Safe Repository: Features e análise com ASVS

Diogo Costa 112714

Bruno Tavares 113372

SIO 24/25

Prof. João Paulo Barraca

Introdução

Neste documento iremos apresentar o report das features implementadas no âmbito do projeto da unidade curricular Segurança Informática e nas Organizações, bem como uma análise em conformidade com os critérios ASVS de um dos capítulos.

O objetivo do projeto era implementar um software que permitisse o armazenamento de dados relacionados com organizações, como membros e ficheiros, de uma forma que garantisse a segurança.

Segurança

A segurança no repositório é assegurada de várias maneiras: os ficheiros são armazenados no servidor após serem encriptados e as comunicações entre o cliente e o servidor são encriptadas. Para garantir uma comunicação segura entre o servidor e o cliente, as comunicações entre estas entidades em que ocorre a troca de informação sensível foram encriptadas. Foi gerado um par de chaves para o servidor, sendo que a chave pública foi armazenada como variável de ambiente (o cliente tinha acesso a esta chave), e a privada foi armazenada num documento com a extensão ***pem*** no diretório do servidor, sendo que só este tinha acesso a esta chave. Para além disso, toda a manipulação dos dados armazenados nos ficheiros json é toda efetuada no servidor, que previamente verifica se o cliente tem permissões para realizar a ação com sucesso. Além disso, a função encrypt_file, utiliza HMAC para verificar a integridade dos dados,

garantindo que informações sensíveis não sejam alteradas durante a transmissão.

Cliente -> Servidor

Após criado o payload (em json) com todas as informações a enviar para o servidor, este era criptografado localmente usando o algoritmo **AES** em modo **CBC**, com recurso a uma chave gerada dinamicamente. Para além disso, o vetor de inicialização é concatenado ao conteúdo criptografado para garantir que a desencriptação funciona do lado do servidor. Após este passo, a chave AES usada para encriptar os dados a transmitir ao servidor é encriptada com **RSA** utilizando a chave pública do servidor, o que garante que a sua transmissão é segura (apenas o servidor pode acessá-la), e adicionada a outro payload com os dados encriptados e a chave AES encriptada.

Já do lado do servidor, este usa a sua chave privada para desencriptar a chave AES, usando-a posteriormente para desencriptar os dados enviados pelo cliente.

Servidor -> Cliente

Neste sentido, apenas são criptografadas as comunicações em que há a transmissão de dados sensíveis, dado que as restantes são apenas de feedback de sucesso ou erro da operação efetuada.

Tal como no sentido contrário, o servidor cria um payload com os dados a transmitir. Só que desta vez, usa a chave pública do cliente para encriptar a chave AES usada para encriptar o payload. Após a transmissão dos dados criptografados, é pedido ao cliente que introduza o ficheiro onde está armazenada a sua chave privada e a password, visto que no comando **rep_subject_credentials**, a password é usada para gerar a chave privada, e esta só é capaz de desencriptar quando usada em conjunto com a password. Depois de lida a chave privada do utilizador, é desencriptada a chave AES, que depois é usada para desencriptar os dados. Os dados só são expostos corretamente quando as credenciais (ficheiro de chave privada e password) válidas são introduzidos.

Comandos implementados

Nesta secção, iremos apresentar uma breve descrição de todos os comandos que implementámos, divididos em diferentes categorias com base num cenário específico de uso e nas permissões necessárias para cada um deles. As diferentes categorias são:

1. **Comandos locais**: estes comandos são executados em ambiente local do utilizador sem depender de chamadas para a API
2. **Comandos que utilizam a API anónima**: permitem o acesso de qualquer cliente a funcionalidades básicas da API. Uma vez que não requerem qualquer tipo de autorização, não permitem o acesso a dados sensíveis
3. **Comandos que utilizam a API autenticada:** exigem que o utilizador seja autenticado no sistema, garantindo que apenas estes têm acesso a dados sensíveis.
4. **Comandos que utilizam a API autorizada:** não só exigem autenticação por parte do utilizador, mas também como permissões para realizar certas operações de gestão de cada organização.

Comandos locais

- **rep_subject_credentials <password> <credentials file>:**
  Este comando é responsável por criar um par de chaves para o utilizador, sendo estas armazenadas no diretório do cliente em ficheiros **pem** separados, garantindo um fácil armazenamento e acesso futuro. A chave privada é protegida com recurso à password

fornecida pelo cliente e é utilizada a técnica de derivação PBKFD2 com salt para aumentar a segurança.

- **rep_decrypt_file <encrypted file> <encryption metadata>:**
  Este comando realiza a desencriptação de um ficheiro criptografado usando o algoritmo AES-GCM-SHA256, com base em metadados fornecidos num ficheiro json, que incluem a chave de criptografia, o vetor de inicialização (IV) e, opcionalmente, um hash para verificação de integridade. Após ler o arquivo criptografado e aplicar o processo de descriptografia, o código remove o padding do texto, verifica se o hash gerado do conteúdo desencriptado corresponde ao hash esperado, e exibe os conteúdos do ficheiro original.

Comandos que usam a API anónima

- **rep_create_org <organization> <username> <name> <email> <public key file>:**
  Este comando é responsável pela criação de uma nova organização no servidor. Do lado do cliente é efetuada uma validação se todos os parâmetros necessários são fornecidos e envia-os para o servidor. Já o servidor, utiliza os valores transmitidos do cliente para criar uma nova organização. É efetuada uma verificação se a organização já existe. Caso esta verificação seja bem-sucedida, é criada uma nova organização com um subject com os dados fornecidos pelo cliente. Este subject tem como role na organização manager.

- **rep_list_orgs:**
  Este comando permite a um utilizador saber que organizações estão atualmente registadas no servidor.

- **rep_get_file <file handle> [file]:**
  Este comando assegura a transferência de um ficheiro encriptado armazenado no servidor. O conteúdo do ficheiro armazenado no servidor (encriptado) é copiado para um ficheiro no diretório do cliente

- **rep_create_session <organization> <username> <password> <credentials file> <session file>:**
  Este comando cria uma sessão autenticada para um utilizador autenticado numa organização. O servidor verifica se os dados enviados pelo cliente são válidos e valida a password com a chave privada que consta do ficheiro com as credenciais. A sessão é armazenada num ficheiro json no diretório do servidor.

Comandos que usam a API anónima

- **rep_assume_role <session file> <role>**
  Este comando permite a um utilizador assumir um role dentro da sua organização. O servidor verifica se o utilizador pertence à organização e se o papel que pretende assumir está disponível para ele e se o papel se encontra ativo ou não (não é permitido assumir um papel que não esteja ativo). O papel é atualizado na sessão e na organização são adicionadas ao utilizador as permissões desse papel. Não é permitido ao utilizador assumir um papel quando já tem um ativo.

- **rep_drop_role <session file> <role>:**
  Este comando tem a função contrária à do anterior: um utilizador que execute este comando, perde as permissões associadas ao papel que tinha ativo. Na sessão, todos os papeis que o utilizador dá drop são guardados de forma a garantir a persistência dos dados.

- **rep_list_roles <session file> <role>:**

Este comando permite ao utilizador saber todos os papeis que já teve na sessão atual. O servidor devolve ao cliente os papeis armazenados na sessão, tanto o ativo como os que estão atualmente inativos.

- **rep_list_subjects <session file> [username]:**
  Este comando permite ao utilizador saber todas as pessoas que pertencem à sua organização. Através do ficheiro de sessão, o servidor obtém a organização a que o utilizador pertence e lista todos os utilizadores que pertencem a essa mesma organização. Aceita um filtro que pesquisa por um username em específico.
- **rep_list_role_subjects <session file> <role>:**
  Este comando lista todos os utilizadores que na organização à qual o cliente pertence com um papel específico nessa mesma organização. O servidor obtém a organização do cliente no ficheiro de sessão e no acl dessa organização obtém os utilizadores com o papel especificado.

- **rep_list_subject_roles <session file> <username>:**
  Este comando lista os papeis que o cliente tem associados a si na organização em que tem uma sessão ativa. O servidor obtém no ficheiro da sessão a organização à qual o cliente pertence e no ficheiro das organizações procura pelo utilizador e devolve todos os roles associados ao username. Os papeis devolvidos pelo servidor são todos os que o cliente pode assumir dentro da organização.
- **rep_list_role_permissions <session file> <role>:**
  Este comando lista todas as permissões associadas a um papel dentro de uma organização. O servidor obtém no ficheiro da sessão a organização à qual o cliente pertence e no ficheiro das organizações procura pelo papel fornecido pelo cliente, devolvendo uma lista de todas as permissões associadas ao mesmo.

- **rep_list_permission_roles <session file> <permission>:**
  Ao contrário do comando acima, este comando devolve os papeis que têm uma dada permissão na organização à qual o cliente pertence. O servidor obtém no ficheiro da sessão a organização à

qual o cliente pertence e no ficheiro das organizações, procura pelos papeis que têm associado a si a permissão dada pelo cliente.

- **rep_list_docs <session file> [-s username] [-d nt/ot/et date]**

  Este comando permite ao utilizador listar todos os documentos disponíveis na organização na qual tem uma sessão ativa. No servidor, a organização é obtida no ficheiro de sessão. No ficheiro das organizações, obtém todos os ficheiros que foram adicionados à organização e devolve uma lista destes para o cliente.

Comandos que usam a API autorizada

- **rep_add_subject <session file> <username> <name> <email> <credentials file>:**

  Este comando permite adicionar um novo utilizador à organização, adicionando uma nova entrada aos subjects na organização com os dados do novo utilizador, sem papeis associados e sem permissões. Antes de adicionar, o servidor verifica se o utilizador que efetuou a requisição tem a permissão SUBJECT_NEW.

- **rep_suspend_subject <session file> <username> e rep_activate_subject <session file> <username>**

  Este par de comandos efetua operações em contrário, de ativação e suspensão de um utilizador associado à organização do requerente. O servidor na organização, procura pelo user a suspender/ativar e muda o seu status. O requerente tem de ter a permissão necessária (SUBJECT_UP para ativar e SUBJECT_DOWN para suspender)

- **rep_add_role <session file> <role>:**
  Este comando adiciona um papel à organização à qual o cliente pertence. Após obter a organização no ficheiro de sessão, o servidor, no ficheiro das organizações, adiciona o papel à organização, sendo que este é adicionado sem permissões associadas.

- **rep_suspend_role <session file> <role> e rep_activate_role <session file> <role>:**
  Este par de comandos tem ações contrárias: uma ativa e outro suspende um papel numa organização. Quando suspenso, um papel não pode ser assumido por outro utilizador. Após obter a organização no ficheiro de sessão, o servidor, no ficheiro das organizações, procura pelo role em específico e muda o seu status. Requere permissão ROLE_DOWN para suspender e ROLE_UP para ativar.

- **rep_add_permission <session file> <role> <username> e rep_remove_permission <session file> <role> <username>:**
  Este par de comandos tem ações contrárias e destinam-se à adição/remoção de um papel disponível para um utilizador da organização. Ele obtém a organização no ficheiro de sessão e depois encontra-a no ficheiro com as organizações. Verifica se o papel existe na organização e atribui-o/remove-o do utilizador pretendido, sendo que o utilizador em questão apenas fica com o papel disponível e não com as permissões associadas ao papel, esta ação só se concretiza quando esse utilizador efetua o comando

rep_assume_role. Só é permitida a realização destes comandos a quem tenha a permissão ROLE_MOD.

- **rep_add_permission <session file> <role> <permission> e rep_remove_permission <session file> <role> <permission>**

  Este par de comandos é muito parecido ao anterior, só que em vez de associar um papel a um utilizador, adiciona uma permissão a um papel. Tal como o par anterior, só quem tenha a permissão ROLE_MOD consegue executar estes comandos.

- **rep_add_doc <session file> <document name> <file>:**

  Este comando adiciona um documento a uma organização. Após obter a organização do cliente, este adiciona o nome do documento à organização. O documento é depois encriptado e criado um ficheiro json com a metadata. Tanto o ficheiro encriptado como o ficheiro com a metadata são guardados no diretório do servidor. Apenas sujeitos com a permissão DOC_NEW conseguem adicionar novos ficheiros.

- **rep_get_doc_metadata <session file> <document name>:**

  Este comando devolve ao utilizador a metadata pública do ficheiro (caso este exista na organização do cliente). O servidor procura pela organização do cliente e no ficheiro das organizações pelo documento. Caso este exista, envia para o cliente a metadata pública do ficheiro. Este comando só pode ser executado por utilizadores com permissão DOC_READ.

- **rep_get_doc_file <session file> <document name> [file]:**

  Este comando é uma combinação de **rep_get_file, rep_get_doc_metadata** e **rep_decrypt_file.** Dado um ficheiro de uma organização, o servidor procura pelo documento e a sua metadata, desencripta o ficheiro e envia o seu conteúdo para o cliente. O resultado é impresso no terminal ou escrito num ficheiro passado pelo cliente (opcional). Requer a permissão DOC_READ.

- **rep_delete_doc <session file> <document name>:**
  Dado um documento de uma organização, o servidor impede a obtenção do ficheiro por parte de outros utilizadores, ao eliminar a file_handle da metadata do ficheiro. A execução deste comando requer a permissão DOC_DELETE.

- **rep_acl_doc <session file> <document name> [+/-] <role> <permission>:**
  Este comando permite adicionar ou remover permissões associadas a um papel específico para um documento em uma organização. Ao executar o comando, o servidor verifica se o utilizador tem permissão (DOC_ACL) para modificar a ACL do documento. O sinal + é utilizado para adicionar uma permissão a uma role para um documento, enquanto o sinal - é utilizado para removê-la. O servidor faz algumas validações antes de executar o comando (verifica se o documento existe na organização, se a role existe na organização, se a permissão está associada à role e por fim adiciona o documento ao dicionário da organização, se não existir).

ASVS

Em sistemas com múltiplos utilizadores, é crucial garantir que cada pessoa tenha acesso apenas aos recursos e funcionalidades que lhe foram atribuídos, protegendo dados sensíveis e evitando elevação de privilégios.

Problema:
Sem um controlo robusto de permissões:

- Os utilizadores podem aceder a funcionalidades ou dados que não lhes são permitidos.
- Ataques de elevação de privilégios podem comprometer o sistema.
- A confidencialidade e integridade dos dados ficam em risco.

Gestão de Roles e Permissões

- O sistema utiliza um modelo baseado em **roles** (ex.: manager) para definir permissões específicas para cada tipo de utilizador.
- Estas roles são configuradas na organização e associadas aos utilizadores através de um ficheiro JSON (organizations.json).

Controlo Centralizado

- Todas as permissões são definidas e verificadas no lado do servidor, garantindo que os dados não possam ser manipulados pelo cliente.

Tópicos OWASP V4 - Aplicáveis e Não Aplicáveis

4.1.1 Verify that the application enforces access control rules on a trusted service layer, especially if client-side access control is present and could be bypassed.

- **Aplicável**:

- No método /subjects/add, o servidor valida se o utilizador tem permissões antes de executar a operação:

```
requester_username = session['username']
org_data = organizations[organization]
for subject in org_data['subjects']:
    if subject['username'] == username:
        return jsonify({'error': f"Subject with username '{username}' already exists in the organization."}), 409
for subject in org_data['subjects']:
    if subject['username'] == requester_username:
        user_permissions = subject['permissions']
        break
if 'SUBJECT_NEW' not in user_permissions:
    return jsonify({'error': 'Permission denied'}), 403
```

Figura 1 – endpoint /subjects/add

4.1.2 Verify that all user and data attributes and policy information used by access controls cannot be manipulated by end users unless specifically authorized.

- **Aplicável**
  - As permissões e roles são armazenadas no ficheiro JSON e geridas centralmente.


```
{
    "UA": {
        "name": "UA",
        "subjects": [
            {
                "username": "admin",
                "name": "Antonio",
                "email": "antonio@ua.pt",
                "public_key": "-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4ux05LDP5eI+UuGsBPas\n8e30RwS50IhsyY4eyoZ/Lfq6dothm5V+MF3w8Z16bgXKQKm6P/PPHb
                "status": "active",
                "roles": [
                    "manager"
                ],
                "permissions": [
                    "ROLE_ACL",
                    "ROLE_DOWN",
                    "ROLE_UP",
                    "ROLE_NEW",
                    "ROLE_MOD",
                    "SUBJECT_NEW",
                    "SUBJECT_UP",
                    "SUBJECT_DOWN",
                    "DOC_NEW",
                    "DOC_READ",
                    "DOC_DELETE",
                    "DOC_ACL"
                ]
            },
```


Figura 2 - Organizations.json

4.1.3 Verify that the principle of least privilege exists - users should only be able to access functions, data files, URLs, controllers, services, and other resources, for which they possess specific authorization. This implies protection against spoofing and elevation of privilege. (C7)

- **Aplicável**
  - Apenas utilizadores com permissões adequadas podem realizar operações sensíveis, como criar utilizadores ou suspender contas (imagem acima).

4.1.5 Verify that access controls fail securely including when an exception occurs. (C10)

- **Aplicável**
  - O sistema captura exceções e evita que dados sensíveis sejam divulgados.
    No método /subjects/add, se ocorrer uma exceção, uma mensagem genérica é devolvida:

```
    if 'SUBJECT_NEW' not in user_permissions:
        return jsonify({'error': 'Permission denied'}), 403
    new_subject = {
        'username': username,
        'name': name,
        'email': email,
        'public_key': public_key,
        'status': 'active',
        'roles': [],
        'permissions': []
    }
    org_data['subjects'].append(new_subject)
    save_json(ORG_FILE, organizations)

    return jsonify({'message': f"Subject '{username}' added successfully."}), 201

except Exception as e:
    return jsonify({'error': str(e)}), 500
```

Figura 3 - endpoint subjects/add

4.2.1 Verify that sensitive data and APIs are protected against Insecure Direct Object Reference (IDOR) attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records.

- **Aplicável**
  - O sistema valida permissões antes de permitir operações em dados sensíveis, por exemplo no método /subjects/suspend, verifica-se a permissão SUBJECT_DOWN antes de suspender utilizadores:

```
    org = organizations[sessions['organization']]
    if not org:
        return jsonify({'error': 'Organization not found'}), 404

    subjects = org['subjects']
    sub_permissions = []
    for subject in subjects:
        if subject['username'] == sessions['username']:
            sub_permissions = subject['permissions']
            break
    if 'SUBJECT_DOWN' not in sub_permissions:
        return jsonify({'error': 'Permission denied'}), 403

    for subject in org['subjects']:
        if subject['username'] == username:
            if subject['status'] == 'suspended':
                return jsonify({'message': f"Subject '{username}' is already suspended."}), 400
            subject['status'] = 'suspended'
            save_json(ORG_FILE, organizations)
            return jsonify({'message': f"Subject '{username}' suspended successfully."}), 200

    return jsonify({'error': 'Subject not found'}), 404

except Exception as e:
    return jsonify({'error': str(e)}), 500
```

Figura 4 - endpoint /subjects/suspend

4.2.2 Verify that the application or framework enforces a strong anti-CSRF mechanism to protect authenticated functionality, and effective antiautomation or anti-CSRF protects unauthenticated functionality.

- **Não aplicável**
  - A aplicação não utiliza frameworks que necessitem de proteção contra CSRF. Todas as operações são realizadas por chamadas de API autenticadas.

4.3.1 Verify administrative interfaces use appropriate multi-factor authentication to prevent unauthorized use.

- **Não aplicável**
  - O sistema utiliza autenticação baseada em senhas e chaves privadas (RSA), mas não implementa multifator.

```
if not validate_password(password, private_key_pem, salt):
    return jsonify({'error': 'Invalid password'}), 400
```

Figura 5 - autenticação RSA

4.3.2 Verify that directory browsing is disabled unless deliberately desired. Additionally, applications should not allow discovery or disclosure of file or directory metadata, such as Thumbs.db, .DS_Store, .git or .svn folders.

- **Não aplicável**

4.3.3 Verify the application has additional authorization (such as step up or adaptive authentication) for lower value systems, and / or segregation of duties for high value applications to enforce anti-fraud controls as per the risk of application and past fraud.

- **Não aplicável**
  - O sistema implementa uma segregação de permissões e papéis, garantindo que os utilizadores só possam executar ações autorizadas com base nos seus papéis (por exemplo no endpoint/session/assume_role, onde se verifica se a role existe, garante se a role está ativo e depois disso é que se realiza a função). No entanto, esta segregação não considera explicitamente o risco associado às operações ou o histórico de fraudes.