Identifica-se pelo seu id.

- **Session:** permite aos subjects interagir com o repositório. Refere-se sempre a uma organização e é criada através de login. É possível que um subject mantenha sessões simultâneas com diferentes organizações no repositório. Após algum período de inatividade, a sessão expira. É robusta contra diversos ataques (iremos mais à frente explicar como) e identifica-se pelo seu id.
- **Role:** definem os direitos que os subjects possuem na organização. Cada role apresenta um conjunto de permissões e subjects. Distingue-se pelo seu nome e organização.

Há, ainda, outras entidades que servem como auxiliares para as principais acima mencionadas.

## Features e decisões

### Local Commands

#### **rep\_subject\_credentials <password> <credentials file>**

Este comando cria uma chave assimétrica com RSA, gerando uma chave privada, encriptada com a palavra-passe recebida e uma chave pública. Estas são armazenadas em 2 ficheiros diferentes. Os 2 ficheiros são fáceis de identificar, porque o ficheiro da chave pública tem o mesmo nome do ficheiro da chave privada, mas com o prefixo '\_pub'.

A chave privada, em conjunto com a password, é utilizada para autenticar o sujeito na organização. A chave pública identifica-o na organização.

Optámos pela chave assimétrica, porque permite confidencialidade sem prévia troca de segredos, e é um método eficaz de autenticação.

Decidimos utilizar chaves RSA, uma vez que têm uma estrutura interna complexa com propriedades matemáticas específicas e permitem-nos encriptar comunicações e assinar mensagens.

#### **rep\_decrypt\_file <encrypted file> <encryption metadata>**

Este comando descripta o texto cifrado, sendo este conteúdo de um ficheiro, ou seja, passado diretamente como argumento. O algoritmo de descriptação usado é o AES CBC (também é o único usado para encriptar). Os metadados fornecidos devem ter a informação necessária para descriptá-lo (salt, iv, key).

O *file\_handle* também é fornecido para controlo de integridade do conteúdo. A função de *digest* usada para gerar o *file\_handle*, a partir do conteúdo do ficheiro, vai ser aplicada no

conteúdo fornecido, para verificar se o conteúdo foi alterado. Se o resultado for igual ao *file\_handle*, os dados não foram corrompidos.

Optámos por usar apenas um algoritmo de descriptação, porque é o correspondente ao único algoritmo de encriptação utilizado.

Usámos uma função de *digest*, porque é impossível obter o conteúdo do ficheiro a partir desta, e porque é impossível haver 2 textos diferentes com a mesma digest, logo, caso a *digest* seja igual ao *file\_handle*, podemos garantir que o conteúdo não foi alterado.

## Anonymous API

### Verificações

Para nenhum dos comandos desta secção é necessário ter uma sessão criada.

### Comandos

**rep\_create\_org <organization> <username> <name> <email> <public key file>**

Este comando cria uma organização e define o seu primeiro subject. A informação é descriptada e usada para verificar se a mesma organização já existe. Caso o criador (subject) não exista, este é adicionado à base de dados. De seguida, a organização é também adicionada à BD e o subject adicionado à tabela *organization\_member*, que contém a sua chave pública. Finalmente, é atribuído o role de “manager” ao subject, que contém todas as permissões possíveis nessa organização, sejam de roles, organização, ou de um documento criado nessa organização.

**rep\_list\_orgs**

Usado para listar todas as organizações existentes, correspondem à tabela *organization*.

**rep\_create\_session <organization> <username> <password>  
<credentials file> <session file>**

Dado um id da organização, criamos a sessão de um sujeito na respetiva organização. Isto só é possível se este sujeito tiver sido previamente adicionado à organização, ou seja, o seu criador.

O username, a password e o credentials file são utilizados para autenticar o sujeito na organização. O credentials file deve conter apenas a chave privada do sujeito para, em conjunto com a password, gerar a chave pública correspondente.

Na função *request\_challenge*, identificamos o sujeito pelo seu id, obtido com base no username com queries na base de dados. Caso o sujeito não exista na organização, ou este sujeito esteja suspenso, o início de sessão é interrompido. Caso contrário, verificamos se a chave pública guardada na BD para este sujeito é igual à que foi gerada. Se forem iguais, procede-se à verificação da validade do challenge, senão, as credenciais são inválidas e impede-se o início de sessão.

O challenge consiste num número aleatório gerado pelo repositório (através de `os.random(32)`). Este número funciona como uma OTP, uma vez que tem um período curto de vida (7 segundos) e só serve para uma função, que é a autenticação dos dois intervenientes naquela comunicação. O número é encriptado com a chave pública do utilizador e enviado ao mesmo. Para o utilizador responder ao desafio, este tem de descriptar o desafio com a sua chave privada e voltar a enviar para o repositório, mas desta vez encriptado com a chave do repositório. Se o desafio recebido for igual ao criado, o repositório gera uma chave de autenticação para a sessão, de forma a esta autenticar-se em comunicações futuras.

A sessão do utilizador é armazenada na base de dados na tabela *session* e as chaves, chave de sessão e chave de autenticação, são guardadas na respetiva tabela *session\_keys*. Caso já exista uma sessão para o respetivo sujeito na respetiva organização, atualiza-se os dados da tabela com os da nova sessão.

As informações da sessão são também escritas num ficheiro do repositório, para uso futuro, concluindo, assim, o início de sessão. Esta sessão expira após um certo tempo de inatividade e terá de ser criada uma nova através deste comando.

## **rep\_get\_file <file handle> [file]**

O conteúdo de um ficheiro pode ser obtido diretamente através do seu *file\_handle*. O conteúdo do mesmo pode ser escrito no terminal ou no ficheiro correspondente ao último argumento (opcional).

Para fazermos get do ficheiro, selecionamos o conteúdo da tabela *file\_info*, distinguida pelo seu *file\_handle* (id).

Não é feita qualquer descriptação do conteúdo do ficheiro neste comando. Para isto, usa-se o comando **rep\_decrypt\_file**, passando-lhe como argumento o conteúdo cifrado (em texto, ou o ficheiro que o contém) e os metadados necessários para a descriptação.

# Authenticated API

## Verificações

Para todos os comandos desta secção e da secção seguinte, são realizadas as seguintes verificações:

- **Validade da sessão**
- **Assinatura**
- **Nonce**
- **Seq\_num**

### Validade da sessão:

A sessão apresenta um lifetime definido no repositório. Se este tempo terminar, a sessão expira e é apagada, sendo necessário criar sessão de novo. A sessão também possui uma chave de autenticação. Sempre que é estabelecida uma comunicação com o repositório (no contexto de uma sessão), o utilizador envia uma assinatura, que é comparada com a chave daquela sessão que está armazenada na BD. Se a verificação da assinatura não se verificar, a comunicação é descartada.

### Verificação de assinatura:

Esta verificação permite garantir a autenticidade da mensagem, ou seja, quem a enviou. O sujeito assina a mensagem com a sua chave privada, e o repositório valida a autenticidade através da chave pública do sujeito. É, ainda, verificado se a mensagem enviada e recebida é igual. O repositório, por sua vez, realiza o mesmo com a resposta, mas utilizando a sua chave (assina com a sua chave privada, e o sujeito verifica com a chave pública).

### Verificação de nonce:

O *nonce* é um número aleatório (gerado através de `os.random(16)`). Cada mensagem tem um *nonce* atribuído. O repositório armazena o *nonce* de cada mensagem e, sempre que recebe uma nova, compara o nonce desta com os que tem armazenados (tabela *nonce* da BD) e verifica se o recebido é igual a algum dos outros. Caso isto se verifique, a mensagem é descartada, de modo a prevenir ataques de repetição, já que uma mensagem repetida terá o mesmo *nonce* da qual repetiu.

### Validação de seq\_num:

As sessões possuem um *seq\_num* armazenado no respetivo ficheiro de sessão, que é incrementado sempre que o sujeito envia uma mensagem para o repositório e permite ver se a ordem das mensagens está correta. Para isto, o repositório compara o valor do *seq\_num* com o último valor armazenado na respetiva tabela *session* da base de dados,



para verificar se aquela mensagem é válida (se o valor for menor ou igual ao valor guardado na BD, a mensagem é descartada, uma vez que pode ser o resultado de um ataque de repetição).

Se alguma das verificações mencionadas falhar, ocorre um erro, e não é possível realizar a operação. No final de cada operação, a informação passa pelo processo inverso, sendo encriptada e assinada.

## Roles

É importante explicar a forma como utilizamos as roles no nosso sistema, para contextualizar a implementação dos comandos seguintes.

Na nossa interpretação, as roles são utilizadas como um meio para realizar uma ação no repositório. Por exemplo, se quero criar um documento, preciso de assumir uma role com essa permissão para realizar esta ação.

Desta forma, definimos que cada sujeito pode ter várias roles, mas apenas ter uma assumida de cada vez. Quando o sujeito quer assumir uma role, tem de verificar os seguintes pré-requisitos:

- tem de ter, pelo menos, uma role associada.
- não pode ter nenhuma role assumida, na correspondente sessão.
- a role que assume não pode estar suspensa.

Para trocar a role assumida, o sujeito tem de primeiro largar a role que tem assumida (comando *rep\_drop\_role*).

Esta abordagem implementa o *Least Privilege Principle*, limitando, assim, as permissões do sujeito à ação que pretende fazer.

## Comandos

### **rep\_assume\_role <session file> <role>**

Este comando permite que um sujeito assumira uma role na sessão. Após a informação ser descriptada, é necessário verificar se há alguma role já ativa. Se for o caso, é preciso largar esta primeiro. Se a organização existir e a role fizer parte da lista das roles do sujeito, e a role estiver ativa, esta pode ser assumida na sessão.

Para implementar isto, guardamos na tabela *session\_role* da BD a role assumida para essa sessão.

### **rep\_drop\_role <session file> <role>**

Como mencionado anteriormente, pode ser necessário desativar uma role. Se esta estiver ativa, então é possível removê-la da sessão. Desta forma, removemos da base de dados o tuplo da tabela *session\_role* correspondente.

### **rep\_list\_roles <session file>**

Este comando lista as roles todas da organização correspondente à sessão que efetua o comando. Para implementar isto, faz-se uma query para listar da tabela *role* o nome de todas as roles da respetiva organização.

### **rep\_list\_subjects <session file> [username]**

Este comando lista todos os sujeitos da organização correspondente à sessão (tabela *organization\_member*). São retornados dados sobre cada sujeito, incluindo o seu estado atual (ativo/suspenso). Pode ser filtrado para listar apenas a informação do sujeito com o dado username, caso este exista.

### **rep\_list\_role\_subjects <session file> <role>**

Lista todos os sujeitos (tabela *organization\_member*) que possuem a role (passada no argumento) na organização correspondente à sessão.

### **rep\_list\_subject\_roles <session file> <username>**

Lista todos as roles (tabela *member\_roles*) que um sujeito (passado no argumento) possui na organização correspondente à sessão. Pode-se verificar as roles de cada sujeito na organização através da tabela *member\_roles*.

### **rep\_list\_role\_permissions <session file> <role>**

Lista todas as permissões que um role (passado no argumento) possui na sessão. Lista tanto as permissões na organização, como as permissões em cada documento da respetiva organização. Estas consultas podem ser feitas efetuando queries nas tabelas *role\_permissions* e *document\_role\_permissions*, respetivamente.

### **rep\_list\_permission\_roles <session file> <permission>**

Apresenta todos as roles que possuem uma determinada permissão (passada no argumento) na organização correspondente à sessão.

Se a permissão for ao nível das roles ou da organização (role ACL ou ACL de organização), basta listar as roles com essa permissão na organização. Pode-se fazer isto através de uma query na tabela *role\_permissions*.

Se a permissão for ao nível dos documentos, lista-se cada documento da organização correspondente à sessão, e as respetivas roles que contêm essa permissão para o respetivo documento. Pode-se implementar isto através da tabela *document\_role\_permissions*.

**rep\_list\_docs <session file> [-s username] [-d nt/ot/et date]**

Lista todos os documentos da organização da sessão. Se o username for passado como argumento, lista os documentos criados pelo mesmo. Pode-se realizar este comando através da tabela *document* da BD.

É, ainda, possível filtrar os documentos pela sua data de criação (mais antiga, mais recente, ou igual à data passada no argumento).

## Authorized API

### Verificações

Nesta secção, para além de todas as verificações previamente realizadas, são ainda verificadas as devidas permissões que um sujeito tem para efetuar alguma operação, através da sua role assumida (tabela *session\_role*).

### Comandos

**rep\_add\_subject <session file> <username> <name> <email>  
<credentials file>**

Este comando permite que um sujeito, sujeito da sessão (*session file*), adicione um novo sujeito com as respectivas informações nos argumentos (*username*, *name*, *email*, *credentials file*). O *username* deve ser único na organização e as credenciais previamente geradas através do comando *rep\_subject\_credentials*. O ficheiro passado como argumento deve ser o da chave pública do sujeito.

Implementou-se isto, ao armazenar as informações do sujeito nas tabelas *subject* e *organization\_member*.

Para efetuar esta operação, o sujeito da sessão necessita de uma role assumida com a permissão **SUBJECT\_NEW** para esta organização. Caso contrário, a ação não é permitida e o sujeito é alertado.

## **rep\_suspend\_subject <session file> <username>**

Este comando permite que um sujeito, sujeito da sessão (session file), suspenda outro sujeito da organização pelo seu username. Implementou-se isto, atualizando o atributo *subject\_status* da tabela *organization\_member* para 'suspended' para o sujeito que vai ser suspenso.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso o role assumido pelo sujeito da sessão não tenha a permissão **SUBJECT\_DOWN**
- caso o outro sujeito não exista na organização
- caso o sujeito se pretenda suspender a si próprio
- caso o sujeito pretenda suspender um sujeito com o role 'manager' nessa organização
- caso o sujeito tente suspender um sujeito que já se encontra suspenso.

Sujeitos suspensos não conseguirão criar sessão na organização onde foram suspensos.

## **rep\_activate\_subject <session file> <username>**

Este comando permite que um sujeito, sujeito da sessão (session file), ative outro sujeito da organização que tenha sido previamente suspenso. Implementou-se isto, atualizando o atributo *subject\_status* da tabela *organization\_member* para 'active' para o sujeito que vai ser ativado.

Esta operação não será efetuada caso se verifique pelo menos uma das seguintes condições:

- caso o role assumido pelo sujeito da sessão não tenha a permissão **SUBJECT\_UP**
- caso o outro sujeito não exista na organização
- caso o sujeito se pretenda ativar a si próprio
- caso o sujeito pretenda ativar outro sujeito que já se encontra ativo.

## **rep\_add\_role <session file> <role>**

Este comando permite que um sujeito, sujeito da sessão (session file), adicione uma nova role à organização (sem permissões). Implementou-se isto, adicionando os dados da role à tabela *role* da BD.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_NEW**
- caso já exista uma role com este nome.

## **rep\_suspend\_role <session file> <role>**

Este comando permite que um sujeito, sujeito da sessão (session file), suspenda uma role da organização. Implementou-se isto, atualizando o atributo *role\_status* da role correspondente da tabela *role* para 'SUSPENDED'.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_DOWN**
- caso a role não exista na organização
- caso a role já esteja suspensa.

**Importante:** O código deste comando foi corrompido no ficheiro repository.py na função *suspend\_role* ao realizar um merge na 2ª entrega. Para se conseguir verificar a veracidade da nossa implementação, é possível ver uma versão anterior do código na branch *edu*, exatamente no mesmo ficheiro e na mesma função a funcionar corretamente, da maneira aqui descrita.

## **rep\_reactivate\_role <session file> <role>**

Este comando permite que um sujeito, sujeito da sessão (session file), reative uma role da organização previamente suspensa. Implementou-se isto, atualizando o atributo *role\_status* da role correspondente da tabela *role* para 'ACTIVE'.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_UP**
- caso a role não exista na organização
- caso a role já esteja ativada.

## **rep\_add\_permission <session file> <role> <username>**

Este comando permite que um sujeito, sujeito da sessão (session file), atribua uma role da organização a outro sujeito da organização. Implementou-se isto, inserindo um tuplo na tabela *member\_roles*, com os respectivos ids do sujeito que recebe a role e da organização da sessão, e o nome da role.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_MOD**
- caso não exista nenhum sujeito com o dado username na organização
- caso a role não exista na organização
- caso o sujeito que irá receber a nova role já a tenha.

### **rep\_remove\_permission <session file> <role> <username>**

Este comando permite que um sujeito, sujeito da sessão (session file), retire uma role da organização a outro sujeito da organização. Implementou-se isto, apagando um tuplo na tabela *member\_roles*, com os respectivos ids do sujeito que recebe a role e da organização da sessão, e o nome da role.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_MOD**
- caso não exista nenhum sujeito com o dado username na organização
- caso a role não exista na organização
- caso o sujeito ao qual irá ser retirada a role, não a tenha associada.

### **rep\_add\_permission <session file> <role> <permission>**

Este comando permite que um sujeito, sujeito da sessão (session file), atribua uma permissão a uma role da organização. Implementou-se isto, inserindo um tuplo na tabela *role\_permissions*, com o id da organização da sessão, o nome da role e a permissão passada como argumento.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_MOD**
- caso a role não exista na organização
- caso a role que irá receber a nova permissão já a tenha.

### **rep\_remove\_permission <session file> <role> <permission>**

Este comando permite que um sujeito, sujeito da sessão (session file), remova uma permissão a uma role da organização. Implementou-se isto, apagando um tuplo da tabela *role\_permissions*, com o id da organização da sessão, o nome da role e a permissão passada como argumento.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **ROLE\_MOD**
- caso a role não exista na organização
- caso a role à qual irá ser removida a permissão não a tenha.

## **re\_add\_doc <session file> <document name> <file>**

Este comando permite que um sujeito, sujeito da sessão (session file), adicione um documento à organização, sendo ele próprio o *creator*. Este documento é guardado na bd, com a respetiva informação (tabela *document*). O ficheiro é guardado na tabela *file\_info* com o algoritmo de encriptação (alg), key, e *file\_key* como chave primária. O documento, por sua vez, apresenta o *file\_handle* como chave primária, que referencia a *file\_key* do *file\_info*.

O conteúdo do ficheiro é encriptado da seguinte forma: geramos uma chave aleatória e derivamo-la usando PBKDF2, em conjunto com um *salt* (também aleatório). Criamos um *iv* aleatório e usamo-lo com a chave, de modo a encriptar o ficheiro com o algoritmo AES CBC.

O algoritmo AES foi escolhido por ser um padrão bastante confiável da criptografia simétrica. O método CBC (Cipher Block Chaining) foi escolhido porque, neste método, o mesmo texto plano não gera o mesmo texto cifrado, devido ao facto de cada bloco de texto plano ser combinado com o bloco de texto cifrado anterior quando é criptografado. Desta forma, conseguimos dificultar ataques de repetição e análise de padrões.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **DOC\_NEW**

## **rep\_get\_doc\_metadata <session file> <document name>**

Este comando permite que um sujeito, sujeito da sessão (session file), aceda aos metadados de um documento. Esta informação inclui os metadados públicos e privados, guardados nas tabelas *file\_info* e *document*.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **DOC\_READ**
- caso não exista nenhum documento com o nome dado na organização

## **rep\_get\_doc\_file <session file> <document name> [file]**

Este comando permite que um sujeito, sujeito da sessão (session file), aceda ao conteúdo descriptado do ficheiro de um documento. Este comando acede aos metadados necessários para descriptar o ficheiro e ao conteúdo cifrado do ficheiro, guardados nas tabelas *file\_info* e *document*. Estes metadados são a chave, o algoritmo, *iv*, *salt* e o *file\_handle* e aplica-se o método de descriptação já referido anteriormente.

O conteúdo descriptado do ficheiro pode ser escrito num ficheiro (caso seja passado como argumento) ou no terminal.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso não exista nenhum documento com o nome dado na organização
- caso a role assumida pelo sujeito da sessão não tenha a permissão **DOC\_READ**
- caso o *file\_handle* não exista neste documento (tenha sido apagado, o ficheiro passa a ser apenas acessível diretamente pelo seu *file\_handle*)

### **rep\_delete\_doc <session file> <document name>**

Este comando permite que um sujeito, sujeito da sessão (session file), apague um documento. Para o efeito, na tabela *document*, o *file\_handle* fica a null e o *subject\_id* passa a ser o *deleter\_id*. É retornado um tuplo com o *file\_handle*, alg e key.

O ficheiro passa a ser acessível apenas diretamente com o comando *rep\_get\_file*, através do seu file handle, e apenas poderá ser descriptado por quem possuir o ficheiro encriptado e os metadados necessários, com o comando *rep\_decrypt\_file*.

Esta operação não será efetuada caso se verifique, pelo menos, uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **DOC\_DELETE**
- caso não exista nenhum documento com o nome dado na organização

### **rep\_acl\_doc <session file> <document name> [+/-] <role> <permission>**

Este comando permite que um sujeito, sujeito da sessão (session file), mude a ACL de um documento na organização. São verificadas as permissões na tabela *document\_role\_permissions*. Caso não exista, uma permissão pode ser adicionada a esta tabela. Caso exista, pode ser removida.

Esta operação não será efetuada caso se verifique pelo menos uma das seguintes condições:

- caso a role assumida pelo sujeito da sessão não tenha a permissão **DOC\_ACL**
- caso não exista nenhum documento com o nome dado na organização
- caso se tente remover uma permissão de um subject com role “manager”
- caso se tente adicionar uma permissão que já esteja na lista de permissões
- caso se tente remover uma permissão que não esteja na lista de permissões



## Medidas de Segurança

- **Eavesdropping:** Todas as mensagens trocadas com o repositório são encriptadas com a chave pública do destinatário. Assim, só a pessoa com a chave privada (do par RSA) é que consegue descriptar a mensagem. Logo, mesmo que os pacotes correspondentes àquela mensagem sejam capturados, o atacante não os vai conseguir descriptar, na teoria.
- **Impersonation:** Este tipo de ataques são impedidos com a assinatura de todas as mensagens trocadas. Assim, o receptor consegue garantir a origem da mensagem, comparando a mesma com a assinatura feita.
- **Manipulation:** Para este tipo de ataques, nós temos duas maneiras de os verificar. Podemos utilizar o mecanismo de assinatura descrito no ataque anterior, ou podemos utilizar o *mac*. Sempre que encriptamos uma mensagem, também calculamos o *mac* da mensagem encriptada (utilizamos uma chave aleatória com 32 bits). Na função de descriptar, nós verificamos se o *mac* corresponde com a mensagem recebida. Assim, podemos detetar qualquer tipo de alterações que possam ter sido feitas à mensagem original.
- **Replay:** O *nonce* e o *seqnum*, como mencionámos anteriormente, previnem estes ataques.

## Features a melhorar

Na nossa implementação dos comandos há alguns lapsos, sendo possível melhorar alguns aspetos a nível de segurança, nomeadamente:

- **Suspensão de sujeito:** Antes de realizar qualquer comando, deveria ser verificado se o subject está suspenso. De momento, não é realizada essa verificação, o que implica que um sujeito que tenha sido suspenso mas ainda esteja numa sessão válida, possa efetuar qualquer operação permitida. O sujeito apenas perde o acesso ao repositório da próxima vez que tentar criar uma sessão. (verificação apenas feita no *rep\_create\_session*).
- **Verificações da password:** Garantir que a password escolhida pelo utilizador tem mais de 12 caracteres e menos de 128. Também podíamos dar um feedback ao utilizador, relativamente à força da password, dando sugestões para poder melhorar o nível de segurança da mesma.
- **Guardar tentativas de login sem sucesso:** Alterar a BD para conseguirmos registar todas as tentativas falhadas de login e podermos criar um limite para bloquear novas tentativas dentro de um certo intervalo de tempo.

# Segurança (OWASP)

O Capítulo do ASVS que o nosso grupo escolheu foi o **V2 Authentication**. Escolhemos este capítulo, porque consideramos que é um dos que apresenta maior relevância, relativamente ao trabalho realizado. O nível de NIST AAL utilizado na verificação dos controlos foi o L3, tal como especificado no enunciado do projeto.

## V2.1 Password Security

### 2.1.1 - Verify that user set passwords are at least 12 characters in length (after multiple spaces are combined)

Este controlo não é cumprido. Na implementação realizada na entrega anterior, não existe qualquer tipo de verificação feita relativamente à password passada no comando `rep_subject_credentials <password> <credentials file>`. O impacto desta falha é reduzido, devido às medidas de segurança implementadas no trabalho.

Este lapso permite que o utilizador escolha passwords menos seguras e mais fáceis de decifrar/adivinhar (usando força bruta por exemplo).

Para corrigir este lapso, bastava acrescentar uma verificação no momento em que o programa recebe os inputs. O seguinte código apresenta um exemplo de como o fazer.

```
if __name__ == '__main__':
    if len(sys.argv) != 3:
        print(f'Usage: rep_subject_credentials <password> <credentials_file>')
        sys.exit(1)

    password = sys.argv[1]

    if len(password) < 12:
        print(f'Password must have at least 12 characters')
        sys.exit(1)

    credentials_file = sys.argv[2]
    private_key, public_key = create_key_pair(password)
    write_privkey_credentials(credentials_file, private_key)
    save_public_key(public_key, credentials_file)

    sys.exit(0)
```

Imagem 2: Exemplo da implementação do controlo

### 2.1.2 - Verify that passwords of at least 64 characters are permitted, and that passwords of more than 128 characters are denied

Este controlo não é cumprido. Tal como referido no controlo anterior, a nossa implementação não aplica restrições relativamente à palavra passe introduzida. Quanto maior for a password, maior é a dificuldade de a decifrar por força bruta. Por outro lado,

passwords muito grandes são difíceis de armazenar (devido ao espaço que vão ocupar), e difíceis de recordar para o utilizador (caso este não utilize outras tecnologias para guardar as suas passwords). Assim, é necessário haver um meio termo relativamente ao tamanho de uma password.

O código seguinte mostra um exemplo de como cumprir este controlo.

```
if __name__ == '__main__':
    if len(sys.argv) != 3:
        print(f'Usage: rep_subject_credentials <password> <credentials_file>')
        sys.exit(1)

    password = sys.argv[1]

    if len(password) < 12 or len(password) > 128:
        print(f'Password must be between 12 and 128 characters')
        sys.exit(1)

    credentials_file = sys.argv[2]
    private_key, public_key = create_key_pair(password)
    write_privkey_credentials(credentials_file, private_key)
    save_public_key(public_key, credentials_file)

    sys.exit(0)
```

Imagem 3: Código de verificação do comprimento da password

### 2.1.3 - Verify that password truncation is not performed. However, consecutive multiple spaces may be replaced by a single space

Este controlo não é cumprido. O programa aceita qualquer tipo de password. Para corrigir isto, era necessário acrescentar uma verificação para substituir todos os espaços duplos por um espaço só. Uma forma mais simples de implementar esta verificação é separar a password pelos espaços, e voltar a juntar a mesma, mas só usando um espaço simples. O código seguinte apresenta um exemplo dessa implementação.

```

if __name__ == '__main__':
    if len(sys.argv) != 3:
        print(f'Usage: rep_subject_credentials <password> <credentials_file>')
        sys.exit(1)

    password = sys.argv[1]

    if len(password) < 12 or len(password) > 128:
        print(f'Password must be between 12 and 128 characters')
        sys.exit(1)

    password = " ".join(password.split())

    credentials_file = sys.argv[2]
    private_key, public_key = create_key_pair(password)
    write_privkey_credentials(credentials_file, private_key)
    save_public_key(public_key, credentials_file)

    sys.exit(0)

```

Imagem 4: Código para garantir que a password só tem espaços simples

#### 2.1.4 - Verify that any printable Unicode character, including language neutral characters such as spaces and Emojis are permitted in passwords

Este controlo é cumprido. Uma vez que não temos nenhuma verificação para a password, qualquer tipo de caractere é permitido.

```

• (venv) edu@edu:~/Desktop/SIO/sio-2425-project-113626_103070_113736/delivery2$ ./rep_subject_credentials "hello there🤔"
"credentials/subject_credentials.pem"

```

Imagem 5: Exemplo de uma password

#### 2.1.5 - Verify users can change their password

Este controlo é cumprido, mas, no contexto do nosso projeto, mudar a password implica gerar novas credenciais, através do comando *rep\_subject\_credentials*. Alterar só a password das credenciais não é possível, uma vez que não existe um comando para esse fim. Mudar as credenciais implica atualizar as credenciais públicas na organização onde ele utilizou as mesmas que alterou (para as verificações de segurança não falharem, visto que os valores passam a ser diferentes).

#### 2.1.6 - Verify that password change functionality requires the user's current and new password

Este controlo não é cumprido, uma vez que o utilizador não precisa da password antiga para conseguir alterar a password. Para conseguir alterar, basta executar o comando *rep\_subject\_credential* com a nova password. Como referido no controlo anterior, não existe um comando específico para alterar a password.

**2.1.7 - Verify that passwords submitted during account registration, login, and password change are checked against a set of breached passwords either locally (such as the top 1,000 or 10,000 most common passwords which match the system's password policy) or using an external API. If using an API a zero knowledge proof or other mechanism should be used to ensure that the plain text password is not sent or used in verifying the breach status of the password. If the password is breached, the application must require the user to set a new non-breached password**

Este controlo não é cumprido. Verificar se a password está na lista de passwords já descobertas é uma forma de diminuir a probabilidade desta ser descoberta por ataques de força bruta. Para implementar este controlo, tínhamos de integrar uma API no nosso repositório ou ter uma lista local das palavras mais frequentes já descobertas. Sempre que o comando *rep\_subject\_credentials* fosse utilizado, a password era comparada com as passwords presentes na lista (ou através da API).

**2.1.8 - Verify that a password strength meter is provided to help users set a stronger password**

Este controlo não é cumprido. Dar feedback ao utilizador relativamente à segurança da sua password é uma boa maneira de diminuir ataques por força bruta, uma vez que garante que as passwords são mais complexas (isto se o utilizador seguir as dicas fornecidas em relação à força da password).

Para implementar este controlo, seria necessário criar uma medida de forma a poder comparar as passwords introduzidas e avaliar o seu nível de força. (Poderíamos comparar com a lista de passwords já descobertas e calcular um fator de semelhança). Consoante o nível de força da password, dar sugestões para tornar a password mais forte.

Em termos de código, isto seria alcançado através de um ciclo while, onde o mecanismo de saída era a password atingir o nível máximo de força, ou o utilizador inserir a tecla de saída do ciclo, dando o conhecimento que a sua password pode não ser a mais segura. Entre cada ciclo, é enviado ao utilizador o output, que consiste no nível de força, acompanhado com algumas sugestões para melhorar (caso seja necessário).

**2.1.9 - Verify that there are no password composition rules limiting the type of characters permitted. There should be no requirement for upper or lower case or numbers or special characters**

Este controlo é cumprido, uma vez que não existe nenhum tipo de restrições para a password na nossa implementação. Assim, o utilizador tem liberdade total na escolha da password que quer utilizar para proteger as suas credenciais.

```
• (venv) edu@edu:~/Desktop/SIO/sio-2425-project-113626_103070_113736/delivery2$ ./rep_subject_credentials "hello there!" "
credentials/subject_credentials.pem"
• (venv) edu@edu:~/Desktop/SIO/sio-2425-project-113626_103070_113736/delivery2$ ./rep_subject_credentials "Hello There!" "
credentials/subject_credentials.pem"
```

Imagem 6: Exemplo de possíveis passwords

#### **2.1.10 - Verify that there are no periodic credential rotation or password history requirements**

Este controlo é cumprido. Sempre que o utilizador quiser alterar a password, também tem de gerar novas credenciais. O repositório não tem qualquer conhecimento do histórico de passwords utilizadas pelo sujeito, e estas não são precisas em nenhuma parte da interação entre o utilizador e o repositório.

#### **2.1.11 - Verify that "paste" functionality, browser password helpers, and external password managers are permitted**

Este controlo é cumprido. O utilizador pode copiar e colar a password no terminal quando está a executar o comando `./rep_subject_credentials`. Uma vez que o projeto é feito para utilizar o terminal, qualquer tipo de ajuda do browser ou aplicações externas não se aplicam.

#### **2.1.12 - Verify that the user can choose to either temporarily view the entire masked password, or temporarily view the last typed character of the password on platforms that do not have this as built-in functionality**

Este controlo não é cumprido, uma vez que o utilizador vê sempre a password na íntegra, visto que esta é passada como input no momento de executar o comando (mostrado em imagens em controlos anteriores). Assim, o utilizador não tem opções de visionamento da password.

## **V2.2 General Authenticator Security**

#### **2.2.1 - Verify that anti-automation controls are effective at mitigating breached credential testing, brute force, and account lockout attacks. Such controls include blocking the most common breached passwords, soft lockouts, rate limiting, CAPTCHA, ever increasing delays between attempts, IP address restrictions, or risk-based restrictions such as location, first login on a device, recent attempts to unlock the account, or similar. Verify that no more than 100 failed attempts per hour is possible on a single account**

Este controlo não é cumprido. A maior parte deste controlo não se aplica ao projeto, mas a parte de verificar o número de tentativas falhadas de login podia ser implementada. Para isso, primeiro teríamos de alterar a BD, adicionando uma nova tabela para guardar as tentativas de login falhadas (esta tabela guardaria o id do utilizador, o id da organização e um timestamp da tentativa de login). Assim, sempre que há uma tentativa de login, o repositório começa por verificar o número de tentativas falhadas por aquele sujeito, naquela organização. Se este número for superior ao limite, o repositório bloqueia a tentativa de login.

**2.2.2 - Verify that the use of weak authenticators (such as SMS and email) is limited to secondary verification and transaction approval and not as a replacement for more secure authentication methods. Verify that stronger methods are offered before weak methods, users are aware of the risks, or that proper measures are in place to limit the risks of account compromise**

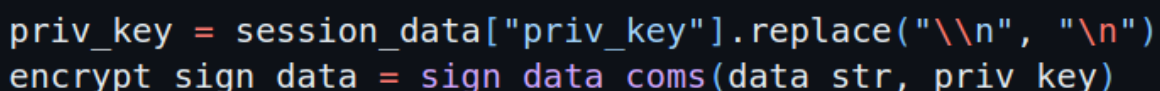
Este controlo não se aplica ao projeto, uma vez que não são utilizados nenhum tipo de autenticadores como aqueles referidos no controlo (no projeto não são utilizados autenticadores adicionais).

**2.2.3 - Verify that secure notifications are sent to users after updates to authentication details, such as credential resets, email or address changes, logging in from unknown or risky locations. The use of push notifications - rather than SMS or email - is preferred, but in the absence of push notifications, SMS or email is acceptable as long as no sensitive information is disclosed in the notification**

Este controlo não se aplica, visto que não existe nenhum sistema de notificações implementado. As únicas alterações que o utilizador pode fazer relevantes para este controlo é alterar a password e gerar novas credenciais, mas este comando não interage com o repositório, ou seja, as alterações são feitas localmente, logo, não existe necessidade de notificar o utilizador.

**2.2.4 - Verify impersonation resistance against phishing, such as the use of multi-factor authentication, cryptographic devices with intent (such as connected keys with a push to authenticate), or at higher AAL levels, client-side certificates**

Este controlo é cumprido. O utilizador assina todas as mensagens no contexto da sessão, o que impede ataques por impersonation, uma vez que o repositório só aceita mensagens assinadas por aquele utilizador (assinadas com as chaves da sessão que consistem num par RSA)

A screenshot of a code editor showing two lines of Python code. The first line is 'priv\_key = session\_data["priv\_key"].replace("\\n", "\\n")' and the second line is 'encrypt\_sign\_data = sign\_data\_coms(data\_str, priv\_key)'. The code is displayed in a dark-themed editor with syntax highlighting.

```
priv_key = session_data["priv_key"].replace("\\n", "\\n")
encrypt_sign_data = sign_data_coms(data_str, priv_key)
```

Imagem 7: Mensagem assinada

**2.2.5 - Verify that where a Credential Service Provider (CSP) and the application verifying authentication are separated, mutually authenticated TLS is in place between the two endpoints**

Este controlo não se aplica. Neste projeto não é utilizado nenhum CSP.

### 2.2.6 - Verify replay resistance through the mandated use of One-time Passwords (OTP) devices, cryptographic authenticators, or lookup codes

Este controlo é cumprido. As sessões possuem um seq\_num armazenado no sessionFile, que permite ver se a ordem das mensagens recebidas no repositório está correta. Para isso, o repositório compara o valor do seq\_num com o último valor armazenado na base de dados, com a intenção de verificar se aquela mensagem é válida (se o valor for menor ou igual ao valor guardado na BD, a mensagem é descartada, uma vez que pode ser o resultado de um ataque de replay). O seq\_num é incrementado sempre que o sujeito envia uma mensagem ao repositório. Outra medida de segurança implementada para impedir ataques de replay foi o nonce, que consiste num valor numérico único (gerado através de os.random(16)). O repositório armazena todos os nonces que recebe e, sempre que recebe uma nova mensagem, compara o nonce atribuído a esta com todos os que tem registados na BD (se houver algum que seja igual, a mensagem é descartada). Cada mensagem tem um nonce atribuído, gerado automaticamente. Isto ajuda a impedir ataques de replay, uma vez que os nonces da mensagem original e da repetida seriam iguais.

```
request = {
    "org_id": session_data['org_id'],
    "sub_id": session_data['sub_id'],
    "document": document_name,
    "nonce": os.urandom(16).hex(),
    "seq_num": session_data["seq_num"],
    "signature": session_data["signature"]
}

#encrypt the data
rep_pubkey = load_rep_pubkey()
encrypted_data = encrypt_data(json.dumps(request), rep_pubkey)
data_str = bytes_to_string(encrypted_data)

priv_key = session_data["priv_key"].replace("\\n", "\n")
encrypt_sign_data = sign_data_coms(data_str, priv_key)

if os.path.exists(session_file):
    with open(session_file, 'w') as f:
        session_data["seq_num"] += 1
        json.dump(session_data, f)
```

Imagem 8: Nonce e seq\_num

### 2.2.7 - Verify intent to authenticate by requiring the entry of an OTP token or user-initiated action such as a button press on a FIDO hardware key



Este controlo é cumprido. Para criar uma sessão, o utilizador e o repositório têm de se autenticar. Para isso, o utilizador solicita um desafio que consiste num número aleatório gerado pelo repositório (através de `os.random(32)`). Este número funciona como uma OTP, uma vez que tem um período curto de vida (7 segundos) e só serve para uma função, que é a autenticação dos dois intervenientes naquela comunicação. O número é encriptado com a chave pública do utilizador e enviado ao mesmo. Para o utilizador responder ao desafio, este tem de descriptar o desafio com a sua chave privada e voltar a enviar para o repositório, mas, desta vez, encriptado com a chave do repositório. Se o desafio recebido for igual ao criado, o repositório gera uma chave de autenticação para a sessão, de forma a esta se autenticar em comunicações futuras.

```
request = {
    "org_id": organization,
    "username": username,
    "session_pub_key": public_key_pem.decode('utf-8'),
    "challenge": challenge
}

#encrypt the data
rep_pubkey = load_rep_pubkey()
encrypted_data = encrypt_data(json.dumps(request), rep_pubkey)
data_str = bytes_to_string(encrypted_data)

response = requests.post("http://localhost:5000/session", json=data_str)
```

```
if pub_key != user_pub_key:
    conn.close()
    return jsonify({"error": "Invalid credentials or Invalid user"}), 401

# verify if the challenge is still valid
verify_challenge = conn.execute("SELECT challenge FROM challenge WHERE org_id = ? AND subject_id = ?", (org_id, sub_id)).fetchone()
verify_timestamp = conn.execute("SELECT last_interaction FROM challenge WHERE org_id = ? AND subject_id = ?", (org_id, sub_id)).fetchone()

if not verify_challenge:
    challenge = os.urandom(32)
    last_interaction = datetime.datetime.now()

    conn.execute("INSERT INTO challenge (org_id, subject_id, challenge, last_interaction) VALUES (?, ?, ?, ?)", (org_id, sub_id, challenge, last_interaction))
    conn.commit()

else:
    time = datetime.datetime.timestamp(datetime.datetime.now())
    time_object = datetime.datetime.strptime(verify_timestamp[0], DATE_FORMAT)
    last_time = int(time_object.timestamp())

    if (time - last_time) > CHALLENGE_LIFETIME:
        challenge = os.urandom(32)
        last_interaction = datetime.datetime.now()

        conn.execute("UPDATE challenge SET challenge = ?, last_interaction = ? WHERE org_id = ? AND subject_id = ?", (challenge, last_interaction, org_id, sub_id))
        conn.commit()

    else:
        challenge = verify_challenge[0]
```

```
if challenge.hex() != user_challenge:
    conn.close()
    return jsonify({"error": "Authentication failed"}), 401

# generate auth key for session and sign it
auth_key = os.urandom(32)
```

Imagens 9, 10 e 11: Desafio

## V2.3 Authenticator Lifecycle

**2.3.1 - Verify system generated initial passwords or activation codes SHOULD be securely randomly generated, SHOULD be at least 6 characters long, and MAY contain letters and numbers, and expire after a short period of time. These initial secrets must not be permitted to become the long term password**

Não se aplica ao projeto, uma vez que não é necessário gerar passwords iniciais. O sujeito gera as suas credenciais sem ter de interagir com o repositório.

**2.3.2 - Verify that enrollment and use of user-provided authentication devices are supported, such as a U2F or FIDO tokens**

Não se aplica ao projeto, visto que não podemos utilizar tecnologia pré-existente para a proteção de comunicações.

**2.3.3 - Verify that renewal instructions are sent with sufficient time to renew time bound authenticators**

Não se aplica ao projeto, uma vez que o utilizador não precisa de interagir com o repositório para alterar a password/credenciais.

## V2.4 Credential Storage

Este conjunto de controlos não se aplica ao nosso projeto, uma vez que não guardamos qualquer tipo de credenciais na base de dados.

## V2.5 Credential Recovery

Este conjunto de controlos não se aplica ao nosso projeto. Tal como referido no ponto anterior, não existe armazenamento de credenciais, logo, não é necessário fazer recuperação das mesmas.

## V2.6 Look-up Secret Verifier

Este conjunto de controlos não se aplica ao nosso projeto, visto que não utilizamos lookup secrets.

## V2.7 Out of Band Verifier

Este conjunto de controlos não se aplica ao nosso projeto, visto que não utilizamos Out of Band Verifiers.

## V2.8 One Time Verifier

### 2.8.1 - Verify that time-based OTPs have a defined lifetime before expiring

Este controlo é cumprido. A nossa única OTP é o desafio utilizado no momento de criar uma sessão. Este tem um tempo de vida de 7 segundos (tal como dá para ver na imagem seguinte)

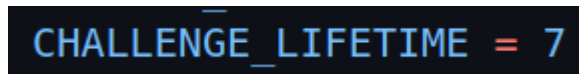


Imagem 12: Tempo de vida do desafio

### 2.8.2 - Verify that symmetric keys used to verify submitted OTPs are highly protected, such as by using a hardware security module or secure operating system based key storage

Este controlo não se aplica ao projeto, uma vez que nós não utilizamos chaves simétricas para verificar as OTPs.

### 2.8.3 - Verify that approved cryptographic algorithms are used in the generation, seeding, and verification of OTPs

Este controlo é cumprido. O desafio é gerado com o `os.random(32)` e este é verificado através da sua encriptação com a chave pública do utilizador e a sua desencriptação com a chave privada.

```
if not verify_challenge:
    challenge = os.urandom(32)
    last_interaction = datetime.datetime.now()

    conn.execute("INSERT INTO challenge (org_id, subject_id, challenge, last_interaction) VALUES (?, ?, ?, ?)", (org_id, sub_id, challenge, last_interaction))
    conn.commit()

else:
    time = datetime.datetime.timestamp(datetime.datetime.now())
    time_object = datetime.datetime.strptime(verify_timestamp[0], DATE_FORMAT)
    last_time = int(time_object.timestamp())

    if (time - last_time) > CHALLENGE_LIFETIME:
        challenge = os.urandom(32)
        last_interaction = datetime.datetime.now()

        conn.execute("UPDATE challenge SET challenge = ?, last_interaction = ? WHERE org_id = ? AND subject_id = ?", (challenge, last_interaction, org_id, sub_id))
        conn.commit()

    else:
        challenge = verify_challenge[0]

conn.close()

#encrypt data
data_encrypted = encrypt_data(json.dumps({"challenge": challenge.hex()}), pub_key)
data_str = bytes_to_string(data_encrypted)
```

Imagem 13: Geração do desafio

### 2.8.4 - Verify that time-based OTP can be used only once within the validity period

Este controlo não é cumprido. O desafio só é utilizado uma única vez dentro do período de tempo válido, mas isto não é verificado pelo programa. Para cumprir este controlo, teríamos de guardar todos os desafios gerados pelo repositório na BD e se este foi respondido com sucesso ou não.

**2.8.5 - Verify that if a time-based multi-factor OTP token is re-used during the validity period, it is logged and rejected with secure notifications being sent to the holder of the device**

Este requisito não é cumprido. O mesmo desafio pode ser enviado ao utilizador mais do que uma vez, se este for válido (menos de 7 segundos de vida) e não tenha sido respondido. No nosso projeto não existe um sistema de notificações, logo, para cumprir este controlo, todos os acessos ao desafio deviam ser registados (indicando se é a primeira vez que estão a aceder ou não).

**2.8.6 - Verify physical single-factor OTP generator can be revoked in case of theft or other loss. Ensure that revocation is immediately effective across logged in sessions, regardless of location**

Este controlo não se aplica, visto que não utilizamos geradores de OTPs.

**2.8.7 - Verify that biometric authenticators are limited to use only as secondary factors in conjunction with either something you have and something you know**

Este controlo não se aplica, uma vez que nós não utilizamos dados biométricos.

## **V2.9 Cryptographic Verifier**

Este conjunto de controlos não se aplica ao nosso projeto, visto que não utilizamos dispositivos externos para geração de chaves.

## **V2.10 Service Authentication**

**2.10.1 - Verify that intra-service secrets do not rely on unchanging credentials such as passwords, API keys or shared accounts with privileged access**

Este controlo é cumprido. Os segredos da nossa aplicação não dependem de chaves. Só o acesso feito por um utilizador aos dados guardados no repositório é que necessita de chaves.

**2.10.2 - Verify that if passwords are required for service authentication, the service account used is not a default credential. (e.g. root/root or admin/admin are default in some services during installation)**

Este controlo é cumprido. No nosso projeto não existem contas default. Para iniciar o repositório, é necessário uma password (escolhida pelo grupo), que não advém previamente da aplicação ou de uma conta default.

**2.10.3 - Verify that passwords are stored with sufficient protection to prevent offline recovery attacks, including local system access**

Este controlo não se aplica ao projeto, uma vez que nós não armazenamos passwords.

**2.10.4 - Verify passwords, integrations with databases and third- party systems, seeds and internal secrets, and API keys are managed securely and not included in the source code or stored within source code repositories. Such storage SHOULD resist offline attacks. The use of a secure software key store (L1), hardware TPM, or an HSM (L3) is recommended for password storage**

Este controlo não se aplica, pela mesma razão que o anterior.

# Bibliografia

Nesta secção são apresentados as referências utilizadas na realização deste projeto.

- Slides teóricos disponíveis na página da UC no elearning
- <https://cryptography.io/en/latest/>
- <https://medium.com/@igorfilatov/hybrid-encryption-in-python-3e408c73970c>