

Segurança de Sistemas Informáticos
Trabalho pratico nº1
Relatório de Desenvolvimento

Pedro Gomes (a84220)

Filipa Silva (a81015)

20 de Novembro de 2020

Contents

1	Introdução	3
1.1	<i>Data Flow Diagram</i>	3
1.2	Aplicação <i>mID</i>	4
1.3	Aplicação leitora	4
1.4	<i>Backend</i>	5
1.5	Objetivo	5
2	Modelo de ameaças	6
2.1	Modelo <i>STRIDE</i>	6
2.2	Aplicação <i>mID</i>	6
2.3	Aplicação leitora	8
2.4	<i>BackEnd</i>	9
3	Modelo de vulnerabilidades	10
3.1	<i>Django</i>	10
3.2	<i>PostgreSQL</i>	11
3.3	<i>Ubuntu</i>	11
3.4	<i>Docker</i>	12
4	Conclusão	13

1 Introdução

Nesta secção vamos identificar os componentes da aplicação e resumir a sua função na aplicação e como comunicam entre si. Bem como, o objetivo primordial do trabalho prático.

1.1 *Data Flow Diagram*

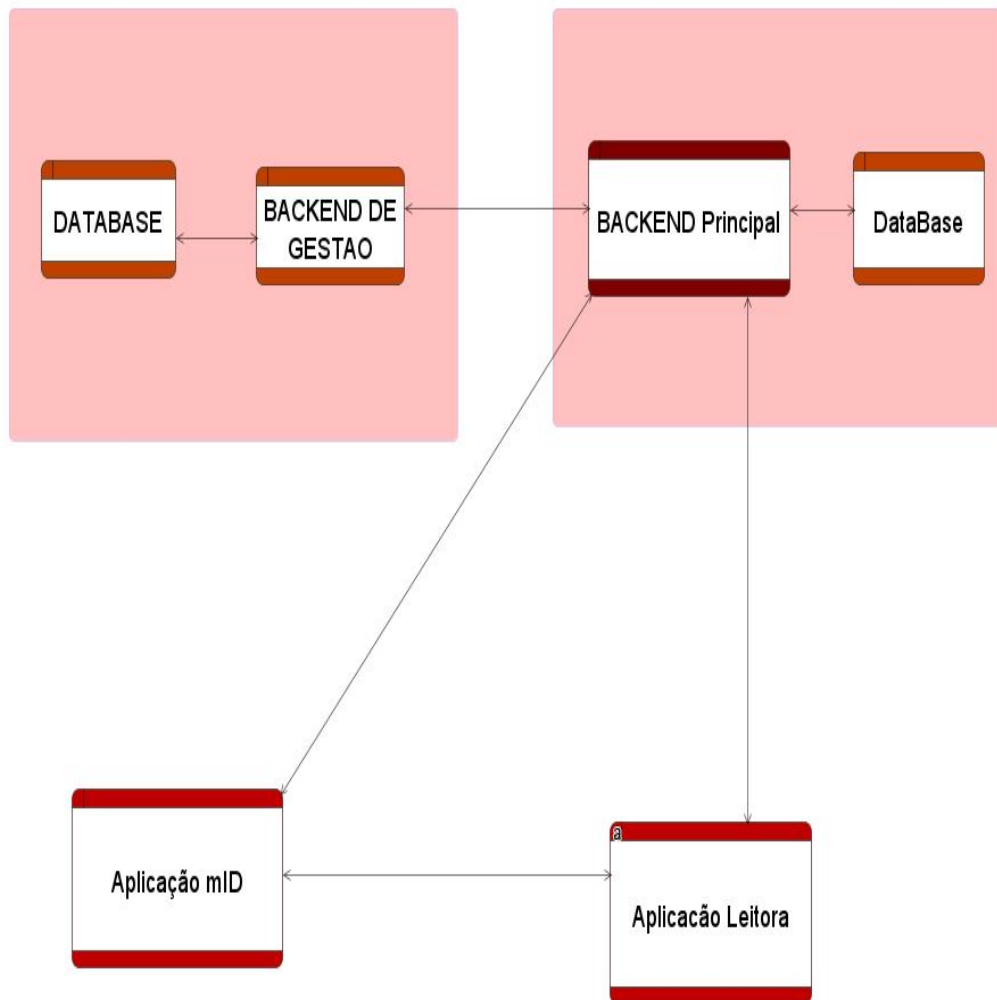


Figure 1: Dataflow diagram da aplicação.

1.2 Aplicação *mID*

A aplicação móvel *mID* para sistemas operativos Android e IOS consiste em armazenar dados de documentos de identificação pessoal e adicionalmente elementos que verifiquem a autenticidade e integridade dos mesmos dados.

Na primeira utilização esta aplicação comunica com a infraestrutura da entidade emissora dos documentos usando conexão TCP/IP para fazer o *download* de todos os dados associado com o documento em questão. Para isso o utilizador necessita de se autenticar no serviço. Esta operação é repetida periodicamente para garantir atualizações dos dados e estes são transferidos no formato *JSON*.

A aplicação também comunica com a aplicação de leitura, que verifica os dados, e pode ocorrer de duas maneiras: *off-line* e *on-line*. Em qualquer dos casos a comunicação entre os dois dispositivos é feita através de *Bluetooth low energy*, *Near field Communication* ou *Wifi-Aware*.

Quando a conexão estiver estabelecida o portador escolhe enviar os dados que quer aquando do pedido do leitor. Estes mesmos dados são enviados no formato *Concise Binary Object Representation (CBOR)*.

1.3 Aplicação leitora

Tal como a aplicação *mID*, a aplicação leitora também é para Android e IOS e com esta aplicação o verificador estabelece comunicação com o portador e solicita os dados suficientes para identificar no âmbito do serviço em específico.

Esta aplicação também precisa de suportar os protocolos já indicados para a aplicação *mID*. Além disso, cabe ao verificador decidir se a operação será em modo *off-line* ou *on-line*. É necessário também, que forneça dados para permitir a auditoria das operações por parte do portador, para isso terá que realizar a autenticação.

1.4 *Backend*

A parte do *backend* da aplicação corresponde à entidade com poder de emitir e conferir autenticidade de documentos de identificação pessoal. Além disso, esta entidade também é responsável por fornecer mecanismos que garantem a autenticidade e integridade dos documentos digitais que transmite aos portadores.

Como foi dito anteriormente, esta entidade comunica com a aplicação de leitura e do portador através de TCP/IP e é imperativo que esteja a todo o tempo disponível.

1.5 **Objetivo**

Após ter sido apresentado o enquadramento teórico daquilo que é a aplicação *mID*, o objetivo será estudar as possíveis vulnerabilidades e riscos que o funcionamento desta aplicação poderá apresentar para qualquer uma das entidades.

Para o realizar, não basta identificá-los, mas sim proceder à sua análise e descrição detalhada, para que os restantes membros que integram a equipa de trabalho entendam a que ameaças o sistema poderá estar associado.

2 Modelo de ameaças

Nesta secção vamos identificar as ameaças a que o sistema poderá estar exposto e para isso vamos usar o modelo **STRIDE** para catalogar essas mesmas ameaças.

2.1 Modelo **STRIDE**

- *Spoofing* - Fazer-se passar por alguém ou por algo que não é verdadeiro, violando a autenticação do sistema;
- *Tampering* - Modificar dados num disco, rede ou memória não persistente, violando a integridade do sistema;
- *Repudiation* - Conseguir recusar a confirmação sobre a autoria de um dado ato, violando a não-repudição do sistema;
- *Information Disclosure* - Divulgar informação confidencial para uma entidade que não tem autorização à mesma, violando a confidencialidade do sistema.
- *Denial of Service* - Esgotar recursos necessários para o funcionamento do sistema/serviço, violando assim a disponibilidade do sistema
- *Elevation of Privilege* - Permitir a uma entidade fazer alguma ação a que não tem privilégios para a fazer, violando assim a autorização do sistema.

2.2 Aplicação *mID*

Uma vez que a aplicação móvel está em contacto com dados pessoais do documento do utilizador, isto cria uma ameaça elevada e permanente em relação a esses mesmos dados. Visto que, há motivação para terceiros roubarem os dados, por exemplo *data miners*, assim como o próprio utilizador, que pode ter a motivação de alterar os próprios dados, adicionando mais pontos na carta, por exemplo.

Estes mesmos dados podem ser alterados/roubados quando estão armazenados no dispositivo, quando são transferidos da entidade reguladora, ou quando são transferidos para a aplicação leitora. Deste modo, estes 3 momentos são identificados como situações críticas e requerem atenção especial, pois a segurança deve ser máxima.

De seguida com o auxílio do modelo **STRIDE**, especifica-se a que tipo de ameaça a aplicação está vulnerável, nos 3 momentos anteriormente descritos.

- ***Spoofing*** - Um dos momentos críticos na segurança da informação é no momento de transferir os dados para a aplicação leitora. No caso do *Bluetooth low energy* encontramos vulnerabilidades recentes, **CVE-2020-11957**, que reportam que o momento de paridade entre os dois dispositivos não é seguro e permite a um atacante executar um ataque no tipo "man in the middle" o que levaria o atacante fazer-se passar por uma das aplicações ou as duas e roubar os dados que estão a ser transferidos sem nunca ser detetado. No entanto esta ameaça de um atacante interetar as comunicações entre as aplicações não é exclusiva ao *Bluetooth low energy*, o *Near field communication* e o *Wifi-Aware* também serão alvo dos mesmos tipos de ataques por isso os autores acreditam que uma implementação de um protocolo de encriptação, independentemente do tipo de comunicação usada, será fundamental para garantir a segurança da informação.
- ***Tampering*** - Esta ameaça é real devido ao utilizador da aplicação e portador dos dados, ter motivação para alterar os dados do próprio documento para benefício próprio pondo em causa a integridade do mesmo. Para isso, é fundamental que os dados tenham um mecanismo de segurança, que garanta ao leitor que os dados são autênticos e que são realmente da entidade reguladora, e não fabricados pelo utilizador.
- ***Repudiaton*** - Esta ameaça existe pois, tanto o próprio utilizador da aplicação como o leitor que fez o pedido dos dados, podem ter motivação de negar que dada transferência de dados ocorreu para benefício próprio. Portanto, é imperativa a existência de um mecanismo que permita auditar todas as interações ocorridas com dispositivos leitores, com todos os metadados necessários (ex.: data, hora, entre outros) e garantir a integridade dos mesmos metadados.
- ***Information Disclosure*** - Esta ameaça é real uma vez que as comunicações com a entidade reguladora são efetuadas por TCP/IP. O que não garante, por si só, a privacidade dos dados e, por isso, um atacante pode aceder aos dados e roubá-los. Logo, na perspetiva dos autores, é necessária a implementação do protocolo *OpenSSL* para encriptar os dados por forma a garantir a segurança na comunicação.

2.3 Aplicação leitora

A aplicação leitora está em contacto com dados pessoais do documento do portador da aplicação *mID*, isto cria uma ameaça elevada e permanente em relação a esses mesmos dados. Visto que, há motivação para terceiros roubarem os dados, por exemplo *data miners*, assim como o próprio utilizador, que pode ter a motivação roubar dados do portador para benefício próprio.

De seguida com o auxílio do modelo *STRIDE*, especifica-se a que tipo de ameaça a aplicação está vulnerável, nos 3 momentos anteriormente descritos.

- ***Spoofing*** - A aplicação leitora comunica com a aplicação do portador com os métodos que já acima mencionados (*BLE*, *NFC*, *Wifi-Aware*), e, tal como se explicou, deve ser implementado um protocolo de encriptação independentemente do tipo de comunicação usada.
- ***Tampering*** - Esta ameaça é real uma vez que no modo *on-line* o utilizador da aplicação *mID* dá permissão sob quais os atributos que quer que sejam visíveis. De seguida, o leitor estabelece o pedido desses mesmos atributos à *backend* para assegurar que os dados estão atualizados. A ameaça existe na medida em que o leitor pode alterar os atributos permitidos pelo utilizador *mID*, e assim ter acesso a dados do utilizador não autorizados. Deste modo considera-se que é imperativo que haja um mecanismo de segurança que não permita a modificação dos dados por parte do leitor.
- ***Information Disclosure*** - A aplicação leitora, tal como a aplicação do portador, também comunica com a entidade reguladora por TCP/IP o que não garante, por si só, a privacidade dos dados e, por isso, um atacante pode aceder aos dados e roubá-los. Logo, na perspetiva dos autores, a solução passaria pela implementação do protocolo *OpenSSL* para encriptar os dados para garantir a segurança na comunicação, tal como foi referido acima.
- ***Elevation of Privilege*** - Esta ameaça é real devido ao utilizador da aplicação leitora poder ter motivação de realizar auditorias, sem ter permissões para tal, para conseguir acesso a dados do utilizador para benefício próprio. Deste modo, sugere-se que seja implementado um sistema de autenticação para garantir que o leitor tem permissões para fazer dada auditoria e confirmar que o leitor é realmente quem diz ser, uma opção para tal problema seria o *JSON Web Token*.

2.4 *BackEnd*

O modelo de segurança para esta parte do *backend* é complicado, dado que para acontecer qualquer tipo de ameaça a esta parte do sistema, tem que existir primeiro uma quebra na autenticação do utilizador/administrador do sistema que tem acesso que tem acesso à *infraestrutura* e a todos os serviços inerentes a esta.

- **STRIDE**

Consoante o tipo de *login* necessário para ter acesso à entidade emissora, o que pode acontecer é um **Web Spoofing**, isto é, existe uma página que finge ser a do *login* do administrador, e assim consegue obter as informações inseridas pelo mesmo, em espécie de formulários. Para contornar este tipo de situações é possível utilizar outros métodos de autenticação, como por exemplo, a impressão digital.

Se a dificuldade anteriormente mencionada não for ultrapassada, todo o núcleo do sistema será ameaçado, e consequentemente podem existir ameaças de todos os tipos.

Primeiramente, se o atacante conseguiu obter privilégios que não teria e conseguiu manipular a *infraestrutura* por forma a enviar informações que são certificadas pela entidade emissora, estamos perante **Elevation of Privilege**. Com este acesso total, o atacante pode facilmente estagnar o serviço fornecido por esta entidade, aos seus utilizadores. Há várias alternativas para esgotar os serviços, neste caso, podem ser inseridos, exageradamente, inúmeros dados na base de dados, que tornem o sistema incapaz de receber e fornecer a base de conhecimento, inicia-se aqui, um **Denial of Service**.

A partir do momento em que a base de dados de um sistema se encontra em *check*, o sistema está numa situação bastante grave. Assim sendo, a visualização dos dados pode levar a um acesso às chaves criptográficas privadas em memória e, a um posterior acesso às credenciais de acesso dos utilizadores. Existindo este cenário, estamos claramente perante um **Information Disclosure**.

No seguimento do exemplo anterior, o atacante tem agora o poder de efetuar ações "em nome" dos vários utilizadores, e ainda de eliminar os *logs* de ações já efetuadas pelos mesmos. Naturalmente, isto quebra o sistema ao nível da **Repudiation**.

Em jeito de conclusão, é também possível existir **Tampering** visto que todos os caminhos de dados, processos e de rede vão dar à *infraestrutura* e assim sendo, podem ser alterados ficheiros do sistema, forjando a veracidade dos mesmos.

3 Modelo de vulnerabilidades

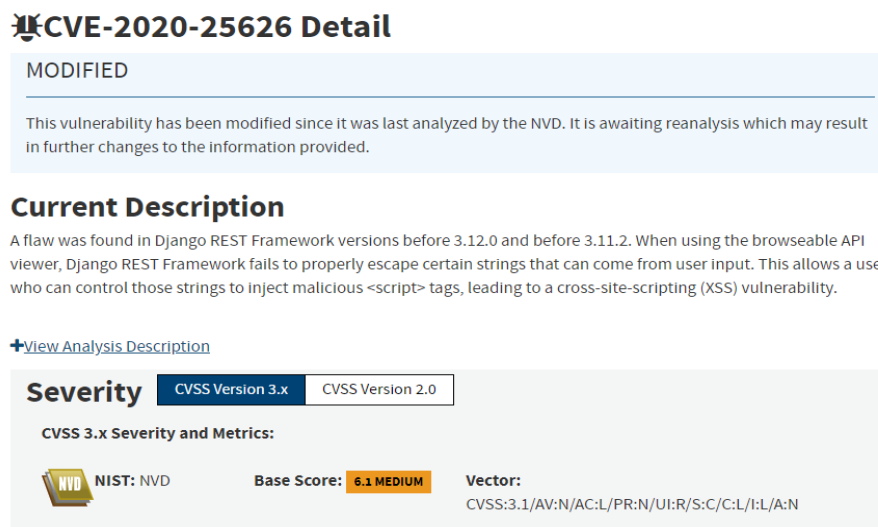
Nesta seção, identificam-se vulnerabilidades que existam na parte do *backend* da aplicação, ou seja, esta é a única estrutura onde se sabem quais os *softwares* específicos implementados, incluindo as versões dos mesmos. É através dessas mesmas versões que serão estudadas as vulnerabilidades presentes e o risco que elas representam.

Naturalmente, serão feitos apenas os levantamentos de riscos que os autores considerem se enquadrar numa utilização conjunta e não desses softwares em separado.

3.1 *Django*

Após ser feito um enquadramento das vulnerabilidades existentes neste *software* de *backend* e na versão a ser utilizada, concluiu-se que há uma CVE ao qual se deve prestar atenção. Isto porque, ao utilizar a API deste *software*, podem ser deixadas escapar algumas *strings* que provêm do *input* do utilizador, desta forma essas *strings* podem dar acesso a que o utilizador injete *tags* `<script>` maliciosas e existam vulnerabilidades entre sites.

Pode acrescentar-se ainda que esta exploração é realizada através da rede e que as condições de ataque são baixas, o que significa que aumenta a probabilidade de sucesso. Em contrapartida, sabe-se que é baixa a perda da confidencialidade e da integridade.



CVE-2020-25626 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

A flaw was found in Django REST Framework versions before 3.12.0 and before 3.11.2. When using the browsable API viewer, Django REST Framework fails to properly escape certain strings that can come from user input. This allows a user who can control those strings to inject malicious `<script>` tags, leading to a cross-site-scripting (XSS) vulnerability.

[View Analysis Description](#)

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD Base Score: 6.1 MEDIUM Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N

Figure 2: Vulnerabilidade referente ao Django.

3.2 PostgreSQL

Esta falha vem sendo frequente ao longo das demais versões lançadas pelo PostgreSQL e ainda se encontra a ser analisada, porém deve ser tida em consideração uma vez que promete ameaçar a confidencialidade e integridade dos dados, bem como a disponibilidade do sistema. Aqui, um atacante com permissões, cria objetos não temporários no *schema* e executa funções SQL sob a identidade de um super utilizador.

CVE-2020-25695 Detail

AWAITING ANALYSIS

This vulnerability is currently awaiting analysis.

Description

A flaw was found in PostgreSQL versions before 13.1, before 12.5, before 11.10, before 10.15, before 9.6.20 and before 9.5.24. An attacker having permission to create non-temporary objects in at least one schema can execute arbitrary SQL functions under the identity of a superuser. The highest threat from this vulnerability is to data confidentiality and integrity as well as system availability.

Severity

CVSS Version 3.x

CVSS Version 2.0

CVSS 3.x Severity and Metrics:



NIST: NVD

Base Score: N/A

NVD score not yet provided.

Figure 3: Vulnerabilidade referente ao PostgreSQL.

3.3 Ubuntu

O pacote *libvirt*, na versão 20.04 do Ubuntu, criou um socket de controlo com permissões globais de leitura e escrita. Isto abre caminho, a que um atacante o use para substituir arquivos arbitrários ou executar código arbitrário.

Estando esta vulnerabilidade classificada com uma pontuação alta, considera-se que é crítica e portanto é necessário tê-la em conta. Uma vez que a complexidade do ataque é baixa e tem uma alta probabilidade de perda total de confidencialidade e integridade, bem como capacidade de negar o total acesso aos recursos no destino.

CVE-2020-15708 Detail


Current Description

Ubuntu's packaging of libvirt in 20.04 LTS created a control socket with world read and write permissions. An attacker could use this to overwrite arbitrary files or execute arbitrary code.

[+View Analysis Description](#)

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

 **NIST:** NVD **Base Score:** 7.8 HIGH **Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H


 **CNA:** Canonical Ltd. **Base Score:** 9.3 CRITICAL **Vector:** CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H

Figure 4: Vulnerabilidade referente ao Ubuntu.

3.4 Docker

Uma vulnerabilidade de divulgação de informações e, por isso, que afeta a confidencialidade dos dados, foi encontrada em *containers* a usar a API Varlink compatível com Docker. Portanto, se forem criados vários *containers* de curta duração, as variáveis do primeiro irão passar para os próximos. Deste modo, um atacante que tenha controlo sobre os *containers* subsequentes, pode usar essa falha para obter informações confidenciais armazenadas nessas variáveis.

CVE-2020-14370 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

An information disclosure vulnerability was found in containers/podman in versions before 2.0.5. When using the deprecated Varlink API or the Docker-compatible REST API, if multiple containers are created in a short duration, the environment variables from the first container will get leaked into subsequent containers. An attacker who has control over the subsequent containers could use this flaw to gain access to sensitive information stored in such variables.

[+View Analysis Description](#)

Severity CVSS Version 3.x CVSS Version 2.0

CVSS 3.x Severity and Metrics:

 **NIST:** NVD **Base Score:** 6.5 MEDIUM **Vector:** CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Figure 5: Vulnerabilidade referente ao Docker.

4 Conclusão

A modelação de ameaças, de acordo com a definição de um estudo da **Microsoft**, é uma técnica de engenharia que pode ser usada para identificar ameaças, atacantes, vulnerabilidades e contra medidas que poderão afetar a aplicação. Esta modelação tem como objetivo moldar o design da aplicação, torná-la segura e reduzir o risco.

Na aplicação *mID*, essa modelação foi baseada numa descrição detalhada dos seus componentes principais e das tecnologias de comunicação utilizadas. Com base nesta explicação, realizou-se um *Data Flow Diagram* que fornece uma visão esquemática daquilo que é o sistema na sua globalidade. Posto isto, procedeu-se, então, ao desenvolvimento propriamente dito da modelação de ameaças, através do modelo STRIDE. Este modelo indica claramente as permanentes ameaças que o sistema vai enfrentar, quem poderão ser os eventuais atacantes e quais os seus motivos. Por fim, considerou-se relevante identificar algumas vulnerabilidades da *backend*, tendo em conta a informação dos diversos *softwares* e das respetivas versões.

O grupo entende que a maior dificuldade foi enquadrar as situações analisadas no tipo de ameaça correspondente, e ainda, identificar qual a contramedida ou possível solução que deveria ser sugerida.

Em suma, após este trabalho, os autores consciencializaram-se dos perigos que uma aplicação *Web* enfrenta a nível de segurança e, em futuros projetos, tanto a nível pessoal como, eventualmente, a nível profissional será certamente um fator a ter em conta.