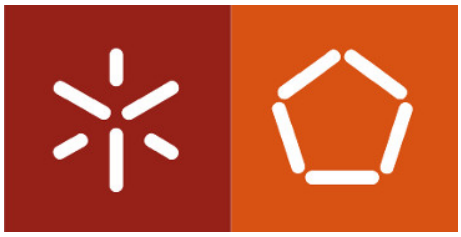
Tecnologia de Segurança

23 de Novembro de 2020

Trabalho Prático 1

a83899	André Morais
a84485	Tiago Magalhães

Análise e *Threat Model* da Aplicação mID



Mestrado Integrado em Engenharia Informática
Universidade do Minho

Conteúdo

1 Introdução

Neste trabalho prático, desenvolvido no âmbito da Unidade Curricular de Tecnologia de Segurança, foi-nos proposto tecer uma análise que identifique e descreva potenciais problemas de segurança e suas respetivas soluções.

Posteriormente vai ser feito um ***Threat Model*** para analisar mais detalhadamente as falhas/vulnerabilidades de um sistema de identificação digital e móvel.

Estando perante um sistema que seja utilizado por múltiplos intervenientes, torna-se necessária e crítica a existência de algum tipo de análise no que toca a segurança desta aplicação, de modo a garantir o bom funcionamento e a própria proteção de dados dos utilizadores.

No texto recorre-se, por vezes, ao uso de siglas tais como: **CWE**: *Common Weakness Enumeration*, **CVSS**: *Common Vulnerability Scoring System*, **CVE**: *Common Vulnerabilities and Exposures*, **DoS**: *Denial of Service* e **STRIDE** que se refere a um modelo para identificar ameaças na segurança computacional. É um acrónimo que representa as ameaças à segurança divididas em 6 categorias.

2 Descrição do sistema

Podemos dividir o sistema em três grandes componentes :

- **Aplicação do portador**

  Corresponde a uma aplicação móvel para sistemas **Android** e **IOS**. Esta aplicação armazena os dados de um documento de identificação e os elementos necessários para a aplicação **leitor** verificar a sua integridade e autenticidade.

  Numa primeira fase, o portador da aplicação tem de se inscrever partilhando os dados com uma entidade emissora, sendo estes armazenados.

  O portador pode atualizar os seus dados, estando em **mode online**, sem recorrer a uma autenticação mais explicita ao sistema.No **mode offline** apenas pode transferir os dados para a **aplicação leitora** através de *Bluetooth Low Energy, NFC* ou *WiFi-Aware*, sendo a informação mais recentemente atualizada, partilhada.

  Uma vez estabelecida uma conexão entre os dois dispositivos, o verificador envia um pedido com os atributos que deseja conhecer. A este pedido, o portador pode aceitar a transferência na sua totalidade, ou apenas um subconjunto destas. De relembrar, que às autoridades superiores, nenhuma informação pode ser negada.

- **Aplicação Leitora**

  É também uma aplicação móvel para sistemas **Android** e **IOS** ou para qualquer outro dispositivo que suporte os 3 protocolos de comunicações e operações aqui definidos.

  Para haver troca de informação entre os dois dispositivos, primeiramente, ambos têm de aceitar estabelecer uma comunição. Depois, através desta aplicação, o **verificador** solicita um pedido para a prova de maioridade, por exemplo, tendo o portador que aceitar ou rejeitar.

  Esta *app* pode atuar em dois modos: o **mode online**, em que o verificador escolhe uma lista de atributos que quer conhecer, e se eventualmente o portador aceitar, gera um token de autorização para que o verificador consulte diretamente a entidade emissora do documento, garantindo assim, dados mais recentes sobre o respetivo cidadão; o **mode offline**, faz se exatamente igual ao do outro modo, só que aqui, não se consegue verificar os dados à entidade emissora e portanto obtemos a informação que o portador tem armazenada no seu telemóvel, estejam os dados recentemente atualizados ou não. Pode pensar que o portador neste caso, se não atualizar os dados, sai beneficiado, mas não. Hoje em dia, muito dificilmente há falhas de rede e hipoteticamente falando nada disto seria pior do que o sistema de cartão físico

- **Entidade emissora(Back-end)**

  A **Entidade emissora** tem o poder de emitir e conferir autenticidade a um documento de identificação pessoal. Neste serviço, é também a responsável por fornecer os mecanismo que garantem integridade dos documentos digitais transmitidos tanto no **mode online**, como no **mode offline**.

3 Modelo do sistema

Sabendo os requisitos de segurança a serem considerados, também é necessário fazer uma modelação de sistema para podermos compreender onde cada vulnerabilidade se pode encontrar.

O **Threat Model** é um processo iterativo que permite identificar as principais ameaças e mitigações, pelo qual iremos começar pela modelação do sistema de maneira a perceber as principais componentes e fluxo de dados.

Esta aplicação pode ser usada em dois modos, o **mode offline** e o **mode online**. Para isso foi feita a modelação do sistema para estes dois tipos de modos.

3.1 Modo Online

Começamos por identificar as entidades do nosso sistema no **mode online**:

- **Entidade externas :** Utilizadores das aplicações (Portador e Verificador)
- **Processos :** Entidade Emissora/Back-end, Aplicação Leitora e Aplicação Portadora
- **Base de Dados :** PostgreSQL 12.1 (Base de dados de gestão) e PostgreSQL 12.4 (Base de dados principal)

Tendo em conta as entidades e processos caracterizados acima, procedemos para o desenvolvimento do modelo do sistema, apresentado na Figura 1

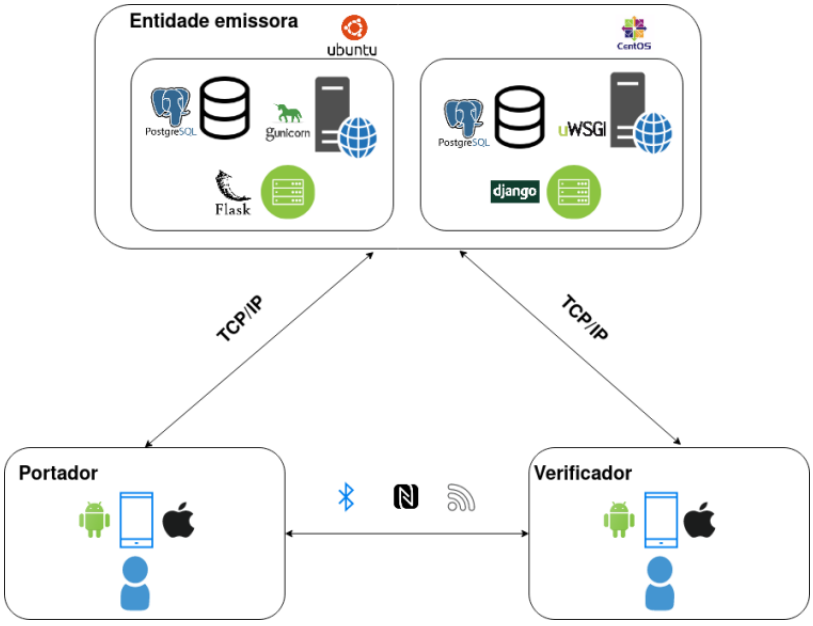


Figura 1: Modelo de Sistema Online

3.2 Modo Offline

No **mode offline** não conexão à rede, isto é, não há uma ligação direta com a entidade emissora, deixando de haver a participação desta.

- **Entidade externas :** Utilizadores das aplicações (Portador e Verificador)
- **Processos :** Aplicação Leitora e Aplicação Portadora
- **Base de Dados :** Dispositivo móvel do Portador

Especificadas as características do modelo, do **mode offline**, a **Figura 2** representa o modelo de sistema neste modo.

Figura 2: Modelo de Sistema Offline

4 Modelação de ameaças

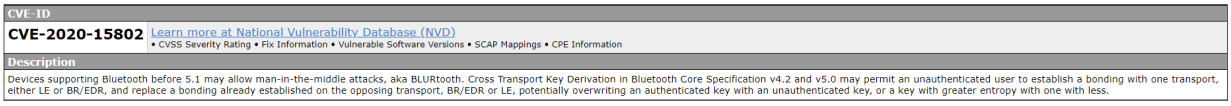
Após modelação do sistema usaremos o método ***STRIDE***, de maneira a identificar e catalogar ameaças e vulnerabilidades, bem como formas de mitigação.

Através da modelação do sistema anteriormente realizada podemos observar que existem três conexões/fluxos de dados: **(Aplicação Portador - Aplicação Leitora)**, **(Aplicação Portador - Entidade Emissora)** , **(Entidade Emissora - Aplicação Leitora)**, que apresentam um elevado risco de ameaça.

Na parte do *Back-end*, devido a um maior grau de informação e detalhe deste, existe também um risco de ameaça e por isso iremos aplicar o mesmo método a estas componentes e relações entre elas. A seguir encontram-se descrições de ameaças e vulnerabilidades para cada uma das categorias do **STRIDE**.

4.1 Aplicação Portador - Aplicação Leitora

- ***Spoofing***
  - **Descrição :** Atacante produz um novo *QRCode* que pode introduzir vulnerabilidade no dispositivo leitor, bem como dados não esperados para o *back-end*.
  - **Solução :** Validação de *input* e uso de uma assinatura digital no *QRCode*, uma vez que permite verificar o originador do código (não repúdio) e a integridade.
- ***Tampering***
  - **Descrição :** Um atacante pode explorar uma vulnerabilidade presente na componente CTKD no *Bluetooth Low Energy* de forma a alterar as chaves de autenticação do dispositivo.
  - **Solução :** Atualização do sistema.


CVE-ID

CVE-2020-15802 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

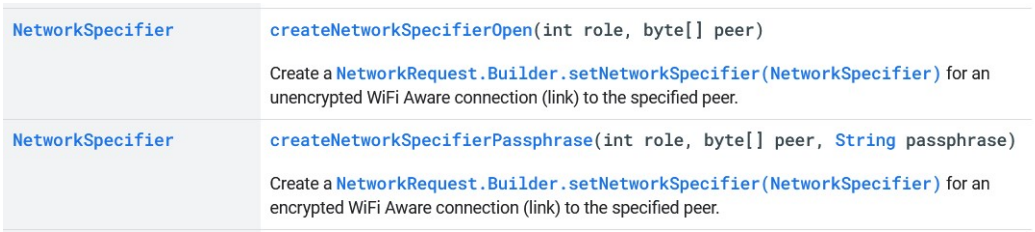
Description

Devices supporting Bluetooth before 5.1 may allow man-in-the-middle attacks, aka BLURtooth. Cross Transport Key Derivation in Bluetooth Core Specification v4.2 and v5.0 may permit an unauthenticated user to establish a bonding with one transport, either LE or BR/EDR, and replace a bonding already established on the opposing transport, BR/EDR or LE, potentially overwriting an authenticated key with an unauthenticated key, or a key with greater entropy with one with less.

Figura 3: Vulnerabilidade do BLE

- ***Information Disclosure***
  - **Descrição :** Desenvolvimento do WiFi-Aware não ter especificação para o estabelecimento de comunicações cifradas, por exemplo o método *createNetworkSpecifierPassphrase()* cria comunicações sem cifragem, o que leva a que a informação não esteja decifrada e possa haver fugas de informação.
  - **Solução :** Usar o método *createNetworkSpecifierOpen()* .
  - **Descrição :** Um atacante pode usar uma antena para obter informação de comunicação NFC. apesar de serem comunicações muito próximas o risco continua presente.

– **Solução :** Estabelecer canais seguros entre comunicação NFC.



`NetworkSpecifier`	`createNetworkSpecifierOpen(int role, byte[] peer)` Create a `NetworkRequest.Builder.setNetworkSpecifier(NetworkSpecifier)` for an unencrypted WiFi Aware connection (link) to the specified peer.
`NetworkSpecifier`	`createNetworkSpecifierPassphrase(int role, byte[] peer, String passphrase)` Create a `NetworkRequest.Builder.setNetworkSpecifier(NetworkSpecifier)` for an encrypted WiFi Aware connection (link) to the specified peer.

Figura 4: *Wifi-Aware Android Development*

- ***Denial of Service***
  - **Descrição :** É possível afetar o **BLE** no nível protocolo e isso pode ser usado para ameaçar a disponibilidade do sistema. Alguns dos BLE chipset não fornecem *"real time resolving of private adresses"*, e por isso ficam expostos ao ataque onde é possível um *master* malicioso se conectar a um *slave*.
  - **Solução :** Optar por outro meio de transferência, quando este se encontrar indisponível.

4.2 Aplicação Portador - Entidade Emissora

- ***Spoofing***
  - **Descrição :** Um atacante pode fazer pode fazer-se passar por algum cidadão, através de um ataque *man in the middle*, uma vez que há necessidade de confidencialidade dados é necessário o uso de criptografia, o atacante pode colocar-se no meio da comunicação e forjar a chave pública de um cidadão se esta não for verificada.
  - **Solução :** Uso de uma infraestrutura de chaves públicas, que oferece uma maior certificação e credibilidade das chaves públicas.
  - **Descrição :** Um atacante pode fazer pode fazer-se passar por algum cidadão,
  - **Solução :** Uso de uma autenticação forte como certificados de chave pública, tanto do lado do cliente como da entidade emissora.
- ***Tampering***
  - **Descrição :** A ameaça descrita acima pode aplicar-se aqui também, assim como utilizadores mal intencionados podem alterar o dados armazenados localmente que foram transferidos, comprometendo a integridade, esta é uma das debilidades mais comuns no desenvolvimento de *software* relativamente a integridade de dados (**Figura 5**).

- **Solução :** Uso de *checksum* no dados transferidos permitindo saber que a cópia dos dados é genuína e íntegra (sem corrupção).

Figura 5: CWE-353: *Missing Support for Integrity Check.*

- ***Repudiation***
  - **Descrição :** Um atacante pode dizer que não enviou determinados dados, o uso de *JSON* não garante o não repúdio de um pedido que foi realmente realizado por este.
  - **Solução :** Uso de *Json Web Signature* permite que exista uma assinatura no dados transferidos garantindo que foi gerada por quem os enviou, não podendo repudiar.
- ***Information Disclosure***
  - **Descrição :** Um atacante que esteja a usar *sniffer* de pacotes consegue ver as mensagens trocadas em *plain-text*, dado que uma das vulnerabilidades do protocolo TCP/IP é não possuir mecanismos de cifragem de dados.
  - **Solução :** Cifrar os dados transmitidos nas comunicações.
- ***Denial of Service***
  - **Descrição** : Um atacante pode comprometer a disponibilidade do sistema, através das ligações TCP/IP entre as aplicações e a entidade emissora através de ataques DoS, tais ataques podem ser *SYN flood attacks*, *Volumetric attacks* entre outros, uma vez que as comunicações TCP/IP são conhecidas pelas sua vulnerabiliades a DoS.
  - **Solução** : Como neste sistema não é suposto que o utilizadores enviem grandes quantidades de dados, por isso o uso e boa configuração de uma *firewall* pode ser uma solução e para complementar o uso de *SYN cookies*, que permite que o servidor só após verificação da conexão através do uso de uma *cookie* é que aloca os recursos necessários em vez de responder logo e ficar com a conexão em aberto ficando exposto a DoS.

4.3 Aplicação Leitora - Entidade Emissora

As ameaças apresentadas anteriormente também se aplicam aqui, a seguir serão apresentadas ameaças adicionais.

- ***Spoofing***
  - **Descrição :** Um atacante pode montar uma aplicação leitor falsa.
  - **Solução :** Uso de uma autenticação forte como certificados de chave pública, tanto do lado do cliente como da entidade emissora.
- ***Tampering***
  - **Descrição :** Um atacante pode alterar a lista de atributos fornecidos pelo portador.
  - **Solução :** Uso de *checksum* no dados transferidos permitindo saber que a lista dos atributos é genuína e integra (sem corrupção).
- ***Repudiation***
  - **Descrição :** Um atacante com o dispositivo leitor pode dizer que não solicitou determinados atributos ou realizou determinada verificação.
  - **Solução :** Uso de um sistema de *logs* de auditoria cronoloógicos.

4.4 Back-end

Devido à existência de uma maior descrição das tecnologias utilizadas no *back-end*, iremos abordar mais vulnerabilidades que ameaças.

- ***Tampering***
  - **Descrição :** A versão 19.4.5 do Gunicorn contem uma vulnerabilidade (**Figura 6**), em que um atacante pode enviar cabeçalhos HTTP arbitrários e levar a alteração de dados (integridade),isto é consequência de uma debilidade no desenvolvimento software (**Figura 7**).
  - **Solução :** Atualização da versão do Gunicorn para 19.5.0, onde a vulnerabilidade se encontra corrigida.

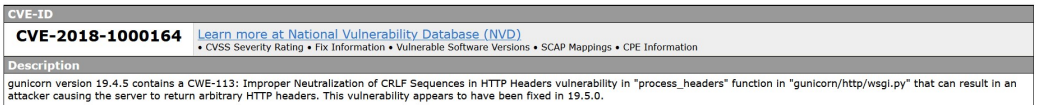


Figura 6: CVE-2018-1000164

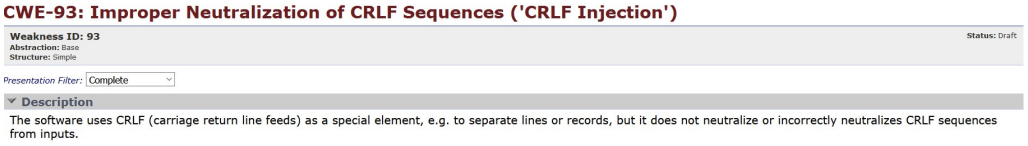


Figura 7: CWE-93

- ***Repudiation***
  - **Descrição :** UWSGI permite configurar o *back-end* da aplicação web dinamicamente através do protocolo UWSGI que usa determinadas variáveis. Se a porta UWSGI for exposta, atacante podem construir pacotes UWSGI e especificar as variáveis de maneira a injetar código arbitrário. Isto é uma da vulnerabilidades mais comuns no densenvolvimento de *software OS Command Injection* (**Figura 8**). Uma vez que é aplicação a executar o código arbitrário o atacante, isto permite o repúdio do atacante.
  - **Solução :** A porta do uWSGI não pode ser publicamente acessível. UWSGI deve ser configurado apenas para "ouvir"apenas na interface local (127.0.0.1).

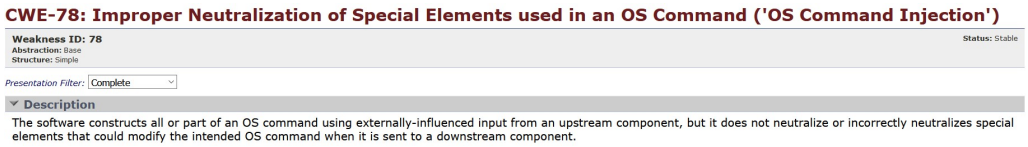
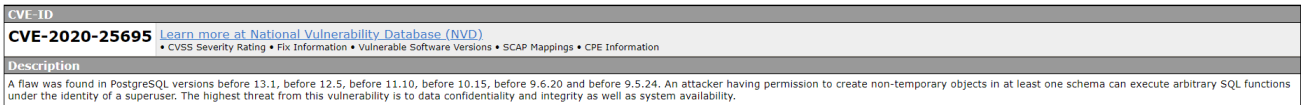


Figura 8: CWE-78: *OS Command Injection*

- ***Information Disclosure***
  - **Descrição :** Existem várias vulnerabilidades associadas ao CentoOS *web panel* [1] que pode levar a *information disclosure* (**Figura 10**). Uma vez que este sistema operativo irá ter um servidor web, isto apresenta um risco, se o gestor do servidor não estiver ciente destas vulnerabilidades pode comprometer o sistema.
  - **Solução :** Uma solução é restringir o acesso com o serviço apenas a maquinas confiáveis, isto pode ser obtido com regras de *firewall* acerca da permissões.
  - **Descrição :** Uma das últimas vulnerabilidades expostas ao público sobre o *PostgreSQL* foi a de um atacante puder executar funções SQL sob a identidade de um **superuser**.
  - **Solução :** Uma das estratégias a optar é o ***Input Validation***, isto é, assumir que todos os inputs são maliciosos. Neste caso, usar uma validação "aceitar o que é bom input", por exemplo, usar uma lista de inputs que cumprem com certas especificações

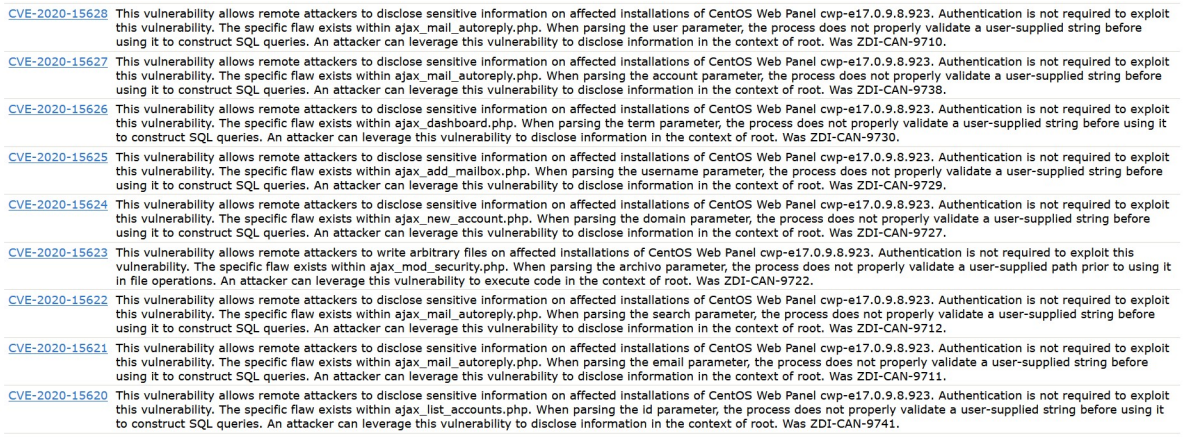

CVE-ID

CVE-2020-25695 Learn more at National Vulnerability Database (NVD)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

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.

Figura 9: CVE-2020-25695 PostgreSQL



CVE-2020-15628	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_mail_autoreply.php. When parsing the user parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9710.
CVE-2020-15627	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_mail_autoreply.php. When parsing the account parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9738.
CVE-2020-15626	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_dashboard.php. When parsing the term parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9730.
CVE-2020-15625	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_add_mailbox.php. When parsing the username parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9729.
CVE-2020-15624	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_new_account.php. When parsing the domain parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9727.
CVE-2020-15623	This vulnerability allows remote attackers to write arbitrary files on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_mod_security.php. When parsing the archivo parameter, the process does not properly validate a user-supplied path prior to using it in file operations. An attacker can leverage this vulnerability to execute code in the context of root. Was ZDI-CAN-9722.
CVE-2020-15622	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_mail_autoreply.php. When parsing the search parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9712.
CVE-2020-15621	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_mail_autoreply.php. When parsing the email parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9711.
CVE-2020-15620	This vulnerability allows remote attackers to disclose sensitive information on affected installations of CentOS Web Panel cwp-e17.0.9.8.923. Authentication is not required to exploit this vulnerability. The specific flaw exists within ajax_list_accounts.php. When parsing the id parameter, the process does not properly validate a user-supplied string before using it to construct SQL queries. An attacker can leverage this vulnerability to disclose information in the context of root. Was ZDI-CAN-9741.

Figura 10: Vulnerabilidades CentOs

- ***Elevation of Privilege***
  - **Descrição :** Todas as bases de dados têm diferentes tipos de permissões. Se um atacante conseguir os privilégios de administrador, toda a informação lá contida fica comprometida.

[1] painel de controle para gerenciar servidores virtuais

  - **Solução :** Implementação de um bom sistema de controlo de acessos.

- **Vulnerabilidades e ameaças gerais** : Quanto ao uso da tecnologia *docker* deve-se sempre utilizar imagens neste caso PostgreSQL oficiais e usar ferramentas como *Docker Bench for Security* para detetar vulnerabilidades, com isto pode-se evitar vulnerabilidades introduzidas despropositadamente.

5 Análise de Risco

Do nosso ponto de vista o recurso mais importante é a entidade emissora, uma vez que é esperado que em grande parte das transações esta componente esteja disponível, em virtude de o modo online representar maior parte do das transações que serão efetuadas, também é na entidade emissora que se encontram as bases dados com documentos/dados importantes dos cidadãos, daí ser importante garantir que o *back-end* tenha controlos de segurança, tais como a nível da disponibilidade garantir que o sistema de gestão não falha, o Gunicorn permite definir um *cluster*, consequentemente isto possibilita a mitigação de que em caso de falha existam réplicas. No entanto tanto o Gunicorn como o UWSGI são utilizados como servidores de aplicação e nas suas documentações recomendam o uso de um *proxy* de modo a evitar ataques DoS, um *proxy* como por exemplo NGINX será útil para garantir a disponibilidade, uma vez que lida melhor com uma maior quantidade de tráfego.

Ao nível das aplicações portador/leitor muitos riscos são reduzidos devido a utilização de tecnologias de transferência de contacto próximo e também de que no modo *online* os dados são obtidos através da entidade emissora o que garante uma maior fiabilidade. Contudo no modo *offline* é importante garantir a integridade dos dados armazenados no portador.

6 Novos elementos para a infra-estrutura

De forma complementar aos elementos do sistema já existentes adicionaríamos, já pelas soluções apresentadas anteriormente o uso de ***firewall*** e ***proxy*** para garantir disponibilidade, **infraestrutura de chaves públicas** para garantir a confidencialidade e autenticação e um **sistema de *logs*** para permitir auditar.

7 Conclusão

A realização deste trabalho permitiu aprender a elaborar um *Threat Model*. A concretização deste é muito importante para o sistema mesmo antes de o criar, pois permite identificar ameaças e encontrar soluções para estas, além de que, possibilita atingir os requisitos de segurança, que por vezes são ignorados no desenvolvimento em vários projetos.

Deste modo compreendemos o quão se torna importante para sistemas/projectos a realização destes modelos, porque para além de documentação, permitem compreender o sistema e saber onde se encontram as falhas destes e possíveis soluções. Com isto, em trabalhos futuros certamente iremos estar mais conscientes da utilização deste modelo e requisitos de segurança.

Referências

[1] Common Vulnerabilities and Exposures Disponível em:
`https://cve.mitre.org/`

[2] Common Vulnerabilities and Exposures Disponível em:
`https://blog.finjan.com/tcpip-vulnerabilities/`

[3] Common Vulnerabilities and Exposures Disponível em:
`https://www.imperva.com/learn/ddos/syn-flood/`

[4] QR Code Security: A Survey of Attacks and Challenges for Usable Security
https://publications.sba-research.org/publications/llncs.pdf

[5] Denial of Service Attack on Bluetooth Low Energy
https://www.researchgate.net/publication/317063884_Denial_of_Service_Attack_on_Bluetooth_Low_Energy