DEPARTAMENTO DE INFORMÁTICA E SISTEMAS

Cibersegurança – Automação de Testes de Intrusão

Relatório de Estágio

Autor

Fábio Capobianchi de Souza

Orientadores

Eng. Amâncio Santos **ISEC**

Eng. João Perdigoto **G9Telecom**

Licenciatura em Engenharia Informática
Ramo de Redes e Administração de Sistemas

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, julho e 2023

Agradecimentos

Gostaria de agradecer à G9Telecom, pela oportunidade de realização deste estágio e pelo acolhimento e apoio que me foram oferecidos, especial ao Eng. João Perdigoto, pela orientação apoio prestados. Também gostaria de agradecer ao meu supervisor Pedro Paiva, por todo acompanhamento oferecido e conhecimento partilhado, e a toda G9Telecom, por me fazerem sentir parte da equipa.

Agradeço ao Instituto Superior de Engenharia de Coimbra, a todos seus professores e funcionários, especialmente ao professor Amâncio Santos pela orientação deste estágio e por toda atenção prestada durante o meu percurso pelo ISEC. Por fim, gostaria de agradecer à minha família e amigos, em especial à minha esposa Gizelli Almeida Silva, que sempre mostrou seu apoio e compreensão durante meu percurso académico, sendo peça fundamental para a conclusão desta etapa da minha vida, e aos meus filhos Gabriel Capobianchi e Helena Capobianchi, que com seu carinho, me renovam as forças frente as dificuldades.

Resumo

O estágio realizado na empresa G9Telecom, tinha como propósito a realização do projeto "Cibersegurança – Automação de Testes de Intrusão", este projeto consiste no desenvolvimento de um *Application Programming Interface* (API) WEB com recurso para automatização de testes de intrusão, e que possa auxiliar no processo de rastreio de vulnerabilidade de sistemas.

Para a criação da API, foi necessário a escolha do ambiente de desenvolvimento, esse engloba o sistema operacional, a linguagem de programação e software necessários. Foi definido que na execução do projeto, deveriam ser utilizados apenas software de código aberto, deste modo foi decidido que o Linux seria a base de desenvolvimento, entretanto foi necessário testar algumas distribuições para decidir qual melhor atende as necessidades da API, como Ubuntu, Centos, Kali e Fedora. Após os testes, O Ubuntu server demonstrou ser o mais indicado para hospedar a API.

A seguir foi necessário a escolha de um software servidor e linguagem de desenvolvimento e método de armazenamento dos dados. Nessa fase, foi sugerido pelo meu supervisor na empresa, que a linguagem mais indicada para a criação da API, seria o PHP. Com essa indicação optei por uma instalação *Linux, Apache, MySQL, PHP* (LAMP) que oferece um ambiente completo. Na fase seguinte, em conjunto com a empresa acolhedora, foi definida a arquitetura da API, suas funcionalidades, e a estrutura da organização dos dados, nessa fase foi criado o diagrama da base de dados da API.

Após a definição de todos os parâmetros, teve início o desenvolvimento da API, são criados os métodos, a base de dados e interface web, a seguir, foram realizados testes exaustivos ao funcionamento da API de forma a eliminar erros de execução e falhas de verificação dos parâmetros. Na fase final foram feitos testes em vários servidores e comparados os resultados, de modo a demonstrar a utilidade da API no auxilio de verificações de vulnerabilidades de um sistema.

Palavras-chave: Arquitetura Sistema, Cibersegurança, Automatização, Scandata.

Abstract

The internship carried out at the company G9Telecom, had as its purpose the realization of the project "Cybersecurity - Automation of Intrusion Tests", this project consists of the development of a WEB API with a resource for automating penetration tests, and that can assist in the process of tracking system vulnerabilities.

For the creation of the API, it was necessary to choose the development environment, which includes the operating system, the programming language and the necessary software. It was defined that in the execution of the project, only open source software should be used, so it was decided that Linux would be the basis of development, however it was necessary to test some distributions to decide which best meets the needs of the API, such as Ubuntu, Centos, Kali and Fedora. After testing, Ubuntu server proved to be the most suitable for hosting the API.

Next, it was necessary to choose a server software and development language and data storage method. At this stage, it was suggested by my supervisor at the company that the most suitable language for creating the API would be PHP. With this indication I opted for a *Linux, Apache, MySQL, PHP* (LAMP) installation that offers a complete environment. In the next phase, together with the host company, the architecture of the API, its functionalities, and the structure of the data organization were defined, in this phase the diagram of the API database was created.

After defining all the parameters, the development of the API began, the methods, the database and the web interface were created, then exhaustive tests were carried out on the operation of the API in order to eliminate execution errors and parameter verification failures. In the final phase, tests were carried out on several servers and the results were compared, in order to demonstrate the usefulness of the API in helping to check the vulnerabilities of a system.

Índice

Lista de Figuras

Lista de Tabelas

Siglas e Acrónimos

ACK *Acknowledgement*.(Reconhecimento)

API *Application Programming Interface*.

AWS *Amazon Web Services*.

CISO *Chief Information Security Officer*.

GUI *Graphical User Interface*.

ICMP *Internet Control Message Protocol*.

ISO 27001 Norma que constitui o padrão internacional para um Sistema de Gestão da Segurança da Informação.

LAMP *Linux, Apache, MySQL, PHP*.

NIST *National Institute of Standards and Technology*.

PCIDDS *Payment Card Industry Data Security Standard*.

PSK *Phase-shift keying*.(Encerra a ligação)

RST *Reset*.(Encerra a ligação)

SSL *Secure Sockets Layer*.

SWOT *Strengths, Weaknesses, Opportunities and Threats*.

SYN *Synchronize*.(Sincronizar)

TCP/IP *Transmission Control Protocol Internet Protocol*.

WEP *Wired Equivalent Privacy*.

Wifi *Wireless Fidelity*.

WPA *Wi-Fi Protected Access*.

Capítulo 1

Introdução

Este relatório visa documentar o estágio desenvolvido ao longo do segundo semestre do ano letivo de 2022/2023, no âmbito da unidade curricular Projeto ou Estágio da Licenciatura em Engenharia Informática — ramo de Redes e Administração de Sistemas do Instituto Superior de Engenharia de Coimbra. A entidade de acolhimento que proporcionou o estágio foi a G9Telecom, sediada em Coimbra.

1.1 Enquadramento

A crescente complexidade das redes e sistemas de informação tem trazido desafios significativos na área de cibersegurança. Com o aumento das ameaças cibernéticas e vulnerabilidades, torna-se essencial ter soluções eficazes para monitorizar e proteger as infraestruturas de sistemas contra potenciais ataques. Nesse contexto, a automatização de testes de segurança tornou-se uma prática fundamental para garantir a integridade e a confidencialidade dos dados.

O presente relatório descreve o desenvolvimento e a implementação da *Application Programming Interface* (API) Scandata, uma ferramenta projetada para automatizar testes de vulnerabilidade em infraestruturas de sistemas. O objetivo principal dessa API é fornecer uma solução eficiente para a realização de testes automatizados em clientes e serviços, permitindo a

deteção de vulnerabilidades e ameaças.

1.2 Entidade de Acolhimento

A G9Telecom é uma empresa portuguesa com mais de 15 anos de experiência em engenharia e telecomunicações. Inicialmente focada em projetos e consultoria para operadores de telecomunicações, a empresa evoluiu e hoje é um Operador de Telecomunicações de Voz Fixa, oferecendo soluções de voz e dados integradas, além de serviços de consultoria e gestão de comunicações. Com uma rede IP nacional própria, a G9Telecom garante conectividade de alta qualidade para seus clientes, investindo em pesquisa e desenvolvimento para lançar regularmente produtos e soluções inovadoras no mercado.

A G9Telecom contempla na sua estratégia um investimento sistemático em investigação e desenvolvimento (I&D) que se tem refletido, ao longo dos anos, na superior autonomia da empresa nos sistemas críticos à sua operação e no lançamento regular de produtos e soluções inovadoras.[4]

1.3 Objetivos

O estágio teve como principal objetivo, o desenvolvimento de uma ferramenta para automatização de testes para deteção de vulnerabilidades na segurança de sistemas. Para que esses objetivos pudessem ser alcançados, foram precisos ultrapassar os seguintes passos:

- Aprofundar o conhecimento em cibersegurança: Durante o estágio, o objetivo principal foi aprimorar os conhecimentos na área de cibersegurança, compreendendo conceitos fundamentais e práticas para garantir a segurança de sistemas e redes.

- Desenvolver habilidades práticas em testes de segurança: O estágio teve como objetivo proporcionar experiência prática em testes de segurança, incluindo a realização de análises de vulnerabilidades, testes de intrusão e análise de resultados.

- Contribuir para o desenvolvimento de uma ferramenta de automação de testes: O objetivo principal do estágio foi dedicado ao desenvolvimento e aprimoramento de uma ferramenta de automação de testes de segurança.

- Adquirir experiência em um ambiente profissional: O estágio teve como objetivo proporcionar uma experiência real de trabalho em um ambiente profissional, permitindo a aplicação dos conhecimentos académicos em situações reais.

No geral, os objetivos do estágio na G9Telecom foram alcançados com sucesso, proporcionando uma valiosa oportunidade de aprendizagem e aplicação prática dos conhecimentos adquiridos ao longo da Licenciatura em Engenharia Informática no ISEC.

1.4 Estrutura do Relatório

O relatório foi estruturado e dividido nos cinco capítulos que se seguem:

Capítulo 1, é apresentada uma visão geral do projeto, onde se contextualiza a importância da cibersegurança e a necessidade de ferramentas que possam automatizar os testes de segurança em infraestruturas.

Capítulo 2, são discutidos os principais conceitos relacionados à cibersegurança, serviços de rede, e automatização de testes, destacando a importância de práticas e ferramentas para detetar e prevenir ameaças cibernéticas.

Capítulo 3, é detalhada a arquitetura da API Scandata, enfatizando a escolha criteriosa dos componentes do ambiente de desenvolvimento, como o sistema operacional, o servidor web e o base de dados. Bem como a escolha da linguagem de desenvolvimento e a utilização de software NMAP. Nesse capitulo é apresentada uma análise SWOT para fundamentar essa escolhas.

Capítulo 4, são apresentados os detalhes da utilização da API, como a criação e execução de testes bem como a comparação dos seus resultados.

Conclusão, com base nos resultados obtidos durante a implementação e execução de testes, nesse capítulo são apresentadas as conclusões sobre o projeto .

Capítulo 2

Segurança nos Serviços de Redes

Os serviços de redes são fundamentais na era digital, com a elevada necessidade de interligação entre dispositivos e a troca de informações, se mostram indispensáveis para a eficiência e produtividade.

Estes serviços englobam um conjunto de recursos e tecnologias que permitem a comunicação entre diferentes sistemas, possibilitando o partilha de dados, acesso à Internet, conexão remota, entre outras funcionalidades.

2.1 Conceitos

Serviços de rede referem-se a softwares e ferramentas que permitem a comunicação e partilha de recursos entre dispositivos conectados por uma rede. Esses serviços são projetados para simplificar o gestão da rede e melhorar a eficiência e a segurança do tráfego de dados.[6]

Cada serviço de rede é composto no mínimo por quatro elementos:

- Servidor: computador que realiza a parte principal do serviço, usando seus recursos locais e/ou outros serviços.
- Cliente: computador que solicita o serviço através da rede; geralmente o cliente age a pedido de um ser humano, através de uma interface de utilizador, mas ele também pode ser o representante de outro sistema computacional.

- Protocolo: é a definição do serviço propriamente dito, ou seja, os passos, o conjunto de mensagens e os formatos de dados que definem o diálogo necessário entre o cliente e o servidor para a realização do serviço.

- Middleware: é o suporte de execução e de comunicação que permite a construção do serviço. Em geral o *middleware* é composto por sistemas operacionais e protocolos de rede encarregados de encaminhar os pedidos do cliente para o servidor e as respostas ao cliente.

2.1.1 Tipos de Arquiteturas

De um ponto de vista da arquitetura, os sistemas que constroem serviços de rede podem se organizar de várias formas. As arquiteturas de serviços de rede mais frequentes na Internet são as seguintes:

- *Two-Tier*: esta arquitetura tem dois componentes: o servidor, responsável pela execução do serviço, e o cliente, responsável pela apresentação dos resultados e interação com o utilizador, como pode ser verificado na figura 2.1.

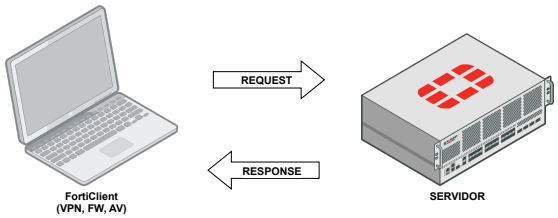


Figura 2.1: Arquitetura Two-Tier.

- *Three-Tier*: conhecida como arquitetura de três camadas, é um modelo que divide um sistema de software em três componentes distintos. O cliente é responsável pela interface com o utilizador. O servidor

concentra a lógica da aplicação, processa as informações. Por fim os repositórios de dados que armazenam e gerem a informação de forma estruturada. Essa abordagem oferece uma organização mais eficiente e escalável, permitindo o desenvolvimento de aplicações robustas e de fácil manutenção, como apresentada na figura 2.2.

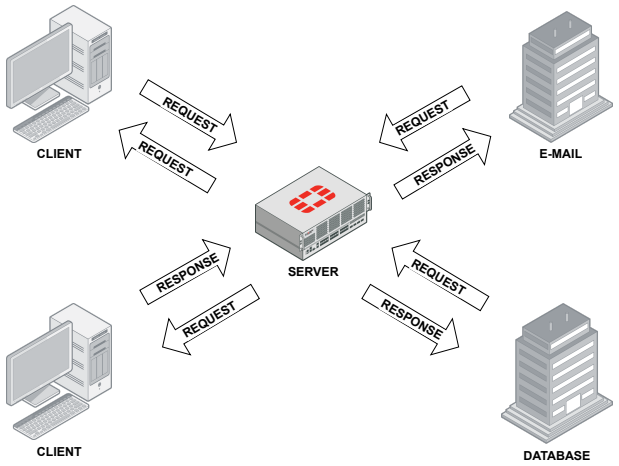


Figura 2.2: Arquitetura Three-Tier.

- *Peer-to-Peer*: nesta arquitetura todos os participantes são ao mesmo tempo servidores (oferecem serviços e recursos) e clientes (usam serviços e recursos) uns dos outros. Muitos serviços de partilha de arquivos e de comunicação entre utilizadores como poder observado na figura 2.3, se estruturam dessa forma.

2.1.2 Serviços

Existem muitos tipos diferentes de serviços de rede, incluindo serviços de correio eletrónico, serviços de mensagens instantâneas, serviços de armazenamento em nuvem, serviços de partilha de arquivos e serviços de videoconferência, entre outros. Cada um desses serviços é projetado para atender a uma necessidade específica de comunicação ou colaboração.

São apresentados aqui alguns exemplos de serviços de rede:

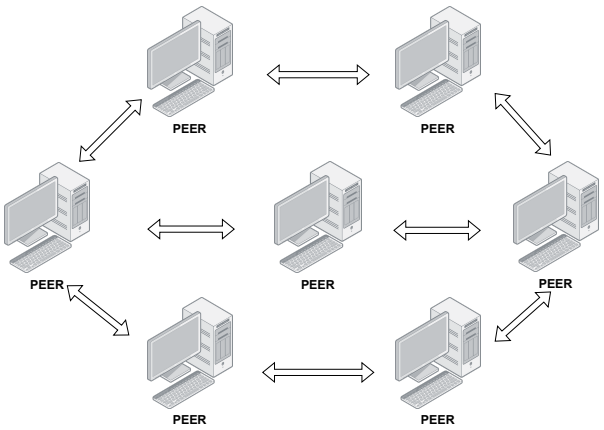


Figura 2.3: Arquitetura Peer to Peer.

- Serviços de gestão de rede e configurações, DNS (*Domain Name System*), DHCP (*Dynamic Host Configuration Protocol*) e SNMP (*Simple Network Management Protocol*), que possibilitam gestão e monitorizar a rede.

- Serviços de partilha de informação, como HTTP (Hyper Text Transfer Protocol), HTTPS (Hyper Text Transfer Protocol Secure) e FTP (File Tranfer Protocol).

- Serviços de partilha de arquivos e impressoras, como NFS (Network File System) e IPP (Internet Print Protocol), que permitem que vários utilizadores acessem e usem arquivos e impressoras em uma rede.

- Serviços de segurança de rede, como firewalls e sistemas de deteção de intrusão, que protegem a rede contra ataques externos.

- Serviços de correio eletrónico e mensagens instantâneas, como SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol v3) e IMAP (Internet Message Access Protocol) que permitem que os utilizadores se comuniquem uns com os outros por meio da rede.

- Serviços de acesso remoto, como VPN (Virtual Private Network), SSH

(Secure Shell) e Telnet, que permitem que os utilizadores possam aceder a rede de forma segura e remota de qualquer lugar com uma ligação à Internet.

Estes serviços, desempenham um papel importante no suporte e no desenvolvimento da infraestrutura de rede moderna e permitem a comunicação e colaboração entre utilizadores e dispositivos em todo o mundo. A figura 2.4 demonstra a estrutura do funcionamento dos protocolos de rede.

Figura 2.4: Protocolos de Redes

Os serviços de rede, são executados sobre o protocolo *Transmission Control Protocol Internet Protocol* (TCP/IP), cada serviço é associado a um número chamado porto que é onde o servidor espera pelas conexões dos computadores clientes. Uma porto de rede pode se referenciada tanto pelo número como pelo nome do serviço[1].

Abaixo, alguns exemplos de portos padrões usadas em serviços TCP/IP:

- 21/TCP - FTP (File Transfer Protocol - Protocolo de transferência de arquivo) - porto do Protocolo de Transferência de Arquivos.
- 22/TCP, UDP – SSH (Secure Shell - Shell seguro) - Usada para logins seguros, encriptação.
- 23/TCP, UDP - Telnet (terminal virtual remoto) - Comunicação de texto sem encriptação.

- 25/TCP, UDP - SMTP (Simple Mail Transfer Protocol - Protocolo simples de envio de e-mail) envio de e-mails.
- 53/TCP, UDP - DNS (Domain Name System - Sistema de nome de domínio) resolução de nomes.
- 67/UDP - BOOTP (BootStrap Protocol) server; também utilizada por DHCP (Protocolo de configuração dinâmica do Host), porto de escuta.
- 68/UDP - BOOTP client; também utilizada por DHCP, porto de escrita.
- 79/TCP - Finger (detalhes sobre utilizadores do sistema).
- 80/TCP - HTTP (HyperText Transfer Protocol) - transferência de páginas WEB.
- 110/TCP - POP-3 (Post Office Protocol version 3): Protocolo de Correio Eletrônico, versão 3 - usada para recebimento de e-mail.
- 119/TCP - NNTP (Network News Transfer Protocol) (Protocolo de transferência de notícias na rede) - usada para recebimento de mensagens de newsgroups.
- 143/TCP, UDP – IMAP4 (Internet Message Access Protocol) (Protocolo de acesso a mensagem da interne).
- 161/TCP, UDP – SNMP (Simple Network Management Protocol) (Protocolo simples de gestão de rede).
- 220/TCP, UDP - IMAP, Interactive Mail Access Protocol, version 3 (Protocolo de acesso interativo ao mail).
- 389/TCP, UDP - LDAP (Lightweight Directory Access Protocol) Protocolo de acesso a diretório lightweight.

Os serviços de rede desempenham um papel crítico na comunicação e na transferência de dados entre dispositivos ligados a uma rede. Eles permitem

que os utilizadores partilhem recursos, como arquivos, impressoras e ligações á Internet, realizem tarefas complexas, como comunicação em tempo real e colaboração[5].

2.2 Cibersegurança

A definição de cibersegurança, são um conjunto de medidas e ações de prevenção, monitorização, deteção, reação, análise e correção que visam manter o estado de segurança desejado e garantir a confidencialidade, integridade e disponibilidade da informação, das redes digitais e dos sistemas de informação no ciberespaço, e das pessoas que nele interagem.[3]

Cibersegurança é o conjunto de práticas, tecnologias e medidas adotadas para proteger sistemas, redes, dispositivos e dados contra ataques cibernéticos, roubo de informações e outras ameaças relacionadas à segurança digital.

A segurança cibernética é uma preocupação crescente, com o crescente aumento de dados armazenados digitalmente e de conectividade entre dispositivos e sistemas. Ciberataques podem resultar em perda de informações confidenciais, perda financeira, danos à reputação e até mesmo riscos à segurança física.

Algumas das práticas e tecnologias comuns usadas em cibersegurança incluem criptografia, autenticação de utilizadores, firewalls, deteção de intrusões, antivírus, backups regulares de dados e testes de intrusão para identificar e corrigir vulnerabilidades. Além disso, é importante que as empresas adotem políticas e procedimentos de segurança eficazes e capacitem seus funcionários com treinamento e consciencialização sobre segurança cibernética.

2.2.1 Testes de cibersegurança

Os testes de cibersegurança são realizados para validar se as configurações de segurança implementadas estão a funcionar como pretendido.

As ameaças na área da cibersegurança estão em constante mudança e evolução. Novas ameaças são constantemente identificadas, e as configurações do sistema podem ser corrompidas, portanto, devem ser realizados testes

periódicos de cibersegurança. Os testes de avaliação de vulnerabilidade e intrusão compõem um conjunto de técnicas para identificar vulnerabilidades em softwares e sistemas.

2.2.2 Avaliação de Vulnerabilidades

Esta avaliação consiste no processo de utilização de softwares e outros processos automatizados para encontrar vulnerabilidades conhecidas em sistemas. As verificações de vulnerabilidade da rede são realizadas para identificar possíveis pontos de acesso e configurações de segurança obsoletas. Estes testes são realizados em LANs e WLANs. As análises de bases de dados são conduzidas para detetar fraquezas numa base de dados e para encontrar ambientes de desenvolvimento inseguros em sistemas.

2.2.3 Testes de intrusão

Teste de intrusão é um ciberataque realizado com a permissão do proprietário do sistema. Isto inclui reconhecimento para recolher dados e informações sobre o sistema alvo, após o qual esta informação é usada para ter acesso e para manter o acesso ao sistema. Depois de explorar o alvo, o testador divulga as suas descobertas para a organização. Os testes de intrusão devem ser realizados por um profissional com o mínimo de conhecimento prévio possível sobre o sistema testado a fim de encontrar vulnerabilidades que possam ter ocorrido na implementação do sistema. O processo de testes de intrusão pode ser dividido em cinco fases principais, como apresentado na figura 2.5:

Figura 2.5: Pentest stages.

1. **Planeamento e Reconhecimento**, definir o escopo e métodos de análise, quais os testes a serem realizados e os objetivos a serem alcançados. Nesta fase devem ser reunidas toda a informação possível sobre o sistema, como: topologia, nomes de hosts, redes e domínios, sistemas operativos, servidores de email, gerência de contas e outras informações relevantes.

2. **Varredura**, com base nos dados recolhidos na fase de reconhecimento, tem início a fase de varredura, onde são utilizadas uma série de ferramentas para identificar e verificar as vulnerabilidades e a existência de possíveis pontos de entrada, como portos abertos e elementos como o tráfego na rede alvo.

3. **Obter Acesso**, uma vez que todas as vulnerabilidades tenham sido identificadas e listadas, tem início a fase de Exploração. Nesta fase, consiste em acessar o sistema de destino e explorar as vulnerabilidades listadas com o objetivo de conseguir um aumento de privilégios no sistema, de modo a obter acesso total ao sistema. Para esta fase são utilizadas ferramentas para uma simulação de ataque real.

4. **Manter o Acesso**, esta fase mostra o impacto de um ataque, utilizando privilégios de acesso. Após ter acesso ao sistema, é preciso o manter, para isso deve-se tentar obter o nível de máximo de privilégios, informações da rede e obter acesso ao maior número de sistemas e identificar quais serviços podem ser acedidos. Nesta fase, é possível perceber quais seriam as consequências de um ataque real ao sistema.

5. **Relatório**, após identificar e explorar as vulnerabilidades, classificar os pontos críticos, recolher evidências das falhas existentes, deve ser gerado um relatório reportando os itens encontrados, apontando erros como má configuração, ferramentas que conseguem aceder o sistema com sucesso, aplicações em conflito, problemas de autenticação, atualizações de sistemas, falhas na rede e tudo que for crucial para a melhoria na segurança.[2]

2.2.4 Vantagens da realização dos Testes

Os principais benefícios da realização de testes de intrusão incluem a manutenção da conformidade e a adesão a entidades de controle, de modo a manter um padrão ou licença ativa, como o Norma que constitui o padrão internacional para um Sistema de Gestão da Segurança da Informação (ISO 27001) e *Payment Card Industry Data Security Standard* (PCIDDS). Prevenir ataques cibernéticos e violações de segurança cibernética também são benefícios que impactam diretamente a reputação, o desempenho financeiro e os níveis de confiança.

Tornar a organização menos suscetível a ataques de hackers é um benefício importante. A realização regular de testes de intrusão ajuda o *Chief Information Security Officer* (CISO) e os profissionais de tecnologia a se manterem atualizados e atualizados sobre as medidas de defesa de segurança cibernética mais recentes.

2.3 Distribuições Linux

2.3.1 O que é uma distribuição Linux?

Uma distribuição Linux, é um sistema operacional compilado a partir de componentes desenvolvidos por vários projetos e criadores de código aberto. Cada distribuição inclui o kernel do Linux (a base do sistema operacional), os utilitários Linux shell (a interface do terminal e os comandos), o servidor X (para um ambiente de trabalho gráfico), o ambiente de área de trabalho, um sistema de gestão de pacotes, um instalador e outros serviços. Muitos componentes são desenvolvidos independentemente uns dos outros e são distribuídos na forma de código-fonte. As distribuições também incluem um navegador, ferramentas de gestão e outros softwares, como o hipervisor KVM. Uma única distribuição Linux pode conter milhares de pacote de software, utilitários e aplicações.

As distribuições do Linux compilam esses pacotes de software, criando um único sistema operacional que pode ser instalado e inicializado. As distribuições do Linux estão disponíveis para computadores pessoais, servidores

sem interface gráfica, para supercomputadores, dispositivos móveis entre outros. Por ser um software de código aberto, qualquer pessoa pode criar sua própria distribuição Linux, a partir do código-fonte ou modificando uma distribuição existente. Atualmente, mais de trezentas distribuições Linux são mantidas ativamente. A seguir a figura 2.6 apresenta o Tux, pinguim mascote da Linux.

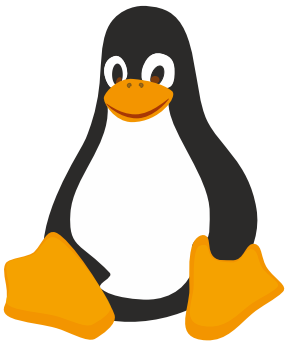


Figura 2.6: Tux pinguim mascote do Linux.

Abaixo são apresentadas as principais distribuições de código aberto Linux utilizadas para testes e monitorização.

Kali Linux

O Kali Linux é a distribuição Linux mais popular entre os profissionais da área de cibersegurança, utilizada para testes de intrusão e hacker ético. É uma distribuição baseada em Debian, de código aberto, desenvolvida pela Offensive Security. O Kali oferece mais de seiscentas ferramentas frequentemente atualizadas, para hacker ético, forense digital, pesquisa de segurança e engenharia reversa.

Ubuntu Server

O Ubuntu Server é uma distribuição de código aberto baseada no sistema operacional Ubuntu, desenvolvida para atender ás necessidades de servidores e ambientes de computação em nuvem. Com uma série de recursos e funcionalidades voltadas para desempenho, segurança e escalabilidade, o Ubuntu Server é uma escolha popular para hospedagem de serviços e aplicativos na rede.

Uma das principais vantagens do Ubuntu Server é a sua estabilidade. Essa distribuição é conhecida por oferecer um ambiente robusto, capaz de lidar com cargas de trabalho intensivas e de garantir alta disponibilidade para os serviços hospedados. Além disso, o Ubuntu Server é suportado por uma vasta comunidade de utilizadores e desenvolvedores, o que facilita a obtenção de suporte e a resolução de problemas.

O Ubuntu Server é compatível com uma ampla gama de tecnologias e serviços, permitindo integrações com outras ferramentas e soluções para melhorar ainda mais a funcionalidade da sua *Application Programming Interface* (API).

Em termos de ameaças, embora o Ubuntu Server seja conhecido por ser um ambiente seguro, é fundamental manter o sistema e as ferramentas atualizados com as últimas correções de segurança e seguir as melhores práticas para mitigar possíveis riscos.

Back Box

BackBox Linux é uma distribuição de código aberto, baseada em Ubuntu desenvolvida para realizar testes de intrusão e avaliações de segurança. Foi concebido para ser rápido e fácil de utilizar. Fornece um ambiente de trabalho básico, mas completo. O BackBox possui repositório de software próprio, frequentemente atualizado para as versões mais recentes e estáveis das ferramentas de hacker ético e teste de intrusão, mais utilizadas. Esta distribuição é bastante utilizada, devido ao fato de oferecer um conjunto de ferramentas para análise de rede, teste de aplicações da WEB , análise de vulnerabilidade de segurança e exploração. Também é possível optar pelo BackBox baseado

em nuvem na *Amazon Web Services* (AWS), independente do dispositivo utilizado.

SO Parrot Security

O Parrot Security OS é outra das distribuições Linux muito populares para hacking e testes de intrusão. Desenvolvidos pela equipe Frozenbox, oferece um ambiente de área de trabalho GNOME2 com uma ótima interface e de simples utilização. Embora seja leve e fácil de usar, também possui uma série de ferramentas úteis de teste de intrusão, que garantem o anonimato durante a execução de tarefas de hacking. Esta distribuição também é muito utilizada por desenvolvedores de software, pois possui ferramentas para defesa de privacidade e teste de segurança de software para aplicações. Portanto, é uma distribuição voltada para hacker ético e testes de intrusão.

BlackArch

O BlackArch é uma das principais distribuições Linux. Baseada no Arch Linux foi desenvolvida com foco em testes de intrusão, muito utilizada por profissionais em segurança e hackers éticos . Uma de suas características mais marcantes é seu repositório com mais de duas mil ferramentas de hacker ético que são testadas antes do lançamento. A instalação dessas ferramentas pode ser feita por pacotes, ou completa.

DEFT Linux

DEFT Linux , que significa Digital Evidence and Forensics Tool, é outra das melhores distribuições Linux para hacker ético baseada no Ubuntu. Esta distribuição Linux oferece uma enorme variedade de ferramentas forenses e guias de utilizador para iniciantes. Suas funcionalidades visam principalmente a análise forense e recolha de evidências digitais. Portanto, é mais comum entre as autoridades policiais e agências relacionadas.

Bugtraq

Bugtraq é uma distribuição popular de Linux, voltada para a comunidade, usada principalmente para hacker ético, forenses digitais e tarefas relacionadas com a cibersegurança. O Bugtraq oferece uma variedade de ferramentas de teste de intrusão, incluindo análise forense móvel e ferramentas de teste de software malicioso. A comunidade desta distribuição Linux, também desenvolve muitos softwares e ferramentas que podem ser utilizadas para executar várias outras tarefas de hacker ético. O Bugtrap também oferece uma versão para Android.

Samurai WEB Testing

O Samurai WEB Testing Framework é uma distribuição Linux pré-configurada baseada no Ubuntu que é usada especificamente para testes de intrusão de aplicações WEB, entre outras tarefas de teste de segurança interessantes. Esta distribuição trás uma variedade de ferramentas de hacker ético gratuitas e de código aberto, para deteção e exploração de vulnerabilidades em aplicações WEB. Embora o Samurai seja uma máquina virtual e não um sistema operacional completo, está incorporado com todas as ferramentas necessárias para testes abrangentes de aplicações WEB. O Samurai é considerado a melhor distribuição Linux para testes de intrusão em aplicações WEB.

Caine *(Computer Aided Investigative Environment)*

Caine significa Ambiente de Investigação Auxiliado por Computador. Esta distribuição Linux baseada no Ubuntu, oferece muitas ferramentas para análise forense de computadores e análise de segurança. Pode ser executado como um sistema operacional portátil diretamente de uma unidade USB de arranque, ou pode ser instalado. O Caine inclui ferramentas para auditoria de memória, análise de rede, análise de banco de dados e análise forense digital. Esta distribuição oferece utilitários básicos como navegador da WEB, um cliente de email, um editor de imagens, entre outros.

Fedora Security Spin

O Fedora Security Spin é uma das melhores distribuições Linux lançadas como uma variação do Fedora, desenvolvida pela Security Lab, foi projetada para testes de segurança e para fins de ensino. Seu principal objetivo é fornecer suporte a professores e alunos, sobre segurança da informação , análise forense e segurança de aplicações WEB. Como outras distribuições Linux importantes, O Fedora traz uma série de ferramentas para auditoria de segurança, teste de intrusão e resgate do sistema.

Network Security Toolkit

O Network Security Toolkit (NST) é uma unidade USB de arranque baseado em Linux que fornece um conjunto de ferramentas gratuitas e de código aberto para segurança de computador e rede para executar tarefas rotineiras de segurança e diagnóstico de rede e monitorização. O NST possui recursos de gestão de pacotes semelhantes ao Fedora e mantém seu próprio repositório de pacotes adicionais.

2.4 Automatização de Testes

A automatização de testes de cibersegurança, consiste num processo em que são utilizadas ferramentas e técnicas de automatização, para avaliar a segurança de um sistema. Através da automatização de testes, é possível realizar testes repetitivos, de forma mais eficiente que do método manual.

Existem diversas ferramentas para a realização de testes de segurança automatizados, como ferramentas de varredura de vulnerabilidades, ferramentas de teste de intrusão e ferramentas de análise de código. Essas ferramentas são capazes de identificar potenciais vulnerabilidades do sistema, permitindo que os responsáveis possam corrigi-las antes que sejam exploradas e corrompidas por terceiros. A figura 2.7 apresenta o cadeado, que representa segurança.

A automação de testes de segurança é uma parte importante do processo de desenvolvimento de um sistema seguro, e deve ser realizada em conjunto

Figura 2.7: Testes Automatizados

com outras práticas de segurança, como revisões de código, testes manuais e auditorias à segurança.

É importante lembrar que as ferramentas de teste automatizado, não substituem a necessidade de profissionais em segurança para analisar e interpretar os resultados destes testes e tomar decisões sobre como corrigir as vulnerabilidades identificadas.

2.4.1 Softwares para Automatização

Existem uma grande oferta de software para automatização de testes, tanto para uso comercial como gratuitas. A seguir são apresentados alguns dos *softwares* de código aberto mais utilizados.

1. **Nmap**

   O Nmap é a ferramenta ideal para reconhecimento, este software é capaz de verificar rapidamente redes de grande dimensão e pode ser executado em todos os principais sistemas operativos.

Com esta ferramenta é possível descobrir quais *hosts* estão disponíveis na rede; quais serviços estão a se executados; quais versões do seu sistema operacional; que tipo de filtro de pacotes e *firewalls* estão em uso; e outras informações úteis necessárias antes de iniciar um teste de segurança. O Nmap permite a execução de *scripts* para automatização de tarefas.

O Nmap oferece uma ampla variedade de recursos avançados com documentação abrangente e muitos tutoriais disponíveis que abrangem a linha de comandos e as versões da *Graphical User Interface* (GUI).

2. **Wireshark**

   O Wireshark é um analisador de protocolo de rede muito utilizado, e também pode se executado nos principais sistemas operativos. O Wireshark faz capturas em tempo real, suporte de desencriptação e análise *off-line*, para todos os principais protocolos de rede. Existe documentação abrangente e tutoriais em vídeo.

3. **Legion**

   Legion é uma ferramenta de teste de intrusão de rede extensível e semi automatizada. A documentação é esparsa, mas a GUI possui menus e painéis de contexto, facilitando a conclusão de muitas tarefas. A funcionalidade modular o torna personalizável e vincula automaticamente os CVEs (*Common Vulnerabilities and Exposures*) descobertos com exploits no *Exploit Database*.

4. **Jok3r**

   Outra ferramenta interessante para infraestrutura de redes e testes de intrusão na *web* é o Jok3r. É uma compilação de mais de 50 ferramentas e *scripts* de código aberto que podem ser executadas automaticamente, para reconhecimento, verificação de vulnerabilidades e ataques de exploração. A documentação encontra-se em em andamento, mas a sua combinação de módulos a torna uma ferramenta poderosa.

5. **Ataque de Proxy Zed**

O Zed Attack Proxy (ZAP) da OWASP verifica as aplicações da Web em busca de vulnerabilidades. Atuando como um *proxy man-in-the-middle* entre o navegador de quem estiver a executar o teste e o aplicação web, o ZAP pode intercetar solicitações, modificar conteúdos e encaminhar pacotes. Ele oferece muitos recursos e complementos disponíveis gratuitamente. Existem versões disponíveis para cada sistema operativo, bem como para o *Docker*.

6. **Nikto2**

   Nikto2 executa verificações, em que pode identificar as falhas mais comuns encontradas em servidores web. Executado a partir da linha de comando, não possui interface gráfica. A documentação ainda não é particularmente detalhada, mas não é difícil de usar.

7. **OpenSCAP**

   O sistema OpenSCAP*Security Content Automation Protocol* é uma coleção de ferramentas de código aberto para implementar e aplicar o Protocolo de Automação de Conteúdo de Segurança (SCAP), um padrão dos EUA mantido pelo *National Institute of Standards and Technology* (NIST) que se concentra no monitorização contínuo, gestão de vulnerabilidade e conformidade com políticas de segurança. As ferramentas oferecem configuração automatizada, verificação de vulnerabilidades, patches e avaliação contínua da infraestrutura para conformidade com a segurança. Cada ferramenta é acompanhada por documentação e orientação abrangentes.

8. **SQLmap**

   A injeção de SQL é um vetor de ataque comum contra aplicações Web orientadas a dados, uma ferramenta como o sqlmap, pode automatizar o processo de deteção e exploração de falhas de injeção de SQL. O Sqlmap pode ser executado em sistemas Windows e Linux/Unix e possui exemplos úteis em sua extensa documentação. Oferece suporte a vários tipos de base de dados e inclui recursos de teste de intrusão,

como quebra de password, escalonamento de privilégios de utilizador e execução arbitrária de comandos.

9. **Scapy**

   Scapy é um programa de criação de pacotes que possui uma documentação particularmente boa.É necessário conhecimento profundo das estruturas de pacotes de protocolo e camadas de rede para aproveitar ao máximo a ferramenta. Essa ferramenta pode simular ou descodificar um grande número de pacotes de protocolo e pode lidar facilmente com tarefas como varredura, rastreio da rota de um pacote, sondagem, testes de unidade, ataques e descoberta de rede.

10. **CrackStation** CrackStation é um projeto de segurança desenvolvido pela Defuse Security. O seu objetivo é aumentar a consciencialização sobre o armazenamento inseguro de senhas em aplicações Web e fornecer orientações para melhorias nos sistemas de autenticação.

    Existem alguns software para descobrir senhas gratuitos, mas o CrackStation é um dos mais rápidos, pois usa tabelas de pesquisa que consistem em mais de 15 bilhões de entradas retiradas de vários recursos online.

11. **Aircrack-ng**

    Aircrack- ng é um conjunto completo de ferramentas para avaliação de segurança, específico para *Wireless Fidelity* (Wifi). Entre as capacidades do Aircrack-ng estão: Monitorização: Captura de pacotes e exportação de dados para arquivos de texto para posterior processamento por ferramentas de terceiros. Ataque: Ataques de repetição, quebras de autenticação, pontos de acesso falsos e outros via injeção de pacotes. Teste: Verificação de placas Wifi e recursos de drivers (captura e injeção). Cracking: *Wired Equivalent Privacy* (WEP) e *Wi-Fi Protected Access* (WPA) *Phase-shift keying* (PSK) (WPA 1 e 2). Todas as ferramentas são executadas em linha de comandos, o que permite *scripts* pesados, mais muitas GUIs utilizam seus recursos. Ele foi pen-

sado para Linux, mas pode ser utilizado no Windows, macOS e outros sistemas operacionais.

12. **Metasploit**

    O Metaesploit é uma ferramenta muito poderosa que pode ser utilizada para investigar vulnerabilidades em redes e servidores. Por ser um software de código aberto, pode ser personalizada e utilizada com a maioria dos sistemas operativos. Pode ser utilizado em uma rede para identificar pontos fracos, bem como a reconhecimento de ameaças, uma vez que as falhas são identificadas e documentadas, as informações podem ser utilizadas para a resolução dos problemas encontrados.

A tabela 2.1 e 2.2 apresentam um comparativo destas ferramentas, que pode auxiliar na escolha do software mais indicado para cada situação.

Tabela 2.1: Comparativo das Ferramentas de Testes de Segurança e Automatização

Ferramenta	**Escopo e Finalidade**	**Facilidade de Uso**	**Suporte e Comunidade**
Nmap	Ideal para reconhecimento de redes	Curva de aprendizado	Comunidade ativa
Wireshark	Analisador de protocolo de rede	Complexo para iniciantes	Comunidade sólida
Legion	Teste de intrusão de rede semi-automatizado	GUI amigável, documentação escassa	Não especificado
Jok3r	Combinação de várias ferramentas de código aberto	Documentação em andamento	Não especificado
ZAP	Verificação de vulnerabilidades em aplicações web	Interface intuitiva	Suportado pela comunidade OWASP

Tabela 2.2: Comparativo das Ferramentas de Testes de Segurança e Automatização (Cont.)

Ferramenta	Escopo e Finalidade	Facilidade de Uso	Suporte e Comunidade
Nikto2	Identificação de falhas comuns em servidores web	Execução por linha de comando	Comunidade ativa
OpenSCAP	Implementação e aplicação do SCAP	Configuração complexa	Não especificado
SQLmap	Detecção e exploração de falhas de injeção de SQL	Execução em sistemas Windows e Linux/Unix	Comunidade ativa
Scapy	Criação e decodificação de pacotes de rede	Requer conhecimento avançado	Boa documentação
CrackStation	Verificação de senhas inseguras em aplicações web	Simples de usar	Não especificado
Aircrack-ng	Conjunto de ferramentas para avaliação de segurança de redes Wi-Fi	Utilização em linha de comandos	Suporte em Linux, Windows e macOS
Metasploit	Ferramenta poderosa para investigar vulnerabilidades em redes e servidores	Personalizável e suportado em vários sistemas operativos	Suportado pela comunidade

Capítulo 3

Arquitetura do Sistema

A arquitetura do sistema descreve a estrutura e organização dos componentes do sistema, bem como as interações e fluxos de dados entre esses componentes.

3.1 Etapas para arquitetura de um Sistema

O planeamento da arquitetura de um sistema é um processo que envolve vários passos, o primeiro é a definição dos objetivos do sistema deve atingir. Isso pode garantir que o sistema seja desenvolvido de maneira eficaz e eficiente.

Após a definição dos objetivos do sistema segue a identificação dos requisitos necessários, é importante estabelecer uma arquitetura sólida para que o sistema seja robusto.

A seguir são definidas as tecnologias a serem utilizadas, como qual o sistema operativo, linguagem de programação, base de dados e tipo de hospedagem. Nessa fase também deve ser feito o esboço da estrutura de dados.

Concluídas as fases anteriores, tem inicio a fase de desenvolvimento da aplicação, nesta fase ainda pode ser alterada a arquitetura do sistema.

A figura 3.1 demonstra os passos para definição da arquitetura de um sistema.

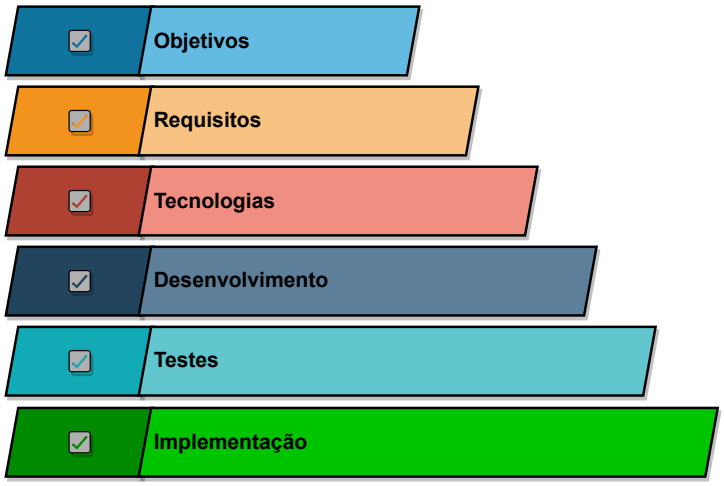


Figura 3.1: Arquitetura Sistema

3.1.1 Modelos de Arquiteturas

Existem várias abordagens e tipos de arquiteturas que podem ser utilizadas, dependendo das características e necessidades específicas do sistema. Alguns dos modelos mais utilizados incluem a arquitetura em camadas, a arquitetura cliente-servidor, a arquitetura orientada a serviços (SOA) e a baseada em micro-serviços.

- A arquitetura em camadas é um modelo em que o sistema é dividido em camadas distintas, com cada camada responsável por funções específicas. Isso permite uma separação clara de responsabilidades e facilita a manutenção e evolução do sistema.

- A arquitetura cliente-servidor é um modelo em que o sistema é dividido em duas partes principais, o cliente, que é responsável pela interface com o utilizador, e o servidor, que fornece os serviços e processamento de dados. Essa arquitetura é amplamente utilizada em sistemas distribuídos e baseados em rede.

- A arquitetura orientada a serviços (SOA) é um modelo em que o sistema é projetado como um conjunto de serviços independentes que se

comunicam por meio de interfaces bem definidas. Essa abordagem promove a reutilização de componentes e permite uma maior flexibilidade e escalabilidade do sistema.

- A arquitetura baseada em micro-serviços é uma evolução da arquitetura orientada a serviços, em que o sistema é dividido em pequenos serviços autónomos, cada um executando uma função específica. Essa abordagem facilita a manutenção e o desenvolvimento ágil do sistema, permitindo que cada micro-serviço seja desenvolvido, implantado e escalado de forma independente.

Além da escolha do modelo de arquitetura, é importante considerar outros aspetos do sistema, como a escolha das tecnologias e plataformas a serem utilizadas, o gestão de dados e a segurança. A arquitetura deve ser projetada de forma a garantir a eficiência, escalabilidade, confiança e segurança do sistema como um todo.

Em seguida, é importante identificar todos os requisitos necessários para atingir esses objetivos. Isso pode incluir requisitos técnicos, requisitos de recursos humanos, requisitos financeiros e outros. A identificação cuidadosa dos requisitos pode garantir que o sistema seja implementado de maneira adequada e que as necessidades sejam atendidas. Nesta fase uma análise *Strengths, Weaknesses, Opportunities and Threats* (SWOT) é fundamental.

3.1.2 Análise SWOT

Análise SWOT é uma ferramenta de gestão estratégica utilizada para avaliar os pontos fortes (*Strengths*), pontos fracos (*Weaknesses*), oportunidades (*Opportunities*) e ameaças (*Threats*) de uma organização, produto ou projeto. Essa análise permite identificar os fatores internos e externos que podem afetar o desempenho de um projeto. A análise SWOT é uma ferramenta simples, que pode ajudar na tomada de decisões, a estar preparado para enfrentar os desafios e aproveitar as oportunidades que possam surgir.

- Pontos Fortes(*Strengths*): Isso pode incluir recursos avançados de segurança, interface amigável, suporte técnico eficiente, historial de de-

sempenho confiável, entre outros. A identificação dos pontos fortes é essencial, para que se possa avaliar qual a que melhor atende necessidades de segurança, desempenho e fiabilidade.

- Pontos Fracos(*Weaknesses*): Considerando as possíveis fraquezas que o projeto possa apresentar, como: falta de recursos avançados, dificuldade de aprendizado e utilização, suporte técnico deficiente ou inexistente, historial de falhas de segurança, entre outros. É fundamental entender as limitações e avaliar se são aceitáveis para a realização do projeto.

- Oportunidades (*Opportunities*): Nas oportunidades que cada projeto pode oferecer, estão incluídos: recursos inovadores, integração com outras ferramentas de segurança, atualizações frequentes de segurança, compatibilidade com plataformas a maioria das plataformas, entre outros. Reconhecer as oportunidades pode ajudar a escolher uma opção que tenha potencial de crescimento e possa se adaptar necessidades futuras.

- Ameaças (*Threats*): Tendo em conta as ameaças associadas a cada opção de sistema operacional e softwares a serem utilizados, devem ser identificados possíveis problemas de segurança e vulnerabilidades conhecidas, falta de suporte ou atualizações, incompatibilidade com outros sistemas e softwares. Identificar as ameaças, pode evitar escolhas que possam comprometer a segurança ou causar problemas de compatibilidade.

A realização da análise SWOT para a escolha de um sistema operacional e software, pode ser determinante para uma decisão informada, considerando todos os aspetos relevantes para necessidades de segurança e funcionamento do projeto.

A seguir é apresentada na figura 3.2, a análise SWOT, para fundamentar a escolha do Ubuntu Server e Nmap na implementação do projeto Scandata.

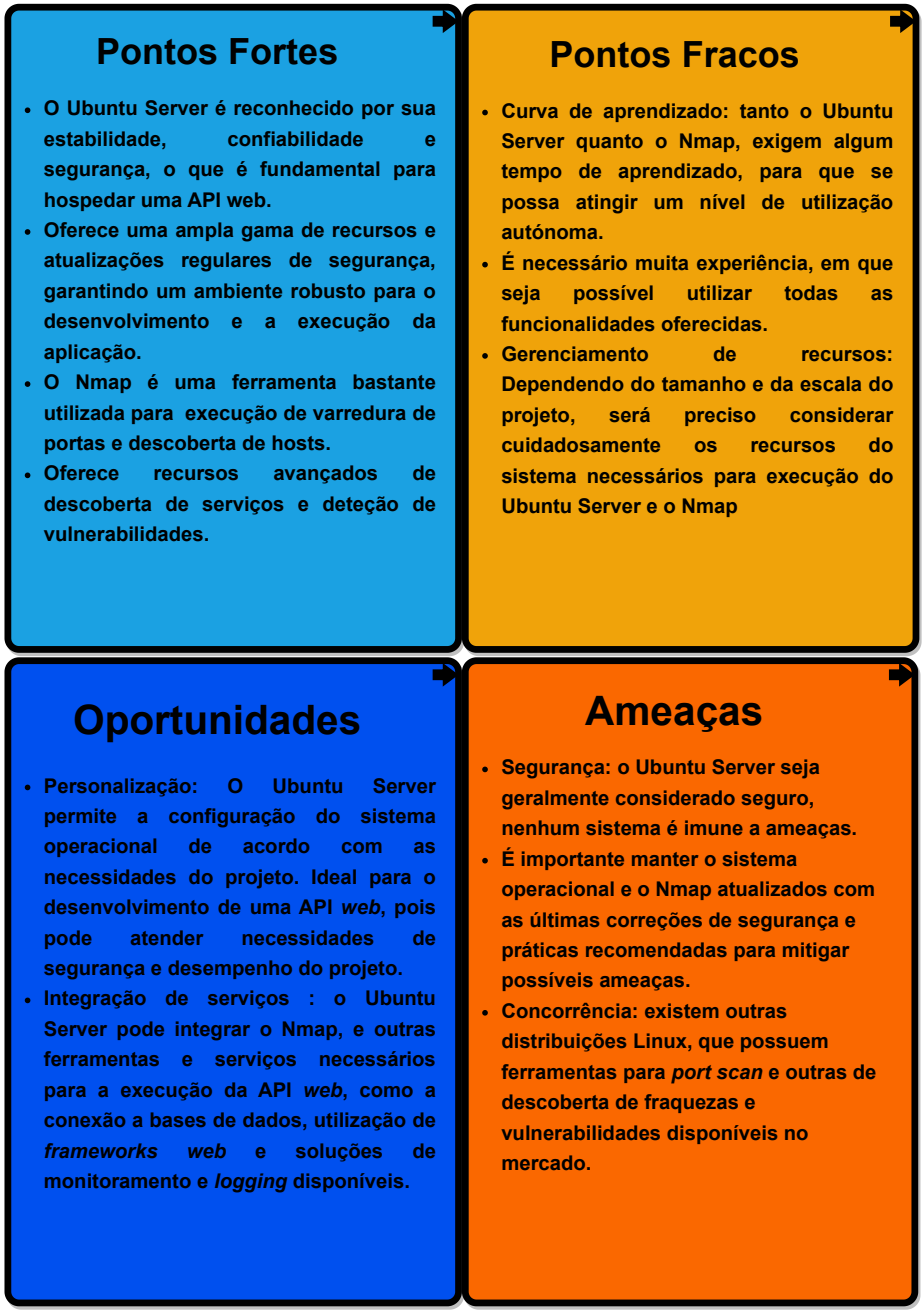


Figura 3.2: Análise SWOT

3.2 Arquitetura da API Scandata

A *Application Programming Interface* (API) Scandata foi desenvolvida de raiz, no contexto do Projeto de Estágio Curricular, com o objetivo de conclusão da Licenciatura em Engenharia Informática no Instituto Superior de Engenharia de Coimbra. Entre uma série de propostas de grande interesse, o projeto **"Cibersegurança – Automatização de testes"** apresentado pela empresa G9Telecom, foi o qual me despertou maior interesse.

A Scandata, tem como principal funcionalidade, a execução de testes automatizados, em sua grande maioria a servidores, denominados como clientes, que devem ser adicionados previamente para a execução de testes. Estes testes são análises do tipo*port scan*, em que a API utiliza recursos externos como Nmap, onde são recolhidas as informações sobre os serviços encontrados, bem como seus estados e outras informações relevantes, a seguir os dados são armazenados em base de dados, para que possam ser utilizados em comparações com testes subsequentes.

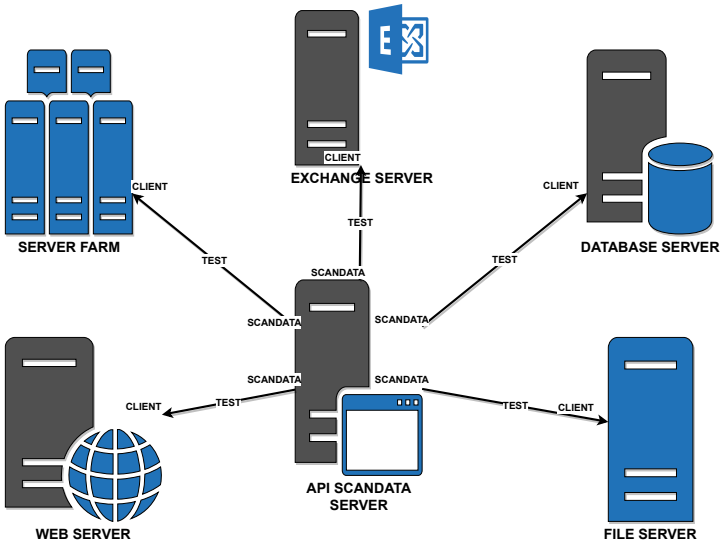


Figura 3.3: Arquitetura Scandata.

3.2.1 Ambiente da API

Uma das etapas mais importantes, é escolha dos componentes do ambiente para o desenvolvimento de uma API, foram considerados parâmetros como suporte contínuo, fiabilidade e curva de aprendizagem, mas o primeiro parâmetro definido, foi a escolha apenas de softwares de código aberto.

A seguir são apresentados os softwares escolhidos, bem como algumas das razões dessas escolhas.

3.2.2 Ambiente de Desenvolvimento

- Como *host* foi criada uma máquina virtual com sistema base Linux Debian, onde foi instalada e configurada uma distribuição Ubuntu Server 22.04. Como servidor, foi utilizado o Apache versão 2.4.52, um servidor muito utilizado para aplicações web. Para o desenvolvimento da API, foi necessário a instalação de alguns pacotes de aplicações Linux, como net-tools e telnet e o cron, um serviço que é executado no arranque do sistema, que permite agendar a execução de comandos e processos por tempo indeterminado. Bem como a instalação de softwares de terceiros como Nmap 7.91, PHP 8.2. e Python 3. Foi disponibilizado uma imagem da máquina virtual com toda configuração disponível em Scandata.ova.

- A API foi desenvolvida em PHP versão (8.2). O PHP (*Hypertext Preprocessor*) é uma linguagem de programação de código aberto, que ainda é muito utilizada para o desenvolvimento web, possui uma curva de aprendizagem relativamente reduzida em comparação com outras linguagens e oferece vasta documentação e suporte contínuo. Também possibilita fácil integração com sistema Ubuntu e Mysql.

- O tipo de arquitetura escolhida, foi a cliente - servidor. Esse tipo de arquitetura, é muito utilizado para hospedagem de aplicações WEB. Possui um historial de fiabilidade reconhecido, e uma implementação relativamente simples. O seu funcionamento é baseado em pedidos

feitos ao servidor e respostas devolvidas aos clientes, de modo que essa abordagem, mostrou-se como a mais indicada a realização do projeto.

- Na escolha do software de gestão de base de dados, o Mysql foi a opção mais indicada, possui segurança e fiabilidade comprovadas e uma comunidade que oferece vasta documentação. Os principais pontos a serem levados em conta, foram a sua completa integração com o sistema Linux, possibilidade da instalação de extensões Mysql incorporadas ao PHP e por estar relativamente familiarizado com o Mysql.

3.2.3 Arquitetura da Base de Dados

Para armazenar as informações dos testes e resultados, a API utiliza um base de dados relacional desenvolvida em Mysql. A arquitetura da base de dados, foi desenhada de modo que possa garantir a integridade e a persistência dos dados armazenados.

Descrição Base de Dados

1. Tabela clientes: armazena informações sobre os clientes.
   - `clientes_id`: id do cliente, auto-increment primary-key.
   - `n_test`: número de testes que foi alvo.
   - `tgt_ip`: IP do cliente alvo.
   - `clientes_descri`: descrição do cliente.
   - date: data da criação do cliente.
2. Tabela perfis: armazena as informações sobre perfis de teste, como nome do ISP e *gateway*.
   - `prof_id`: id do perfil, auto-increment primary-key.
   - profile: nome do perfil.
   - isp: nome do ISP.
   - gate: *gateway* a utilizar no teste.

3. Tabela tests: essa tabela armazena os tipos de testes que podem ser executados aos clientes.

    - `tests_id`: id do tipo de teste, auto-increment primary-key.
    - *date*: data de criação.
    - `prof_id`: id do perfil a ser utilizado no teste, *foreign key* da tabela perfis.
    - *ports*: portos que devem ser analisadas nesse no teste.

4. Tabela *`schedule_tests`*: Quando os testes são agendados, estes ficam armazenados nesta tabela enquanto não forem executados.

    - `scheduled_id`: id do agendamento do teste, auto increment primary key.
    - `cliente_id`: id do cliente, *foreign key* da tabela clientes.
    - `test_id`: id do teste a ser realizado, *foreign key* da tabela tests.
    - `testes_id`: id da entrada na tabela onde serão armazenados os resultados dos testes, *foreign key* da tabela `test_results`.
    - *date*: define a data que o teste deve ser executado.

5. Tabela *services*: Tabela que armazena as informações recolhidas no primeiro teste executado a cada cliente.

    - `service_id`: id do serviço, auto-increment primary-key.
    - *date*: data da execução do teste.
    - `cliente_id`: id do cliente, *foreign key* da tabela clientes.
    - `test_id`: id do teste que foi executado, *foreign key* da tabela `tests`.
    - *service*: nome do serviço.
    - *port*: monitorização que o serviço está ativo.
    - *state*:estado do serviço.

- proto: protocolo (tcp
  udp).
- *banner*: mensagem padrão do serviço.

6. Tabela `test_result`: Essa tabela tem armazenadas as informações sobre os testes realizados.
   - `testes_id`: id do teste efetuado, auto-increment primary-key.
   - *date*: data da excução do teste.
   - `execution_time`: tempo de execução do teste.
   - `cliente_id`: id do cliente, *foreign key* da tabela clientes.
   - `test_id`: id do tipo de teste executado, *foreign key* da tabela tests.
7. Tabela `test_descri`: Armazena os detalhes dos testes.
   - id: id da descrição, auto-increment primary-key.
   - `testes_id`: id do teste, *foreign key* da tabela `test_result`.
   - descrição: descrição efetiva dos detalhes dos testes.
8. Tabela `state`: Nessa tabela são armazenadas os detalhes das alterações encontradas nos testes.
   - `state_id`: id da alteração, auto-increment primary-key.
   - *date*: data da alteração.
   - `service_id`: id do serviço, *foreign key* da tabela `services`.
   - *state*: estado do serviço.
   - `cliente_id`: id do cliente, *foreign key* da tabela clientes.
   - obs: campo para observações.

Para melhor perceção da arquitetura da base de dados, a figura 3.4 apresenta o Diagrama ER (Entidade e Relacionamento) da API, que serviu como base do seu planeamento.

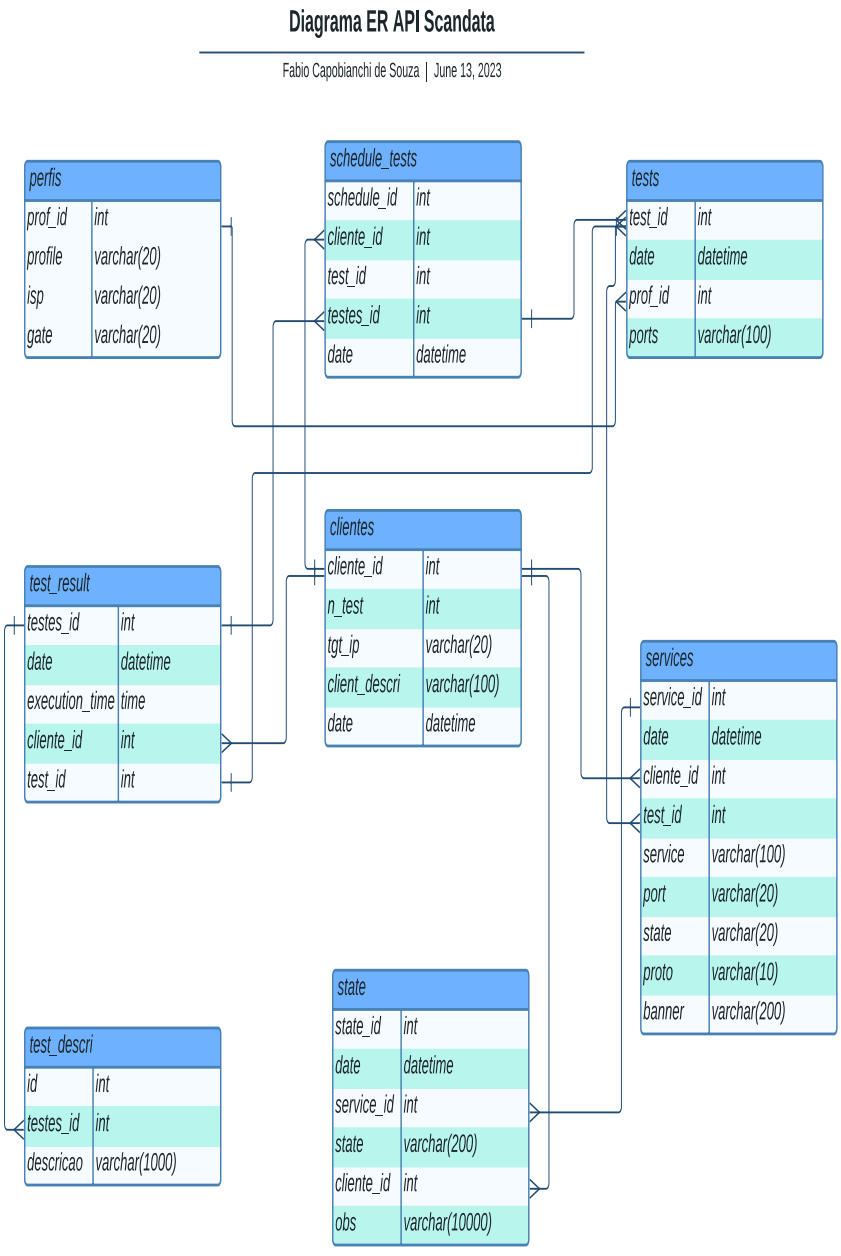


Figura 3.4: Diagrama ER.

3.2.4 Componentes da API

A seguir, uma breve apresentação das principais funcionalidades da API:

Gestão de Clientes: Este módulo permite a criação e gestão de clientes, incluindo informações como nome, número de testes executados, endereço de IP, descrição do cliente e data da criação. Quando um cliente é criado, no campo host pode ser inserido um endereço IP ou um nome, caso seja um nome, a API tenta fazer a resolução do nome para endereço IP, se for possível, ele cria o cliente caso contrário retorna um erro especificando o motivo, entretanto se no campo host for inserido um IP, a criação do cliente só terá sucesso se a máquina que tem esse endereço atribuído, estiver online e acessível através de uma ligação *Internet Control Message Protocol* (ICMP) (Ping).

Gestão de Perfis: Permite a criação de perfis de testes bem como a sua gestão, inclui endereço de IP do gateway, data de criação e descrição do ISP. No perfil se encontram as definições referentes ao gateway a utilizar em determinado teste, se necessário a API insere uma entrada na tabela de encaminhamento, com o IP alvo do teste a apontar para o gateway definido no perfil.

Gestão de Testes: Neste módulo, é possível definir e

configurar os tipos de testes que serão executados nos clientes, permite definir se o teste será a todas as portos ou a um porto especifico e também define o perfil a ser utilizado no teste. Os tipos de testes devem ser definidos de modo a atender as necessidades de cada cliente.

Agendamento de Testes: Os testes podem ser agendados, ou executados de imediato. O agendamento é feito indicando o id do cliente, o id do teste e a data da realização.

No primeiro teste realizado em um cliente, o processo é relativamente mais simples, é executado um port scans ao cliente, recolhe informações como porto, serviço, estado, e protocolo, bem como executa tentativas de ligação por telnet em cada porto encontrada, com o estado aberta, e obtém a mensagem padrão do serviço. Por fim, todos resultados obtidos são armazenados em uma base de dados, para análise posterior.

Comparação de Resultados: A API evoca esta funcionalidade sempre que um teste é executado a um cliente existente, e que foi alvo do primeiro teste anteriormente.

Esta funcionalidade, é responsável por comparar os resultados do teste atual, com os resultados do primeiro teste. Esta função foi desenvolvida de modo que possa detetar qualquer alteração dos serviços, no estado das portos, nos tipos de protocolo e na mensagem padrão de cada serviço.

Caso sejam detetadas quaisquer alterações, são gerados alertas específicos, para que se possa analisar cada situação específica. A seguir, a função assinala os serviços alterados e quais alterações, então atualiza toda a informação na base de dados. Caso contrário assinala como inalterado e atualiza na base de dados.

3.2.5 Funcionamento da API

Esta secção foi reservada para uma exposição mais detalhada sobre o funcionamento da API. Antes da fase de execução de testes e comparação dos resultados, é necessário uma preparação do ambiente de execução, como criação de clientes, a definição de perfis e tipo de testes.

Acesso a API

Em contexto académico, a Scandata só pode ser acedida em uma rede local, com um endereço IP privado, a API é executada sobre o protocolo HTTP, a sua implementação é relativamente mais simples, pois não exige a criação de certificados *Secure Sockets Layer* (SSL).

O acesso a API Scandata é feito através do navegador, na barra de endereços deve ser inserido o endereço IP do servidor, o nome do ficheiro da página pública da API e o método referente a página inicial.

A página inicial apresenta uma interface gráfica simples, que foi desenvolvida de modo que tanto a utilização da API, como a apresentação de resultados, sejam mais amigáveis. Na figura 3.5 segue um exemplo de acesso à página inicial e no apêndice A é apresentada uma imagem global da API.

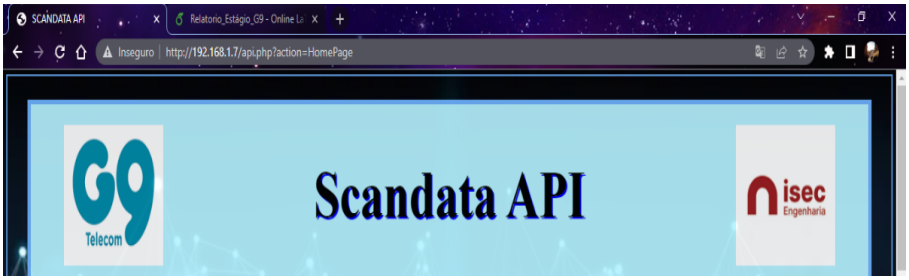


Figura 3.5: ScandataAccess

Criação de Clientes

Para efeitos de demonstração da execução de testes, foi criada uma nova máquina virtual Ubuntu server 22.04 para ser utilizada como alvo de testes, nesta máquina foram configurados alguns serviços, bem como a *firewall*, de modo a permitir que a API possa executar a consulta aos seus portos. Nesse mesmo servidor, foi desenvolvido um *script* em linguagem *Python*, que recebe como parâmetros o nome do serviço e o estado desejado *(on/off)*, após a execução retorna o nome e o novo estado do serviço.

Para a criação de um cliente, na página principal da API, na label *"Create Functions"*, no metodo createClient, deve ser inserido no campo host/IP, um nome ou um endereço IP válidos e a descrição. Para obter sucesso na criação

de um cliente, este deve estar *online* e alcançável. A a imagem 3.6 apresenta o processo de criação.

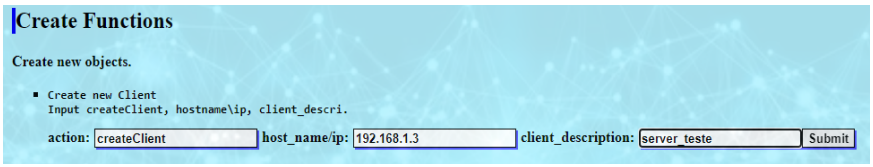


Figura 3.6: CreateClient

Criação de Perfis

A criação de perfis, foi desenvolvida a pensar na possibilidade da utilização de múltiplos ISPs (*Internet Service Provider*). No início da execução de um teste, é definido o gateway a utilizar, a API insere uma entrada na tabela de encaminhamento com o IP do alvo do teste e o *gateway* referente ao perfil, no término do teste, essa mesma entrada é removida. Como é apresentado na figura 3.7.

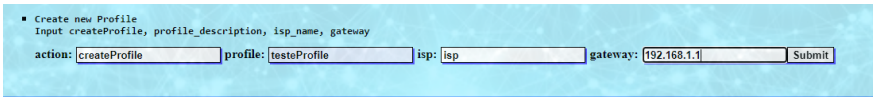


Figura 3.7: CreateProfile

Criação de Testes

Essa funcionalidade, define os parâmetros a serem utilizados na execução do teste, na criação de um tipo de teste, é necessário a escolha de um perfil criado previamente e a opção referente aos portos a serem verificadas. Existem

dois tipos de teste, um que verifica todos os portos e serviços que forem identificados, e uma segunda opção que verifica apenas um porto. Este último parâmetro indica a API quais métodos que o NMAP irá utilizar na verificação dos portos, na figura 3.8 pode ser observado um exemplo da criação de um teste.

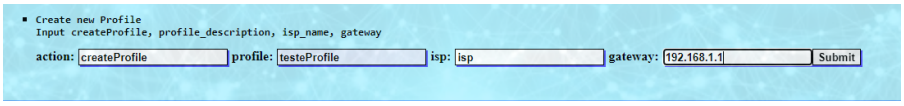


Figura 3.8: CreateTeste

Agendamento de Testes

Essa função é responsável por toda lógica da execução de um teste. A sua utilização é simples, devem ser inserido o id do cliente alvo, o id do tipo de teste, no caso de execução imediata a palavra *"now"*, ou para o agendamento futuro, a data no formato indicado. A seguir a esses passos, a informação é armazenados em uma base de dados, que é monitorada por um *script* em php, executado através de uma crontab do Cron no início de cada minuto. Sempre que o script encontra um teste com o horário do seu agendamento inferior a a data e hora atual, envia os dados do teste para API e a seguir remove a entrada desse teste na tabela de agendamento. O processo de agendamento é apresentado na figura 3.9.

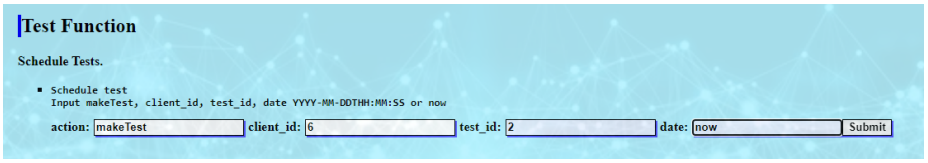


Figura 3.9: agendamentoTeste

Funções de Consulta Básicas

A API possui quatro funções de consulta, que apresentam toda informação armazenada, uma para cada método de criação. Essas funções são relativamente simples, mas facilitam na criação do agendamento de um teste, bem como na verificação dos objetos criados. Abaixo são apresentadas as funções e o separador de acesso pode ser observado na figura 3.10.

- showClients: retorna os dados de todos os clientes.
- showProfiles: apresenta todos os perfis criados.
- showTests: essa função devolve todos os tipos de testes criados.
- showScheduled: mostra todos os testes agendados que ainda não foram executados.

Figura 3.10: showFunctions

Funções de Consulta Avançada

Essas funções beneficiam-se das mais valias que uma base de dados relacional oferece, podendo assim executar consultas mais elaboradas e obter dados de múltiplas tabelas utilizando apenas uma única identificação ("ID").

- getClient: apresenta os dados relativos a um cliente, o número e o id dos testes que foi alvo.

- getInfoTest: utilizando o "ID"de um dos testes realizados, devolve todos os dados sobre o teste, e uma descrição detalhada sobre o estado dos serviços.
- getServices: essa função retorna todos os serviços que o cliente possui.
- getState: com o id do cliente, apresenta as alterações e o horário em que foram detetadas.

Na figura 3.11 é apresentado o separador com as funções de consulta avançada.

Figura 3.11: getFunctions

Eliminação de objetos

Esses métodos formam um grupo com quatro funções, que fazem a eliminação dos clientes, perfis, tipo de testes e agendamentos. Estas funções se encontram no separador *"Delete Functions* apresentado na figura 3.12, e utilizam o id dos objetos para sua eliminação.

Figura 3.12: deleteFunctions

Capítulo 4

Testes e Comparação de Resultados

Esse capítulo foi reservado para a demonstração da execução de testes e comparação de resultados. Bem como a descrição de todo o processo.

4.1 Ambiente de Testes

Para essa apresentação, como foi referido no capítulo anterior, irão ser utilizados dois servidores Ubuntu, instalados e configurados em máquinas virtuais, o primeiro com a instalação da *Application Programming Interface* (API) SCANDATA com endereço IP 192.168.1.9 e a segunda com 192.168.1.8, configurada com serviços ativos, e com possibilidade de mudança de estado dos mesmos, através da utilização de um script instalado nessa máquina.

Nesta fase serão realizados testes ao mesmo servidor, o primeiro com todos os serviços ativos, e na realização do segundo e terceiro testes, alguns serviços irão ser desativados, de modo a ser possível demonstrar que a API é capaz de detetar as alterações efetuadas.

4.2 Preparação

O primeiro passo a para a demonstração, é a criação de um cliente. Para esse efeito irá ser utilizado o servidor de testes, o nome utilizado foi serverTestes e o IP já definido.

O segundo é a criação de um perfil para os testes, foi escolhido como nome do perfil: "testeProfile", nome do isp: "interna"e gateway 192.168.1.1, o terceiro e último passo, é a definição de um tipo de teste, foi utilizado o id "1", que é referente ao perfil criado anteriormente, e o segundo parâmetro: "all", para que sejam verificadas todos os portos e serviços padrão. Após esses passos, estão criadas as condições para o agendamento da execução do primeiro teste ao cliente criado.

A figura 4.1 apresenta o processo de preparação.

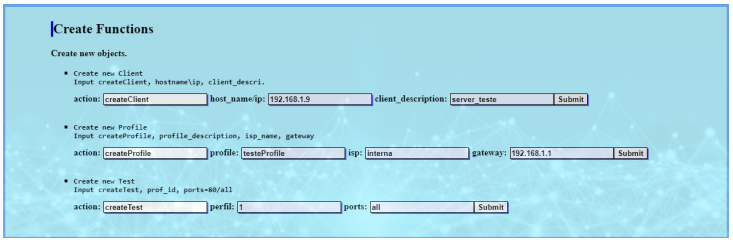


Figura 4.1: Test Prepare

4.3 Agendamento do Teste

Para o agendamento da execução do primeiro teste, foram utilizados o id do cliente, nesse caso "cliente-id 2", o id do tipo de test "test-id 1", que define também o perfil a utilizar. O último parâmetro é a data da execução do teste, para que o teste seja executado de imediato, irá ser utilizado a palavra *"now"*, a API reconhece a instrução e insere no teste a data e hora atual. A figura 4.2 demonstra o procedimento.

O agendamento de um teste para ser bem sucedido, passa por uma série de etapas de verificações e configurações. A primeira é a definição da *Time-Zone* como *('Europe/Lisbon')* para que a hora do agendamento esteja sempre

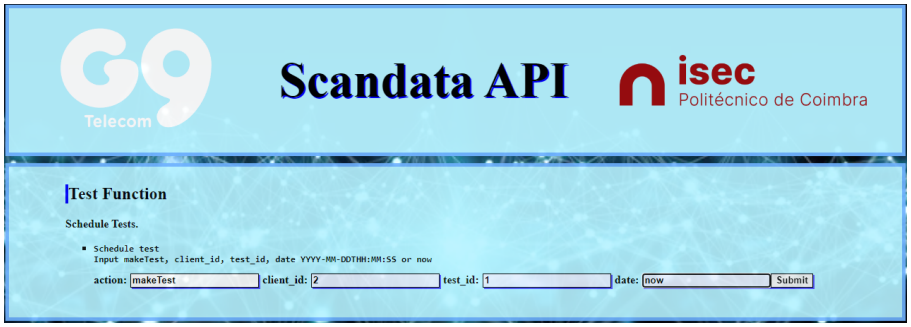


Figura 4.2: Make Test

atualizada, a seguir verifica se o cliente e o tipo de teste existem na base de dados, então executa uma verificação do formato da data e se é superior a data e hora atuais.

Quando o agendamento é concluído com sucesso, a API retorna uma mensagem a confirmar, que inclui o código "202"e a descrição *"Created"*, que indica que a operação foi executada sem erros, e o número do "id"do teste, o qual deve ser posteriormente utilizado na consulta dos resultados. Após a confirmação, no dia e hora agendadas, é iniciada a execução do teste.

A confirmação que o teste foi criado com sucesso, é apresentada na figura 4.3.

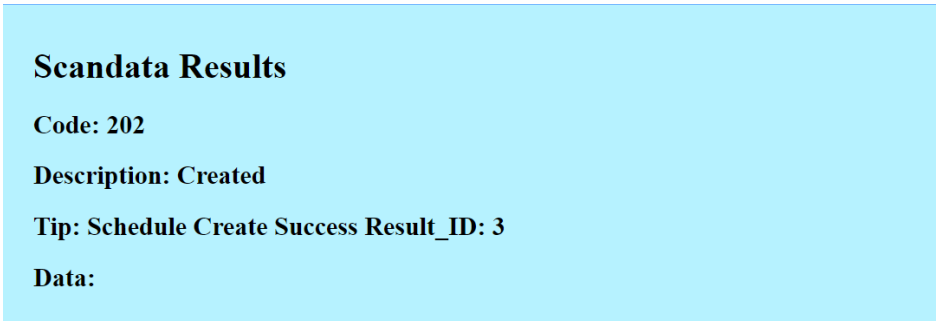


Figura 4.3: test-comfirm

4.4 Execução dos Testes

Sempre que um teste é iniciado, a função *scheduleTest* é chamada. Esta função é responsável pela gestão do fluxo de um teste, passando por várias fases, em que só é possível avançar para próxima fase se a anteriores foram executadas com sucesso, caso contrário, o teste não irá ser concluído.

Em primeiro, com recurso ás funções auxiliares a função *scheduleTest* obtém o endereço IP do *gateway* a utilizar, o ip do cliente alvo e o tipo de teste a ser executado. A seguir a função insere uma entrada na tabela de encaminhamento com o IP do cliente e o *gateway* a utilizar no teste. Então após essas fases ultrapassadas com sucesso, é chamada a função testScan, a qual é responsável pela maior parte da lógica do teste.

No contexto da função testScan, que o teste propriamente dito é executado, esta recebe os parâmetros recolhidos pela *scheduleTest*, e inicia a sua execução. A primeira tarefa da função é enviar um comando para o sistema, através da função *shell_exec()* com os parâmetros e o endereço IP do cliente, para que o Nmap execute a verificação dos portos do alvo escolhido com apresentado na figura 4.4.

```
// Execute the nmap command with the IP address parameter to scan TCP/UDP services
$command = 'sudo nmap -sS -sU -O -T4 ' . $ip;
$resultado = shell_exec($command);
```

Figura 4.4: Nmap-comand

Como foi referido anteriormente, a API Scandata utiliza o NMAP para a verificação dos portos (*port_scan*), com os parâmetros -sS -sU -O -T4, a seguir é apresentada uma breve explicação sobre cada parâmetro utilizado.

- -sS: A utilização desse parâmetro indica o uso do scan TCP *Synchronize* (SYN). Essa técnica é utilizada para verificar quais os portos estão

abertas em um cliente alvo. O Nmap envia pacotes SYN para as portos de destino, e analisa as respostas para determinar se o porto está aberta, fechada ou filtrada por um firewall. Se receber um pacote SYN + *Acknowledgement* (ACK), o porto está aberto, se for um pacote *Reset* (RST) + ACK, o porto está fechado.

- -sU: Essa opção habilita o scan UDP. Nesse caso são enviados pacotes UDP para portos específicas no cliente alvo, e como no -SS analisa as respostas para determinar se o porto está aberta ou fechada. Os portos UDP são usadas por serviços que não exigem uma ligação TCP estabelecida, como o DNS e o SNMP. O porto é considerada aberta se receber um pacote de resposta ICMP ou uma resposta do serviço no porto.

- -O: Esse parâmetro possibilita a deteção de sistemas operacionais. O Nmap tentará determinar o sistema operacional do host de destino com base em características e respostas de pacotes recebidos. Isso pode ser útil para identificar o sistema operacional de um cliente alvo.

- -T4: Essa opção define o tempo de execução do Nmap. O valor "4"indica uma velocidade maior. O Nmap usa intervalos menores entre os envios dos pacotes, o que acelera o processo de verificação, mas pode tornar o teste mais intrusivo.

De modo a obter os resultados do *port scan*, foi criada uma expressão regular para que através da função `preg_match_all` foi possível converter os dados em um array (conjunto de dados) ordenado, facilitando assim a separação dos campos em variavéis para melhor manipulação. Essa separação acontece em um ciclo, que me simultâneo, para cada porto encontrada com estado "aberto", a função tenta uma ligação por *Telnet*, e obtém a mensagem padrão que o serviço envia como resposta.

Nesse caso como é o primeiro teste feito ao cliente, esses dados devem ser utilizados como termo de comparação em testes futuros. Então a função armazena a informação obtida sobre os serviços na tabela *services*, e atualiza os dados na tabela "`test_result`"que mantém os resultados dos testes, essa tabela quando consultada utilizando o id que foi retornado no momento do agendamento do teste, apresenta toda a informação sobre o mesmo. Na figura 4.5, demonstra uma apresentação parcial do resultados do primeiro teste.

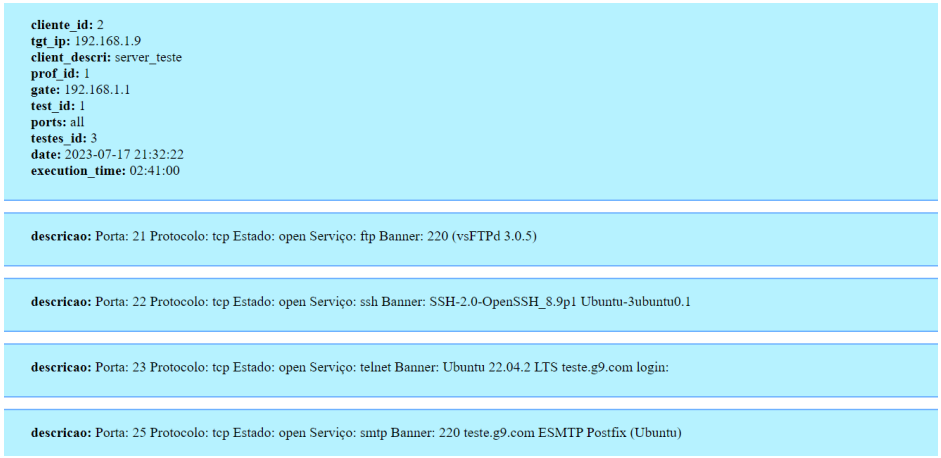


Figura 4.5: test-result

4.5 Comparação de Resultados

Nos testes seguintes a API altera seu fluxo de execução, após obter os dados da realização do *`port_scan`*, a Scandata inicia as fases de verificação dos serviços e comparação com o resultado inicial.

Na primeira fase a API consulta toda a informação sobre os serviços que foi armazenada na execução do primeiro teste, a seguir, compara cada serviço encontrado no resultado do *port scan*, com os dados armazenados na tabela *services*. Todos os serviços são verificados, e os resultados são atualizados na tabela "`test_result`", caso o serviço não apresente alteração, é assinalado com *"service ok"*, caso contrário o serviço é marcado com *"service altered"*, e os detalhes da alteração. A API é capaz de detetar alterações no estado do

serviço, no porto que o serviço utiliza, no protocolo, no estado e na mensagem padrão.

Nesta fase a API faz uma nova verificação, compara novamente os resultados, de modo a detetar a presença de um novo serviço, ou a falta de um existente. Nas duas situações as alterações são registadas na tabela dos resultados, e no caso de existir um novo serviço, este é adicionado ao s resultados que são utilizados como referência para os testes.

Na última etapa, a API atualiza toda a informação sobre o teste, inclusive o tempo de execução, então é removida a entrada na tabela de encaminhamento que define o *gateway* a ser utilizado no teste. O teste é dado como concluído, após a execução com sucesso, de todas as etapas, caso aconteça algum erro, a API interrompe a execução, e insere na tabela que armazena os resultados do teste, motivo da interrupção.

Para que fosse possível apresentar alterações, no resultado do segundo teste, os serviços *HTTP/HTPPs* foram desativados, como pode ser verificado nos resultados apresentados na figura 4.6.

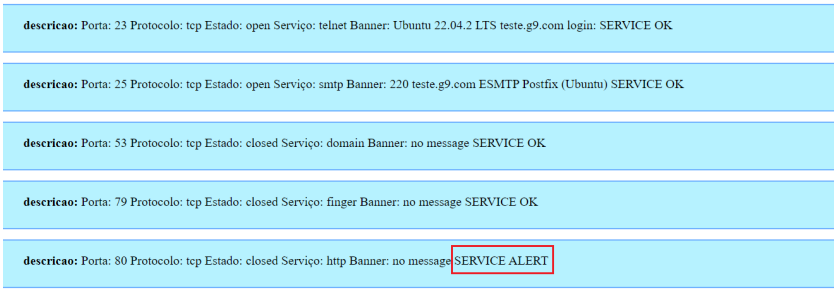


Figura 4.6: test-result2

De modo a confirmar o funcionamento da API, foi realizado um terceiro teste ao servidor, nesta execução foram introduzidas novas alterações, os serviços *HTTP/HTPPs* foram repostos e o serviço IMAP desativado. Após a conclusão com sucesso de todas as etapas de execução, o teste é concluído.

Como esperado, o resultado da execução do teste apresentado na figura 4.7, comprova a deteção das alterações.

O resultado de um teste, apresenta a informação obtida sobre os serviços. Como mencionado anteriormente, são adicionados alertas aos serviços, para

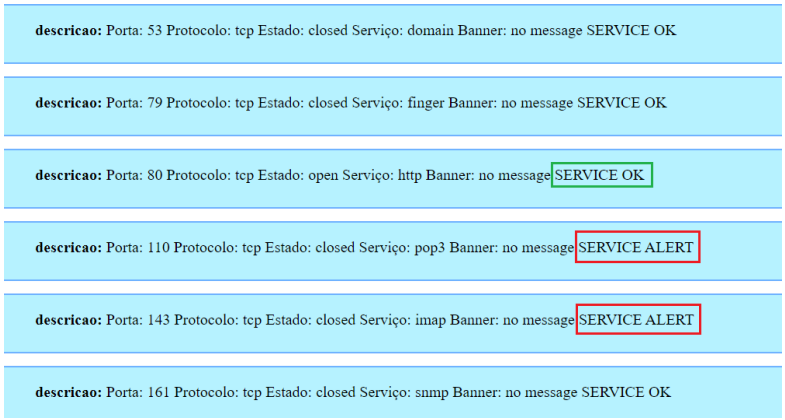


Figura 4.7: test-result3

que se torne mais intuitiva a identificação dos serviços alterados, essas alterações podem ser verificadas com a utilização do método de consulta *getState*, após inserir o "Id"do cliente, a função retorna a informação mais detalhada sobre as alterações encontradas nas verificações. Estas podem ser verificadas na figura 4.8.

Neste capítulo, foram realizados testes utilizando a API SCANDATA com o objetivo de verificar a eficiência e a precisão das funcionalidades implementadas. Os testes foram conduzidos em dois servidores Ubuntu, configurados em máquinas virtuais, permitindo a avaliação da API em um ambiente controlado. Foram agendados testes sequenciais no mesmo servidor para demonstrar a capacidade da API em detetar alterações nos serviços em tempo real.

A execução dos testes demonstrou a eficácia da API em utilizar o Nmap para verificar as portos no cliente alvo. Foram detetadas e registadas alterações nos serviços, como mudanças no estado, protocolo, porto e mensagem padrão. A API também foi capaz de identificar a falta de serviços existentes e a adição de novos serviços, fornecendo uma visão abrangente do estado da segurança do cliente.

Considerando o cenário dos testes e os resultados obtidos, podemos concluir que a API SCANDATA demonstrou ser uma ferramenta interessante e eficiente para monitorização dos serviços. Podendo proporcionar aos administradores de sistemas e equipes de segurança uma ferramenta de apoio na

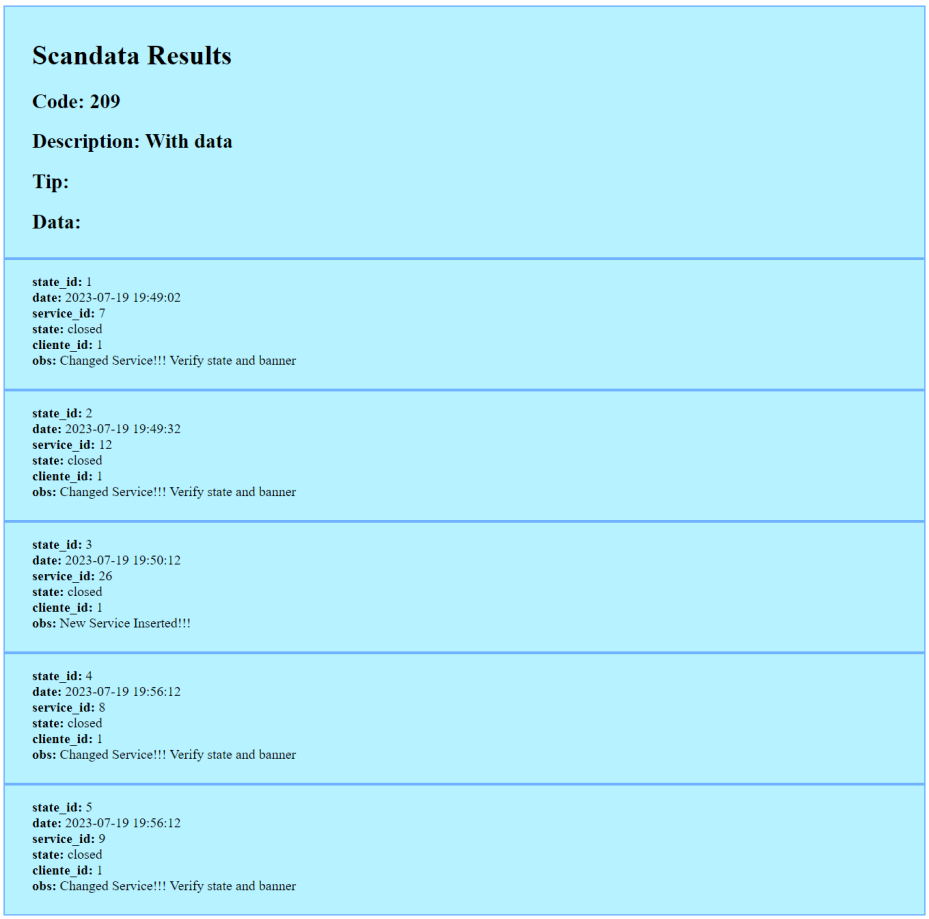


Figura 4.8: stateTest-details

manutenção e proteção de infraestruturas.

Capítulo 5

Conclusão

O desenvolvimento da *Application Programming Interface* (API) Scandata proporcionou uma solução eficiente para a automatização de testes de segurança em infraestruturas de sistemas. Ao longo deste relatório, foram apresentados conceitos de redes, cibersegurança, automatização de testes e distribuições Linux, mas o foco central, foram os principais aspetos da arquitetura, o funcionamento e a implementação da API, bem como os testes e resultados obtidos.

5.1 Objetivos Alcançados

No primeiro capítulo, foi apresentada uma introdução geral do projeto, apresentação da entidade acolhedora e o plano de execução do projeto.

O Capítulo segundo, foi discutido a importância da segurança de sistemas e os principais conceitos relacionados à cibersegurança. Destacamos a importância de ferramentas e práticas que possam detetar vulnerabilidades e ameaças em um ambiente de forma autónoma.

Para o desenvolvimento do terceiro capítulo, foi explicada com detalhes a arquitetura da API Scandata, destacando a escolha cuidadosa dos componentes do ambiente de desenvolvimento, como o sistema operacional Ubuntu Server, o servidor Apache e o banco de dados MySQL. Também foram explicados os principais módulos e funcionalidades da API, como a gestão de

clientes, perfis, testes e a comparação de resultados. A arquitetura em camadas e a abordagem cliente-servidor foram escolhidas para garantir uma estrutura sólida e escalável.

A seguir, é apresentada a execução de testes e a comparação de resultados. Foi possível demonstrar como a API é capaz de executar testes automatizados em clientes, utilizando o Nmap como ferramenta de verificação de portos. Através da comparação dos resultados dos testes, a API deteta e alerta sobre alterações nos serviços dos clientes, fornecendo informações valiosas para a manutenção e proteção de infraestruturas de sistemas.

Em conclusão, a API Scandata mostrou-se uma ferramenta poderosa e eficiente para a automatização de testes de segurança em infraestruturas. Através da sua implementação, é possível obter uma visão abrangente do estado e do nível da segurança dos clientes, detetar vulnerabilidades e alterações nos serviços em tempo real. A integração com o Nmap permite a execução de testes precisos e confiáveis, garantindo a eficácia da API.

Dessa forma, a API Scandata pode ser uma ferramenta valiosa para administradores de sistemas e equipes de segurança, proporcionando um meio eficaz de monitorizar e proteger infraestruturas de sistemas contra potenciais ameaças e vulnerabilidades. Com um desenvolvimento contínuo e melhorias adicionais, a API Scandata tem o potencial de se tornar uma ferramenta indispensável na área de cibersegurança, contribuindo para a criação de ambientes mais seguros e resilientes.

5.2 Trabalhos Futuros

Como trabalho futuro, sugere-se o desenvolvimento de novos métodos de testes, visando explorar as capacidades que o Nmap pode oferecer, seria de grande interesse, a aplicação de outras técnicas de verificação de vulnerabilidades. Também seria interessante a integração de outros softwares, como WireShark, e o Aircrack-ng que possui ferramentas especificas para redes Wifi.

Com essas alterações a API Scandata pode ser tornar uma ferramenta poderosa e indispensável na manutenção de redes e sistemas.

Apêndice A

Proposta de estágio

Departamento de Engenharia Informática e de Sistemas

Proposta de Projeto/Estágio

Ano Letivo de 2022/2023
2º Semestre

Cibersegurança – Automatização de testes

SUMÁRIO

Pretende-se automatizar testes de intrusão, previamente autorizados pelo destinatário, através da criação de uma ferramenta baseada numa máquina virtual, usando SO Linux e software *opensource* (a definir, por ex: Kali).

O trabalho terá uma primeira fase de identificação do software existente e da sua adequação aos requisitos dos testes a efetuar. Posteriormente, deverá ser configurado o sistema de modo a automatizar a bateria de testes de verificação de vulnerabilidades para serviços e redes predefinidas. O resultado dos testes deverá ser confrontado com um *template* de segurança. Os *templates* de segurança deverão ser definidos em função das redes de origem.

ESTAGIÁRIO (*indicar o destinatário da proposta se já estiver definido*)
Número de aluno:
Nome completo:

RAMO (*indicar o(s) ramo(s) em que se enquadra*)

- ☐ Desenvolvimento de Aplicações
- ☒ Redes e Administração de Sistemas
- ☒ Sistemas de Informação

ENTREVISTA/PROCESSO DE SELEÇÃO (*informar se o candidato indicado pelo DEIS-ISEC será submetido a uma entrevista ou outro tipo de processo de seleção antes da sua admissão efetiva*)

- ☐ Não
- ☐ Talvez Especificar:
- ☒ Sim Especificar: Pequena entrevista para avaliar nível de conhecimentos específicos.

1. ÂMBITO

A política de cibersegurança e as respetivas medidas de controlo são essenciais nas organizações atuais. Com este projeto pretende-se customizar uma pequena plataforma de teste automatizados de baixo custo que permita às PME controlarem a sua política definida de segurança.

2. OBJECTIVOS

O presente estágio pretende atingir os seguintes objetivos genéricos:

- Análise de requisitos;
- Seleção, Instalação, Configuração da Plataforma;
- Testes;

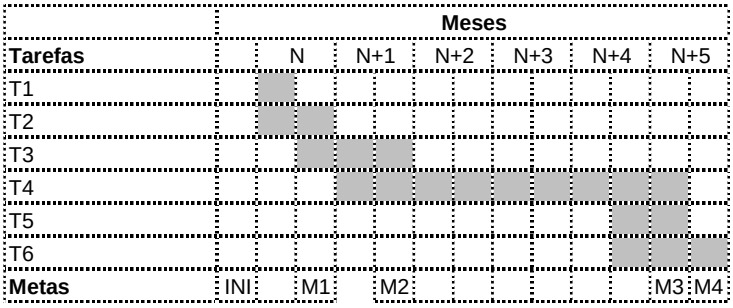
3. PROGRAMA DE TRABALHOS

O projecto/estágio consistirá nas seguintes actividades e respectivas tarefas:

- ***T1*** – *Integração* – Integração e conhecimento da empresa. Perspetiva técnica do serviço a desenvolver.
- ***T2*** – *Identificação de Necessidades e dos Planos de Teste* – Identificação de requisitos relativas ao projeto
- ***T3*** – *Análise* – Definição da análise do projeto.
- ***T4*** – *Desenvolvimento* – Desenvolvimento da plataforma tecnológica.
- ***T5*** – *Teste* – Execução do plano de testes.
- ***T6*** – *Documentação* – Elaboração da documentação e do relatório de estágio.

4. CALENDARIZAÇÃO DAS TAREFAS

O plano de escalonamento das tarefas é apresentado em seguida



INI		Início dos trabalhos
M1	(INI + 4 Semanas)	Tarefa T2 terminada
M2	(INI + 8 Semanas)	Tarefa T3 terminada
M3	(INI + 22 Semanas)	Tarefa T4 e T5 terminadas
M4	(INI + 24 Semanas)	Tarefa T6 terminada

5. LOCAL, HORÁRIO DE TRABALHO E CONDIÇÕES OFERECIDAS

O estágio decorrerá nas instalações da G9Telecom (https://www.g9telecom.pt/), na rua Bernardim Ribeiro, 76, em Coimbra, em horário laboral e dias úteis. Caso seja do interesse do estagiário, o local poderá ser em Viana do Castelo.

6. TECNOLOGIAS ENVOLVIDAS

Serão utilizados PHP, MySQL, assim como serão necessários conhecimentos de sistemas operativos LINUX e respectivas aplicações e servidores HTTP (Apache); São essenciais conhecimentos fortes de TCP/IP e XML/JSON.

7. METODOLOGIA

É necessário a elaboração de um documento de análise do projeto, assim como a elaboração da documentação sobre o código desenvolvido ou manuais de instalação e configuração dos serviços. O trabalho será acompanhado por um responsável técnico pela área, sendo promovidas reuniões de ponto de situação semanais entre o estagiário, o responsável técnico e o orientador da empresa, podendo ser chamados à reunião outras pessoas que contribuam para a melhoria do projeto.

8. ORIENTAÇÃO

Entidade de acolhimento:

João Perdigoto <joao.perdigoto@g9telecom.pt>

Administrador

DEIS-ISEC (*caso já esteja definido*):

Email do orientador <nome@isec.pt>

Categoria Profissional

Apêndice B

API Scandata

Figura B.1: API Scandata

Referências

[1] Andrei L. *Protocolos de Rede: O Que São, Como Funcionam e Tipos de Protocolos de Internet*. https://www.weblink.com.br/blog/tecnologia/conheca-os-principais-protocolos-de-internet/. Accessed: 2023-02-20. 2020.

[2] Borislav Kisprin. *The 5 penetration Testing Phases*. https://crashtest-security.com/penetration-test-steps/. Accessed: 2023-02-21. 2021.

[3] Diário da República n.º 108/2019, Série I de 2019-06-05. *Resolução do Conselho de Ministros n.º 92/2019, de 5 de junho*. https://dre.pt/dre/detalhe/resolucao-conselho-ministros/92-2019-122498962. Accessed: 2023-02-20. 2019.

[4] G9Telecom. *Quem Somos*. https://www.g9telecom.pt/Quem-Somos.html. Accessed: 2023-07-21. 2023.

[5] Joe Touch; Eliot Lear, Kumiko Ono, Wes Eddy, Brian Trammell, Jana Iyengar, Michael Scharf. *Service Name and Transport Protocol Port Number Registry*. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml. Accessed: 2023-02-20. 2017.

[6] Mazieiro, Carlos. *Introdução aos Serviços de Rede*. https://wiki.inf.ufpr.br/maziero/doku.php?id=espec:introducao. Accessed: 2023-02-17. 2021.