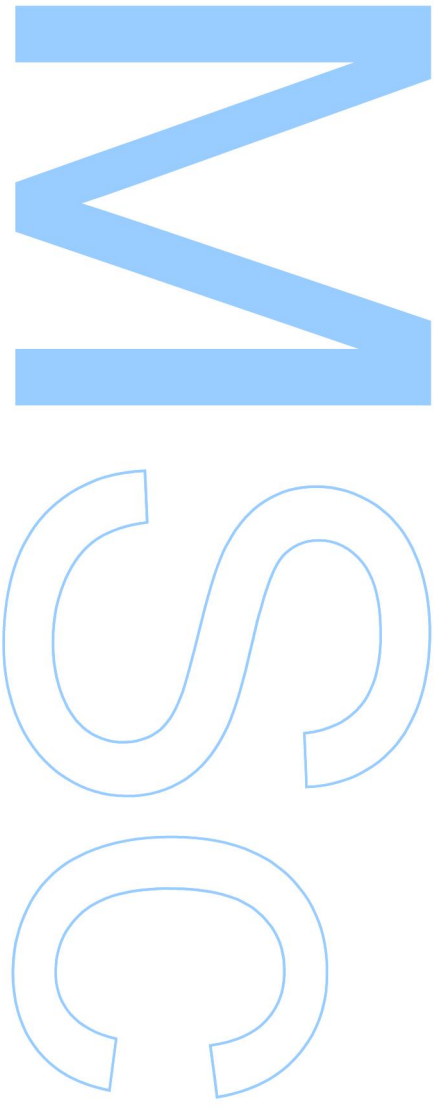
Deteção e Alerta de Vulnerabilidades e Exposições Comuns (CVE)

Luís Ricardo Pereira Passos
Mestrado em Segurança Informática
Departamento de Ciência de Computadores
2020

Orientador
Luís Filipe Coelho Antunes, Faculdade de Ciências da Universidade do Porto

Supervisora
Ana Cristina de Melo Carvalho, TekPrivacy



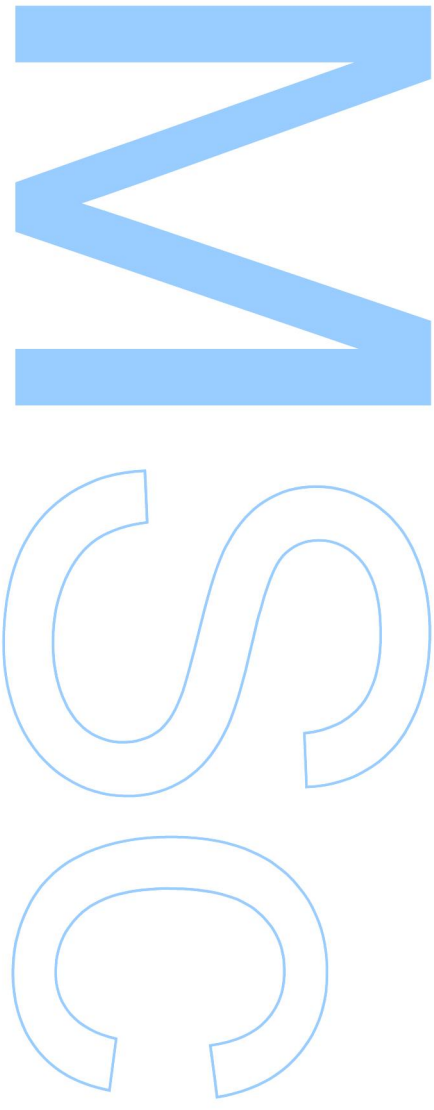
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____



Agradecimentos

Eu gostaria de agradecer aos meus orientadores de tese, Professor Luís Antunes e Ana Carvalho por toda a orientação, ajuda e suporte que permitiram que todos os objetivos fossem cumpridos e pelos quais estou muito agradecido.

Estou também muito grato à minha família, em especial aos meus pais, por todo o suporte e motivação.

Abstract

As a consequence of the growing number of computer systems, it is necessary to be updated regarding new vulnerabilities in order to ensure that the system is not accessed in an unauthorized way where private information, including personal data can be compromised. These vulnerabilities introduce different levels of risk for an organization, depending on the existence of exploits and ease of use and can compromise the confidentiality, integrity or availability of the system or the entire organization.

New vulnerabilities are usually made available through public databases, and MITRE manages one of the largest public dictionaries called CVE (Common Vulnerabilities and Exposures) as basis for several scanners allowing different organizations to develop, prioritize and resolve patches for critical vulnerabilities.

The developed platform proposes an approach that allows a quick discovery and resolution of vulnerabilities, comparing all users' systems with vulnerabilities made available by MITRE. The platform allows the user to configure several systems and send alerts based on the entries added or updated in the NVD list. Fractioning the project into several modules allows easy integration with future implementations, new alert systems and technologies such as Redis that are also used to ensure that the project is scalable and performance-oriented.

Vulnerability detection is dependent on the systems known to the platform, which are typically systems that were previously vulnerable. New systems will be added as new vulnerabilities are discovered, with the user being able to choose from an established list of manufacturers and products.

The results obtained from this platform allowed the detection of vulnerabilities that were to be expected as well the execution time of the scripts to obtain them.

Keywords: Vulnerability Management; Data Protection; System Security; Vulnerability Analysis; SCAP Components

Resumo

Como consequência do crescente número de sistemas informáticos é necessário estar atualizado quanto a novas vulnerabilidades, garantindo que o sistema não é acedido de forma não autorizada onde informações privadas, incluindo dados pessoais, são comprometidas. Estas vulnerabilidades apresentam diferentes níveis de risco para uma organização, dependendo da existência de *exploits* e da facilidade de utilização e podem comprometer a confidencialidade, integridade ou disponibilidade do sistema ou de toda a organização.

Novas vulnerabilidades são normalmente disponibilizadas através de bases de dados públicas, sendo que a MITRE gere uma das maiores denominada CVE (*Common Vulnerabilities and Exposures*), sendo a base para diversos scanners e que permitem diversas organizações desenvolver,prioritizar e resolver patches para vulnerabilidades criticas.

A plataforma desenvolvida propõem uma abordagem que possibilita uma rápida descoberta e resolução de vulnerabilidades, comparando todos os sistemas de um utilizador com vulnerabilidades disponibilizadas pela MITRE. A plataforma extrai todas as entradas adicionadas ou atualizadas na lista NVD e permite o envio de alertas para diversos sistemas configurados pelo utilizador. A separação do projeto em diversos módulos permite uma fácil integração com futuras implementações, novos sistemas de alertas e são ainda utilizadas tecnologias como Redis que permitem garantir que o projeto é escalável e desempenho.

A deteção de vulnerabilidades está dependente dos sistemas conhecidos pela plataforma, sendo estes tipicamente sistemas anteriormente vulneráveis. Novos sistemas serão adicionados conforme novas vulnerabilidades são descobertas, sendo que o utilizador poderá escolher entre uma lista de fabricantes e produtos estabelecida.

Os resultados obtidos pela plataforma permitiram a deteção das vulnerabilidades esperadas, bem como o tempo de execução dos *scripts* para as mesmas.

Palavras chave: Gestão de Vulnerabilidades; Proteção de Dados; Segurança de Sistemas; Análise de Vulnerabilidades; Componentes SCAP

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Blocos de Código

Acrónimos

CVE Common Vulnerabilities and Exposures

JSON JavaScript Object Notation

NVD National Vulnerability Database

CVSS Common Vulnerability Scoring System

CPE Common Platform Enumeration

WFN Well Formed CPE Naming

CWE Common Weakness Enumeration

SW Software

HW Hardware

UDP User Datagram Protocol

CNA CVE Numbering Authority

VMS Vulnerability Managment Systems

USA United States of America

SCAP Security Content Automation Protocol

NIST National Institute of Standards and Technology

XML Extensible Markup Language

OVAL Open Vulnerability and Assessment Language

XCCDF Extensible Configuration Checklist Description Format

OCIL Open Checklist Interactive Language

ARF Asset Reporting Format

CCSS Common Configuration Scoring System

TMSAD Trust Model for Security Automation Data

AID Asset Identification

SMS Short Message Service

PHP PHP: Hypertext Preprocessor

API Application Programming Interface

SQL Structured Query Language

SHA Secure Hash Algorithm

MB Megabyte

CPU Central Process Unit

RAM Random-access Memory

Capítulo 1

Introdução

Todos os dias são identificadas e disponibilizadas novas vulnerabilidades que podem comprometer todo o sistema informático de uma organização, sendo importante uma rápida intervenção para a resolução desta vulnerabilidade. O aumento de utilizadores e serviços ligados à internet faz com que estas vulnerabilidades sejam cada vez mais comuns, sendo necessária a utilização de um sistema de gestão de vulnerabilidades para alertar e atualizar quaisquer serviços que se encontrem vulneráveis [20].

Estas falhas surgem tipicamente de erros no desenvolvimento de software e são oportunidades para qualquer utilizador mal intencionado obter acesso não autorizado a redes e dados privados de utilizadores, sendo procurados emails, passwords e até cartões de crédito. Todas as falhas de segurança informática conhecidas encontram-se documentadas e identificadas numa lista pública (CVE) lançada pela MITRE em 1999 com o objetivo de desenvolver uma catalogação que pudesse ser utilizada pelas organizações para melhorar a segurança das suas aplicações, sistemas operativos e *hardware*, providenciando uma forma de avaliar o nível de segurança das ferramentas da organização e, desta forma, garantir uma estrutura uniforme para que vários serviços possam comunicar entre si numa linguagem comum [19].

Segundo a MITRE, para que uma falha seja documentada na lista de CVE, esta deve ser um erro em código corrigível de forma independente e que permita ao atacante um acesso direto para um sistema ou rede. Apesar de um registo CVE conter informações muito limitadas sobre a vulnerabilidade ou exposição comum, estão ainda disponíveis bases de dados que permitem catalogar todos as vulnerabilidades e incluir dados sobre produtos e versões afetadas, bem como possíveis resoluções, riscos associados e vectores de ataque. Estas bases de dados ajudam a manter a confidencialidade dos sistemas e dados, validando a sua proteção, sendo um aspeto importante das politicas de segurança a

adoptar [22].

1.1 Motivação

Com a aplicação do Regulamento Geral sobre a Proteção de Dados (RGPD), e consequentemente a responsabilização das instituições nas medidas e nas atividades realizadas, a proteção de dados passou a ter uma componente importante de gestão de risco.

Como parte integrante da gestão de risco de proteção de dados é necessário analisar e monitorizar os ativos (papel/digital) que suportam os dados pessoais.

No caso específico dos ativos em suporte digital a análise deve ser constante devido a vulnerabilidades que possam surgir. Dado que o número de incidentes de cibersegurança tem aumentado de ano para ano resultando em várias organizações reconhecidas a perder dados de utilizadores, muito frequentemente devido a vulnerabilidades já conhecidas mas que não foram resolvidas por falta de um sistema de alerta eficiente. Um ataque que ocorreu no dia 3 de Maio de 2020 à plataforma de blogging Ghost.org e afetou 750.000 utilizadores teve origem numa falha já conhecida e avaliada como vulnerabilidade crítica mas que não foi prontamente resolvida [30].

Assim, numa análise de risco de proteção de dados deve constar a segurança dos sistemas de dados que dão suporte ao tratamento de dados pessoais. Essas avaliações aos sistemas de informação são dinâmicas, aumentando o risco quando são encontradas novas vulnerabilidades ao sistema de informação, ou à plataforma que suporta o sistema de informação, por forma a obter um rápido alerta para qualquer falha encontrada em qualquer activo disponível numa determinada organização.

1.2 Solução Proposta

Como resultado de um aumento de tecnologias e produtos utilizados pelas organizações é agora difícil estar atualizado quanto a novas vulnerabilidades identificadas ou atualizadas, comprometendo todos os dados, sistemas e aplicações e podendo causar impacto negativo em toda a organização e utilizadores, caso esta venha a ser alvo de um ataque direccionado ao sistema comprometido.

Esta solução é um sistema para gestão de vulnerabilidades e risco associado que terá como base a lista de vulnerabilidades disponibilizada pelo NVD, permitindo manter sempre informadas as organizações de novas falhas de segurança, bem como atualizações que possam acontecer em

todos os seus sistemas, sendo possível um tempo de resposta significativamente mais rápido. A plataforma permite listar todas as diferentes aplicações, sistemas operativos e hardware de acordo com fabricante, produto e versão para os quais pretende ser informado de novos CVEs. Quando um novo CVE é identificado ou atualizado, um novo relatório é disponibilizado com todos os detalhes do mesmo através de um portal onde várias ações para o seu tratamento, prioritização e resolução estão disponíveis. É disponibilizado ainda um sistema de alertas configurável que permite informar em tempo real e através de diversos meios, a existência de um novo CVE aplicável aos sistemas identificados.

A plataforma é facilmente extendida com novas funcionalidades, sendo dividida atualmente em três principais componentes que compreendem o processamento e interpretação de dados provenientes de ficheiros JSON, um portal para utilizadores e finalmente um sistema de atualização e alerta de utilizadores.

1.3 Contribuições no mundo real

O tema da presente tese foi proposto em parceria com a TekPrivacy, empresa que centra a atividade no desenvolvimento de soluções tecnológicas de apoio à atividade do EPD.

Por forma a que os Encarregados de Proteção de Dados, utilizadores da plataforma, maioritariamente juristas, tenham uma maior perceção do risco a que os dados tratados na sua organização estão sujeitos devido aos riscos de sistemas de informação, e conseguir intervir no momento.

Deste modo, o resultado da solução proposta será posteriormente implementado na plataforma TekPrivacy como uma nova funcionalidade.

1.4 Estrutura da tese

Esta tese está estruturada da seguinte forma:

- No capitulo 2 é apresentado um estudo da arte atualmente existente;
- No capítulo 3 são detalhadas as principais tecnologias utilizadas no desenvolvimento do projeto;
- O capitulo 4 detalha o desenvolvimento do projeto e todos os seus componentes;

- No capitulo 5 são detalhadas as experiências e testes efetuados nos diversos componentes da plataforma desenvolvida, bem como os resultados obtidos;
- O capitulo 6 apresenta a conclusão e trabalho futuro que poderá ser realizado por forma a melhorar a plataforma.

Capítulo 2

Estado da Arte

2.1 Vulnerabilidades

É importante que todas as organizações mantenham um registo de todos os seus ativos e conheçam o risco associado a uma vulnerabilidade. Algumas ferramentas tentam procurar vulnerabilidades e notificar a organização assim que estas sejam encontradas, para que possam ser o mais rapidamente abordadas. Esta abordagem poderá passar pela resolução da falha com uma atualização de software, mitigação até que seja encontrada uma solução, ignorar a falha dependendo da severidade ou remover o sistema afetado dos ativos da empresa.

As vulnerabilidades podem ser anunciadas pelo proprietário de uma determinada tecnologia ou por investigadores independentes. Estas vulnerabilidades podem ser publicadas imediatamente e sem uma resolução, após a resolução estar disponível ou após algum tempo sem resolução aparente.

Estes casos podem levar a que pessoas mal intencionadas procurem desenvolver ferramentas de ataque automatizado antes da resolução estar disponível para o público [29].

2.1.1 Falsos Positivos / Falsos Negativos

Devido às diversas formas e termos para representar uma falha, é comum softwares de gestão de vulnerabilidades apresentarem resultados distintos muitas vezes também por falha na documentação para a apresentação de certas configurações.

Estas características podem levar à identificação de falsos positivos ou seja vulnerabilidades que

não apresentam quaisquer riscos para o ativo em questão, sendo importante desenvolver mecanismos para a diminuição destas ocorrências, uma vez que podem originar uma investigação em problemas inexistentes. Alguns softwares de gestão de vulnerabilidades oferecem a possibilidade de excluir vulnerabilidades que não apresentam risco ou cuja resolução ainda não foi encontrada pelo proprietário.

O pior caso é a existência de falsos negativos, onde um software falha em identificar uma vulnerabilidade, tipicamente devido à falta de informação. Apesar deste problema ser cada vez menos comum, várias organizações dependem de várias soluções para identificar falhas de segurança [18].

2.1.2 Remediação e Mitigação

Quando uma falha é apresentada sob a forma de CVE esta pode ser uma falha de configuração ou uma falha de código, e dependendo do nível de severidade da falha ela pode ser mitigada ou remediada. Mitigação é tipicamente a alteração de uma configuração através de um *patch* que procura corrigir e mitigar o risco associado. Apesar de tipicamente as falhas de configuração serem rápidas de resolver, podem também atribuir privilégios e acesso a sistemas de forma muito simples para qualquer utilizador mal intencionado. Tipicamente, para mitigar uma falha, o procedimento é mais complexo e exige que o *software* ou *firmware* seja atualizado para que a vulnerabilidade possa ser resolvida, sendo tipicamente um problema no código [16].

2.1.3 *Exploits*

Caso uma vulnerabilidade não seja rapidamente resolvida, um atacante com conhecimento da existência do sistema vulnerável pode recorrer ao uso de um *exploit* em que poderá tentar obter vantagem do sistema com a finalidade de obter dados sobre os utilizadores ou obter privilégios sobre todo o sistema. [14]

Os *exploits* podem ser utilizados para a instalação de *malware*, *trojans* ou outro sistema que permita o atacante manter-se indetetável, sendo para isso útil um sistema de gestão de vulnerabilidades de forma a minimizar o risco [16].

- *Zero Day Vulnerabilities* - Vulnerabilidades para as quais ainda não é conhecida uma resolução pública, sendo que *exploits* para estas vulnerabilidades são utilizados por criminosos na tentativa de acesso indevido aos sistemas. [12]

2.2 Gestão de Vulnerabilidades

A gestão de vulnerabilidades é o processo que permite estar atento a novas ameaças de forma e definir um plano de ação para que novas vulnerabilidades sejam resolvidas de forma rápida e eficiente. Este processo tipicamente consiste em identificar, classificar, prioritizar ,resolver e reavaliar o risco de uma determinada vulnerabilidade, resultando na correção da falha encontrada e gestão do risco associado.

Este processo é conseguido com a ajuda de sistemas de gestão de vulnerabilidades e *scanners* de vulnerabilidades, sendo importante para prevenir atacantes de entrar na rede de uma empresa e consequente roubo de informações.

Para que este processo seja eficaz, é necessário definir um conjunto de ações sequenciais de forma a definir quais os ativos que devem ser protegidos, o seu nível de risco de acordo com o impacto que possam ter na organização e para cada ativo deve ser listada a funcionalidade, quais as portas abertas e ainda processos e serviços que possam ter impacto ou que não devam ser utilizados neste ativo.

Deve ainda ser definida uma estrutura com técnicas de mitigação baseadas no risco associado a cada ativo e gerado um relatório que será atualizado sempre que falhas em sistemas sejam encontradas ou resolvidas.

2.2.1 Avaliação de Vulnerabilidades

Avaliação de Vulnerabilidades é o processo que permite identificar, classificar e prioritizar os diferentes ativos da organização, sendo normalmente conseguido com as seguintes fases:

- **Verificação inicial** – A primeira fase para um processo de gestão de vulnerabilidades deve ser a fase de preparação onde devem ser definidos os ativos a proteger e o valor que o ativo tem para a organização, bem como o impacto no negócio e, de acordo com os dados obtidos deve ser definida uma lista de práticas para mitigar o risco encontrado. Nesta fase deve ser ainda definido quem tem acesso a um determinado ativo.
- ***Scanning* de vulnerabilidades** – Nesta fase deve ser utilizado um sistema para verificar potenciais vulnerabilidades de acordo com os ativos e o risco encontrado na verificação inicial. Este processo pode ser configurado e agendado de acordo com os requisitos da equipa, tendo em atenção a importância do ativo e a frequência com que um proprietário do *software* lança novas actualizações.

- **Geração de relatório** – Finalmente deve ser gerado um relatório contendo todas as vulnerabilidades encontradas, sistemas afetados, data, descrição detalhada, risco baseado em CVE, o responsável pela falha encontrada e se possível, uma prova de conceito de que a vulnerabilidade foi efetivamente corrigida ou contramedidas caso a vulnerabilidade ainda se encontre ativa. [19]

2.3 *Scanners* de Vulnerabilidades

Um *scanner* de vulnerabilidades tem diversas tarefas que ajudam a encontrar vulnerabilidades consistindo tipicamente nas seguintes tarefas:

- Procurar ativos na rede ou verificar se os ativos detalhados pela organização estão acessíveis;
- Identificar possíveis serviços ou portas abertas;
- Obter informação sobre os sistemas e tentar autenticação se possível;
- Procurar vulnerabilidades conhecidas de acordo com a informação recolhida.

2.3.1 BurpSuite

BurpSuite é um software desenvolvido pela PortSwagger e subdividido em diversas ferramentas que permitem explorar e testar todo um sistema desde o mapeamento dos seus ativos até análise de aplicações e sistemas, procurando encontrar e simular ataques de acordo com vulnerabilidades conhecidas. Com o BurpSuite é possível automatizar tarefas permitindo ainda efetuar ações de forma manual para um maior controlo sobre o ambiente. [1]

2.4 Sistemas de Gestão de Vulnerabilidades

Um sistema de Gestão de Vulnerabilidades (VMS) é um sistema desenvolvido com a funcionalidade de gerir vulnerabilidades, identificando atualizações em falta e configurações inseguras nos diversos sistemas operativos, aplicações ou qualquer outro ativo garantindo que todos os problemas são resolvidos rapidamente [16].

Um VMS deve ser escolhido com base em diversas capacidades como a capacidade de cobrir o máximo de ambientes e ativos, suporte por parte da equipa técnica para ajuda em caso de

incidentes, capacidades de rapidamente reportar uma *Zero Day vulnerability* com relatórios detalhados e customizáveis pelo cliente e capacidade de apresentar o mínimo de falsos positivos possíveis [23].

2.4.1 Snyk

Snyk [8] é uma empresa baseada em Londres e estabelecida em 2015 com a finalidade de ajudar a desenvolver software de forma mais rápida e manter o código seguro encontrando, resolvendo, alertando e prevenindo vulnerabilidades nas dependências de projetos em aplicações desenvolvidas em diversas linguagens e utilizando diversos gestores de pacotes. Além de manter uma monitorização constante sobre as dependências do projeto através de ferramentas como GitHub e Bitbucket, ainda permite apresentar relatórios detalhados com soluções e um painel onde é possível gerir todo o projeto, bem como um suporte dedicado.

A ferramenta pode ser utilizada por linha de comandos ou incluída no sistema de integração/desenvolvimento continuo, facilitando o desenvolvimento seguro de aplicações.

O Snyk apenas tem como finalidade o alerta de vulnerabilidades em dependências de código fonte, não sendo atualmente utilizado para sistemas operativos, aplicações ou hardware. Esta plataforma não requer qualquer instalação ou atualização.

2.4.2 Saucs

A Saucs [7] é uma plataforma que pretende alertar sobre novas vulnerabilidades através de filtros que poderão ser adicionados de acordo com proprietário ou *software* utilizado e apresentar novas vulnerabilidades de acordo com os últimos CVE publicados ou atualizados. Esta plataforma recorre a standards como CWE, CPE, CVSS e CVE para a apresentação de resultados e utiliza emails customizados para a divulgação de novos CVE para as plataformas que o utilizador está subscrito.

Contrariamente ao Snyk esta plataforma procura erros em qualquer sistema ou aplicação mas não detalha a versão do sistema em questão, resultando em diversos falsos positivos. Além da plataforma possuir detalhes como histórico de alterações ela não parece estar atualizada, resultando numa falha para as organizações que dependam deste sistema.

2.4.3 NucleusSec

Nucleus [5] procura ser uma ferramenta com a finalidade de efetuar o *scan* de vulnerabilidades, prioritizar decisões através do risco, automatizar a resolução de vulnerabilidades e apresentar um relatório para as organizações indicando o estado atual dos diferentes ativos identificados. Inicialmente o Nucleus procura agregar e prioritizar todas as possíveis fontes de vulnerabilidades na organização, sendo que poderá ser configurado um relatório para alertas em vários sistemas de suporte que permitem enviar a informação para os diversos utilizadores no formato mais rápido e automatizado possível, garantindo uma resposta 10x superior em relação a outros sistemas.

A ferramenta permite utilizar mais de 50 *scanners* como Nessus, Qualys, OWASP Dep Checker ou Snyk para obter dados mais confiáveis de todos os ativos da organização desde aplicações até redes e sistemas operativos, podendo apresentar os dados por ativos mais vulneráveis.

2.4.4 Tenable

Tenable é uma organização fundada em 2002 com o desenvolvimento do software de segurança Nessus, sendo hoje o software de resolução de vulnerabilidades mais utilizado. Este software é composto por *scanners* apresentando resultados das avaliações de vulnerabilidades em tempo real e possibilita que o utilizador configure a forma como reportes são apresentados. Um *scan* efetuado com a ferramenta Nesus poderá ser realizado através de *scanning* tradicional em que um *scanner* procura correr em vários ativos na rede de uma organização ou para resultados mais rápido poderão ser utilizados agentes Nessus que correm em cada máquina de forma individual e reportam os dados para o Nessus Manager [28].

2.4.5 CVE Details

CVE Details [2] é uma base de dados que contem todas as vulnerabilidades detalhadas e especificadas por fabricante, produto e versão, sendo utilizados dados enviados pelos *feeds* XML da NVD, bem como dados de *exploits*, confirmações de fabricantes e módulos Metasploit que possam ser publicados por fontes externas. Apesar da plataforma oferecer um conjunto de dados que permitem verificar se os ativos de uma organização estão seguros, não oferece qualquer sistema para gestão da vulnerabilidade ou mecanismo de alerta.

Capítulo 3

Componentes e Métricas

Todos os CVE apresentados na solução proposta para gestão de vulnerabilidades são especificados pela MITRE e detalhados através da base de dados de vulnerabilidades NVD que se encontra totalmente sincronizada, sendo que todos os *updates* aparecem imediatamente na lista de CVE e NVD. A base de dados NVD adiciona informação ao CVE como: resolução dos problemas, sistemas afetados detalhados por proprietário e versão, pontuação de severidade com base na escala CVSS e possível impacto. Estes dados serão utilizados na ferramenta de gestão de vulnerabilidades juntamente com a lista CPE, permitindo obter uma solução confiável para o utilizador.
Neste capítulo pretende - se descrever todos os componentes associados a um CVE.

3.1 *Common Vulnerabilities and Exposures*

Common Vulnerabilities and Exposures (CVE) é um dicionário público que contem todas as vulnerabilidades de software, sistemas operativos e hardware, desenvolvido pela organização sem fins lucrativos, MITRE, em 1999, com o objetivo de desenvolver um *standard* para partilhar dados entre diferentes aplicações e facilitar a descoberta e resolução de novas falhas. [22]

Uma entrada CVE é apresentada com um número de identificação tipicamente detalhado com o ano da ocorrência e um número sequencial atribuído por um CVE Numbering Authority (CNA), sendo este na maior parte dos casos o vendedor do produto em questão. Além deste dado, o CVE ainda possui uma descrição, sendo posteriormente detalhada através da lista NVD. Estes dados são utilizados em bases de dados, ferramentas de gestão de vulnerabilidades e *firewalls* para estabelecer a ligação com possíveis vulnerabilidades. [18]

Para reportar uma nova vulnerabilidade é necessário localizar o CNA responsável pelo sistema ou contactar diretamente a MITRE caso não seja encontrado um CNA diretamente relacionado com o sistema vulnerável. Este contacto terá um formulário para preencher com detalhes necessários acerca da vulnerabilidade e tem a finalidade de obter um novo CVE ID. O CNA irá após verificação da vulnerabilidade, detalhar todas as versões do software afetadas, atribuir uma pontuação com base na *framework* CVSS e apresentar a resolução da mesma. [21]

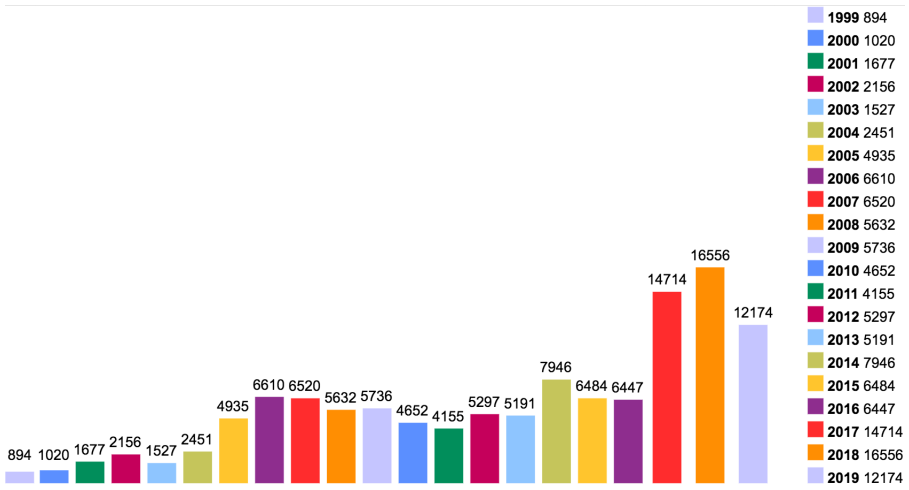


Figura 3.1: Número de CVE por ano

3.2 *National Vulnerability Database*

A *National Vulnerability Database* (NVD) é uma base de dados de vulnerabilidades totalmente sincronizada com a lista CVE e lançada em 2005, sendo um produto da divisão de segurança de computadores NIST e gerida pelo governo dos EUA representando os seus dados de acordo com o protocolo SCAP (*Security Content Automation Protocol*). Esta base de dados utiliza a especificação SCAP e procura detalhar todas as vulnerabilidades adicionando métricas de impacto, *exploits*, soluções, referências úteis e identificação de sistemas afetados pela vulnerabilidade. [11]

Os dados de um CVE presente na base de dados NVD apresentada a seguinte estrutura:

- CVE - Neste campo são detalhados dados detalhando o formato do CVE, ID do CVE, descrição e referências contendo *tags* detalhando o tipo de artigo
- Configurações - Detalhes de todas as configurações e sistemas vulneráveis que podem depois ser associados com a lista CPE para obter uma lista mais detalhada

- Impacto - Avaliação do impacto de acordo com a *framework* CVSS e tipicamente detalhado na versão 2 e 3.1 da *framework*
- Datas de publicação e modificação

3.3 SCAP

O *Security Content Automation Protocol* (SCAP) é uma especificação criada pelo governo dos USA e descrita na publicação especial 800-117 da NIST com o objetivo de estabelecer um standard para a manutenção e interoperabilidade entre os diversos produtos de segurança através da definição de 11 componentes e convenções. Estes componentes têm como origem *frameworks* como CVSS, OVAL, CVE e CPE, e são utilizados pela base de dados NVD. [27]

Atualmente, este protocolo é extensamente utilizado por *software* e *hardware* de segurança, por forma a monitorizar, corrigir e documentar possíveis vulnerabilidades, sendo utilizado no sistema de gestão de vulnerabilidades desenvolvido. [20]

3.4 OVAL

A *Open Vulnerability and Assessment Language* é um *standard* baseado na especificação SCAP com a finalidade de estabelecer um meio comum de transferência de informação entre diversos sistemas de segurança da informação, permitindo descrever vulnerabilidades ou configurações testáveis de sistemas vulneráveis. Esta linguagem consiste em três importantes componentes [26]:

- Oval Definitions - Modelo que define como a *framework* deve apresentar os dados
- Oval System Characteristics - Define como devem ser representados os dados obtidos do sistema alvo
- Oval Results - Modelo para definir como os resultados devem ser apresentados

A OVAL é definida através de XML e desenvolvida por uma ativa comunidade e organizações governamentais, sendo atualmente gerida pelo *OVAL Board*. Esta linguagem define uma forma simples de determinar se existe uma vulnerabilidade, problema de configuração ou *patch* para um determinado sistema e apresenta todos os detalhes específicos para um determinado problema, garantindo que qualquer dependência vulnerável é devidamente especificada [13].

3.5 XCCDF

O *Extensible Configuration Checklist Description Format* (XCCDF) é a linguagem utilizada para descrever a lista de referências de segurança e utilizada no *standard* OVAL para definir um modelo de dados que permita a troca de informação e registar os resultados de uma avaliação num formato padrão.

3.6 *Common Vulnerability Scoring System*

A *Common Vulnerability Scoring System (CVSS)* é uma framework desenvolvida pela FIRST.ORG que permite avaliar as características e severidade de uma falha de segurança, utilizando para isso três grupos de métricas: *base*, *temporal* e *environmental*. Os resultados destas métricas são apresentadas na base de dados NVD, sendo a sua pontuação atribuída de 0 a 10 e permitindo avaliar o produto vulnerável por forma a determinar o impacto em diversos vectores de ataque, bem como impacto na confidencialidade, integridade e disponibilidade do sistema. A métrica temporal ajusta a pontuação base de acordo com a disponibilidade de possíveis *exploits* e métricas ambientais ajustando a pontuação resultante conforme a especificidade do ambiente e possíveis mitigações que podem existir.

Cada uma destas métricas é composta por outras métricas que procuram estabelecer uma pontuação confiável de acordo com a versão de especificação através da atribuição de diversos valores métricos que serão traduzidos em valores numéricos e de seguida definida uma pontuação final com base numa fórmula definida. Todas estas métricas e valores serão apresentadas no ficheiro JSON enviado pela NVD que atualmente apresenta os valores CVSS para a versão 2 e 3.1 da framework [17].

CVSS 3.1

Na versão 3.1 do CVSS é possível definir a métrica Exploitability de acordo com critérios que podem variar entre a existência de um *exploit* disponível, existência de um *exploit* em prova de conceito, possibilidade de ativar a vulnerabilidade sem necessidade de exploit ou inexistência de *exploit*. O CVSS 3.1 permite definir um score decorrente dos três grupos de métricas composto por diversas submétricas de forma qualitativa de acordo com a seguinte tabela:

Tabela 3.1: Métricas e Grupos CVSS

Base Score	*Temporal Score*	*Environmental Score*
Attack Vector (AV)	*Exploit Code Maturity (E)*	*Confidentiality Requirement (CR)*
Attack Complexity (AC)	*Remediation Level (RL)*	*Integrity Requirement (IR)*
Privileges Required (PR)	*Report Confidence (RC)*	*Availability Requirement (AR)*
User Interaction (UI)		*Modified Attack Vector (MAV)*
Scope (S)		*Modified Attack Complexity (MAC)*
Confidentiality (C)		*Modified Privileges Required (MPR)*
Integrity (I)		*Modified User Interaction (MUI)*
Availability (A)		*Modified Scope (MS)*
		Modified Confidentiality (MC)
		Modified Integrity (MI)
		Modified Availability (MA)

A FIRST.ORG permite ainda definir de forma mais fácil o impacto do CVSS e a atribuição das pontuações através de uma calculadora disponível no seu website, onde através de um questionário se consegue obter uma pontuação para cada os grupos de métricas *base*, *temporal* e *environmental*. O resultado final da avaliação irá resultar num vetor que contêm para cada métrica e submétrica o valor atribuido [17].

```
CVSS:3.0/AV:A/AC:H/PR:H/UI:R/S:C/C:L/I:N/A:L/E:U/RL:T/RC:R/CR:M/IR:M/MAV:A/
MPR:N/MUI:N/MS:U/MC:N/MI:N/MA:H
```

Bloco de Código 3.1: CVSS Vector String

3.7 *Common Platform Enumeration*

Common Platform Enumeration (CPE) é um método padrão para descrever e identificar classes de aplicações, sistemas operativos ou *hardware* de um CVE, podendo ser dividido em diversas especificações modulares com diferentes funções e podendo ser utilizado como fonte de informação para verificar que todos os ativos estão devidamente protegidos e respeitam as politicas de segurança

da empresa. As várias especificações trabalham em conjunto para formar o *CPE Naming*, sendo este processo de formação denominado *Well Formed CPE Naming* (WFN). Para que um WFN seja corretamente formado, as seguintes partes devem estar definidas:

- *Part* - Apenas aceita os valores 'a' para Aplicação, 'o' para sistemas operativos e 'h' para dispositivos *hardware*
- *Vendor* - O valor deste campo deve identificar a pessoa ou organização responsável pelo desenvolvimento da aplicação
- *Product* - Este campo deve identificar o produto pelo seu titulo ou nome
- *Version* - Este campo deve identificar o produto pelo conjunto de caracteres alfanuméricos de uma *release* particular
- *Update* - Este valor deve conter o conjunto de caracteres alfanuméricos que identifiquem uma atualização, *service pack* ou outro em particular
- *Edition* - Valor não utilizado na última versão da especificação CPE mas se utilizado deverá traduzir qualquer termos relacionados com a versão
- *Language* - Linguagem utilizada na interface de utilizador através de *tags* e códigos de região
- *SW Edition* - Este campo deve especificar como o produto é atribuído ao utilizador final
- *target SW* - Especificação do tipo de ambiente onde o produto deve correr
- *target HW* - Especificação do tipo de arquitectura onde o produto deve correr
- *other* - Valor que deve traduzir qualquer informação geral ou descritiva do proprietário

O resultado final de um WFN deverá ser o seguinte [15]:

```
cpe:2.3:a:microsoft:internet_explorer:8.0.6001:beta:*:*:*:*:*:*
```

Bloco de Código 3.2: Especificação WFN

3.8 *CPE Match*

A especificação de um CPE nem sempre contem todas as versões, *updates* e edições afetadas por uma vulnerabilidade, sendo que poderá apresentar apenas a primeira versão afetada e a última versão. Para que seja possível determinar se uma versão está incluída nesta lista é necessário recorrer ao *CPE Match*, sendo um documento de dados em formato JSON enviados pela NVD e que permite através da especificação CPE obter todos os detalhes.

Um *CPE Match* pode ser representado através de blocos que procuram relacionar um CPE URI com um ou vários URI presentes no dicionário CPE ou através de um intervalo que específica a versão de inicial e final, garantindo que todas as versões entre ambas se encontram vulneráveis.

Esta base de dados poderá falhar para detalhar alguns CPE, sendo estes dependentes dos dados atuais no dicionário CPE [25].

3.9 *Common Weakness Enumeration*

A *Common Weakeness Enumeration (CWE)* pretende descrever as falhas, *bugs* e vulnerabilidades de um determinado produto de forma a que sejam facilmente entendidas e resolvidas. O CWE pretende ainda aplicar uma *framework* com o objetivo de identificar, mitigar e prevenir qualquer falha, sendo uma lista que qualquer organização de qualquer tamanho poderá consultar para verificar se os seus sistemas estão devidamente protegidos. Atualmente o CWE não integra o SCAP [24].

Tabela 3.2: Componentes SCAP

Componente	Descrição
CVE (Common Vulnerabilities and Exposures)	Catálogo com as vulnerabilidades conhecidas
CCE (Common Configuration Enumeration)	Lista de identificadores relaccionado com um problema de configuração no sistema
CPE (Common Platform Enumeration)	Framework para a definição de sistemas operativos, software ou hardware abrangidos pela falha de segurança
CVSS (Common Vulnerability Scoring System)	Framework para descrever quantitativamente o impacto da vulnerabilidade
XCCDF (eXtensible Configuration Checklist Description Format)	Lista de verificação da configuração em formato XML
OVAL (Open Vulnerability and Assessment Language)	Avaliação do estado de uma vulnerabilidade
OCIL (Open Checklist Interactive Language)	Linguagem para apresentar questões e interpretar respostas
AID (Asset Identification)	Descrição do objetivo de identificação de ativos e normas de utilização
ARF (Asset Reporting Format)	Modelo para transferência de dados de ativos
CCSS (Common Configuration Scoring System)	Métricas de severidade de problemas de configuração
TMSAD (Trust Model for Security Automation Data)	Modelo para aplicar no contexto da automatização de segurança

Capítulo 4

Arquitetura e Desenvolvimento

4.1 Arquitetura

O sistema desenvolvido encontra-se subdividido em diversos projetos que pretendem a extração e tratamento de dados mais recentes e actualizados, a sincronização de vulnerabilidades encontradas com os utilizadores e um sistema de *reporte* e alertas. Todos os dados são armazenados numa base de dados MySQL contendo os CVEs, CPEs, dados de utilizadores, tecnologias utilizadas por cada utilizador e o relatório de vulnerabilidades.

O processo de extração de dados tem como objetivo adicionar e actualizar todos os CVEs e CPEs enviados pela base de dados NVD presente no portal do NIST. A obtenção destes dados é executada através de dois processos configuráveis que após o tratamento de dados obtidos enviam os dados para a base de dados MySQL comum.

Os processos de sincronização e o sistema de reporte e alertas encontram-se relacionados, uma vez que após qualquer alteração no sistema os dados serão sincronizados e no caso de existência de alguma nova vulnerabilidade de um utilizador é activado o mecanismo de alerta e reporte. A sincronização dos dados pode ser executada de forma manual ou automática através de um *Cron Job*.

Esta arquitectura permite que o sistema seja fácil de manter e as tecnologias utilizadas nos diversos componentes tornam o sistema altamente escalável. Com a utilização de Laravel como *framework* que permite o acesso dos utilizadores ao relatório e sistema de alerta, bem como desenvolvimento do processos de atualização de vulnerabilidades é muito simples adicionar novas funcionalidades e alterar as funcionalidades existentes.

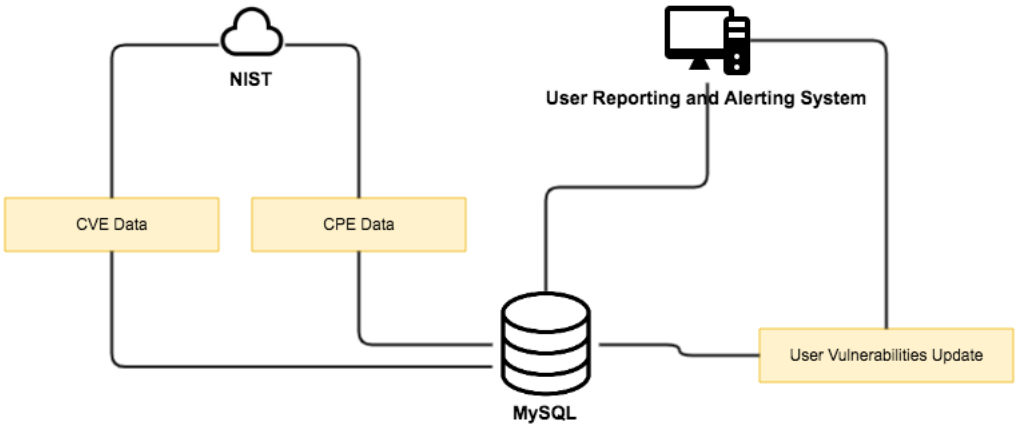


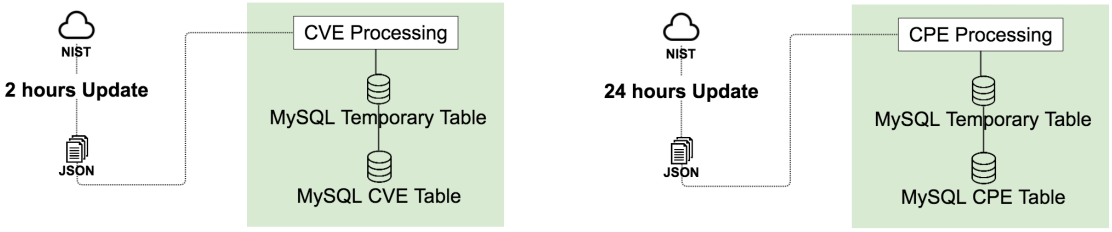
Figura 4.1: Arquitetura

4.1.1 Extração de Dados

A extração de dados recorre ao portal do NIST onde são disponibilizados diversos ficheiros JSON tipicamente organizados por ano e os mais recentes (dados dos últimos dias) para o caso de CVEs e apenas um ficheiro JSON com os dados de todos os CPEs existentes.

A base de dados NVD para CVEs é atualizada aproximadamente a cada duas horas e o ficheiro com dados mais recentes contem dados até 8 dias. O ficheiro com os dados de CPEs é atualizado uma vez por dia. Estes dados ajudam a determinar quão frequentemente o processo terá que correr.

Todos os dados extraídos são enviados para a base de dados onde são apenas seleccionados novos dados ou dados recentemente atualizados.



(a) Processamento de CVE (b) Processamento de CPE

Figura 4.2: Extração de Processamento de Dados

O download, extracção e processamento de todos os dados é realizado com recurso a dois processos Python que correm independentemente, correndo de acordo com o tempo definido num processo Cron.

4.1.2 Atualização de Vulnerabilidades de Utilizadores

O processo que permite atualizar as vulnerabilidades de cada utilizador é um processo que compara dados de sistemas do utilizador e verifica se novas ou atualizadas vulnerabilidades foram encontradas pelo sistema de extração de dados. Esta actualização ocorre na base de dados através de operações Join, sendo que a tarefa de actualização apenas tem como funcionalidade efetuar o pedido e receber os dados.

Este processo corre de acordo com um Cron previamente desenvolvido para funcionalidades Laravel e é processado através de uma tarefa Laravel que permite agendar um processamento através de uma entrada Cron e quais as tarefas a executar.

Após obtenção de todos os dados de utilizadores afetados por uma vulnerabilidade, estes são enviados para uma tabela MySQL com a finalidade de *reporting* e para uma fila Redis para envio de alertas.

O sistema encontra - se sempre sincronizado através de um *timestamp* que permite saber quando foi a última actualização e quais dados precisam ser atualizados.

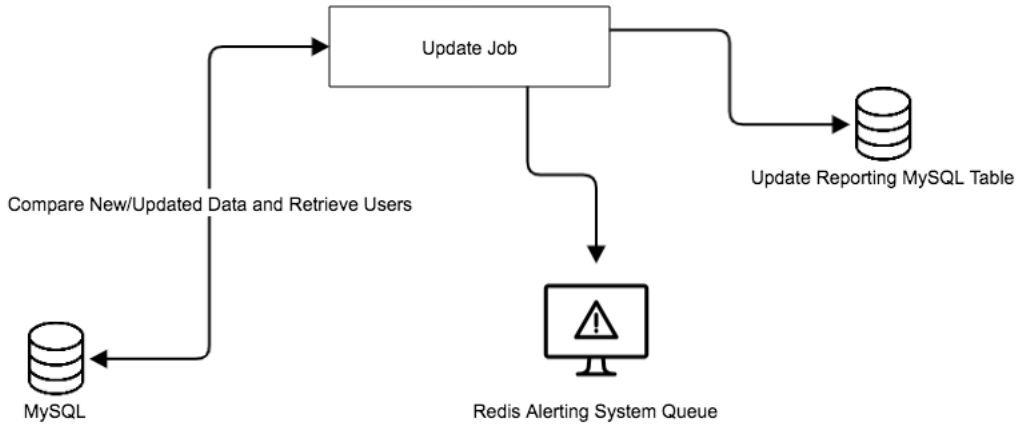


Figura 4.3: Arquitectura do Sistema de Alerta

4.1.3 Sistema de Alertas e Notificações

Após todos os dados referentes a uma actualização serem enviados para uma fila de notificações Redis, estes são consumidos através de um processo externo que se encontra permanentemente à espera de novos dados. Este processo tem como finalidade consumir os dados que vão sendo enviados para a fila com a finalidade de os processar e enviar notificações através de email, Telegram e SMS, de acordo com os registos do utilizador presentes na base de dados.

Os dados consumidos encontram - se em formato JSON e o processamento é efetuado através de um método Laravel que associa automaticamente o utilizador em questão com o dado a ser processado, facilitando a obtenção de todos os dados para envio das notificações.

A mensagem a enviar ao utilizador é estática e dependente do sistema para o qual é enviado, podendo no entanto ser personalizada de acordo com o utilizador. Todas as notificações são enviadas através de um módulo Laravel que pode ser extendido e com suporte para integrar com outros sistemas. É possível ainda definir quantos processos podem ser executados em paralelo para um maior controlo sobre a fila.

Para envio dos emails podem ser utilizados diversos *drivers* pré definidos na *framework* Laravel ou utilizar um SMTP que poderá ser configurado através de variáveis de ambiente.

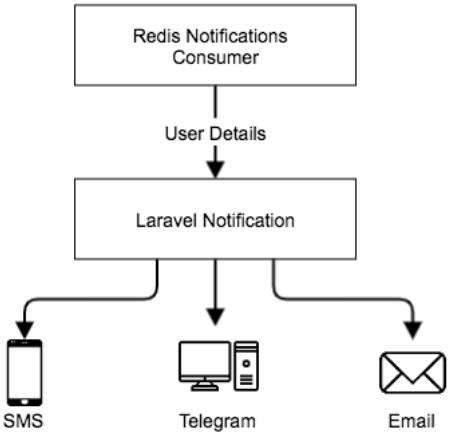


Figura 4.4: Processo de Notificação

4.2 Opções Tecnológicas para Desenvolvimento

As opções tecnológicas para desenvolvimento foram em parte derivadas de sistemas diretamente suportados por Laravel, para o caso de *updates* de vulnerabilidades e notificações e para o processamento dos dados foi utilizada a linguagem de Programação Python, uma vez que permitia uma fácil manipulação e extração de todos os dados necessários.

Laravel

Laravel [3] é uma *framework open-source* desenvolvida na linguagem de programação PHP com o objetivo de tornar a experiência de desenvolvimento mais agradável, proporcionando diversas funcionalidades já desenvolvidas, comuns entre a maior parte dos projetos, bem como integração com diversas bases de dados, servidores de email, entre outras ferramentas. A opção para utilizar Laravel no desenvolvimento tem como finalidade permitir uma fácil integração com o atual sistema TekPrivacy, bem como utilizar as diversas integrações que a framework possui para um desenvolvimento mais rápido e livre de erros.

MySQL

MySQL [4] é uma base de dados relacional lançada em 1995 que utiliza SQL como interface.

Esta linguagem é de fácil integração com PHP e Laravel, sendo o principal motivo para a sua

utilização neste projeto.

Redis

Redis [6] é uma estrutura de dados em memória de duração opcional tipicamente utilizada como base de dados, cache e *message broker*.

Além de uma fácil integração com Laravel, a utilização como *message broker* permite que o envio de todas as notificações possa ocorrer conforme disponibilidade do servidor, garantindo que uma falha não resulta no reinicio de todo o processo.

Twilio

Twilio [10] é uma plataforma de comunicação baseada na *cloud* que possui serviços que permitem facilitar a obtenção de um número e programaticamente enviar e receber chamadas e SMS, bem como aceder a diversas APIs que permitem outras formas de comunicação.

Neste projeto foi utilizado Twilio para envio de alertas de novas ou atualizadas vulnerabilidades automaticamente por SMS.

Telegram

Telegram [9] é um serviço de envio de mensagens instantâneas baseado na *cloud*, onde os utilizadores podem trocar mensagens, fotos ou vídeos de forma totalmente segura assegurando a sua privacidade.

Esta plataforma é utilizada como uma das opções de envio de alertas.

4.3 Desenvolvimento

A extração de dados utiliza Python3 e é realizada em dois processos distintos que permitem sincronizar novos e atualizados CVEs e CPEs com os dados existentes na base de dados.

4.3.1 Sincronização CVE

Neste processo são extraídos dados de todos os CVEs diretamente do portal NDIST podendo estes ser recentes, actualizados ou de anos anteriores. Este processo aceita como parâmetro o inputfile (-i), sendo este o URL de acesso ao ficheiro em formato ZIP.

```
python3 sync.py -i
    https://nvd.nist.gov/feeds/json/cve/1.1/nvdcve-1.1-modified.json.zip
```

Bloco de Código 4.1: Executar Sincronização de CVE

```
    operatorArray = json['configurations']['nodes']
    for operator in operatorArray:
        if operator['operator'] == 'OR':
        ...
                for children in operatorChildren:
                    cpe_match_array = children['cpe_match']
                    for cpeMatch in cpe_match_array:
                        if cpeMatch['vulnerable'] == True:
                                cpe23Uri_set.add(cpeMatch['cpe23Uri'])
                                cpeMatchList.append(cveDictionary)
        elif operator['operator'] == 'AND':
        ...
                for children in operatorChildren:
                    cpe_match_array = children['cpe_match']
                    for cpeMatch in cpe_match_array:
                        if cpeMatch['vulnerable'] == True:
                                cpe23Uri_set.add(cpeMatch['cpe23Uri'])
                                cpeMatchList.append(cveDictionary)
```

Bloco de Código 4.2: Operadores CVE

Após iniciar o processo, este irá efetuar o *download* do ficheiro ZIP para uma nova pasta, sendo eliminado um ficheiro anterior caso esteja disponível. Concluído o *download* do ficheiro este será descompactado e o processamento de todos os documentos JSON presentes é iniciado.

Todos os dados JSON são enviados para duas tabelas temporárias MySQL, sendo uma para a inserção de dados do CVE e outra de dados CPE. Os dados inseridos no CPE incluem a data de publicação, última modificação, pontuação no formato CVSS V3 e descrição, além do documento com todos os detalhes que serão apresentados ao utilizador. A tabela temporária CPE tem como finalidade receber todas as especificações WFN com informações dos sistemas vulneráveis.

Após este processamento estar concluído os dados da tabela temporária não existentes ou com datas actualizadas nas tabelas CVE e CPE são comparados diretamente na base de dados e actualizados.

A linguagem de desenvolvimento deste processo é Python e a sua única dependência é a base de dados MySQL.

```
============
= CVE Sync =
============
Downloading: https://nvd.nist.gov/feeds/json/
cve/1.1/nvdcve-1.1-modified.j
son.zip
Extracting all files
Found: nvdcve-1.1-modified.json
Processing documents...
Processing Done, updating Database
Completed in: 2.078seconds
```

Figura 4.5: Sincronização CVE

O processamento das especificações WFN ocorre dependendo do operador presente nas configurações de cada CVE, podendo este ser 'OR' ou 'AND', indicando se a vulnerabilidade tem dependências para ocorrer a sua replicação. O processamento ocorre para as especificações que se encontrem vulnerávies e não as suas dependências, por forma a evitar falsos positivos/negativos na deteção de falhas.

4.3.2 Sincronização CPE

Este processo pretende extrair os dados das especificações WFN permitindo apresentar ao utilizador quais as versões de uma determinada dependência podem estar vulneráveis, garantindo que o utilizador não irá ser alertado para problemas já resolvidos e sem impacto na infraestrutura atual.

De forma similar ao processo de sincronização CVE, inicialmente é efetuado o download do ficheiro ZIP para uma nova pasta, sendo importante destacar que este ficheiro é único e contem todas as especificações independentemente da data em que foram adicionadas. Após extracção do ficheiro, todos os dados são processados e os seus detalhes como versão e sistema são enviados para uma tabela temporária que garante a inexistência de dados duplicados, sendo posteriormente enviados os dados para a tabela de detalhes onde serão automaticamente relaccionados com um ou vários CVE. Uma vez que, é necessário extrair todos os dados desde ficheiro cada vez que executado são enviados 10.000 dados para a base de dados a cada pedido, por forma a garantir que não existe uma sobrecarga.

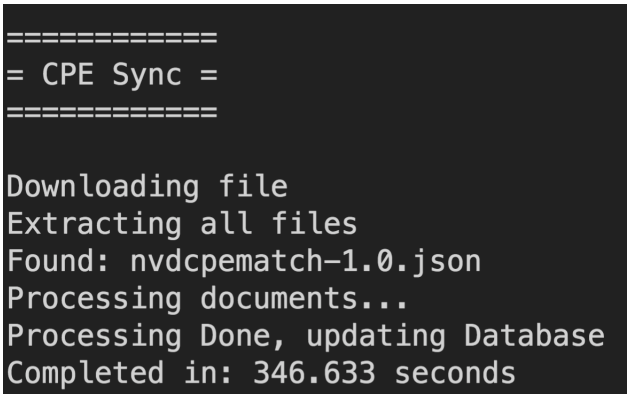


Figura 4.6: Sincronização CPE

4.4 Actualização de Vulnerabilidades de Utilizador

Após obtenção dos dados mais recentes, é necessário sincronizar com os dados de utilizadores procurando por novas vulnerabilidades.

Após inicio da tarefa agendada todos os dados na base de dados serão comparados através de comandos JOIN em SQL e os dados de utilizadores que se encontrem vulneráveis são seleccionados e enviados para uma fila Redis. Por fim existe uma actualização no timestamp de sincronização de dados.

```
public function handle()
{
    $users = DB::select('JOIN TABLES AND GET VULNERABLE USERS');

    foreach($users as $key=>$value) {
            $user = new User;
            $user->id = $value->id;
            $user->name = $value->name;
            $user->email = $value->email;
            $user->phone_number = $value->phone_number;
            ProcessEmailNotification::dispatch($user);
    }

    DB::statement("UPDATE Updated set last_update = CURRENT_TIMESTAMP;");
}
```

Bloco de Código 4.3: Obtenção de Vulnerabilidades de Utilizador

4.5 Sistema de Alertas

Para que um alerta seja enviado é necessário garantir que se encontra sempre um cliente Redis ativo e à espera de novos dados. Para colocar o cliente à escuta é necessário o seguinte comando:

```
        php artisan queue:listen
```

Bloco de Código 4.4: Fila Redis

```
[2020-08-17 14:28:49][ncXC0nFXaXhRx8bWo4BGjMoq2Xnd7CcC] Processing: App\Jobs\ProcessEmailNotification
[2020-08-17 14:28:52][ncXC0nFXaXhRx8bWo4BGjMoq2Xnd7CcC] Processed:  App\Jobs\ProcessEmailNotification
```

Figura 4.7: Prcessamento de Tarefas

Os novos registos devem ser imediatamento consumidos e processados, ativando um sistema de notificações para três plataformas com opções configuráveis. Para que a notificação seja enviada para Telegram foi necessário o desenvolvimento de um bot denominado @tekcve bot sobre os quais serão enviadas as mensagens referentes a novas vulnerabilidades do utilizador subscrito.

O envio de alertas ocorre por uma classe de notificações, onde estão detalhadas as propriedades

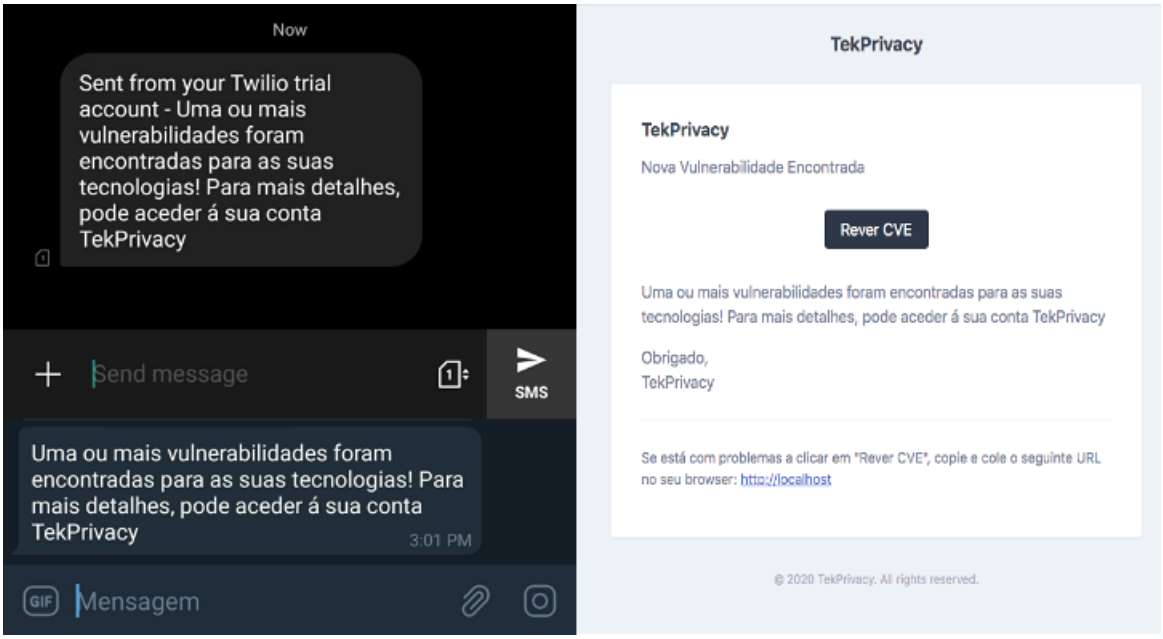


Figura 4.8: Notificações

de cada mensagem e quais as plataformas para o qual devem ser enviadas novas mensagens.

```
public function via($notifiable)
{
    return ['mail', TwilioChannel::class, TelegramChannel::class];
}

public function toMail($notifiable)
{
    return (new MailMessage)
                ->greeting('TekPrivacy')
                ->line('Nova Vulnerabilidade Encontrada')
                ->action('Rever CVE', url('/'))
                ->line('Uma ou mais vulnerabilidades foram encontradas para
                    as suas tecnologias! Para mais detalhes, pode aceder
                    sua conta TekPrivacy');
 }
```

Bloco de Código 4.5: Alertas de Vulnerabilidades por Utilizador

Capítulo 5

Resultados e Análise

O teste das várias funcionalidades do projeto foi realizado de forma modular. Inicialmente foram testados os tempos de todas as etapas para obtenção de novos ou atualizados CVE e CPE, incluindo o *download* do ficheiro desde o portal NVD, a extração de todos os ficheiros, o processamento de dados do ficheiro JSON e o envio de dados para a base de dados e posterior inserção de dados únicos.

Para o ficheiro de sincronização CVE foram verificados os tempos de todas as vulnerabilidades do ano 2019 (3.86MB) e das vulnerabilidades mais recentemente actualizadas (0.23MB).

Tabela 5.1: Performance de Sincronização CVE

	2019 (3.68MB)	Recentes (0.23MB)
Download	4s	0.9s
Extração	0.15s	0.01s
Processamento Python	6s	0.5s
Processamento SQL	14.5s	0.8s

O ficheiro de sincronização CPE apresenta um tempo total de processamento de 5 minutos e 46 segundos, sendo que este tempo é alterado sempre que novos dados são adicionados ao ficheiro. Atualmente o ficheiro JSON com os dados de todos os CPE tem 419.3MB.

A comparação e selecção de utilizadores com uma tecnologia vulnerável tem uma duração de 60 segundos incluindo 141.000 vulnerabilidades conhecidas. A duração deste processo poderia ser melhorado com a utilização de índices em todas as tabelas que são utilizadas e poderia ser testada a utilização de uma estrutura de dados diferente. O envio de notificações através de Redis tem uma duração de 3 segundos por pedido e podem ser definidas o número de notificações processadas

de forma concorrente. Este tempo está dependente do servidor de email, telegram e SMS, sendo utilizado para testes o servidor de email Mailtrap.

Todos os testes foram efetuados utilizando a plataforma Docker com ambientes Redis e MySQL simulados em ambiente Alpine Linux e sob uma máquina com 16GB de RAM e 6 CPU cores, com uma ligação de 500Mbs/s.

Foram ainda efectuados testes para verificar a eficácia do sistema para o alerta de novas vulnerabilidades, sendo criado um utilizador fictício e adicionados sistemas e tecnologias para esse utilizador com a finalidade de verificar que apenas uma mensagem é enviada mesmo que várias vulnerabilidades sejam descobertas no mesmo período e que apenas são enviadas mensagens no caso de um vulnerabilidade com correspondência ser encontrada. Estes testes foram bem sucedidos, garantindo que não são enviados dados duplicados ou incorretos para o utilizador, sendo que a tabela com a última atualização é revista a cada teste efetuado.

Foi efetuada uma comparação de dados com os dados disponibilizados no portal de vulnerabilidades CVE Details e verificou - se uma maior eficácia do sistema perante a detecção de todas as versões vulneráveis para as falhas encontradas.

Capítulo 6

Conclusões

A plataforma desenvolvida é uma ferramenta para deteção de vulnerabilidades que poderá ser bastante útil para gestão de diversos ativos garantindo uma rápida análise do dispositivo. A sua arquitectura modular torna possível adicionar novas funcionalidades de forma muito fácil e o seu *design* permite uma fácil utilização da plataforma por qualquer utilizador.

A performance da plataforma é razoável no processamento de novas vulnerabilidades, sendo esta implementada em Python e o processamento de novos dados com utilizadores existentes é bastante rápido. A utilização de Redis permite que o sistema seja escalável e eficiente, não comprometendo a performance, podendo ser utilizado para processamento de tarefas mais pesadas em futuros desenvolvimentos.

Apesar da deteção de vulnerabilidades em diversos dispositivos, o sistema encontra - se atualmente limitado a *software* ou *hardware* já conhecido e presente na lista de CVEs passada, sendo necessário ao utilizador inserir a sua lista de ativos manualmente e de acordo com opções existentes.

A integração da plataforma é muito rápida devido aos seus componentes modulares, sendo que poderá ser utilizada para proteger qualquer infraestrutura e garantir rápidos alertas após detetada nova falha.

Trabalho Futuro

No futuro e por forma a tornar a plataforma mais completa, seria interessante o desenvolvimento de um sistema que permita verificar se uma vulnerabilidade foi corretamente corrigida, bem como

a possibilidade de adicionar verificação de gestores de pacotes das mais diversas linguagens de programação, procurando dependências vulneráveis. Com a finalidade de garantir que todos os ativos se encontram listados, seria útil uma funcionalidade que permitisse a procura de ativos/serviços de uma organização.

No sistema de alertas será também necessário desenvolver um sistema que permite o utilizador escolher para quais dispositivos pretende ser alertado, bem como configurar para que tipo de problemas deseja receber um alerta.

Finalmente, no processamento de CVE e do *CPE Match Feed* será adicionada a verificação por SHA - 256, garantindo que só ocorre o processamento se novos dados forem adicionados, tornando o sistema mais eficiente.

Bibliografia

[1] Burp Suite - Application Security Testing Software [visited August 2020].

[2] CVE Details HomePage. Technical report [visited June 2020].

[3] Laravel - The PHP Framework for Web Artisans [visited August 2020].

[4] MySQL [visited August 2020].

[5] Nucleus HomePage. Technical report, Nucleus [visited June 2020].

[6] Redis [visited August 2020].

[7] Saucs HomePage. Technical report, Saucs [visited June 2020].

[8] Snyk HomePage. Technical report, Snyk [visited June 2020].

[9] Telegram Messenger [visited August 2020].

[10] Twilio - Communication APIs for SMS, Voice, Video and Authentication [visited August 2020].

[11] Cve and nvd relationship. https://cve.mitre.org/about/cve_and_nvd_relationship.html, August 2019. Accessed on 2020-06-05.

[12] Lillian Ablon and Andy Bogart. *Zero days, thousands of nights: The life and times of zero-day vulnerabilities and their exploits*. Rand Corporation, 2017.

[13] Jonathan Baker, Matthew Hansbury, and Daniel Haynes. The oval language specification. *MITRE, Bedford, Massachusetts (url: http://ebookbrowsee. net/ovallanguage-specification-08-08-2011-pdf-d222972411)*, 2011.

[14] Lee Brotherston and Amanda Berlin. *Defensive security handbook: best practices for securing infrastructure*. "O'Reilly Media, Inc.", 2017.

[15] Brant A Cheikes, Brant A Cheikes, Karen Ann Kent, and David Waltermire. *Common platform enumeration: Naming specification version 2.3*. US Department of Commerce, National Institute of Standards and Technology, 2011.

[16] Building Effective Cyber-Defense, Morey J Haber, and Brad Hibbert. Privileged attack vectors.

[17] FIRST.ORG. Common vulnerability scoring system v3.1: Specification document. https://www.first.org/cvss/specification-document. Accessed on 2020-06-02.

[18] Morey J Haber and Brad Hibbert. *Asset Attack Vectors: Building Effective Vulnerability Management Strategies to Protect Organizations*. Apress, 2018.

[19] Wolfgang Kandek. Vulnerability management for dummies, 2015.

[20] GeonLyang Kim, JinTae Oh, DongIl Seo, and JeongNyeo Kim. The design of vulnerability management system. *International Journal of Computer Science and Network Security (IJCSNS)*, 13(4):19, 2013.

[21] MITRE. Cve - request cve ids. https://cve.mitre.org/cve/request_id.html, June 2020. Accessed on 2020-06-05.

[22] MITRE. Cve - frenquently asked questions. https://cve.mitre.org/about/faqs.html, Mar 2020. Accessed on 2020-05-27.

[23] Sameer Nanda and Umashankar Ghugar. Approach to an efficient vulnerability management program. *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, (6), 2017.

[24] NDIST. Nvd cwe slice. https://nvd.nist.gov/vuln/categories. Accessed on 2020-06-07.

[25] NIST. Known affected software configurations. https://nvd.nist.gov/vuln/Vulnerability-Detail-Pages. Accessed on 2020-06-02.

[26] OpenSCAP. Scap components. https://www.open-scap.org/features/scap-components/. Accessed on 2020-06-07.

[27] SD Quinn. *Technical specification for the security content automation protocol (SCAP): SCAP version 1. 0*, volume 800. DIANE Publishing, 2010.

[28] Tenable. Nessus agent 7.1.x deployment and user guide. 2020.

[29] Michael E Whitman, Herbert J Mattord, David Mackey, and Andrew Green. *Guide to network security*. Cengage Learning, 2012.

[30] Davey Winder. Ghost confirms hack attack: 750,000 users spooked by critical vulnerability. https://www.forbes.com/sites/daveywinder/2020/05/03/ghost-confirms-hack-attack-750000-users-spooked-by-critical-vulnerability/, May 2020. Accessed on 2020-05-27.