Faculdade de Ciências e Tecnologia

Departamento de Engenharia Informática

MECANISMO DE PROTEÇÃO CONTRA PHISHING

Rafael Filipe Pereira Pinho

Relatório de Estágio no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software orientado pelo Professor Doutor Fernando Boavida, pelo Engenheiro António Mendes e pelo Engenheiro Daniel Marcos, e apresentado à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Julho de 2019

UNIVERSIDADE D
COIMBRA

Resumo

Nos últimos anos, temos assistido a uma nova tendência no aumento da comunicação através de chat. As aplicações como WhatsApp, Facebook Messenger, Skype ou iMessage, conquistaram a preferência dos utilizadores. Com o aumento da utilização desta via de comunicação aumenta também o número de ataques a plataformas de messaging tanto para os servidores como para os clientes. Desde 2011 que os ataques de phishing nas plataformas móveis têm aumentado 85% a cada ano e cerca de 56% dos utilizadores são ludibriados e consequentemente atacados usando esta técnica. Os ataques aos aparelhos móveis estão cada vez mais elaborados e as empresas com aplicações de messaging têm cada vez mais responsabilidade na inibição deste tipo de ataques. Este trabalho consiste no desenvolvimento de um protótipo capaz de identificar e prevenir o envio de URLs maliciosos através da plataforma de messaging da WIT, com o intuito de evitar phishing e execução de código malicioso nos aparelhos dos clientes. Como principais fatores diferenciadores, destaca-se o facto de o protótipo desenvolvido utilizar machine learning, fornecer informação sobre os motivos que levaram à classificação de cada URL, possibilitar a configuração consoante o cliente final e expandir os URLs encurtados para classificar.

Palavras-Chave

Classificação, Machine learning, Mensagens maliciosas, Phishing, URLs maliciosos

Abstract

In recent years, we have witnessed a new trend in increasing communication through chat. applications such as WhatsApp, Facebook Messenger, Skype or iMessage, have won the users' preference. With the increase of the use of this communication route also increases the number of attacks on messaging platforms for both servers and clients. Since 2011 that phishing attacks on mobile platforms have increased 85% each year and about 56% of users are deceived and consequently attacked using this technique. Attacks on mobile devices are increasingly elaborate and companies with messaging applications are increasingly responsible for inhibiting this type of attacks. This work consists of the development of a prototype capable of identifying and preventing the sending of malicious URLs through the WIT messaging platform, in order to avoid phishing and execution of malicious code on clients' devices. As main differentiating factors, highlight the fact that the prototype developed use machine learning, provide information about the reasons that led to the classification of each URL, enable the configuration according to the final client and expand shortened URLs to classify.

Keywords

Classification, Machine learning, Malicious messages, Phishing, Malicious URLs

Agradecimentos

A realização deste trabalho contou com o precioso e generoso apoio e incentivo de instituições e pessoas às quais estou eternamente grato.

Começo por agradecer à WIT Software pela oportunidade de realizar este estágio em contexto empresarial, pelo excelente acolhimento, e ainda por todo o apoio técnico e logístico colocado à minha disposição.

Aos orientadores António Mendes e Daniel Marcos pela amizade, simpatia e profissionalismo com que me acolheram e acompanharam ao longo deste trabalho. O seu conhecimento, inteligência e os constantes desafios lançados ao longo do trabalho contribuíram para alcançar um trabalho diferenciador e relevante em termos científicos.

Ao professor Fernando Boavida que sempre acreditou em mim e me deu total liberdade na conceção e realização deste trabalho.

Finalmente, ao meu irmão e aos meus pais que sempre me apoiaram e motivaram ao longo deste percurso, principalmente nos momentos de maior dificuldade.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

A2P	Application to Person
AJAX	Asynchronous JavaScript and XML
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AUC	Area Under Curve
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DEI	Departamento de Engenharia Informática
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FTP	File Transfer Protocol
GI	Ganho de Informação
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
JDBC	Java Database Connectivity
JS	JavaScript
JSON	JavaScript Object Notation
JWT	JSON Web Token
MaaP	Messaging as a Platform
MMS	Multimedia Messaging Service
OEM	Original Equipment Manufacturer
P2P	Person to Person
RAM	Random Access Memory
RCS	Rich Communication Services
REST	Representational State Transfer
RGPD	Regulamento Geral de Proteção de Dados
ROC	Receiver Operating Characteristic
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSD	Solid State Drive
SVM	Support Vector Machines
TCP	Transmission Control Protocol
TLD	Top Level Domain
TN	True Negative
TP	True Positive
TPR	True Positive Rate
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WEKA	Waikato Environment for Knowledge Analysis
WWW	World Wide Web
XML	Extensible Markup Language

1. Introdução

Este relatório descreve todo o trabalho desenvolvido ao longo do ano letivo 2018/2019, no âmbito da unidade curricular de Estágio do Mestrado em Engenharia Informática da Universidade de Coimbra. O estágio insere-se na área de Engenharia de Software e teve lugar na WIT Software, uma empresa especializada no desenvolvimento de soluções avançadas para os operadores de telecomunicações móveis.

1.1. Enquadramento

O paradigma da comunicação mudou radicalmente com a massificação da comunicação por chat. As aplicações como WhatsApp, Facebook Messenger, Skype ou iMessage, conquistaram a preferência dos utilizadores. Com o aumento da utilização desta via de comunicação aumenta também o número de ataques a plataformas de messaging tanto para os servidores como para os clientes. Desde 2011 que os ataques de phishing nas plataformas móveis têm aumentado 85% a cada ano e cerca de 56% dos utilizadores são ludibriados e consequentemente atacados usando esta técnica [1]. O facto de mais de 52% de todo o tráfego da internet ser proveniente de dispositivos móveis, assim como a enorme quantidade de dados pessoais e empresariais presentes nestes dispositivos, contribuiu para que os atacantes tenham voltado a sua atenção para estes dispositivos e para a ampla variedade de aplicações de comunicação que usam, tornando estes utilizadores três vezes mais vulneráveis a phishing [2].

Perante o crescimento dos ataques aos dispositivos móveis as empresas com aplicações de messaging têm cada vez mais responsabilidade na inibição deste tipo de ataques. Neste contexto e com este trabalho, surge a preocupação da WIT em iniciar o estudo e desenvolvimento de um protótipo capaz de detetar e bloquear os ataques de phishing, para integrar na sua plataforma de messaging num cenário de tempo real.

1.2. Problema

Os ataques aos aparelhos móveis de hoje propagam-se rapidamente e estão cada vez mais elaborados. Problematicamente, as técnicas de proteção contra phishing tradicionais baseadas em listas negras de URLs conhecidos, já não são eficazes quando usadas isoladamente, porque os sites não são detetados nem registados em tempo real. O site de phishing está ativo por uma média de quatro horas e milhares de sites de phishing aparecem todos os dias [3], portanto, as técnicas de deteção precisam de funcionar mais rapidamente, a fim de evitar o roubo de dados e informações confidenciais dos utilizadores.

1.3. Objetivo

O objetivo deste estágio consiste no estudo e desenvolvimento de um protótipo capaz de identificar e bloquear qualquer tentativa de phishing e execução de código malicioso nos aparelhos móveis, e usará como base as comunicações que passam nas soluções de messaging da WIT. Este protótipo a desenvolver consistirá num sistema de back-end disponibilizado sob a forma de uma API capaz de analisar e bloquear mensagens maliciosas, com uma interface web para visualização e gestão de conteúdos bloqueados e permitidos. Para evitar ataques de dia zero e ataques futuros, irá recorrer a algoritmos de machine learning para criar modelos de

classificação sobre heurísticas capazes de detetar URLs maliciosos em tempo real com uma grande probabilidade.

Com base nestas informações, as principais etapas do estágio passam por:

- Estudo do conhecimento base, que inclui os conceitos gerais e a técnica de classificação de machine learning;
- Análise do estado da arte relacionado com o tema do trabalho;
- Levantamento e especificação dos requisitos do sistema;
- Desenho da arquitetura do sistema que suporte todos os requisitos especificados;
- Implementação do sistema definido;
- Testes de validação ao sistema desenvolvido.

1.4. Estrutura do relatório

Este relatório de estágio é composto por um total de dez capítulos: Introdução, Gestão do Projeto, Conhecimento Base, Estado da Arte, Especificação de Requisitos, Arquitetura, Implementação, Modelo de Classificação, Testes e Conclusão. No capítulo 1 de Introdução apresentou-se o enquadramento, o problema, o objetivo do projeto e a estrutura do relatório. No capítulo 2 de Gestão do Projeto apresenta-se a metodologia de desenvolvimento e as ferramentas usadas neste projeto, o planeamento do trabalho para o primeiro e para segundo semestre, e a análise de riscos associada ao projeto. No capítulo 3 de Conhecimento Base apresenta-se o estudo dos conceitos gerais de telecomunicações e segurança, e o estudo da técnica de classificação de machine learning, importantes para o desenvolvimento deste trabalho. No capítulo 4 de Estado da Arte apresenta-se o levantamento dos sistemas concorrentes, o estudo de abordagens analíticas para o classificador com base em trabalhos relacionados, e o levantamento e análise de APIs de deteção de URLs maliciosos e de ferramentas de machine learning. No capítulo 5 de Especificação de Requisitos apresenta-se os requisitos funcionais do sistema, os requisitos não funcionais e as restrições técnicas e de negócio. No capítulo 6 de Arquitetura apresenta-se a arquitetura proposta para o sistema através de um conjunto de vistas relevantes que demonstram como o sistema suporta os requisitos especificados. No capítulo 7 de Implementação apresenta-se todos os detalhes de implementação dos módulos dos principais componentes. No capítulo 8 do Modelo de Classificação apresenta-se todo o estudo, análise comparativa e experimentação realizada para o desenvolvimento do modelo de classificação. No capítulo 9 de Testes apresenta-se o plano e a execução dos testes aos requisitos funcionais e não funcionais, e os testes finais de aceitação. Finalmente, no capítulo 10 de Conclusão apresenta-se as conclusões sobre o trabalho realizado bem como as sugestões para trabalho futuro.

2. Gestão do Projeto

Este capítulo descreve a metodologia de desenvolvimento e as ferramentas de gestão do projeto usadas, o planeamento do trabalho para o primeiro e para o segundo semestre e, por fim, a análise de riscos associada ao projeto.

2.1. Metodologia e ferramentas

No desenvolvimento deste projeto optou-se por seguir uma metodologia ágil, baseada em scrum, à imagem do que é utilizado na WIT Software. Cada sprint teve a duração de uma semana e no final de cada um, foi realizada uma reunião com os orientadores da empresa, para avaliar o que foi feito durante o sprint, o que vai ser feito no próximo sprint, assim como discutir dúvidas e dificuldades. Para além disso, mensalmente foram realizadas reuniões com o orientador do DEI para dar a conhecer os avanços realizados e validar a documentação produzida.

Relativamente a ferramentas de gestão de projeto, utilizou-se o GanttProject [4] para fazer o cronograma e calendarizar as tarefas necessárias para concluir o trabalho, e o Redmine para a gestão de sprints. Para fazer o versionamento do relatório e do código utilizou-se o Git através de um repositório privado do GitLab.

2.2. Planeamento

2.2.1. Primeiro semestre

No primeiro semestre o trabalho teve como principal objetivo o estudo dos conceitos relacionados com o projeto e a elaboração da proposta de estágio. A Figura 1 apresenta o diagrama de Gantt com o planeamento e a calendarização das tarefas definidas para o primeiro semestre. Para estimar o esforço das várias tarefas recorreu-se à técnica de estimação por analogia [5], baseada no esforço da realização de tarefas semelhantes em projetos anteriores.

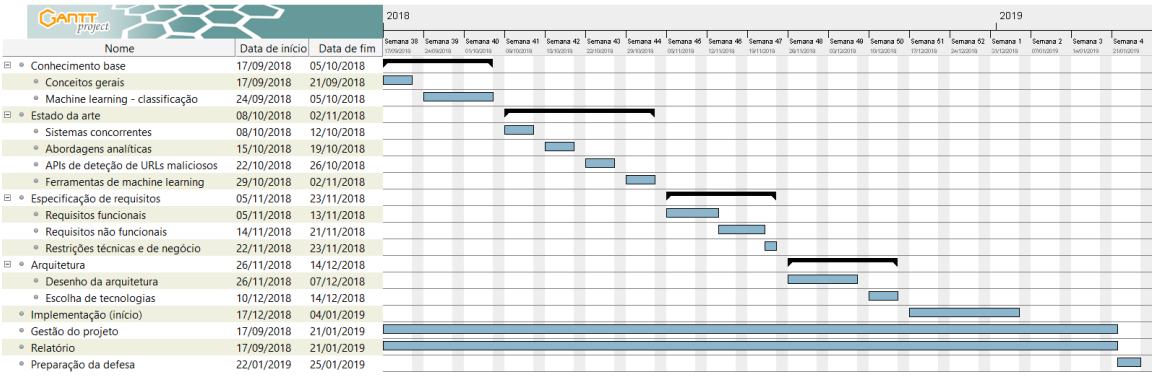


Figura 1 - Diagrama de Gantt do primeiro semestre

Descrição das tarefas do 1º semestre:

- **Conhecimento base**: consiste no estudo dos conceitos gerais associados ao projeto (RCS, phishing, componentes do URL), e no estudo da técnica de classificação de machine learning (pré-processamento de dados, algoritmos de classificação, métricas de avaliação de desempenho).
- **Estado da arte**: esta fase inicia-se após o conhecimento base e consiste no levantamento dos sistemas concorrentes, no estudo de abordagens analíticas para o

classificador com base em trabalhos relacionados, e no levantamento e análise de APIs de deteção de URLs maliciosos e de ferramentas de machine learning.

- **Especificação de requisitos**: esta fase inicia-se após o estudo do estado da arte e consiste em fazer o levantamento e especificação dos requisitos funcionais, dos requisitos não funcionais e das restrições técnicas e de negócio.
- **Arquitetura**: esta fase inicia-se após a especificação dos requisitos e consiste no desenho de uma arquitetura para o sistema que suporte todos os requisitos definidos.
- **Implementação**: esta fase inicia-se após o desenho da arquitetura e consiste no início da implementação do sistema proposto.
- **Gestão do projeto**: realiza-se ao longo do semestre e consiste nas atividades de gestão do projeto (planeamento, gestão de riscos).
- **Relatório**: realiza-se ao longo do semestre e consiste na documentação de todas as tarefas realizadas.
- **Preparação da defesa**: realiza-se após a conclusão da escrita do relatório e consiste na preparação da defesa intermédia.

2.2.2. Segundo semestre

No segundo semestre o trabalho teve como principal objetivo a implementação do sistema proposto, na proposta de estágio elaborada durante o primeiro semestre. A Figura 2 apresenta o diagrama de Gantt com o planeamento e a calendarização das tarefas definidas para o segundo semestre. Para estimar o esforço das várias tarefas recorreu-se também à técnica de estimação por analogia.

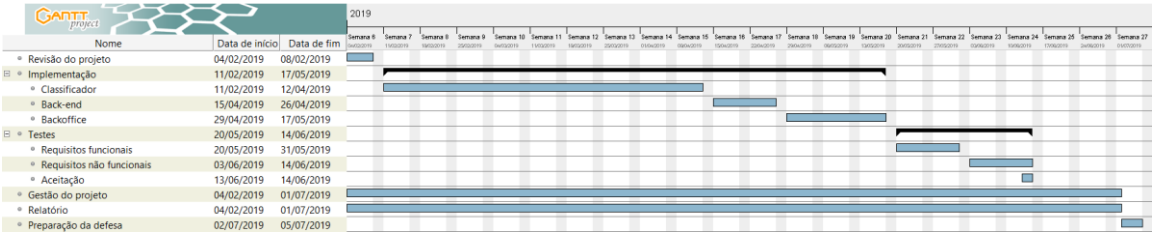


Figura 2 - Diagrama de Gantt do segundo semestre

Descrição das tarefas do 2º semestre:

- **Revisão do projeto**: consiste na revisão geral do projeto tendo em conta os comentários da defesa intermédia.
- **Implementação**: esta fase inicia-se após a revisão do projeto e consiste no desenvolvimento do sistema proposto, começando pelo desenvolvimento do modelo de classificação, seguido do back-end e, por fim, do backoffice de administração.
- **Testes**: esta fase inicia-se após a implementação do sistema e consiste na criação e execução de um plano de testes aos requisitos funcionais e não funcionais do sistema. Por fim, são realizados testes de aceitação ao sistema desenvolvido.
- **Gestão do projeto**: realiza-se ao longo do semestre e consiste nas atividades de gestão do projeto (planeamento, gestão de riscos).
- **Relatório**: realiza-se ao longo do semestre e consiste na documentação de todas as tarefas realizadas.
- **Preparação da defesa**: realiza-se após a conclusão da implementação do sistema e da escrita do relatório, e consiste na preparação da defesa final.

2.2.3. Desvios

Durante o primeiro semestre seguiu-se o planeamento inicialmente definido sem grandes dificuldades, não se tendo registado qualquer desvio. Sempre que ocorreu algum pequeno atraso no término de alguma tarefa, estes foram sempre compensados durante os fins de semana de forma a não atrasar o início das tarefas seguintes. Durante o segundo semestre foram registados ligeiros desvios em relação ao planeamento previamente definido, em particular na fase de implementação, como se pode observar pela análise comparativa da Figura 3, que apresenta o diagrama de Gantt com a execução real das tarefas do segundo semestre, e da Figura 2 acima que apresentou o diagrama de Gantt com o planeamento inicial.

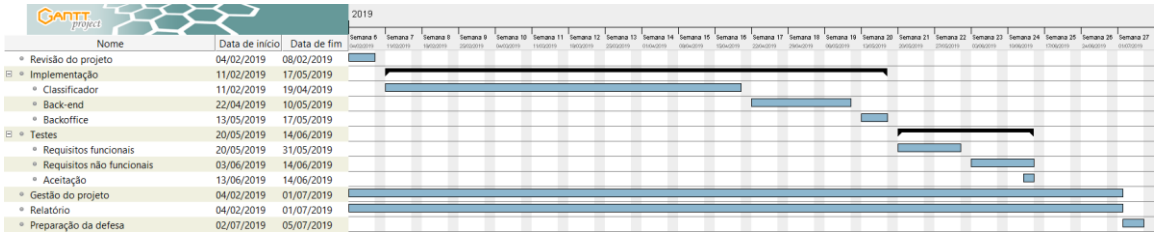


Figura 3 - Diagrama de Gantt final do segundo semestre

De notar que na fase de implementação, o tempo alocado para o desenvolvimento do backoffice foi reduzido em duas semanas, ou seja, passou de três para apenas uma semana. Estas duas semanas extras foram alocadas para o desenvolvimento do classificador, que passou de nove para dez semanas, e do back-end que passou de duas para três semanas. Este desvio deveu-se ao facto do backoffice já ter sido maioritariamente desenvolvido durante o primeiro semestre, tendo necessitado apenas de alguns retoques. Enquanto que no classificador a semana extra foi aproveitada para explorar novos atributos e experimentar novas abordagens de seleção de atributos mais relevantes, no back-end a semana extra foi particularmente útil uma vez que o tempo necessário para o desenvolvimento dos seus componentes foi superior ao planeado.

2.3. Riscos

Nesta secção realiza-se a identificação, análise e gestão dos riscos associados ao projeto. Este processo é bastante importante na gestão de um projeto porque permite monitorizar e minimizar o impacto dos riscos no projeto. A identificação dos riscos tem como base os fatores que podem colocar em causa o sucesso do projeto. Para tal começa-se por definir um limiar de sucesso para o projeto, que é uma fronteira entre o sucesso e o fracasso de um projeto, definido através de metas de qualidade, desempenho e orçamento. Para este projeto o limiar de sucesso é atingido se:

- Os requisitos funcionais de prioridade mais elevada estiverem implementados até à data de entrega final;
- O sistema satisfazer todos os requisitos não funcionais definidos e as restrições técnicas e de negócio;
- O projeto ficar terminado dentro do prazo previsto.

A análise dos riscos [5] é realizada tendo em conta o impacto do risco no projeto, a probabilidade de ocorrência do risco, o intervalo de tempo onde se prevê que o risco possa vir a ocorrer, e a consequência que resulta para o projeto e para a mitigação.

Os níveis definidos para o impacto de um risco no projeto foram:

- Mínimo: o limiar de sucesso pode ser atingindo sem grandes dificuldades;

- Médio: o limiar de sucesso pode ser atingido com algumas dificuldades e esforço adicional;
- Máximo: o limiar de sucesso não pode ser atingido.

Os níveis definidos para a probabilidade de um risco ocorrer no projeto foram:

- Baixa: probabilidade de ocorrência inferior a 30%;
- Média: probabilidade de ocorrência entre 30% e 70%;
- Alta: probabilidade de ocorrência superior a 70%.

Os níveis definidos para o intervalo de tempo de um risco vir a ocorrer no projeto foram:

- Curto prazo: o risco pode ocorrer na fase inicial do projeto;
- Médio prazo: o risco pode ocorrer na fase intermédia do projeto durante a implementação;
- Longo prazo: o risco pode ocorrer na fase final do projeto ou depois de concluído.

2.3.1. Identificação de riscos

De seguida apresenta-se a listagem dos riscos identificados associados ao projeto, com as consequências resultantes, o impacto no projeto, a probabilidade de ocorrência, o intervalo de tempo em que pode vir a ocorrer e o plano de mitigação a utilizar em caso de ocorrência.

ID: R1
Risco: conhecimento reduzido de machine learning
Consequência: põe em risco o sucesso do projeto
Impacto: máximo
Probabilidade: alta
Intervalo de tempo: médio prazo
Plano de mitigação: alocar tempo para estudar o que é o machine learning, assim como todas as etapas envolvidas no processo de desenvolvimento de um modelo de classificação

---

ID: R2
Risco: complexidade e tamanho do projeto
Consequência: poderá não ser possível durante o estágio obter um modelo com uma precisão de classificação razoável
Impacto: médio
Probabilidade: alta
Intervalo de tempo: médio prazo
Plano de mitigação: definir um limite de tempo para o estudo e para as experiências a realizar no desenvolvimento do modelo de classificação na tentativa de obter a melhor precisão possível

---

ID: R3
Risco: inexperiência no uso das tecnologias necessárias à implementação do sistema
Consequência: atraso na implementação do sistema
Impacto: médio
Probabilidade: média
Intervalo de tempo: médio prazo
Plano de mitigação: quando as tecnologias estiverem definidas, começar a fazer tutoriais e pequenos exemplos como forma de ganhar experiência

---

ID: R4
Risco: alterações na arquitetura do sistema
Consequência: atraso na implementação do sistema
Impacto: médio
Probabilidade: média
Intervalo de tempo: médio prazo
Plano de mitigação: estruturar o sistema de forma a permitir realizar alterações, sem que isso exija muito esforço adicional

2.3.2. Análise de riscos

Uma vez que já temos os riscos identificados e analisados, o próximo passo é analisar a exposição do projeto a esses riscos, tendo em conta o seu impacto e a sua probabilidade de ocorrência no projeto para posteriormente se priorizar esses riscos. A Tabela 1 apresenta a exposição do projeto aos riscos identificados.

Exposição	Baixa	Média	Alta	Crítica
Cor				

		Impacto		
		Mínimo	Médio	Máximo
Probabilidade	Alta		R2	R1
	Média		R3, R4	
	Baixa			

Tabela 1 - Exposição do projeto aos riscos

Após a análise da exposição do projeto aos riscos identificados, o próximo passo é priorizá-los. Esta tarefa é feita com base na exposição aos riscos da Tabela 1 e de acordo com o intervalo de tempo em que esses riscos podem vir a ocorrer no projeto. A Tabela 2 mostra a priorização de todos os riscos identificados de forma ordenada, do mais prioritário para o menos prioritário.

ID do risco	Exposição do risco	Intervalo de tempo
R1	Crítica	Médio prazo
R2	Alta	Médio prazo
R3	Média	Médio prazo
R4	Média	Médio prazo

Tabela 2 - Priorização dos riscos

Ao longo do projeto os riscos identificados foram monitorizados semanalmente. No primeiro semestre para evitar consequências no projeto mitigou-se a ocorrência do risco 1, conhecimento reduzido de machine learning, do risco 3, inexperiência no uso das tecnologias necessárias à implementação do sistema, e do risco 4, alterações na arquitetura do sistema. Para mitigar o risco 1, foi alocado tempo ao planeamento do primeiro semestre para o estudo da técnica de classificação de machine learning. Para mitigar o risco 3, quando as tecnologias foram definidas começou-se a fazer tutoriais e pequenos exemplos como forma de ganhar experiência. Para mitigar o risco 4, foram realizadas reuniões regulares com os orientadores da empresa em conjunto com o tutor técnico do lado do produto para validar a arquitetura produzida e, desta forma, garantir que não haveria mais alterações no decorrer do projeto.

Estes três riscos deixaram de representar um risco para o projeto no final do ano de 2018, o que comprova a eficácia dos planos de mitigação criados previamente para cada um destes riscos. Quando ao risco 2 da complexidade e tamanho do projeto, no segundo semestre este risco foi controlado através da realização de várias iterações de experimentação e desenvolvimento do modelo de classificação apenas durante o tempo planeado para o desenvolvimento deste componente, de forma a não pôr em causa a realização das restantes tarefas do projeto. Em suma, tanto a monitorização dos riscos como os planos de mitigação criados foram eficazes na mitigação dos riscos, uma vez que não impediram o projeto de alcançar todos os objetivos definidos.

3. Conhecimento Base

Este capítulo apresenta o estudo de conceitos gerais e o estudo da técnica de classificação de machine learning, importantes para o desenvolvimento deste trabalho.

3.1. Conceitos gerais

Dada a importância de compreender alguns conceitos de telecomunicações e segurança, assim como os componentes do URL que aparecem ao longo deste trabalho, esta secção serve para os apresentar e defini-los desde já. Ao nível de telecomunicações aborda o RCS (Rich Communication Services) e a sua plataforma de messaging. Ao nível de segurança aborda o que é o phishing, concluindo com o tipo de vulnerabilidades a que os utilizadores das plataformas de messaging estão sujeitos durante uma troca de mensagens. Por último descreve os diferentes componentes do URL.

3.1.1. RCS

O RCS [6] é a suite de comunicações da WIT para os operadores de telecomunicações que permite a utilização de funcionalidades avançadas durante as mensagens ou chamadas. Ao nível das mensagens permite substituir as mensagens de SMS tradicionais por mensagens de texto enriquecidas, permitindo que os sistemas de mensagens possuam recursos tais como, receber notificações de leitura, ver quando alguém está a responder a uma mensagem, iniciar conversas em grupo, e enviar mensagens com imagens, sons e vídeos. Ao nível das chamadas permite iniciar chamadas com informação adicional que será apresentada no ecrã do destinatário, mesmo antes de a chamada ser atendida, como o assunto da chamada, a indicação de importância, a localização e fotografia.

A plataforma de messaging do RCS que é o componente que mais importa para este estágio suporta dois tipos de messaging, o P2P (Person-to-Person) e o A2P (Application-to-Person) [7]. O messaging P2P é o processo de troca de mensagens entre dois telemóveis através de uma rede móvel. Um exemplo de comunicação P2P são as mensagens de texto. O messaging A2P é o processo pelo qual as mensagens são enviadas de uma aplicação de negócio, geralmente uma plataforma web, para um utilizador móvel. Exemplos populares de comunicação A2P são as notificações de transações, as confirmações de reservas online, os lembretes de viagem, os esforços de marketing e alertas críticos.

3.1.2. Phishing

O phishing é um tipo de ataque informático em que os hackers tentam obter por via da engenharia social dados confidenciais de utilizadores, como logins, passwords e dados de cartões multibanco. Tipicamente o phishing começa com o utilizador a receber uma mensagem eletrónica de alguém, que se tenta fazer passar por uma entidade credível, por exemplo um banco ou uma loja de compras online. Nessa mensagem, é pedido ao utilizador para clicar num link que o leva para um site falso, de aparência idêntica ao site original sendo a única diferença o URL, onde lhe é pedido que conceda permissão para algo. Inconscientemente o utilizador pensando estar a realizar um login numa página real e segura, está a fornecer os seus dados pessoais ao hacker. Nestes esquemas de phishing, por vezes são também usados URLs de código malicioso, que instalam software malicioso no computador do utilizador com o objetivo de roubar as suas informações confidenciais.

Em suma, pode-se concluir que as mensagens enviadas durante as comunicações de chat, podem conter links para dois tipos distintos de sites maliciosos, e que se pretende detetar no sistema a desenvolver [8]:

- **Sites de phishing** (engenharia social): sites que fingem ser legítimos para enganar os utilizadores, e levá-los a escrever os respetivos logins e passwords ou a partilhar outras informações confidenciais.
- **Sites de malware** (código malicioso): sites que contêm código que instala software malicioso nos computadores dos visitantes, quando o utilizador pensa estar a transferir software legítimo ou sem ter conhecimento dessa instalação. Depois, esse software instalado, é utilizado pelos hackers para capturar e transmitir as informações confidenciais do utilizador.

3.1.3. Componentes do URL

Um URL é um endereço para localizar um determinado recurso disponível na internet, ou seja, localiza o servidor que contém o recurso e especifica onde o recurso está colocado nesse servidor. Um URL pode ser composto por vários componentes diferentes (protocolo, TLD, hostname que contém o domínio e o subdomínio, path, parâmetro(s) e ancora), cada um indicando uma informação específica sobre o recurso. Para exemplificar estes componentes vai-se considerar o seguinte exemplo de URL [9]: https://blog.example.com/web-tutorials/addresses.html?userid=123456#section1

- **Protocolo**: https, que é o protocolo usado para transferir páginas web e a maioria dos outros recursos na internet. A maioria dos websites usa o protocolo HTTP (Hypertext Transfer Protocol) ou o HTTPS, a sua versão segura. No entanto, existem outros protocolos como o FTP (File Transfer Protocol) para transferência de ficheiros, e o SMTP (Simple Mail Transfer Protocol) para o envio de emails.
- **Top Level Domain**: .com, que indica o domínio de nível superior. O .com é geralmente usado para sites comerciais, no entanto existem outros TLDs como por exemplo, o .org usado por organizações sem fins lucrativos, o .gov usado pelo governo dos Estados Unidos, o .edu usado por universidades, etc.
- **Hostname**: blog.example.com, que consiste em um domínio e um subdomínio. O nome do domínio registado (example.com) juntamente com o TLD que identifica o servidor na internet que contém o recurso, e o subdomínio (blog) geralmente usado para especificar subsites dentro de um domínio. O subdomínio mais comum é o www (World Wide Web), usado por alguns domínios para indicar que o conteúdo é acessível publicamente.
- **Path**: /web-tutorials/addresses.html, que indica ao servidor o caminho para o recurso que deve encontrar. Tem duas partes: o diretório (/web-tutorials/) que é a pasta de ficheiros no servidor, e o nome do recurso (addresses.html) que pode ser um nome de uma página ou de um ficheiro.
- **Parâmetro(s)**: ?userid=123456, que são usados para codificar dados específicos no URL para uso pelo servidor. O ponto de interrogação (?) indica o início de uma lista de parâmetros, depois cada parâmetro tem um nome (userid) e um valor (123456), separados por um sinal de igual (=). Quando há vários parâmetros eles são separados pelo sinal e comercial (&), por exemplo: ?userid=123456&color=blue.
- **Ancora**: #section1, que é usado para vincular a um local específico da página, neste caso à secção 1.

Um hacker tem controlo total sobre o subdomínio do hostname, o path, os parâmetros e a ancora, podendo definir quaisquer valores para eles para criar um novo URL.

3.2. Machine learning - classificação

Dada a importância do machine learning neste projeto para a classificação dos URLs que passam numa plataforma de messaging, nesta secção apresenta-se o estudo deste tópico. A classificação [10] é uma técnica supervisionada que permite prever a classe alvo de valor discreto em novas instâncias. É constituída por duas fases, uma de treino para a construção do modelo e uma de classificação. A fase de treino tem como objetivo construir um modelo, representado por um conjunto de regras ou por redes neuronais, a partir de um conjunto de dados de treino previamente classificado. O modelo construído é usado na fase de classificação para classificar novas instâncias. Nas próximas secções descreve-se as etapas envolvidas no processo iterativo de construção de um modelo de classificação (ilustrado na Figura 4) [11] após a recolha dos dados, começando pela fase de pré-processamento dos dados, seguido dos algoritmos mais populares para treino do modelo e, por fim, as métricas de avaliação de desempenho que permitem avaliar a eficácia do modelo produzido.

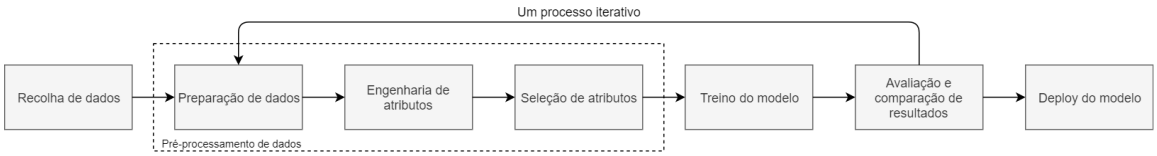


Figura 4 - Etapas do processo de machine learning

3.2.1. Pré-processamento de dados

Para o desenvolvimento de um modelo de classificação, a fase de pré-processamento de dados é crucial para a obtenção de uma boa precisão no modelo de classificação, porque a qualidade dos dados de entrada no modelo tem impacto direto na sua qualidade. Esta fase subdivide-se em três partes, a primeira de preparação do conjunto de dados, a segunda de engenharia de atributos (ou features), e a terceira de seleção dos atributos mais discriminativos.

3.2.1.1. Preparação de dados

De uma forma geral, durante a preparação de dados trata-se os valores em falta, removem-se os outliers, e transformam-se os valores de alguns dos atributos.

Os valores em falta no conjunto de dados de treino são importantes de tratar, para não reduzirem a precisão ou mesmo enviesarem o modelo, o que pode levar a classificações erradas. Existem três técnicas para tratar os valores em falta [10]:

- Eliminar os registos em que algum dos atributos está em falta. É um método simples, mas que reduz o tamanho do conjunto de dados e consequentemente a precisão do modelo.
- Preencher os valores em falta pelo valor mais frequente quando o atributo é nominal, ou pelo valor médio quando o atributo é numérico.
- Substituir os valores em falta por valores predefinidos, por exemplo n/a para valores nominais e 0 para valores numéricos. Este método é usado quando existem atributos com muitos valores em falta.
- Determinar o valor em falta pelo algoritmo K-Nearest Neighbors. Este algoritmo procura no conjunto de dados um registo semelhante para preencher o valor em falta. É muito demorado na análise de grandes conjuntos de dados.

Os outliers são valores que divergem de um padrão geral numa amostra de dados. Estes valores são importantes de tratar, porque podem alterar drasticamente os resultados da análise de dados. Existem três métodos para detetar outliers, um por análise visual, um por

distribuição normal e outro por clustering [10]. O primeiro método consiste na visualização dos dados de forma empírica com recurso a box-plots ou scatter plots para detetar os outliers. O segundo método consiste em detetar todos os valores do atributo que estiverem a mais de três desvios padrões da média da distribuição estimada. O terceiro método consiste em recorrer ao clustering para que os valores semelhantes sejam organizados em grupos ou clusters, intuitivamente, os valores que caírem fora dos conjuntos de clusters podem ser considerados outliers. Depois de identificados, os outliers são tratados de forma semelhante às técnicas de tratamento dos valores em falta descritas em cima.

Por vezes os valores dos atributos são convertidos em formatos apropriados através das operações de discretização e normalização, para facilitar a descoberta e a compreensão dos padrões pelos algoritmos [10]. A discretização consiste em transformar os valores dos atributos numéricos em valores nominais pela categorização dos valores em intervalos. A técnica de Binning é normalmente usada para discretizar os atributos de dados desconhecidos, através da distribuição dos valores do atributo em partições de igual tamanho ou frequência. A normalização aplica-se aos atributos numéricos que tenham intervalos amplos, e consiste em dimensioná-los num intervalo menor, geralmente de 0 a 1. A técnica do Min-Max é normalmente usada para normalizar os dados ao dimensioná-los no intervalo 0 a 1, mapeando o valor v_i do intervalo original do atributo de valor mínimo min_A e valor máximo max_A, para v'_i no novo intervalo de new_min_A a new_max_A definido, calculado através da fórmula:

$$v'_i = \frac{v_i - min_A}{max_A - min_A}\left(new\_max_A - new\_min_A\right) + new\_min_A$$

3.2.1.2. Engenharia de atributos

A engenharia de atributos [12] é o fator chave para um projeto de machine learning ser bem sucedido, e consiste em criar atributos a partir dos dados brutos para que possam ser aprendidos pelos algoritmos de machine learning. A qualidade dos atributos criados é fundamental para a qualidade do modelo. Se tivermos muitos atributos independentes que se correlacionam bem com a classe alvo, o modelo treinado será preciso. Se os atributos não forem representativos o modelo treinado também não será preciso. Esta é a fase onde a maior parte do esforço de um projeto de machine learning é gasto, por ser um processo analítico difícil, e por exigir conhecimento do domínio para criar os atributos.

3.2.1.3. Seleção de atributos

A seleção de atributos num conjunto de dados consiste em encontrar o subconjunto final de atributos mais discriminativos em relação à classe alvo. A seleção destes atributos permite aumentar a precisão da classificação, e tornar o processo de treino e aplicação do classificador mais eficiente, uma vez que o número de atributos e ruído nos dados diminui. Para além disso previne o problema do overfitting, que ocorre quando o modelo se adapta bem aos dados de treino, mas não se revela tão eficaz quando aplicado na classificação de novas instâncias. Existem dois tipos de métodos de seleção de atributos que ajudam a entender a importância dos atributos, os modelos de filtro e os modelos de wrapper que se descreve de seguida.

Os **modelos de filtro** [13] baseiam-se nas caraterísticas dos dados para avaliar os atributos, e consistem em duas etapas. Na primeira etapa, os atributos são hierarquizados com base num determinado critério. Na segunda etapa, são escolhidos os atributos com os rankings mais altos que serão depois usados para treinar o modelo de classificação. Vários critérios de desempenho foram propostos na literatura, sendo o mais popular o baseado na informação mútua devido à sua eficiência computacional, através do cálculo do ganho de informação (GI), que mede a dependência entre os atributos e a classe alvo. O ganho de informação entre o i-

ésimo atributo f_i e a classe alvo C pode ser calculado através da seguinte fórmula: $GI(f_i,C) = H(f_i) - H(f_i|C)$, onde $H(f_i)$ é a entropia de f_i, e $H(f_i|C)$ a entropia de f_i depois de observar C. Os atributos mais relevantes são aqueles que têm um alto ganho de informação.

Os **modelos de wrapper** [13] utilizam um classificador específico para avaliar a qualidade dos atributos. Dado um classificador predefinido, um modelo de wrapper consiste em três etapas. Na primeira etapa pesquisa um subconjunto de atributos. Na segunda etapa avalia o subconjunto de atributos selecionado pelo desempenho do classificador. Na terceira etapa repete os passos um e dois até atingir uma qualidade desejada. Basicamente existe um componente de pesquisa de atributos que produz um conjunto de atributos, e depois o componente de avaliação de atributos usa o classificador para estimar o desempenho, que será devolvido ao componente de pesquisa de atributos para a próxima iteração de seleção de outro subconjunto de atributos. O conjunto de atributos com o melhor desempenho será depois usado para treinar o modelo de classificação. O método de wrapper mais popular na literatura é o de eliminação reversa, que começa com o conjunto completo de atributos e vai eliminando progressivamente o menos relevante em cada iteração, até que nenhuma melhora seja observada na remoção dos atributos.

Normalmente, os modelos de wrapper obtêm melhores estimativas de precisão preditiva do que os modelos de filtro, no entanto, também são mais pesados computacionalmente porque envolvem o treino de modelos [13].

3.2.2. Algoritmos de classificação

Existem três tipos de famílias de algoritmos na literatura para construir modelos de classificação, os baseados em métodos básicos, os baseados em métodos de ensemble, e os baseados em redes neuronais. Os algoritmos baseados em métodos básicos utilizam técnicas estatísticas para induzir o conhecimento. Exemplos destes algoritmos bastantes populares são o Naive Bayes, a Regressão Logística, as Support Vector Machines, e a Árvore de Decisão. Os algoritmos baseados em métodos de ensemble combinam uma série de modelos de classificação individuais. Exemplo destes algoritmos bastantes populares são as Florestas Aleatórias, e o Gradient Boosting. Os algoritmos baseados em redes neuronais artificias são inspirados na forma de pensar do cérebro humano para executar tarefas otimizadas. Um exemplo deste algoritmo é o Multilayer Perceptron. De seguida descreve-se estes algoritmos bastante populares na literatura para a construção de modelos de classificação, de forma a obter uma compreensão de alto nível, e analisando as suas vantagens e desvantagens.

O **Naive Bayes** é um algoritmo multiclasse probabilístico baseado no teorema de Bayes que presume que dois eventos são independentes, ou seja, que a ocorrência de um evento não afeta a probabilidade de outro evento ocorrer. O algoritmo usa os dados de treino para calcular a distribuição de probabilidade condicional de cada um dos atributos em relação à classe alvo através da aplicação do teorema de Bayes. Este modelo apesar de ser muito simplista, tem obtido resultados extremamente favoráveis como classificador multiclasse para grandes conjuntos de dados, tendo inclusive um desempenho melhor do que alguns classificadores mais complexos [10].

A **Regressão Logística** é um algoritmo para estimar uma classe alvo de valor binário a partir de um conjunto de variáveis independentes (os atributos da instância). O algoritmo começa por medir a relação entre a classe alvo e o conjunto de variáveis independentes pela estimação das suas probabilidades. Depois, essas probabilidades são transformadas num valor real no intervalo 0 a 1 usando uma função logística, também chamada de função sigmoide. A função sigmoide é uma curva em forma de S que mapeia as probabilidades num valor real no intervalo 0 a 1. Por fim, o valor real é transformado num valor discreto (0 ou 1) para fazer a previsão,

usando um limiar de classificação. Este algoritmo como é muito eficiente de treinar é normalmente usado para definir a linha base de desempenho do modelo, no entanto, pode ser superado facilmente por outros algoritmos mais poderosos [10].

As **Support Vector Machines** (SVM) são um método que transforma cada instância do conjunto de dados de treino num ponto do espaço com n dimensões, em que n é o número de atributos, e sendo o valor de cada atribulo o valor de uma determinada coordenada do espaço (conhecidos como vetores de suporte). Depois é encontrado o hiperplano ou conjunto de hiperplanos que dividem os dois grupos de dados de treino classificados. Este será o hiperplano tal que, as distâncias do ponto mais próximo em cada um dos dois grupos estará mais distante. Este hiperplano define o classificador, uma vez que quando os dados de teste forem mapeados no espaço multidimensional ficarão de um lado ou do outro do hiperplano, que representa uma classe na qual os novos dados são classificados. Embora o tempo de treino das SVM possa ser lento, elas são altamente precisas devido à sua capacidade de modelar limites complexos de decisão não lineares, e pouco propensos ao overfitting [10].

A **Árvore de Decisão** é um modelo baseado em probabilidades condicionais também usado em problemas de classificação. Representam-se sob a forma de uma estrutura em árvore, na qual cada nó representa um teste a um atributo, cada ramo um resultado de teste, e cada nó folha (ou terminal) uma classe alvo de valor discreto. O nó mais alto na árvore é o nó raiz. Este modelo usa algoritmos recursivos baseados em abordagens gulosas que permitem selecionar o melhor atributo para cada nó da árvore, como o ID3, o C4.5 ou o CART que não serão abordados no relatório. Para classificar uma nova instância, os valores dos seus atributos são testados na árvore de decisão, e um caminho é traçado desde a raiz até a um nó folha, que contém a previsão da classe alvo para a instância [10].

As **Florestas Aleatórias** são um método ensemble de bagging que combinam um conjunto de árvores de decisão independentes, construídas a partir de subconjuntos aleatórios do conjunto de dados de treino de forma a serem diferentes uma das outras. Na classificação de uma nova instância as florestas aleatórias escolhem a classe alvo baseada na votação das classificações de cada árvore de decisão independente. As florestas aleatórias permitem aumentar a eficácia preditiva e prevenir o problema do overfitting, que ocorre quando o modelo se adapta bem aos dados de treino, mas que não se revela tão eficaz quando aplicado na classificação de novas instâncias [10].

O **Gradient Boosting** é um método ensemble de boosting em que uma série de k classificadores (tipicamente árvores de decisão) são aprendidos iterativamente por reforço, através da atribuição de pesos a cada uma das instâncias de treino. Depois que um classificador, Mi, é aprendido, os pesos das instâncias são atualizados para permitir que o classificador subsequente, Mi + 1, se concentre mais nas instâncias de treino que foram classificadas erradamente por Mi. O classificador boosted final, M*, combina os votos de cada classificador individual, em que o peso do voto de cada classificador é uma função da sua precisão. Para classificar uma nova instância, ao contrário do bagging onde cada classificador recebia um voto igual, o boosting atribui um peso ao voto de cada classificador, baseado em quão bem o classificador foi executado. Quanto mais baixa a taxa de erro de um classificador, mais preciso ele é e, portanto, maior o seu peso para a votação final. A previsão da classe alvo para a nova instância é a classe com a soma maior de pesos, resultante da soma dos pesos que cada classificador atribuiu às classes [10].

O **Multilayer Perceptron** é baseado numa rede neuronal artificial feedforward, constituída por pelo menos três camadas de nós (ou neurónios) ligadas entre si. Os nós na camada de entrada representam os dados de entrada. Os restantes nós mapeiam entradas para saídas através de uma combinação linear das entradas com os pesos dos nós e da aplicação de uma

função de ativação. Os valores dos pesos são ajustados iterativamente na fase de treino do modelo, ao passo de uma constante. Esta contante é chamada, learning rate, e representa o tamanho do passo para ajudar os pesos a convergir para uma solução quase ótima. Apesar de ser um algoritmo bastante poderoso, o seu tempo de treino é normalmente mais lento do que os outros algoritmos [10].

3.2.3. Métricas de avaliação de desempenho

Avaliar a eficácia de um modelo de classificação é uma parte essencial de qualquer projeto de machine learning. Para avaliar um modelo começa-se por dividir o conjunto de dados em treino e teste através da técnica de holdout ou de cross-validation, para depois obter a matriz de confusão que descreve o desempenho completo de um modelo. Na **técnica de holdout** [10] os dados são divididos aleatoriamente em dois conjuntos de dados independentes, um de treino com dois terços dos dados e um de teste com um terço dos dados. Depois usa-se o conjunto de treino para derivar o modelo, e o conjunto de teste para obter a matriz de confusão. Na **técnica de cross-validation** [10] com k-fold (10 é o recomendado), os dados iniciais são divididos aleatoriamente em k subconjuntos mutuamente exclusivos (ou folds), D_1, D_2, ..., D_k, de igual tamanho. Depois realiza-se o processo de treino e teste do modelo k vezes. Na iteração i, o subconjunto D_i é reservado para testar o modelo e os restantes subconjuntos para treinar o modelo. Ou seja, na primeira iteração, os subconjuntos D_2, ..., D_k são usados como conjunto de treino para treinar um primeiro modelo, que é testado em D_1, a segunda iteração é treinada nos subconjuntos D_1, D_3, ..., D_k e testada no D_2, e assim sucessivamente. Com base nos resultados das classificações das k iterações é obtida a matriz de confusão. Esta técnica apesar de ser mais pesada computacionalmente, permite avaliar a capacidade de generalização de um modelo. A matriz de confusão obtida é uma matriz N x N, em que N é o número de classes, que regista o número de true positives (TP), false negatives (FN), false positives (FP), true negatives (TN) [10]. Na Tabela 3 apresenta-se um exemplo de uma matriz de confusão com duas classes, uma classe positiva (por exemplo um URL malicioso) e uma classe negativa (por exemplo um URL seguro).

Realidade \ Previsto	**Positivo**	**Negativo**
Positivo	TP (instâncias positivas classificadas como positivas)	FN (instâncias positivas classificadas como negativas)
Negativo	FP (instâncias negativas classificadas como positivas)	TN (instâncias negativas classificadas como negativas)

Tabela 3 - Matriz de confusão

A partir da matriz de confusão podem ser calculadas várias métricas que permitem avaliar a eficácia de um classificador, como por exemplo a accuracy, a precisão, o recall, o F1 score, o TPR (True Positive Rate), o FPR (False Positive Rate) e o AUC (Area Under Curve) que se apresentam de seguida.

A **accuracy** mede a percentagem de instâncias classificadas corretamente pelo classificador, calculada com a fórmula abaixo. No entanto, esta métrica só deve ser utilizada se o número de instâncias em cada classe for equilibrado, caso contrário, se a principal classe de interesse for rara, deve-se utilizar as métricas da precisão e do recall.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

A **precisão** mede a percentagem de instâncias classificadas como positivas que são realmente positivas, calculada com a fórmula abaixo. Esta métrica fornece informações sobre o desempenho de um classificador em relação aos falsos positivos.

$$\text{precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

O **recall** mede a percentagem de instâncias positivas que são identificadas corretamente pelo classificador, calculado com a fórmula abaixo. Esta métrica fornece informações sobre o desempenho de um classificador em relação aos falsos negativos.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Se pretendermos minimizar o número de falsos positivos, então a precisão deves ser o mais próxima possível de 100%. Se pretendermos minimizar o número de falsos negativos, então o recall deve ser o mais próximo possível de 100%. Idealmente ambas as métricas devem ser tão altas quanto possível e com valores aproximados, o que indica uma previsão equilibrada e imparcial.

O **F1 score** combina a precisão e o recall numa única medida que avalia a precisão global de um classificador no intervalo de 0 a 1. Esta métrica indica quão preciso é o classificador (ou seja, quantas instâncias foram classificadas corretamente), além de quão robusto ele é (ou seja, se não perde um número significativo de instâncias difíceis de classificar). O F1 score resulta da média harmónica entre a precisão e o recall, calculado com a seguinte fórmula:

$$\text{F1} = 2 * \frac{\text{precisão} * \text{recall}}{\text{precisão} + \text{recall}}$$

A **TPR** (ou recall) mede a percentagem de instâncias positivas que são identificadas corretamente pelo classificador, calculado com a seguinte abaixo.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

O **FPR** mede a percentagem de instâncias negativas que são erradamente identificadas como positivas pelo classificador, calculado com a fórmula abaixo.

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

O **AUC** combina o TPR e o FPR numa única medida, e é uma alternativa ao F1 score para avaliar a precisão global de um classificador. Esta métrica é calculada pela medição de toda a área bidimensional por baixo da curva de ROC (Receiver Operating Characteristic), e varia no intervalo de 0 a 1. A curva de ROC é um gráfico que traça o TPR vs. FPR em todos os limites de classificação. Uma maneira de interpretar o AUC é a probabilidade de o classificador classificar uma instância positiva aleatória mais alta do que uma instância negativa aleatória, indicando quanto o modelo é capaz de distinguir entre estas classes.

4. Estado da Arte

Neste capítulo apresenta-se a informação recolhida dos sistemas concorrentes, de forma a contribuírem com funcionalidades na definição dos requisitos funcionais do protótipo a desenvolver. De seguida, expõe-se o estudo de abordagens analíticas para implementar o mecanismo de proteção contra phishing, resultante da revisão literária dos artigos publicados nesta área de investigação. De seguida, e uma vez que a obtenção e discriminação de URLs é importante para o desenvolvimento do protótipo, fez-se o levantamento e análise de APIs de deteção de URLs maliciosos. Por fim, e dada a importância do machine learning para a criação do modelo de classificação, fez-se o levantamento e análise de ferramentas de machine learning.

4.1. Sistemas concorrentes

Esta secção apresenta a informação recolhida dos sistemas concorrentes das empresas Zvelo, Kaspersky e Cloudmark, que possuem soluções empresariais para proteger as plataformas de messaging do lado do servidor. Tendo em conta a informação limitada que é disponibilizada por estas empresas sobre os seus produtos e, uma vez que a experimentação está sujeita à solicitação de uma demo, apenas foi possível obter as funcionalidades gerais fornecidas por estes sistemas e as técnicas em que se baseiam.

A **Zvelo** fornece uma solução empresarial para categorizar sites maliciosos em fornecedores de serviços de messaging, chamado Malicious Detection for OEMs [14]. Esta solução consiste numa base de dados constantemente atualizada que contém todos os URLs maliciosos, podendo ser implantada diretamente no datacenter do fornecedor de serviços com ingestão direta de novos dados, ou integrada por meio de uma API ou feed de dados. Esta solução é capaz de detetar vários tipos de ameaças, como URLs fraudulentos de phishing, URLs que distribuem malware, URLs de spam que aparecem frequentemente em mensagens, e também URLs que foram comprometidos para incluir links ou código malicioso (por exemplo, sites de mineração de criptomoedas sem a permissão do utilizador). Diversas técnicas são usadas pela Zvelo na deteção de ameaças maliciosas, combinando os seguintes métodos: análise do URL (domínio, path), análise do conteúdo da página, análise heurística, análise comportamental (sandboxing), e machine learning supervisionado para fornecer uma ampla deteção de sites maliciosos.

A **Kaspersky** fornece uma solução empresarial para a deteção de URLs de phishing em plataformas de messaging, chamado Kaspersky Anti-Phishing Feeds [15]. Esta solução oferece um feed atualizado em tempo real de URLs e IPs de phishing confirmados, fornecendo aos clientes elevada precisão na deteção confiável de ataques de phishing cada vez mais sofisticados, e garantindo reação instantânea a domínios e URLs de phishing gerados dinamicamente. Diversas técnicas de deteção de phishing são usadas para formar o feed atualizado, como estatísticas do Kaspersky Security Network, web crawlers, potentes motores heurísticos baseados em machine learning e análise do conteúdo das páginas.

A **Cloudmark** fornece uma solução empresarial de segurança para servidores de messaging (SMS, MMS, RCS), chamado Cloudmark Security Platform for Mobile Messaging [16]. Esta é uma solução de segurança em tempo real de alto desempenho, que permite detetar e filtrar mensagens nos servidores de messaging que contenham phishing, malware e spam. Diversas técnicas são usadas nesta solução, como algoritmos de filtragem baseados em conteúdo e protocolo, e machine learning para aprender e adaptar-se automaticamente a ameaças sempre crescentes que se transformam e espalham rapidamente. Para além disso o Cloudmark Security

Platform possui dashboards com relatórios históricos atualizados, que permitem ao fornecedor do serviço obter visibilidade em tempo real sobre as ameaças, problemas de entrega e abusos.

Na Tabela 4 apresenta-se um resumo da informação recolhida destes sistemas, comparando as funcionalidades que cada um fornece e as técnicas em que se baseiam.

	Zvelo	Kaspersky	Cloudmark
Integrado do lado do servidor	Sim	Sim	Sim
Deteta URLs de phishing	Sim	Sim	Sim
Deteta URLs de malware	Sim	Sim	Sim
Deteta URLs de spam	Sim	Não	Sim
Baseado em machine learning	Sim	Sim	Sim
Análise do URL, domínio	Sim	Sim	Sim
Análise do conteúdo da página	Sim	Sim	Sim
Análise do comportamento	Sim	Sem informação	Sem informação
Atualizado em tempo real	Sim	Sim	Sim

Tabela 4 - Comparação das funcionalidades dos sistemas concorrentes

A Tabela 4 mostra que os sistemas concorrentes são baseados em machine learning e detetam tanto URLs de phishing como URLs de malware, o que vai de encontro aos dois tipos de URLs maliciosos que se podem encontrar nas mensagens das plataformas de messaging já visto anteriormente. A deteção de spam estará fora do âmbito deste trabalho. No entanto, neste trabalho pretende-se iniciar o desenvolvimento de uma solução à medida dos produtos da empresa, com machine learning aprimorado para detetar com efetividade ataques de dia zero e páginas web comprometidas, que possibilite a configuração do grau de confiança do classificador consoante o cliente final, e com menos custos. Para além disso as plataformas de messaging como WhatsApp ou o Facebook Messenger, apesar de não serem direcionadas para os operadores de telecomunicações optaram também por desenvolver a sua própria solução de filtragem de comunicações, na qual podem confiar, em vez de adquirirem este tipo de sistemas. Já o Skype e o iMessage não possuem qualquer tipo de proteção nas comunicações. Uma vez que não há livros nem informação sobre como desenvolver uma solução de deteção de URLs maliciosos, na próxima secção fez-se a revisão literária de abordagens analíticas para o classificador a partir de referências internacionais, e que servirá como base para a implementação do sistema a desenvolver.

4.2. Abordagens analíticas para o classificador

Existem dois tipos de abordagens analíticas para desenvolver um mecanismo de proteção contra phishing. A mais comum é baseada numa lista negra de URLs, e que é usada pelos browsers modernos. No entanto, esta abordagem baseada numa lista negra de URLs centralizada, sozinha não é adequada para proteger os utilizadores das páginas web de phishing de dia zero, que aparecem aos milhares e desaparecem rapidamente todos os dias, porque normalmente estas listas são atualizadas com uma periodicidade de 24 horas. A outra abordagem é baseada em algoritmos de machine learning, para criar modelos de classificação sobre heurísticas, capazes de detetar qualquer tipo de páginas web de phishing com uma grande probabilidade. Esta abordagem mais dinâmica e semi-automatizada têm a capacidade de generalizar para novos URLs, o que permite superar as abordagens baseadas em listas negras de URLs, e como tal será a abordagem seguida no desenvolvimento do protótipo de forma a fornecer mecanismos de defesa que combatem com efetividade os ataques de phishing [17].

Na próxima subsecção apresenta-se o estudo de trabalhos relacionados com abordagens analíticas para a deteção de URLs maliciosos usando técnicas de machine learning. Na subsecção seguinte apresenta-se o levantamento de várias categorias de atributos dos URLs, e que servirão como base para o desenvolvimento do modelo de classificação.

4.2.1. Trabalhos relacionados

Em 2007, os autores Garera, S., Provos, N., Chew, M., & Rubin, A. D. propuseram uma abordagem para deteção de ataques de phishing [18]. Descobriram que muitas vezes é possível com base em atributos heurísticos da estrutura de um URL dizer se o URL pertence ou não a um ataque de phishing. Nesta abordagem um classificador de regressão logística foi usado em mais de 18 atributos selecionados manualmente para classificar os URLs de phishing. Os atributos incluem a presença de determinadas palavras-chave sugestivas (confirm, banking, signin, ebayisapi, login, webscr, account, secure) de sinalização vermelha no URL, alguns atributos proprietários da infraestrutura da Google como o page rank, a posição da página no índice e a pontuação de qualidade da página. Mesmo que não tenham analisado o conteúdo da página para usar como atributos, eles usaram atributos baseados em páginas pré-calculados da infraestrutura proprietária do Google que chamam de base de dados de rastreamento. Como resultado, esta técnica alcançou uma precisão de classificação de 97.3% com uma taxa de falsos positivos de 1.2% num conjunto de dados de 2.508 URLs, dos quais 1.245 URLs eram de phishing.

Em 2011, os autores Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. propuseram uma abordagem para detetar URLs maliciosos a partir de atributos lexicais e atributos baseados no host dos URLs [19]. Usando estes atributos, eles compararam a precisão dos algoritmos de aprendizagem em batch e online. A abordagem em batch foi avaliada com um conjunto de dados de 30.000 URLs, e o algoritmo de regressão logística mostrou ser o mais eficaz, tendo obtido uma taxa de classificação muito alta (95% -99%) e uma taxa de falsos positivos baixa. Para aplicação em larga escala na deteção de URLs maliciosos, os autores recorreram a algoritmos online para lidar com milhões de URLs cujos atributos evoluem com o tempo. Um sistema em tempo real alimentado por URLs já classificados de um serviço de email foi desenvolvido para recolher atributos de URLs em tempo real. Com esses atributos e a classe alvo dos URLs, treinaram um classificador capaz de detetar URLs maliciosos com uma taxa de precisão de 99%, usando um algoritmo de aprendizagem com ponderação de confiança. Apesar da elevada taxa de precisão, a abordagem proposta utiliza os atributos lexicais segundo um modelo de bag-of-words, e como atinge milhões de palavras é bastante ineficiente.

Em 2014, os autores Basnet, R. B., Sung, A. H., & Liu, Q. apresentaram uma abordagem heurística para classificar URLs de phishing com base em informações e atributos presentes nos URLs [20]. A abordagem pode ser usada para impedir um ataque de phishing em qualquer lugar que incorpore URLs (emails, sessões de chat), através da ocultação do URL de phishing ou alertando o utilizador sobre a possível ameaça. A abordagem proposta foi avaliada em conjuntos de dados reais com mais de 16.000 URLs de phishing e 31.000 URLs legítimos. Foi demonstrado experimentalmente que a abordagem anti-phishing proposta foi capaz de detetar URLs de phishing com uma precisão de 99.4%, e com taxas de falsos positivos e de falsos negativos inferiores a 0.5%. Dos vários algoritmos de machine learning experimentados para criação de modelos a partir de dados de treino, foi o classificador de florestas aleatórias que forneceu a melhor relação entre o desempenho da classificação e o tempo de treino e teste. Apesar da elevada taxa de precisão, a abordagem proposta necessita de mais desenvolvimentos a fim de poder ser usada em tempo real, uma vez que a classificação de um novo URL pode demorar 3 segundos.

Em 2017, os autores Ranganayakulu, D., & Chellappan, C. propuseram uma abordagem para detetar URLs de phishing e maliciosos em emails, através de um método com um conjunto reduzido de atributos lexicais e baseados no host do URL [21]. Os atributos lexicais analisam o formato do URL, o que inclui o comprimento do URL, o comprimento do host, o número de pontos, a presença de caracteres suspeitos como o @, caracteres hexadecimais e outros caracteres especiais como ('.', '=', '$', ' ^ ', etc.). Os atributos baseados no host do URL, como a idade do domínio e o page rank, para fornecer informação relacionada com o proprietário do domínio e como os sites maliciosos são alojados. Para experimentar a abordagem usaram um conjunto de dados com 1000 emails contendo URLs maliciosos e 1000 emails contendo URLs legítimos. Um classificador com o algoritmo naive bayes foi treinado e testado, tendo sido obtida uma precisão de 92.8% e uma taxa de falsos positivos de 0.4%.

Em 2017, os autores Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., & González, F. A, apresentaram uma abordagem para classificar URLs de phishing baseada nas informações e caraterísticas presentes no URL [22]. Foi usado como entrada do modelo uma combinação de atributos lexicais e estatísticos do URL, analisados e selecionados manualmente como por exemplo, o comprimento dos componentes do URL, se o domínio é um IP, se está presente no ranking do Alexa, a contagem de carateres de pontuação, entre outros. Para avaliar esta abordagem de engenharia de atributos foi utilizado um conjunto de dados real contendo 10.000 URLs de phishing e 10.000 URLs legítimos. De seguida foi treinado e testado um classificador de florestas aleatórias com 100 árvores de decisão usando a técnica de cross-validation com 3 folds, tendo sido obtido um índice de F1 score de 93.47%, com uma precisão de 93.63% e um recall de 93.28%.

Os trabalhos [18] e [19] sugerem que o algoritmo de regressão logística se adequa à deteção de URLs de phishing, enquanto que o [20] e [22] sugerem as florestas aleatórias, e o [21] o naive bayes. Pode-se constatar que em todos estes trabalhos foram incluídos atributos lexicais nos modelos de classificação. Uma limitação destes trabalhos é que em nenhum deles foi usado um conjunto de dados relativamente grande de URLs (o maior é de 47.000 URLs usado no trabalho [20]), e também o facto de não terem validado o modelo produzido com outras fontes de dados para verificar se é generalizável. Além de ultrapassar estas limitações, no modelo de classificação a desenvolver será também interessante experimentar mais algoritmos do que os expostos e com diferentes parametrizações, na busca do melhor ajuste possível ao conjunto de dados, que não foi referido em nenhum destes trabalhos. Por último, mas não menos importante, é perceber como é que o modelo de classificação pode ser integrado num sistema para deteção de phishing em tempo real.

4.2.2. Categorias de atributos

Vários tipos de atributos têm sido propostos na literatura para o problema de deteção de URLs de phishing com machine learning, de forma a fornecer informações úteis sobre os URLs e tentar maximizar a precisão da classificação. A partir dos trabalhos relacionados recolheu-se vários tipos de atributos que se agrupou em cinco categorias distintas: atributos baseados no léxico, atributos baseados no host, atributos baseados em palavras-chave, atributos baseados em reputação e atributos baseados no motor de busca que se apresentam de seguida, e que depois serão usados como base para o desenvolvimento do modelo de classificação.

4.2.2.1. Atributos baseados no léxico

Os atributos lexicais também conhecidos como as propriedades textuais do próprio URL, têm sido amplamente utilizados na literatura na deteção de ataques de phishing. Nos artigos [20], [22] e [23] examinaram várias técnicas de ofuscação que os atacantes podem empregar, e derivaram vários atributos lexicais de phishing para usar nos classificadores. Existem 24

atributos nesta categoria, 6 atributos de valor real e 18 atributos de valor binário que se apresenta na Tabela 5 e Tabela 6, respetivamente.

Descrição do atributo
Comprimento do host
Comprimento do path
Comprimento do URL
Número de '.' no host
Número de '.' no path
Número de '.' no URL

Tabela 5 - Atributos lexicais de valor real

Descrição do atributo		
'-' no host		Tem parte do parâmetro
Dígito [0-9] no host		Tem parte da query
Host baseado em IP		'=' na parte da query
Host baseado em hex		Tem parte do fragmento
'-' no path		'@' no URL
'/' no path		'Username' no URL
'=' no path		'Password' no URL
';' no path		Tem porto não padronizado
',' no path		'_' no path

Tabela 6 - Atributos lexicais de valor binário

No caso de existir um porto no URL deve-se verificar se o porto pertence à lista dos portos HTTP padrão: 80, 443, 8080, 21, 70 e 1080. Se o porto não pertencer a esta lista, o URL deve ser sinalizado como sendo potencialmente de phishing.

4.2.2.2. Atributos baseados no host

Os atributos baseados no host têm sido amplamente utilizados na literatura na deteção de URLs maliciosos, uma vez que permitem descrever onde é que os hosts maliciosos são alojados, por quem são geridos, e como são administrados. Nos artigos [19] e [24] examinaram várias atributos relacionados ao host, como as propriedades WHOIS do domínio, a data de registo e de expiração do domínio uma vez que como os sites de phishing são removidos diariamente as suas datas de registo são mais recentes do que as de sites legítimos, e ainda quem é o registador e o registante do domínio para se saber por quem são registados e por quem são geridos os domínios maliciosos. Além destas propriedades, a região geográfica (país/cidade) ao qual o endereço IP pertence, para se saber em que regiões os focos de atividade maliciosa estão concentrados, e a velocidade de conexão ao host, uma vez que alguns sites maliciosos tendem a residir em máquinas residenciais. Na Tabela 7 apresenta-se os atributos baseados no host propostos.

Descrição do atributo
Número de dias desde a data de registo do domínio
Número de dias até à data de expiração do domínio
O registador do domínio
O registante do domínio
País/cidade a que o endereço IP pertence
Velocidade da conexão

Tabela 7 - Atributos baseados no host

4.2.2.3. Atributos baseados em palavras-chave

Os atributos baseados em palavras-chave têm sido amplamente utilizados na literatura na deteção de ataques de phishing, porque algumas palavras presentes nos URLs como por exemplo, login, signin, confirm, verify, free, etc., costumam atrair a atenção dos utilizadores. Dado que só as palavras mais relevantes devem ser usadas como atributos do modelo de classificação, os autores do artigo [20] sugerem a seguinte abordagem de seleção. Começar por dividir cada URL de phishing pelos caracteres não alfanuméricos, e de seguida calcular a frequência de cada um dos tokens. Para não sobrecarregar o algoritmo de machine learning sem gerar benefício de desempenho descartar todos os tokens com comprimento menor que três, como a, b, pt, en, etc. Os tokens de um carater costumam ter elevadas frequências sem oferecer qualquer significado, enquanto que os tokens de dois caracteres são principalmente os domínios de primeiro nível com o código do país. Depois descartar as partes comuns dos URLs, como http, https, www, com, etc., e as extensões dos ficheiros das páginas web, como htm, html, asp, php, etc. Finalmente, descartar todos os tokens de frequência um, uma vez que não são usados com frequência nos URLs e por norma são palavras sem sentido.

Aos tokens restantes, aplica-se a técnica de seleção de atributos baseada na informação mútua (IM), de forma a selecionar os mais discriminativos em relação à classe alvo. A IM mede a quantidade de informação que a presença ou ausência de uma palavra contribui para tomar a decisão correta de classificação em uma classe, e pode ser calculada com a seguinte fórmula:

$$\mathrm{IM}\,(U,C) = \frac{N_{00}}{N}\log_2\frac{NN_{00}}{N_0N_{\cdot 0}} + \frac{N_{01}}{N}\log_2\frac{NN_{01}}{N_0N_{\cdot 1}} + \frac{N_{10}}{N}\log_2\frac{NN_{10}}{N_1N_{\cdot 0}} + \frac{N_{11}}{N}\log_2\frac{NN_{11}}{N_1N_{\cdot 1}}$$

Onde U é uma variável aleatória que recebe valores ep = 1 (se o URL contém a palavra) e ep = 0 (se o URL não contém a palavra), C é uma variável aleatória que recebe valores ec = 1 (se o URL está na classe positiva) e ec = 0 (se o URL está na classe negativa). Os N são as contagens dos URLs que possuem os valores ep e ec indicados pelos dois índices. Por exemplo, N_{10} é o número de URLs que contêm a palavra (ep = 1) e estão na classe negativa (ec = 0). $N_1 = N_{10} + N_{11}$ é o número de URLs que contêm a palavra (ep = 1) e pertencem à classe positiva e negativa ($ec \in \{0, 1\}$). $N = N_{00} + N_{01} + N_{10} + N_{11}$ é o número total de URLs no conjunto de treino.

As palavras com valores altos de IM são as mais relevantes e são bons atributos discriminativos, enquanto que as palavras com valores de IM baixos são menos relevantes. Depois utiliza-se um top reduzido de palavras com valores altos de IM para incluir nos atributos do modelo de classificação.

4.2.2.4. Atributos baseados em reputação

Os atributos baseados em reputação, como os relatórios estatísticos de sites phishing e as listas negras de URLs, também têm sido amplamente utilizados na literatura na deteção de ataques de phishing, com o objetivo de aumentar o grau de confiança na classificação dos URLs. No artigo [20] sugerem o uso dos relatórios estatísticos do PhishTank, uma vez que este site produz todos os meses o top de 10 de domínios, o top 10 de IPs e o top 10 de alvos populares usados em sites de phishing. A ideia é fazer uso de dados históricos sobre os principais domínios e IPs que alojam sites de phishing, ou seja, se um URL tiver outras heurísticas relacionadas a phishing e o seu host pertencer ao top de domínios e/ou ao top de IPs com reputação histórica de alojamento de páginas web de phishing, então pode-se aumentar o grau de confiança na classificação de um determinado URL como phishing. Para complementar os relatórios estatísticos, no artigo [20] também sugerem o uso de listas negras para verificar os URLs, como por exemplo a do PhishTank e a do Safe Browsing. Para ambos os casos se

utilizam atributos binários, ou seja, verifica-se se o relatório estatístico ou a lista contêm o domínio ou o URL em questão, e se contiver sinaliza-se o URL como sendo potencialmente de phishing. Na Tabela 8 apresenta-se os atributos baseados em reputação sugeridos.

Descrição do atributo
Top 10 de domínios em URLs (PhishTank)
Top 10 de IPs em URLs (PhishTank)
Top 10 de alvos populares em URLs (PhishTank)
URL na lista negra do PhishTank
URL na lista negra do Safe Browsing

Tabela 8 - Atributos baseados em reputação

4.2.2.5. Atributos baseados no motor de busca

Quando o URL e o domínio não aparecem nos índices dos motores de busca, é uma indicação de que o domínio foi criado recentemente, logo o URL em questão tem uma probabilidade maior de ser de phishing. Com base neste facto, os autores dos artigos [20] e [25] descrevem uma abordagem heurística para verificar se os URLs são indexados pelos principais motores de busca (Google, Bing, Yahoo) da seguinte maneira. Primeiro, procuram o URL inteiro e recuperam o top 30 de resultados devolvidos pelo motor de busca. Se os resultados contiverem o URL, o URL é considerado como sendo potencialmente legítimo, caso contrário como phishing. Além disso também verificam se a parte do domínio do URL corresponde à parte do domínio de qualquer link obtido nos resultados. Da mesma forma, se houver uma correspondência, o URL é sinalizado como sendo potencialmente legítimo. Caso contrário, consultam o motor de busca novamente apenas com a parte do domínio do URL. Se nenhum dos links devolvidos corresponder ao URL da consulta, o URL é sinalizado como sendo potencialmente phishing.

Esta heurística, parte do princípio de que os motores de busca indexam mais cedo ou mais tarde a maioria dos sites legítimos, uma vez que permanecem online durante mais tempo. Por outro lado, o tempo médio que um site de phishing permanece online é de 1 a 3 dias, e como não haverá muitos links a apontar para esse site, os rastreadores dos motores de busca podem não aceder ao site antes de ser removido. Além disto, esta heurística também tira partido do facto de os motores de busca poderem filtrar links maliciosos conhecidos dos resultados da pesquisa através das suas tecnologias proprietárias. Na Tabela 9 apresenta-se os atributos baseados no motor de busca sugeridos.

Descrição do atributo
URL não está no top de resultados do Google
Domínio não está no top de resultados do Google
URL não está no top de resultados do Bing
Domínio não está no top de resultados do Bing
URL não está no top de resultados do Yahoo
Domínio não está no top de resultados do Yahoo

Tabela 9 - Atributos baseados no motor de busca

Concluído o levantamento destas cinco categorias de atributos, importa referir que no modelo de classificação a desenvolver planeia-se explorar os dados analiticamente e também usar o conhecimento no domínio para criar novos atributos lexicais (por exemplo, a entropia do URL, a contagem de carateres de pontuação, etc.) e de reputação (por exemplo, verificar a presença de domínios frequentes de phishing em outras listas negras, fazer uso da recente ferramenta do Open Page Rank [26] para obter métricas do page rank das páginas web, etc.).

4.3. APIs de deteção de URLs maliciosos

Nesta secção fez-se o levantamento, estudo e comparação das APIs de deteção de URLs maliciosos, concretamente a do Google Safe Browsing, do VirusTotal, do PhishTank, do OpenPhish, do Urlscan, e do URLhaus.

O **Google Safe Browsing** [27] é um serviço que permite que as aplicações clientes verifiquem URLs nas listas de recursos web inseguros da Google. Exemplos de recursos web inseguros são os sites de engenharia social (sites de phishing e enganosos) e os sites que alojam malware ou software indesejado. Para determinar se um URL está nas listas de navegação segura da Google, os clientes podem usar duas opções: o Lookup API que permite que as aplicações clientes enviem URLs para o servidor do Google Safe Browsing verificar se o estado deles é seguro ou inseguro, ou o Update API que permite que as aplicações cliente façam o download das listas de navegação segura para realizar as verificações de URLs no lado do cliente.

O **VirusTotal** [28] é um serviço que agrega informações de diversos programas antivírus, scanners de websites, e de ferramentas de análise de ficheiros e URLs, para verificar ficheiros e URLs suspeitos de conteúdo malicioso. O VirusTotal fornece uma API pública que permite fazer a verificação de URLs em tempo real e também recuperar resultados de verificações existentes. Os resultados apresentados são a saída combinada dos diferentes produtos antivírus e ferramentas de análise. Algumas restrições se aplicam aos pedidos feitos por meio da API, como a solicitação de uma chave de API, e o número de pedidos na versão gratuita ser limitado a apenas quatro por minuto.

O **PhishTank** [29] é um serviço anti-phishing baseado na comunidade, onde os utilizadores enviam suspeitas de phishing e outros utilizadores votam se é um phish ou não. Para facilitar a integração de dados anti-phishing nas aplicações, o PhishTank fornece uma API aberta que permite que os developers descarreguem a base de dados que contém todas as páginas de phishing. Esta base de dados é atualizada a cada hora e está disponível em vários formatos (XML, JSON, CSV) para facilitar a integração.

O **OpenPhish** [30] é um serviço que recebe milhares de URLs não filtrados de diversas fontes da sua rede global de parceiros, e que identifica e seleciona os URLs de phishing com base no seu mecanismo independente automatizado de inteligência de phishing. O OpenPhish possui uma API gratuita que permite fazer o download dos URLs de phishing ativos dos últimos 14 dias. A frequência de atualização da informação é feita a cada hora.

O **Urlscan** [31] é um scanner de websites. Quando um URL é submetido ao Urlscan, um processo automatizado acede ao URL e regista toda a atividade criada pela conexão HTTP estabelecida, que inclui, os domínios e os IPs contactados, os recursos (HTML, CSS, JavaScript, etc.) solicitados, bem como outras informações adicionais sobre a página, como por exemplo as variáveis globais JavaScript e os cookies criados. Por fim, o Urlscan consulta o Google Safe Browsing para determinar se a página é maliciosa. O Urlscan possui uma API que permite enviar URLs para fazer uma verificação e recuperar os resultados dessa verificação até um máximo de trinta submissões por minuto. Além disso, a API também permite recuperar resultados de verificações existentes sem restrições.

O **URLhaus** [32] é um serviço que recolhe, rastreia e partilha URLs maliciosos, para ajudar tanto os developers a protegerem as suas aplicações como os administradores de redes e analistas de segurança a protegerem as suas redes contra ameaças cibernéticas. O URLhaus possui uma API gratuita que permite fazer o download dos URLs maliciosos existentes na sua base de dados em formato CSV. A frequência de atualização da informação é feita a cada hora.

Na Tabela 10 apresenta-se uma síntese da comparação efetuada às APIs de deteção de URLs maliciosos:

	Safe Browsing	VirusTotal	PhishTank	OpenPhish	Urlscan	URLhaus
API de acesso	Sim	Sim	Sim	Sim	Sim	Sim
Endpoint para analisar um URL (em tempo real)	Não	Sim	Não	Não	Sim	Não
Endpoint para verificar um URL (consulta da sua lista)	Sim	Sim	Não	Não	Sim	Não
Endpoint descarga da lista URLs maliciosos	Sim	Não	Sim	Sim	Não	Sim
Deteção de phishing nos URLs	Sim	Sim	Sim	Sim	Sim	Não
Deteção de malware nos URLs	Sim	Sim	Não	Não	Sim	Sim
Versão gratuita	Sim	Sim	Sim	Sim	Sim	Sim
Nº pedidos à API	10.000 /dia	4 /minuto	Sem limite	Sem limite	30 /minuto	Sem limite
Frequência de atualização da API	Sem informação	Não aplicável	A cada hora	A cada hora	Não aplicável	A cada hora

Tabela 10 - Comparação das APIs de deteção de URLs maliciosos

Tendo em conta as caraterísticas apresentadas, pode-se verificar que as APIs do VirusTotal e do Urlscan se destacam em relação às restantes pelo facto de permitirem fazer a análise de um URL em tempo real. A API do Google Safe Browsing permite apenas verificar se um determinado URL se encontra nas suas listas negras que poderão estar desatualizadas. Ou seja, sempre que aparece um novo URL malicioso retorna inicialmente um estado seguro, o que obriga a fazer novos pedidos nas próximas horas para saber se o seu estado entretanto mudou. As APIs do PhishTank, do OpenPhish e do URLhaus permitem descarregar a sua lista negra de URLs. No entanto, o uso de uma destas APIs para discriminar os URLs, sozinha não permite desenvolver uma solução on the fly e em tempo real. Primeiro porque as que permitem analisar um novo URL em tempo real têm quotas limitadas, segundo porque as listas de sites comunitários não contêm os novos URLs maliciosos e podem gerar muitos falsos positivos, terceiro porque constatou-se experimentalmente que URLs maliciosos encurtados nunca são detetados por estas APIs. Ainda assim, estas APIs serão úteis para discriminar os URLs do conjunto de dados de treino e teste do modelo de classificação, e as APIs gratuitas e de sites comunitários que permitem fazer o download da lista de URLs são interessantes de incluir no modelo de classificação como atributos de reputação, por exemplo para verificar se um determinado domínio é frequente de phishing.

4.4. Ferramentas de machine learning

Atualmente existem várias ferramentas de machine learning disponíveis no mercado cada uma com os seus benefícios e as suas limitações, sendo as mais populares, o Scikit-learn [33], o TensorFlow [34], o H_2O [35] e o Weka [36]. De maneira a poder-se efetuar uma comparação e avaliação destas ferramentas definiu-se o seguinte conjunto de critérios:

- Open source;
- Suporte, incluí documentação, fóruns e pequenos exemplos de iniciação;
- Fornece funcionalidades para análise exploratória de dados;
- Fornece funcionalidades para análise de atributos relevantes;

- Disponibiliza algoritmos da técnica de classificação de machine learning;
- Fornece funcionalidades para afinar e avaliar o desempenho dos modelos de classificação desenvolvidos;
- Integração com fontes de dados, como bases de dados e ficheiros;
- Integração com linguagens de programação.

Na Tabela 11 apresenta-se uma síntese da comparação das ferramentas de machine learning usando o conjunto de critérios definidos:

	Scikit-learn	TensorFlow	H_2O	Weka
Open source	Sim	Sim	Sim	Sim
Suporte	Sim	Sim	Sim	Não
Análise exploratória	Sim	Sim	Não	Sim
Análise atributos relevantes	Sim	Sim	Não	Sim
Algoritmos de classificação	Sim	Sim	Sim	Sim
Afinar e avaliar os modelos	Sim	Sim	Sim	Sim
Integração com fontes de dados	Sim	Sim	Sim	Sim
Integração com linguagens de programação	Sim	Sim	Sim	Sim

Tabela 11 - Comparação das ferramentas de machine learning

Tendo em conta as caraterísticas apresentadas, pode-se concluir que o H_2O e o Weka não são escolhas adequadas. O primeiro porque não possui funcionalidades próprias para realizar uma análise exploratória dos dados e dos atributos relevantes, e, uma vez que é uma ferramenta recente que ainda está em desenvolvimento pode não estar madura o suficiente. O segundo porque não tem suporte e a documentação que existe é bastante reduzida, para além disso, é menos flexível para a criação de aplicações do que as restantes. Consideram-se que o Scikit-learn e o TensorFlow seriam as escolhas mais adequadas, uma vez que são bibliotecas de machine learning do Python, logo são mais flexíveis para a análise de dados, visualização gráfica de resultados, otimização de modelos e para a criação de aplicações. No enanto, optando por uma, a escolha recai sobre o Scikit-learn por ser uma biblioteca de mais alto nível, e também porque possui uma maior comunidade online.

4.5. Sumário

Este capítulo apresentou a informação recolhida dos sistemas concorrentes. De seguida, apresentou o estudo de abordagens analíticas para implementar o mecanismo de proteção contra phishing, resultante da revisão literária de artigos publicados nesta área de investigação. De seguida, apresentou o levantamento e análise de APIs de deteção de URLs maliciosos, tanto para discriminar os URLs do conjunto de dados de treino e teste, como para auxiliar o processo de aprendizagem do modelo de classificação a desenvolver. Por fim, e dada a importância do machine learning para a criação do modelo de classificação, apresentou o levantamento e análise de ferramentas de machine learning.

Dito isto, importa reforçar que neste trabalho pretende-se iniciar o desenvolvimento de uma solução à medida dos produtos da empresa, com machine learning aprimorado para detetar com efetividade ataques de dia zero, com possibilidade de configuração consoante o cliente final, e com menos custos. Em relação aos trabalhos relacionados, irá usar-se um conjunto de dados relativamente grande de URLs (mais amplo e abrangente), experimentar mais

algoritmos dos que os expostos, tirar partido das parametrizações dos algoritmos de forma a fazer uma última otimização do modelo, e validar com outros conjuntos de dados que o modelo produzido é generalizável. Em relação aos atributos propostos na literatura, importa referir que como o classificador irá ser pensado como um produto exclui-se a utilização de atributos de host (cada lookup demora um segundo, impossível de obter gratuitamente em metade dos casos) e de atributos relacionados aos motores de busca (risco de bloqueio de IP). Como forma de recuperar algum do desempenho perdido, no modelo de classificação a desenvolver vai-se explorar os dados analiticamente e também usar o conhecimento no domínio para criar novos atributos lexicais (por exemplo, a entropia do URL, a contagem de carateres de pontuação, etc.) e de reputação (por exemplo, verificar a presença de domínios frequentes de phishing em listas negras, fazer uso da recente ferramenta do Open Page Rank [26] para obter métricas do page rank das páginas web, etc.).

Por fim, é também objetivo deste trabalho perceber como é que o modelo de classificação pode ser integrado num sistema para deteção de phishing em tempo real. Como os clientes estão cada vez mais exigentes, outro desafio será explorar o modelo machine learning de classificação desenvolvido para tentar fornecer explicações para as classificações dos URLs.

5. Especificação de Requisitos

O mecanismo de proteção contra phishing desenvolvido, consiste num sistema de back-end disponibilizado sob a forma de uma API REST, com uma interface web para visualização e gestão de conteúdos bloqueados e permitidos. Este capítulo apresenta os requisitos funcionais e não funcionais do sistema desenvolvido, e as restrições técnicas e de negócio atendidas no desenho da arquitetura.

5.1. Requisitos funcionais

Os requisitos funcionais do sistema desenvolvido foram levantados a partir de reuniões internas realizadas, tendo em consideração os objetivos da empresa para o projeto, e também o estudo de sistemas relacionados durante o estado da arte. Depois de levantados, os requisitos foram organizados em back-end e backoffice, e descritos com a seguinte estrutura: um id de identificação, um título, uma descrição e um nível de prioridade definido de acordo com a importância do requisito para o projeto. O nível de prioridade foi definido de acordo com o método MoSCoW [37], que divide os requisitos nas seguintes categorias:

- Must: o requisito deve ser implementado no sistema;
- Should: o requisito deveria ser implementado no sistema;
- Could: o requisito é desejável embora não seja necessário nesta fase;
- Won't: o requisito não precisa de ser implementado nesta fase.

De seguida apresenta-se os requisitos funcionais do sistema de acordo com a estrutura mencionada anteriormente, começando com os requisitos referentes ao sistema de back-end e depois os requisitos referentes ao backoffice de administração.

O sistema de back-end deve ter os seguintes requisitos funcionais:

ID: RF1
Título: Mecanismo de autenticação
Descrição: a API REST que expõe o serviço deve receber um login e uma password de administrador previamente definidos, para devolver um token de acesso válido às funcionalidades da API por um determinado período de tempo.
Prioridade: Must

ID: RF2
Título: Detetor de mensagens maliciosas
Descrição: a API REST que expõe o serviço deve receber como parâmetros de entrada o token de acesso, o tipo de mensagem e o conteúdo da mensagem. Após receber o pedido e validar os parâmetros de entrada, deve ser capaz de detetar URLs em texto, identificar se algum dos URLs presentes é malicioso e, por fim, devolver uma resposta a indicar se a mensagem deve ou não ser bloqueada.
Prioridade: Must

ID: RF3
Título: Classificador de URLs
Descrição: o sistema deve conter um classificador baseado em machine learning capaz de determinar se um URL é malicioso ou seguro.
Prioridade: Must

ID: RF4
Título: Retreinar periodicamente o modelo de classificação
Descrição: o sistema deve retreinar periodicamente o modelo de classificação, para que este esteja continuamente a aprender e se mantenha atualizado ao longo do tempo.
Prioridade: Must

ID: RF5
Título: Reanalisar periodicamente os URLs
Descrição: o sistema deve periodicamente fazer uma reanálise aos URLs da base de dados, para detetar se algum foi comprometido.
Prioridade: Must

O backoffice de administração deve ter os seguintes requisitos funcionais:

ID: RF6
Título: Autenticação de um administrador
Descrição: o administrador do sistema deve poder autenticar-se para ter acesso às funcionalidades do backoffice de administração.
Prioridade: Must

ID: RF7
Título: Visualização da lista de URLs bloqueados e permitidos
Descrição: o administrador do sistema deve poder visualizar a lista de URLs bloqueados e a lista de URLs permitidos.
Prioridade: Must

ID: RF8
Título: Adição e remoção de URLs
Descrição: o administrador do sistema deve poder adicionar e remover URLs tanto na lista de URLs bloqueados como na lista de URLs permitidos.
Prioridade: Must

ID: RF9
Título: Edição e correção de falsos positivos e de falsos negativos
Descrição: o administrador do sistema deve poder editar e corrigir falsos positivos na lista de URLs bloqueados, e editar e corrigir falsos negativos na lista de URLs permitidos.
Prioridade: Must

ID: RF10
Título: Explicação da classificação dos URLs
Descrição: o administrador do sistema deve poder consultar a explicação da classificação de qualquer URL em ambas as listas.
Prioridade: Must

ID: RF11
Título: Ordenamento e filtragem de URLs
Descrição: o administrador do sistema deve poder ordenar e filtrar os URLs de ambas as listas por qualquer um dos atributos.
Prioridade: Must

ID: RF12
Título: Definição das configurações do classificador

Descrição: o administrador do sistema deve poder definir o grau de confiança com que as classificações de URLs são aceites, e a API de verificação a usar quando a confiança das classificações for inferior à definida.
Prioridade: Must

ID: RF13
Título: Monitorização do modelo de classificação
Descrição: o administrador do sistema deve poder monitorizar a precisão e outras métricas do modelo de classificação para alertar a equipa caso estas se estejam a degradar.
Prioridade: Won't

5.2. Requisitos não funcionais

Os requisitos não funcionais [38] também denominados como atributos de qualidade definem-se como caraterísticas que o sistema deve possuir em adição às suas funcionalidades. Foram identificados e considerados cruciais para este sistema os seguintes requisitos não funcionais:

- **Desempenho**: é a capacidade de um sistema cumprir as suas funções no tempo especificado. Sendo este um sistema de análise de mensagens em tempo real para deteção de URLs maliciosos, o desempenho será um aspeto importante neste problema.
- **Escalabilidade**: é a capacidade de um sistema lidar com quantidades crescentes de trabalho. Sendo este um sistema que no futuro poderá vir a lidar com milhares de pedidos em simultâneo, espera-se que a sua arquitetura esteja preparada para suportar escalabilidade horizontal.
- **Interoperabilidade**: é a capacidade de um sistema poder comunicar e trocar informações com outro sistema. Sendo este um novo sistema, espera-se que este possa ser facilmente integrado com qualquer outro sistema desde que o formato da comunicação seja respeitado.

Nas tabelas seguintes apresentam-se alguns cenários para demostrar a forma como os requisitos não funcionais vão ter influência sobre o sistema.

Cenário de desempenho	
Fonte do estímulo	Exterior ao sistema
Estímulo	Chegada de um pedido à API do sistema
Ambiente	Sistema em funcionamento normal na máquina de desenvolvimento
Artefacto	O sistema
Resposta	O sistema processa o pedido corretamente
Medida da resposta	Qualquer pedido é respondido em menos de 100 milissegundos

Tabela 12 - Cenário de desempenho

Cenário de escalabilidade	
Fonte do estímulo	Exterior ao sistema
Estímulo	Chegada de um número elevado de pedidos à API do sistema
Ambiente	Sistema em condições de stress na máquina de desenvolvimento
Artefacto	O sistema
Resposta	O sistema processa os pedidos corretamente, mantendo o desempenho em condições de stress

Medida da resposta	O sistema responde aos pedidos dentro do tempo especificado no cenário anterior quando sobrecarregado com 300 pedidos em simultâneo

Tabela 13 - Cenário de escalabilidade

Cenário de interoperabilidade	
Fonte do estímulo	Exterior ao sistema
Estímulo	Chegada de um pedido à API do sistema
Ambiente	Sistema em funcionamento normal na máquina de desenvolvimento
Artefacto	O sistema
Resposta	O sistema recebe o pedido e processa-o corretamente
Medida da resposta	Todos os pedidos são processados corretamente desde que o formato da comunicação seja respeitado

Tabela 14 - Cenário de interoperabilidade

5.3. Restrições técnicas e de negócio

As restrições técnicas e de negócio são decisões impostas que devem ser atendidas no desenho da arquitetura.

As restrições técnicas que o sistema deve satisfazer são as seguintes:

- Tecnologias open source: o sistema deve ser implementado com tecnologias e linguagens de programação open source, para não acarretar custos para a empresa;
- Estilo arquitetural REST: o sistema deve seguir o estilo arquitetural REST, de modo a permitir a integração futura com outros sistemas;
- Serviço REST: o serviço REST que expõe o serviço de back-end deve ser implementado com recurso à framework Spring do Java;
- Bases de dados: o sistema deve utilizar um motor de base de dados PostgreSQL ou Oracle, para estar alinhado com o que é usado na empresa.

As restrições de negócio que o desenvolvimento do sistema deve satisfazer são as seguintes:

- Testes ao sistema: os testes ao sistema desenvolvido não podem implicar custos em máquinas, nem poderão ser testados em produção;
- Prazo de entrega: o sistema deve estar terminado até dia 14 de Junho, data em que o estágio termina;
- Confidencialidade dos dados: no início do estágio foi assinado um acordo de confidencialidade de produtos e de dados disponibilizados que tem de ser cumprido, e como tal, o acesso a dados apenas pode ser feito dentro da rede interna da empresa;
- Conformidade com o RGPD: o sistema desenvolvido deve estar em conformidade com o RGPD, criado para proteger os utilizadores face ao tratamento de dados pessoais na união europeia. Ou seja, a informação relativa a quem envia e recebe as mensagens nunca poderá ser guardada.

6. Arquitetura

A arquitetura de software de um sistema é o conjunto de estruturas necessárias para se compreender o sistema, que compreende elementos de software e as relações entre eles [38]. O desenho da arquitetura tem várias vantagens, como por exemplo, comunicar aos stakeholders o que está a ser desenvolvido, gerir a evolução e as modificações à medida que o sistema evolui, e permitir ao gestor de projeto prever os custos e fazer o planeamento. Com base no modelo arquitetural C4 [39], neste capítulo descreve-se a arquitetura proposta para o sistema através de um conjunto de vistas relevantes que o dividem em múltiplas representações, e que demonstram como o sistema suporta os requisitos especificados no capítulo anterior.

6.1. Diagrama de contexto

A Figura 5 mostra onde o sistema desenvolvido neste projeto se situa, e os sistemas e utilizadores com os quais interage.

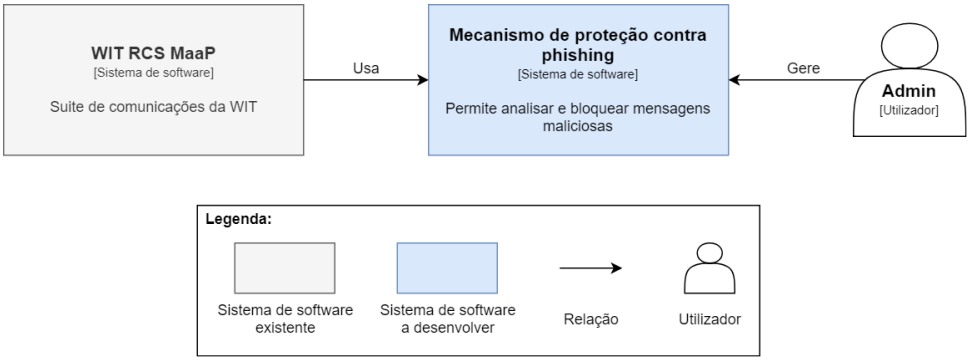


Figura 5 - Diagrama de contexto

O sistema de proteção contra phishing desenvolvido neste projeto será usado pelas comunicações que passam no WIT RCS MaaP, para analisar e bloquear mensagens maliciosas, com o intuito de evitar qualquer tentativa de phishing e execução de código malicioso nos aparelhos dos clientes. Para além disso fornece ao administrador do sistema uma interface web para visualização e gestão de URLs. No futuro quando este sistema for integrado terá como finalidade a análise das mensagens de texto trocadas entre os utilizadores, conhecido como messaging P2P e sem quaisquer conversas de grupo.

6.2. Diagrama de containers

A Figura 6 dá uma visão geral do sistema desenvolvido, dos containers que o compõem e como estes comunicam entre si, e também das principais opções tecnológicas tomadas.

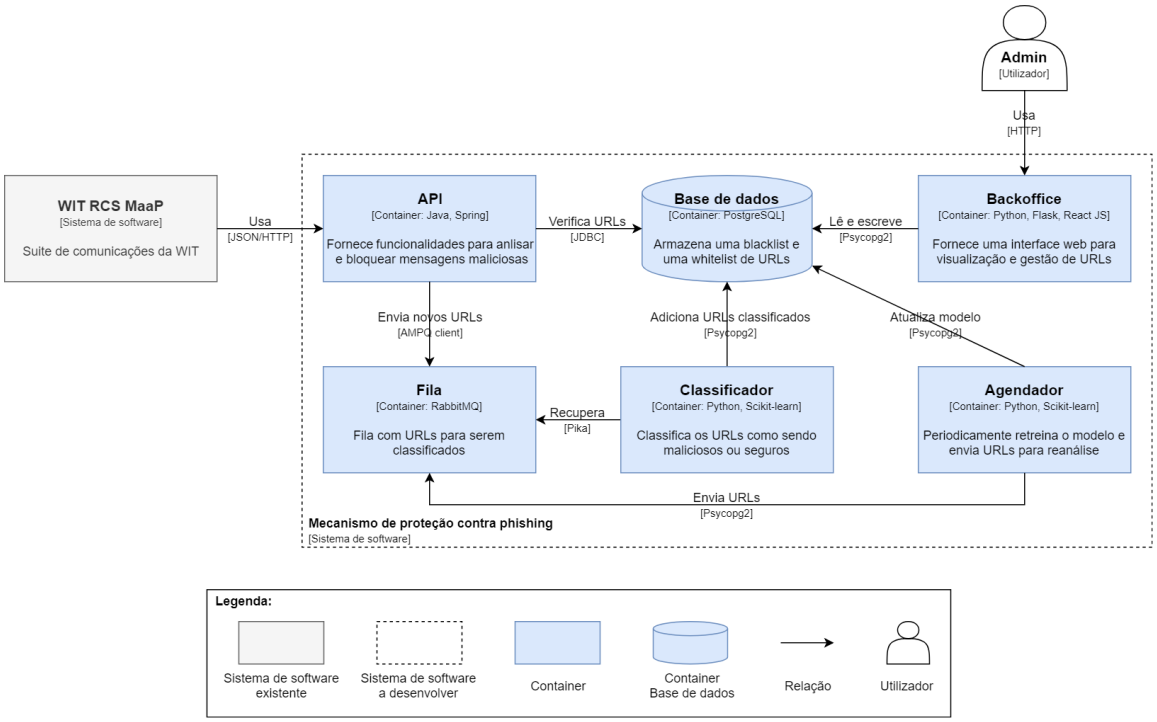


Figura 6 - Diagrama de containers

O sistema de proteção contra phishing desenvolvido é constituído por seis containers, uma API, um backoffice, um classificador, um agendador, uma base de dados e uma fila. A **API** é responsável por fornecer ao sistema WIT RCS MaaP funcionalidades para analisar e bloquear mensagens maliciosas. Este container comunica com dois containers, a base de dados onde são verificados os URLs da mensagem, e a fila para onde são enviados os novos URLs que aparecem para serem analisados e classificados. O **backoffice** é responsável por fornecer ao administrador do sistema uma interface web para visualização e gestão de URLs. Este container comunica com o container da base de dados onde lê e escreve informação. O **classificador** é responsável por classificar URLs como sendo maliciosos ou seguros. Este container recupera URLs para análise da fila, classifica-os e, por fim, adiciona-os à base de dados na respetiva lista. O **agendador** é responsável por periodicamente retreinar o modelo de classificação e enviar URLs para serem reanalisados. Este container comunica com a base de dados onde lê os URLs para retreinar e atualizar o modelo de classificação, e também para ler os URLs que serão enviados para a fila para serem reanalisados. A **base de dados** é responsável por armazenar uma blacklist e uma whitelist de URLs. A **fila** é responsável por manter os URLs para serem analisados e classificados. Este container serve para a API e o agendador (os emissores) comunicarem assincronamente com o classificador (o recetor), evitando que os emissores fiquem bloqueados e que o recetor esteja permanentemente a ler URLs. Desta forma garante-se desacoplamento, tolerância a falhas, escalabilidade e balanceamento de carga uma vez que cada instância do classificador apenas analisa um URL de cada vez.

A comunicação entre os containers é feita da seguinte forma, a comunicação entre o sistema WIT RCS MaaP e a API é feita por API REST através de JSON usando o protocolo HTTP, a comunicação entre a API, o backoffice, o classificador e o agendador com a base de dados é feita por um conector de acesso, e a comunicação entre a API, o classificador e o agendador com a fila é feita através de um cliente de acesso.

Por último, importa referir as principais decisões arquiteturais tomadas de forma a suportar os requisitos não funcionais descritos no capítulo anterior. Quanto ao desempenho, para reduzir a latência na análise de uma mensagem, optou-se em tempo real por verificar os URLs

diretamente na blacklist ao invés de classificar os novos URLs que aparecem, com o custo de deixar passar o primeiro URL que poderá ser malicioso, embora, qualquer URL que resulte do incremento ou de uma reprodução de um URL malicioso já conhecido seja bloqueado à primeira. Esta decisão permite a realização da classificação à posterior com informação mais enriquecida, e ainda expandir os URLs encurtados para poderem ser classificados. Quando à escalabilidade optou-se por uma fila para garantir o balanceamento dos pedidos de classificação de URLs em picos de carga. Esta decisão permite que possam ser lançadas várias instâncias do classificador. A API que expõe o serviço de back-end pode ser também ela escalada no futuro através do lançamento de várias instâncias com um balanceador de carga para distribuir os pedidos. Quando à interoperabilidade optou-se por uma API REST para poder ser invocada pelos sistemas externos, o que também satisfaz a restrição técnica referente ao uso do estilo arquitetural REST.

6.3. Diagrama de módulos da API

A Figura 7 faz zoom-in ao container da API para mostrar os módulos que o constituem, as suas responsabilidades e como interagem entre si.

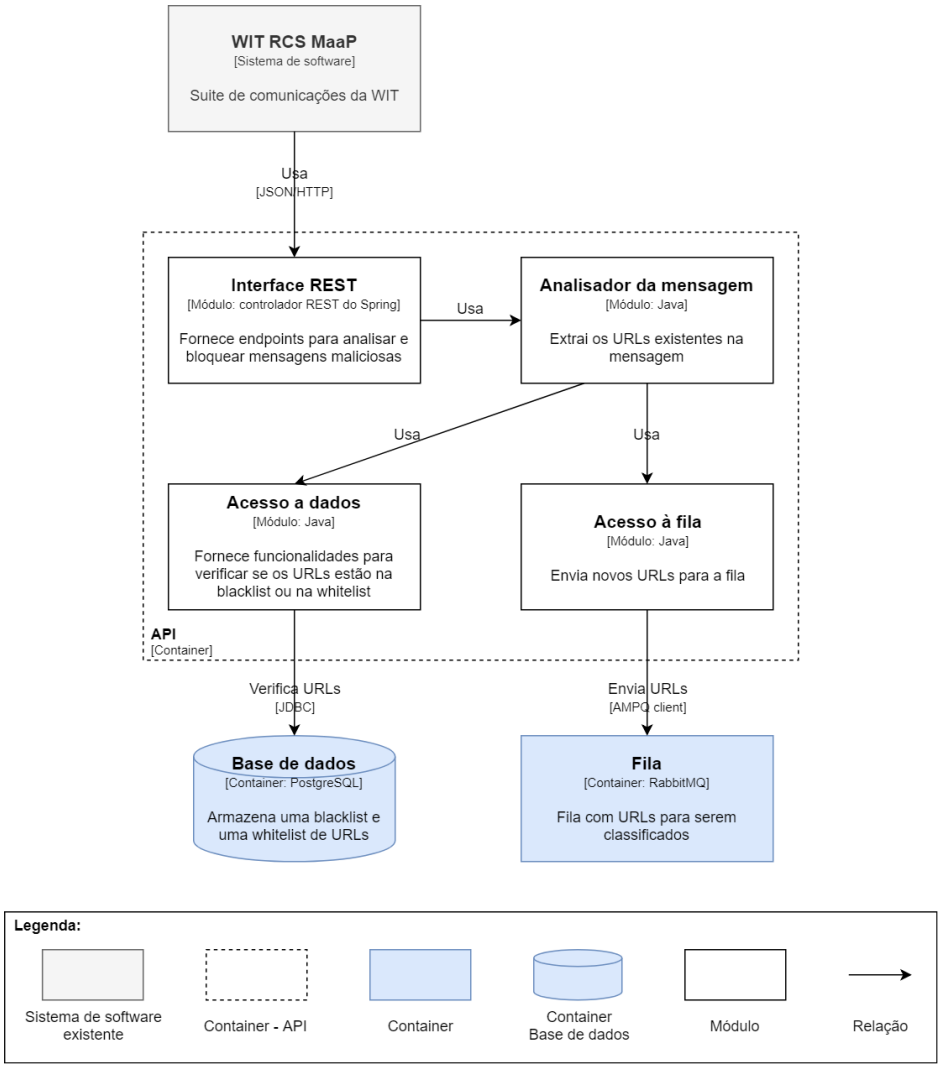


Figura 7 - Diagrama de módulos da API

O container da API é constituído por quatro módulos, uma interface REST, um analisador da mensagem, um acesso a dados, e um acesso à fila. A **interface REST** é responsável por

fornecer ao sistema WIT RCS MaaP um conjunto de endpoints através de uma API REST, para analisar e bloquear mensagens maliciosas. O **analisador da mensagem** é responsável por extrair e verificar todos os URLs existentes na mensagem. O **acesso a dados** é responsável por conectar à base de dados e fornecer funcionalidades para verificar se um determinado URL está na blacklist ou na whitelist. O **acesso à fila** é responsável por conectar à fila e por enviar URLs para a fila. O controlador da interface REST sempre que recebe um pedido, pede ao analisador da mensagem para extrair todos os URLs existentes na mensagem. De seguida verifica esses URLs na blacklist e devolve imediatamente uma resposta ao pedido. Essa resposta indica para bloquear a mensagem se pelo menos um dos URLs estiver na blacklist, caso contrário a mensagem não é bloqueada. Após responder ao pedido, outros URLs existentes na mensagem são verificados na whitelist e, caso não estejam, são enviados para a fila para que sejam analisados, classificados e adicionados à base de dados. No anexo A.1 apresenta-se um diagrama de atividade que explica graficamente o funcionamento da API.

6.4. Diagrama de módulos do backoffice

A Figura 8 faz zoom-in ao container do backoffice para mostrar os módulos que o constituem, as suas responsabilidades e como interagem entre si.

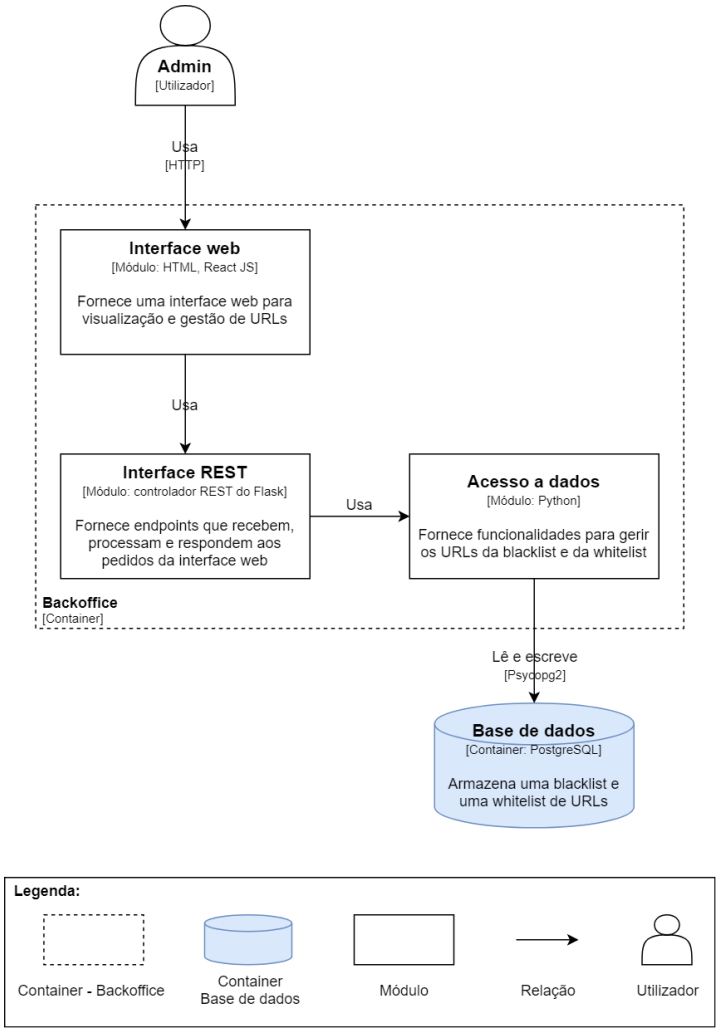


Figura 8 - Diagrama de módulos do backoffice

O container do backoffice é constituído por três módulos, uma interface web, uma interface REST, e um acesso a dados. A **interface web** é responsável por fornecer ao administrador

do sistema uma interface web para visualização e gestão de URLs. A **interface REST** fornece um conjunto de endpoints através de uma API REST para receber, processar e responder aos pedidos da interface web em JSON. O **acesso a dados** é responsável por conectar à base de dados e fornecer à interface REST funcionalidades para gerir os URLs da blacklist e da whitelist.

6.5. Diagrama de módulos do classificador

A Figura 9 faz zoom-in ao container do classificador para mostrar os módulos que o constituem, as suas responsabilidades e como interagem entre si.

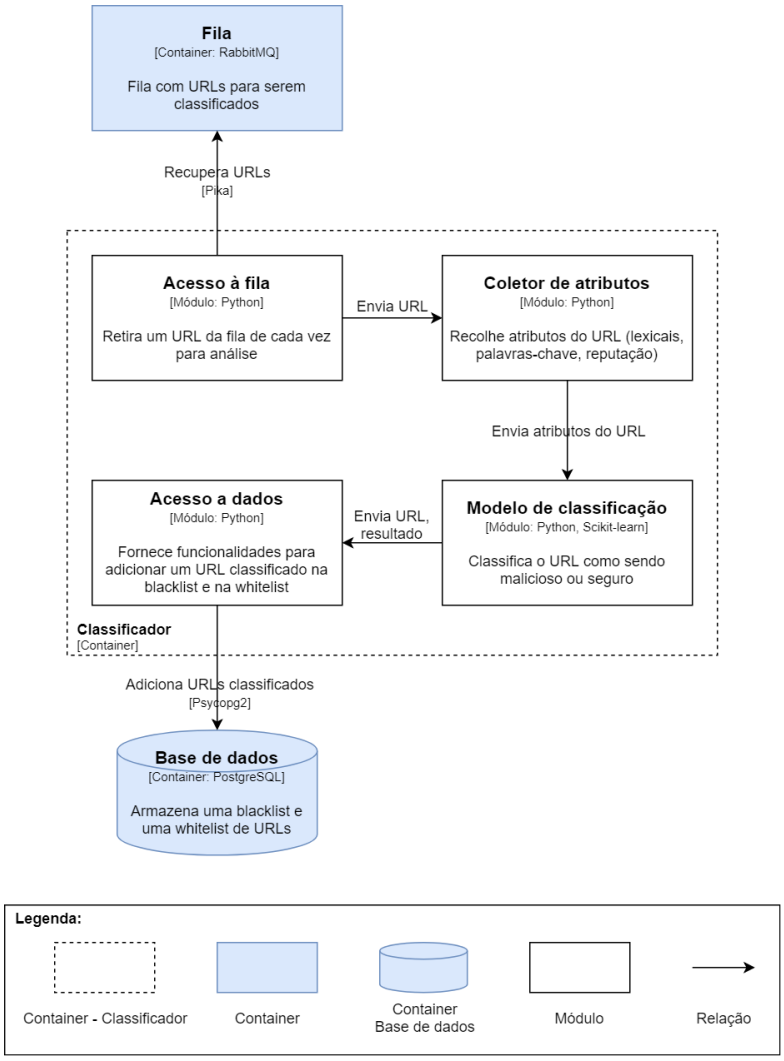


Figura 9 - Diagrama de módulos do classificador

O container do classificador é constituído por quatro módulos, um acesso à fila, um coletor de atributos, um modelo de classificação, e um acesso a dados. O **acesso à fila** é responsável por conectar à fila e retirar um URL da fila de cada vez para análise. O **coletor de atributos** é responsável por recolher os atributos do URL (lexicais, palavras-chave, reputação). O **modelo de classificação** com base no conjunto de atributos recolhidos classifica o URL como sendo malicioso ou seguro. O **acesso a dados** é responsável por conectar à base de dados e fornecer funcionalidades para adicionar um URL classificado na blacklist e na whitelist.

6.6. Diagrama de módulos do agendador

A Figura 10 faz zoom-in ao container do agendador para mostrar os módulos que o constituem, as suas responsabilidades e como interagem entre si.

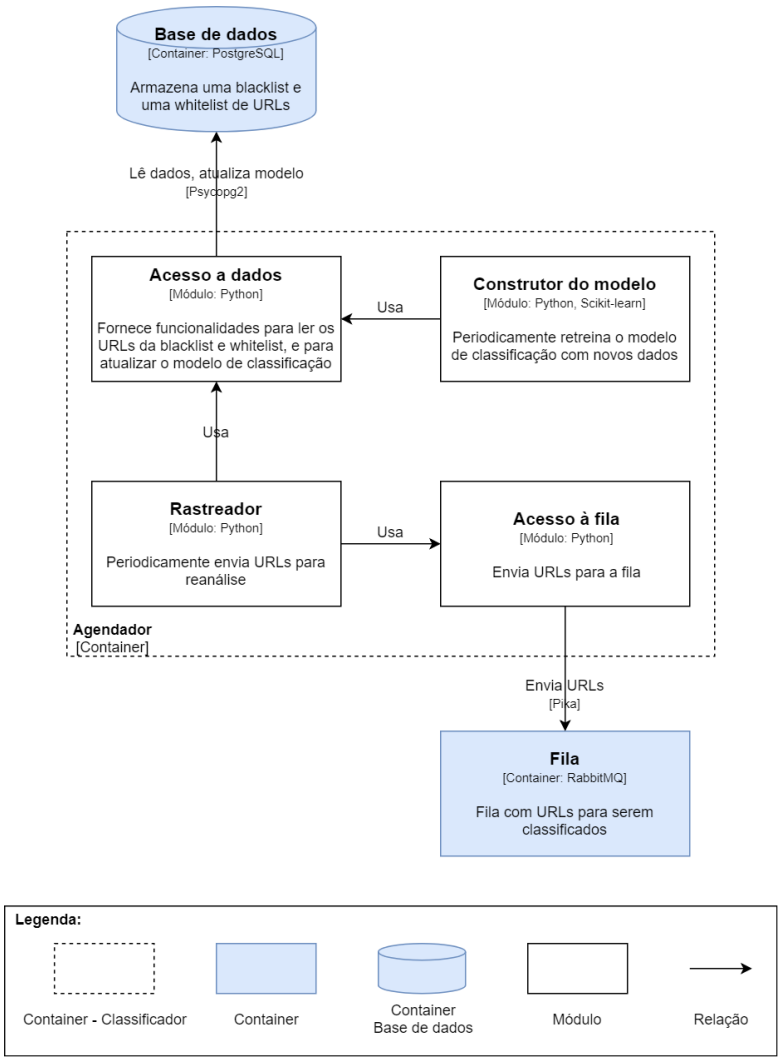


Figura 10 - Diagrama de módulos do agendador

O container do agendador é constituído por quatro módulos, um construtor do modelo, um rastreador, um acesso a dados, e um acesso à fila. O **construtor do modelo** é responsável por periodicamente retreinar o modelo de classificação com novos dados para o manter atualizado ao longo do tempo, e faz uso do acesso a dados para aceder a dados e gravar o modelo na base de dados. O **rastreador** é responsável por periodicamente enviar URLs para serem reanalisados para detetar se algum foi comprometido, e faz uso do acesso a dados para aceder a dados da base de dados e do acesso à fila para enviar URLs para análise. O **acesso a dados** é responsável por conectar à base de dados e fornecer funcionalidades para ler os URLs da blacklist e da whitelist, e para atualizar o modelo. O **acesso à fila** é responsável por conectar à fila e por enviar URLs para a fila.

6.7. Escolha de tecnologias

Apresentada a arquitetura do sistema, esta secção apresenta o racional por detrás da escolha das tecnologias adotadas para a implementação de cada um dos containers. Para o desenvolvimento da **API** que expõe o serviço de back-end de deteção de mensagens

maliciosas optou-se pela framework Spring [40] do Java, para cumprir a restrição técnica descrita na secção 5.3. Para o desenvolvimento do **backoffice** optou-se pelo Python, Flask [41] e React JS [42], por serem open source e por serem bastante flexíveis para a criação de backoffices. Para o desenvolvimento do **classificador** e do **agendador** que treina o modelo de classificação optou-se pela biblioteca Scikit-learn [33] do Python, porque de entre as ferramentas de machine learning analisadas aquando do estudo do estado da arte, esta foi a que mais se destacou. Para o motor de **base de dados** optou-se pelo PostgreSQL [43] para cumprir a restrição técnica descrita na secção 5.3. Para a **fila** optou-se pelo RabbitMQ [44] por ser o intermediário de mensagens open source mais utilizado no mercado segundo [44], e por suportar clientes de acesso tanto em Java como em Python. Para a troca de informações com o exterior optou-se pelo JSON, por ser um formato de dados constituído por uma coleção de pares chave-valor, leve e interoperável [45].

7. Implementação

Este capítulo apresenta o modelo de dados da base de dados e todos os detalhes de implementação dos módulos dos principais containers da arquitetura (API, backoffice, classificador e agendador).

7.1. Modelo de dados da base de dados

A Figura 11 apresenta o modelo de dados da base de dados, constituído por três tabelas, uma para a blacklist de URLs que contém os URLs maliciosos, uma para a whitelist de URLs que contém os URLs seguros, e uma para armazenar o modelo de classificação que é usado pelo classificador.

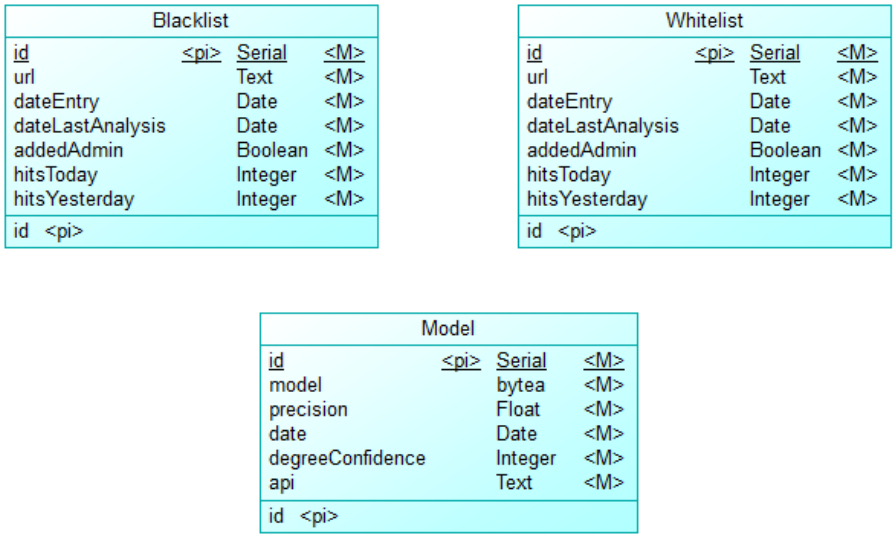


Figura 11 - Modelo de dados da base de dados

A tabela da blacklist e da whitelist são constituídas pelos mesmos sete atributos, um id de identificação, o URL, a data de entrada do URL na lista, a data da última análise efetuada ao URL, um atributo booleano que indica se o URL foi adicionado pelo administrador do sistema através do backoffice, a contagem do número de acessos efetuados ao URL hoje e um atributo para manter o registo do dia anterior. A tabela do modelo de classificação é constituída por seis atributos, um id de identificação, o ficheiro do modelo, a precisão do modelo, a data de criação do modelo, o grau de confiança com que as classificações são aceites, e a API de verificação a usar quando a confiança das classificações for inferior à definida. De referir que as tabelas da blacklist e da whitelist foram estrategicamente replicadas não só separar os dois tipos de URLs, mas também para favorecer o desempenho, uma vez que como a blacklist terá significativamente menos registos do que a whitelist, as queries à blacklist para verificação de URLs maliciosos em tempo real também serão significativamente mais rápidas.

7.2. Container da API

Nesta secção apresenta-se os detalhes de implementação dos quatro módulos (interface REST, analisador da mensagem, acesso a dados, e acesso à fila) que constituem o container da API, responsável por fornecer ao sistema WIT RCS MaaP funcionalidades para analisar e bloquear mensagens maliciosas.

7.2.1. Módulo da interface REST

O módulo da interface REST contém uma API REST que expõe o serviço de back-end, fornecendo endpoints ao sistema WIT RCS MaaP que permitem verificar se uma mensagem é maliciosa. Na Tabela 15 apresenta-se de forma resumida os URIs (Uniform Resource Identifier) fornecidos pela API REST.

Método HTTP	URI	Descrição
POST	/login	Solicita um token de acesso à API
POST	/message	Verifica se uma mensagem é maliciosa

Tabela 15 - URIs da API

Mais detalhes sobre estes URIs da API REST, como os parâmetros recebidos, as respostas JSON e tipos de códigos HTTP que podem ser devolvidos, podem ser consultados no anexo B.1. Este módulo implementa ainda os mecanismos de segurança e de deteção de falhas a fim de:

- Impedir o acesso à API por sistemas externos não autorizados através de um mecanismo de autenticação baseado num token JWT (JSON Web Token) [46]. Ou seja, o sistema externo inicialmente autentica-se perante a API com um login e uma password de administrador previamente definidos para obter um token de acesso às funcionalidades por um determinado período de tempo. O token é depois enviado no cabeçalho dos pedidos HTTP subsequentes e é validado pela API antes de processar o pedido. Caso não seja válido o acesso às funcionalidades protegidas é negado.
- Detetar pedidos com parâmetros de entrada inválidos, para assegurar que os parâmetros de entrada de um determinado pedido são válidos.
- Detetar falhas no lado do servidor, como por exemplo a ocorrência de um erro interno no servidor ou uma falha no acesso à base de dados. Desta forma garante-se que na ocorrência de um erro inesperado o pedido nunca fica sem resposta.

7.2.2. Módulo do analisador da mensagem

O módulo do analisador da mensagem quando invocado pela lógica do endpoint que recebe o pedido começa por verificar a existência de URLs na mensagem. A técnica para verificar e extrair eventuais URLs existentes na mensagem passou por obter todas as palavras do texto da mensagem, e depois verificar se alguma dessas palavras forma um URL válido com recurso à classe URL do Java [47], ou seja, se a instanciação desta classe com uma determinada palavra for bem sucedida é porque a palavra é um URL válido e bem formado, e como tal é adicionado a um vetor. Se o vetor estiver vazio o módulo indica à lógica do endpoint para devolver uma resposta ao pedido que indique que a mensagem não deve ser bloqueada. Se o vetor contiver pelo menos um URL, eles são verificados na blacklist por um método fornecido pelo módulo de acesso a dados. Caso um URL esteja na blacklist ou resulte do incremento de um URL já existente na blacklist, o módulo imediatamente indica à lógica do endpoint para devolver uma resposta ao pedido que indique que a mensagem deve ser bloqueada. Ao mesmo tempo uma thread de uma pool de threads que é inicializada no arranque da API [48], é usada para verificar se algum dos restantes URLs não está nem na blacklist nem na whitelist.

Para detetar que um URL (excetuando encurtados) é um incremento ou uma reprodução de um URL já existente na blacklist, são analisados os URLs da blacklist para verificar se o URL apresenta uma variação em algum dos componentes do URL em que o hacker tem controlo total, nomeadamente o path, os parâmetros e a ancora, onde o hacker pode definir quaisquer valores para eles para criar um novo URL. Deste forma evita-se que seja feito bypass ao

sistema, bloqueando em tempo real qualquer URL que esteja a ser incrementado ou reproduzido a partir de um URL malicioso já conhecido. Quanto ao uso de threads, foi a forma encontrada para verificar assincronamente outros URLs existentes na mensagem e ao mesmo tempo não bloquear o envio da resposta ao pedido. Estes URLs são verificados se não estão nem na blacklist nem na whitelist por um método fornecido pelo módulo de acesso a dados e, caso existam novos, eles são enviados para a fila por um método fornecido pelo módulo de acesso à fila para serem analisados e classificados pelo classificador.

7.2.3. Módulo de acesso a dados

O módulo de acesso a dados é uma camada demarcada, que contém métodos para verificar a presença de URLs na blacklist e na whitelist. De forma a evitar o SQL Injection todos os parâmetros foram introduzidos nas queries através do método setParameter do driver JDBC [49], de acesso à base de dados PostgreSQL. Deste modo assegura-se a segurança no acesso a dados ao não permitir a introdução direta de parâmetros nas queries. Uma vez que a criação de conexões de acesso à base de dados é bastante dispendiosa, optou-se por configurar uma pool de conexões, ou seja, um mecanismo para criar e manter um conjunto de conexões prontas a utilizar e serem reutilizadas entre os diferentes pedidos que precisam de aceder à base de dados. Este mecanismo ajuda não só a reduzir a latência no processamento de determinadas mensagens, mas também melhora o desempenho geral deste container.

7.2.4. Módulo de acesso à fila

O módulo de acesso à fila é uma camada demarcada, que contém um método para enviar os novos URLs para a fila com prioridade elevada, a fim de serem analisados e classificados pelo classificador. Para aceder ao servidor do RabbitMQ recorreu-se ao cliente AMQP do Java [50], e à semelhança do módulo anterior foi também configurado um mecanismo para criar e manter um conjunto de conexões prontas a utilizar e serem reutilizadas entre os diferentes pedidos que precisam de aceder à fila.

7.3. Container do backoffice

Nesta secção apresenta-se os detalhes de implementação dos três módulos (interface web, interface REST, e acesso a dados) que constituem o container do backoffice, responsável por fornecer ao administrador do sistema uma interface web para visualização e gestão de URLs bloqueados e permitidos.

7.3.1. Módulo da interface web

O módulo da interface web foi desenvolvido recorrendo ao React JS [42], uma biblioteca de JavaScript para a construção de interfaces de utilizador baseadas em componentes reutilizáveis. Desta forma o React atualiza e renderiza eficientemente os componentes certos quando os seus dados forem alterados. Para tornar o front-end visualmente apelativo foi utilizada a framework de CSS - Bootstrap [51]. Para a criação de caixas de diálogo e alertas em casos de erro/sucesso programaticamente, usando os modais do Bootstrap recorreu-se à biblioteca do Bootbox.js [52]. Para obter dados ou para executar ações, a interface web invoca os métodos REST (apresentados na subsecção seguinte) utilizando pedidos AJAX com recurso à biblioteca do jQuery [53]. Os screenshots da interface web são apresentados na subsecção 7.3.4, após a descrição dos três módulos que constituem o container do backoffice.

7.3.2. Módulo da interface REST

O módulo da interface REST contém uma API REST que faz a ponte faz a ponte entre o acesso a dados e a interface web, interagindo com o CRUD e fornecendo uma interface REST para receber e responder aos pedidos vindos da interface web. Os recursos fornecidos pelo REST têm identificadores globais (URIs) que são acedidos através de uma interface padrão (HTTP), e trocam representações desses recursos através de objetos JSON. O servidor REST é stateless, ou seja, cada pedido traz todas as informações de contexto. Além disso, este módulo implementa ainda mecanismos de segurança e de deteção de falhas a fim de:

- Impedir o acesso a recursos REST por utilizadores não autorizados através de um mecanismo de autenticação baseado em sessão. Ou seja, quando o administrador faz login no backoffice, o seu identificador é adicionado à sessão. Sempre que um pedido é feito o servidor verifica se o seu identificador existe ou não na sessão. No caso de não existir a informação pretendida não é fornecida. Este mecanismo tem ainda a capacidade de invalidar a sessão ao fim de um determinado período de tempo de inatividade.
- Detetar pedidos com parâmetros de entrada inválidos, tanto para assegurar que os parâmetros de entrada de um pedido são válidos, como para evitar ataques ao servidor.
- Detetar falhas no lado do servidor, como por exemplo a ocorrência de um erro interno no servidor ou uma falha no acesso à base de dados. Desta forma garante-se que na ocorrência de um erro inesperado o administrador do sistema não fique sem resposta.

De seguida apresenta-se de forma resumida os URIs da API REST que foram agrupados em quatro tabelas distintas, a Tabela 16 com os URIs referentes à manipulação da blacklist, a Tabela 17 com os URIs referentes à manipulação da whitelist, a Tabela 18 com os URIs referentes à manipulação do classificador, e a Tabela 19 com os URIs mais gerais referentes às operações sobre o utilizador e à obtenção de páginas web e ficheiros JavaScript.

Método HTTP	URI	Descrição
GET	/blacklist	Obtém os URLs da blacklist
POST	/blacklist	Adiciona um novo URL à blacklist
PUT	/blacklist/{id}	Edita um URL da blacklist
DELETE	/blacklist/{id}	Remove um URL da blacklist

Tabela 16 - URIs da blacklist

Método HTTP	URI	Descrição
GET	/whitelist	Obtém os URLs da whitelist
POST	/whitelist	Adiciona um novo URL à whitelist
PUT	/whitelist/{id}	Edita um URL da whitelist
DELETE	/whitelist/{id}	Remove um URL da whitelist

Tabela 17 - URIs da whitelist

Método HTTP	URI	Descrição
GET	/classifier	Obtém as configurações do classificador
POST	/classifier	Edita as configurações do classificador

Tabela 18 - URIs do classificador

Método HTTP	URI	Descrição
GET	/	Obtém a página de login
POST	/login	Faz o login do utilizador, iniciando a sessão
GET	/logout	Faz o logout do utilizador, removendo a sessão

GET	/explain	Obtém a explicação da classificação de um URL
GET	/html/{path}	Obtém as páginas HTML do servidor
GET	/js/{path}	Obtém os ficheiros JavaScript do servidor

Tabela 19 - URIs gerais

Mais detalhes sobre os URIs da API REST, como os parâmetros recebidos, as respostas JSON e tipos de códigos HTTP que podem ser devolvidos, podem ser consultados no anexo B.2.

7.3.3. Módulo de acesso a dados

O módulo de acesso a dados é uma camada CRUD (Create Read Update Delete) totalmente demarcada, que contém o código que responde ao objetivo do pedido, ou seja, contém métodos para obter e manipular os dados relativos aos URLs da blacklist, aos URLs da whitelist e ao classificador. De forma a evitar o SQL Injection todos os parâmetros foram introduzidos nas queries através do método setParameter do conetor psycopg2 [54], de acesso à base de dados PostgreSQL. Deste modo assegura-se a segurança da camada de dados ao não permitir a introdução direta de parâmetros nas queries.

7.3.4. Screenshots da interface web

A interface web do backoffice está dividida em quatro páginas web distintas, login, blacklist, whitelist e classificador, que contêm os requisitos funcionais referentes ao backoffice de administração de prioridade mais elevada. Na Figura 12, na Figura 13, na Figura 14 e na Figura 15 apresentam-se os screenshots de cada uma destas páginas web desenvolvidas.

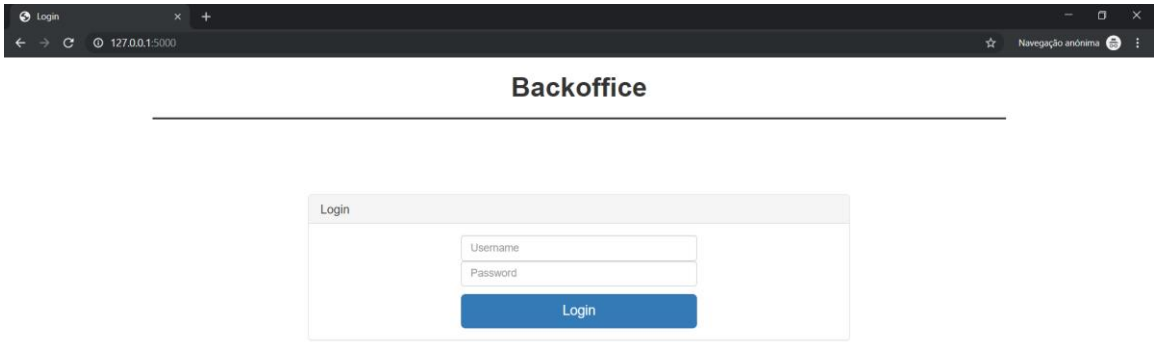


Figura 12 - Página de login

Esta página permite ao administrador do sistema autenticar-se com o login e a password de administrador previamente definidos, para ter acesso às funcionalidades do backoffice de administração.

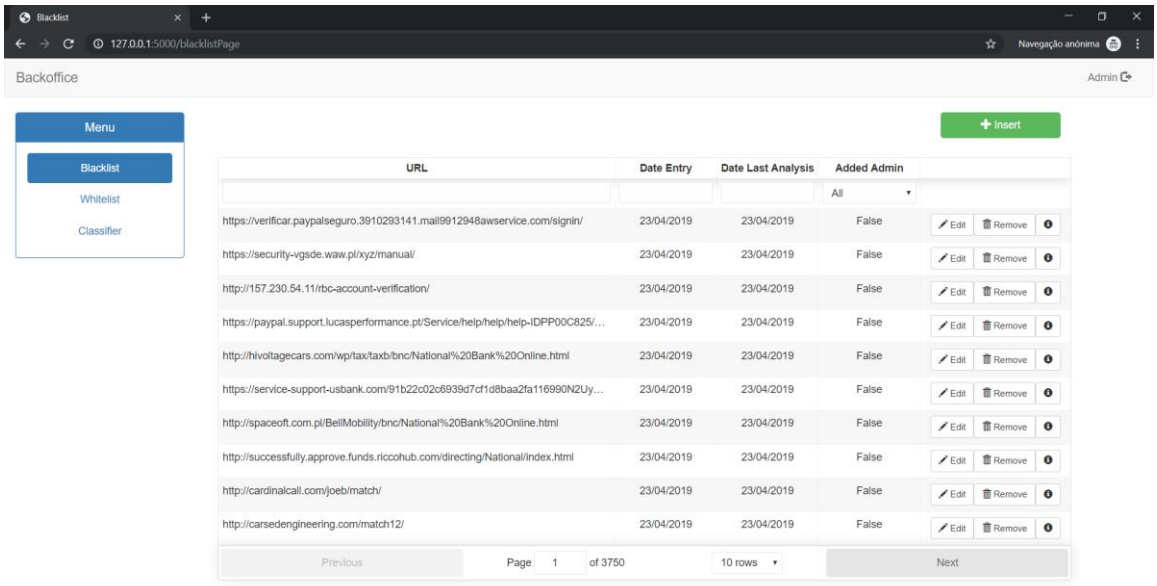


Figura 13 - Página da blacklist de URLs

Esta página está relacionada com a visualização e gestão dos URLs da blacklist, e fornece um conjunto de funcionalidades que permitem: visualizar a lista de URLs bloqueados, adicionar novos URLs, remover um URL, editar um URL, consultar a explicação da classificação de um URL, e ainda funcionalidades para permitir ordenar e filtrar os URLs por qualquer um dos atributos.

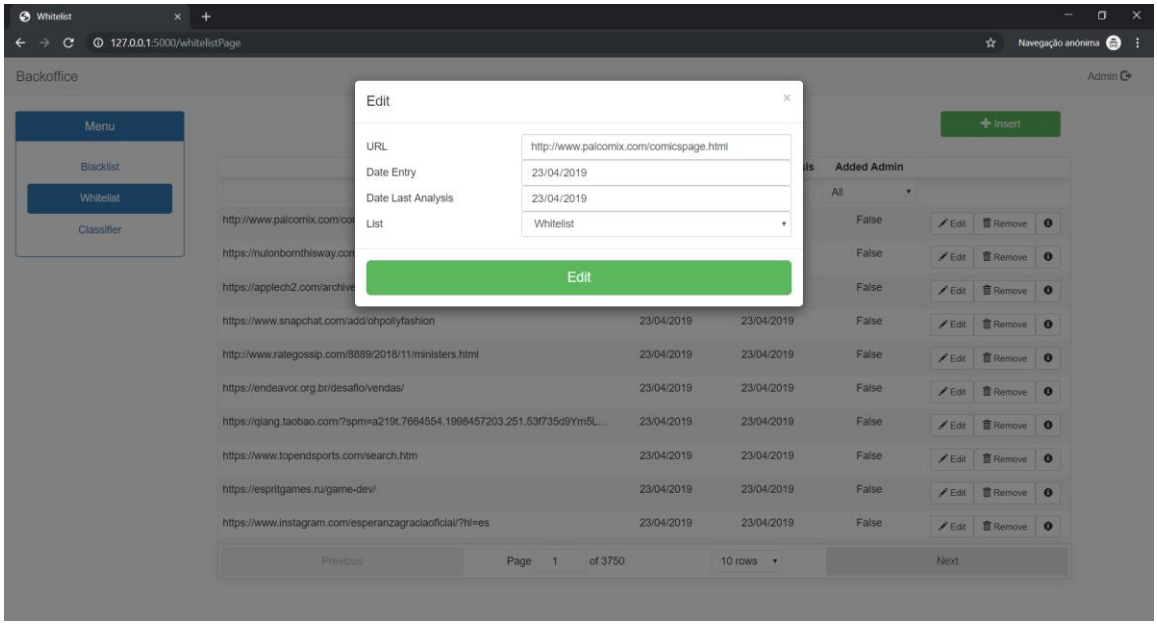


Figura 14 - Página da whitelist de URLs (com a edição de um URL)

Esta página está relacionada com a visualização e gestão dos URLs da whitelist, e fornece um conjunto de funcionalidades que permitem: visualizar a lista de URLs permitidos, adicionar novos URLs, remover um URL, editar um URL, consultar a explicação da classificação de um URL, e ainda funcionalidades para permitir ordenar e filtrar os URLs por qualquer um dos atributos.

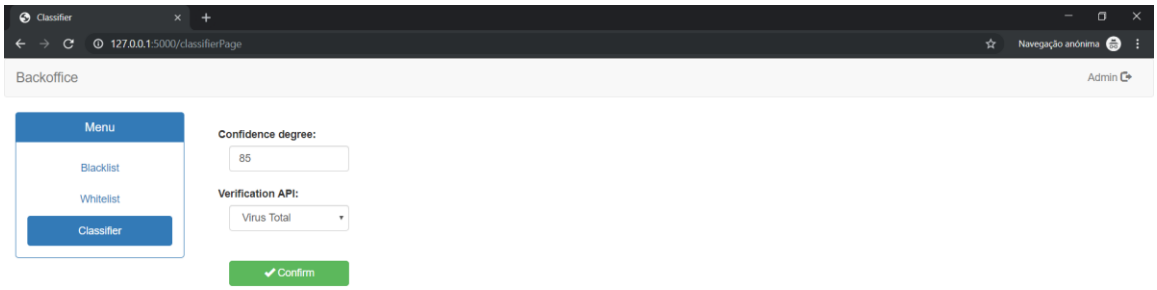
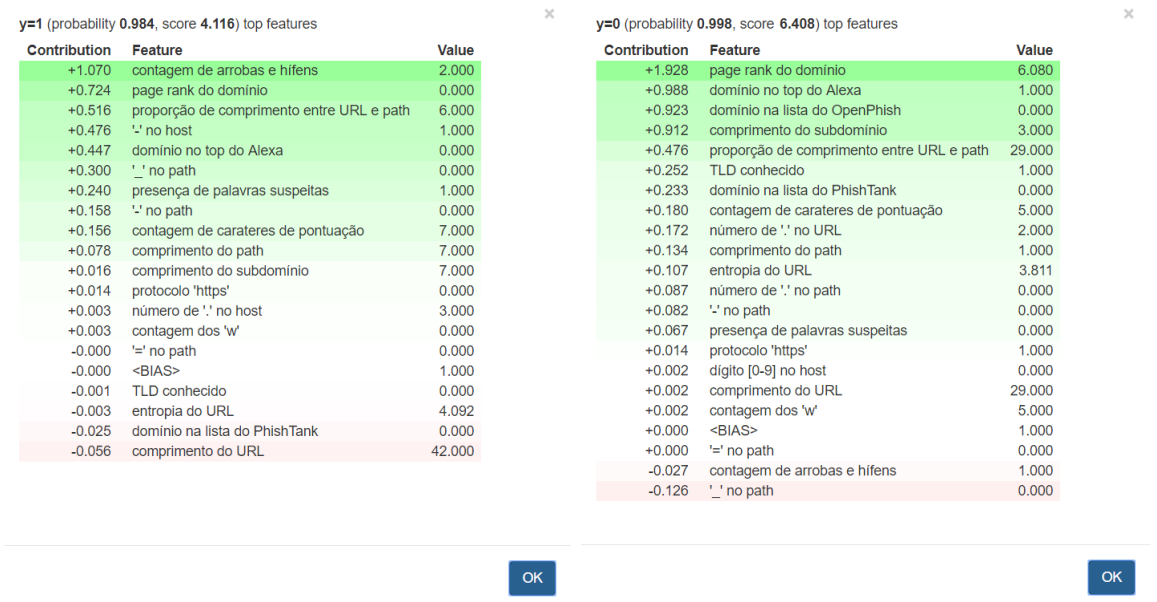


Figura 15 - Página do classificador

Esta página está relacionada com a gestão do classificador de URLs, e fornece um conjunto de funcionalidades que permitem: definir o grau de confiança do classificador e definir a API de verificação a usar quando a confiança das classificações for inferior à definida.

Ao clicar na info de um URL (terceiro botão, à direita do edit e do remove) em qualquer uma das listas é apresentada uma caixa de diálogo com a explicação da classificação do URL, que mostra o caminho da decisão e quais os atributos que contribuíram positivamente e negativamente para essa decisão. Na Figura 16 apresenta-se um screenshot com a explicação da classificação para o URL malicioso http://pay-pal.support-online.co.uk/login/ (caixa de diálogo da esquerda) e para o URL seguro https://www.wit-software.com/ (caixa de diálogo da direita).



y=1 (probability **0.984**, score **4.116**) top features

Contribution	Feature	Value
+1.070	contagem de arrobas e hífens	2.000
+0.724	page rank do domínio	0.000
+0.516	proporção de comprimento entre URL e path	6.000
+0.476	'-' no host	1.000
+0.447	domínio no top do Alexa	0.000
+0.300	'_' no path	0.000
+0.240	presença de palavras suspeitas	1.000
+0.158	'-' no path	0.000
+0.156	contagem de carateres de pontuação	7.000
+0.078	comprimento do path	7.000
+0.016	comprimento do subdomínio	7.000
+0.014	protocolo 'https'	0.000
+0.003	número de '.' no host	3.000
+0.003	contagem dos 'w'	0.000
-0.000	'=' no path	0.000
-0.000	<BIAS>	1.000
-0.001	TLD conhecido	0.000
-0.003	entropia do URL	4.092
-0.025	domínio na lista do PhishTank	0.000
-0.056	comprimento do URL	42.000

OK

y=0 (probability **0.998**, score **6.408**) top features

Contribution	Feature	Value
+1.928	page rank do domínio	6.080
+0.988	domínio no top do Alexa	1.000
+0.923	domínio na lista do OpenPhish	0.000
+0.912	comprimento do subdomínio	3.000
+0.476	proporção de comprimento entre URL e path	29.000
+0.252	TLD conhecido	1.000
+0.233	domínio na lista do PhishTank	0.000
+0.180	contagem de carateres de pontuação	5.000
+0.172	número de '.' no URL	2.000
+0.134	comprimento do path	1.000
+0.107	entropia do URL	3.811
+0.087	número de '.' no path	0.000
+0.082	'-' no path	0.000
+0.067	presença de palavras suspeitas	0.000
+0.014	protocolo 'https'	1.000
+0.002	dígito [0-9] no host	0.000
+0.002	comprimento do URL	29.000
+0.002	contagem dos 'w'	5.000
+0.000	<BIAS>	1.000
+0.000	'=' no path	0.000
-0.027	contagem de arrobas e hífens	1.000
-0.126	'_' no path	0.000

OK

Figura 16 - Caixa de diálogo apresentada para explicar uma classificação

7.4. Container do classificador

Nesta secção apresenta-se os detalhes de implementação dos quatro módulos (acesso à fila, coletor de atributos, modelo de classificação, e acesso a dados) que constituem o container do classificador, responsável por classificar URLs como sendo maliciosos ou seguros.

7.4.1. Módulo de acesso à fila

O módulo de acesso à fila é uma camada demarcada, que contém um método assíncrono para recuperar um URL da fila de cada vez (os de prioridade mais elevada primeiro) e despoletar o processo de classificação que termina com a adição do URL na respetiva lista da base de dados. Para aceder ao servidor do RabbitMQ recorreu-se ao cliente AMQP do Python [55]. Após recuperar um URL o módulo verifica se o URL foi produzido através de serviços de encurtamento como por exemplo o bit.ly [56], o ow.ly [57], o tiny.cc [58], etc., e, em caso afirmativo, expande o URL para o real a fim de poder ser classificado corretamente. O URL é então passado ao módulo seguinte para que seja feita a recolha de atributos.

Para garantir que um URL nunca seja perdido, neste módulo foi implementado um mecanismo de confirmação de mensagens (conhecido como um ack), ou seja, sempre que um URL é recebido, processado e classificado com sucesso, um ack é enviado ao RabbitMQ para dizer que o URL pode ser excluído. Isto permite que se uma instância do classificador morrer (a conexão é fechada ou a conexão TCP é perdida) sem enviar uma confirmação, o RabbitMQ entender que o URL não foi processado completamente, e envia-o rapidamente para outra instância online do classificador.

7.4.2. Módulo do coletor de atributos

O módulo do coletor de atributos faz o parsing do URL nos seus diversos componentes e recolhe atributos lexicais, de palavras-chave e de reputação sobre o URL. Com este conjunto de atributos extraídos é gerado um vetor de atributos num formato legível do modelo de classificação que permita depois ao modelo analisar esses atributos e prever se o URL é malicioso ou seguro. Os atributos de cada uma destas categorias são apresentadas em detalhe no capítulo seguinte.

7.4.3. Módulo do modelo de classificação

O módulo do modelo de classificação com base no conjunto de atributos recolhidos pelo módulo anterior classifica o URL como sendo malicioso ou seguro. Quando a instância do classificador é inicializada, o módulo começa por recuperar da tabela do modelo da base de dados o modelo de classificação e o grau de confiança com que as classificações são aceites. Depois a partir do vetor de atributos que contém as caraterísticas/evidências recolhidas sobre o URL o modelo de classificação prevê as probabilidades da classe positiva e negativa para o URL e podem ocorrer três cenários:

- Se a probabilidade da classe positiva for maior ou igual ao grau de confiança da classificação a classificação é aceite.
- Se a probabilidade da classe negativa for maior ou igual ao grau de confiança da classificação a classificação é aceite.
- Se as probabilidades de ambas as classes forem inferiores ao grau de confiança da classificação significa que o classificador não tem a certeza da decisão, então o URL é verificado externamente com recurso à API do VirusTotal para que o modelo aprenda quando for retreinado.

A preferência pela API do VirusTotal que está pré-definida deveu-se ao facto de esta permitir fazer a análise do URL em tempo real, ao contrário das outras APIs que consultam as suas blacklists que poderão estar desatualizadas. Para além disso, como os resultados apresentados por esta API são a saída combinada de diferentes produtos antivírus e ferramentas de análise, é expetável que a verificação seja bastante precisa e confiável. No entanto, se o cliente quiser terá sempre a possibilidade de selecionar outra API na página do classificador do backoffice. Terminada a classificação o URL é passado ao módulo seguinte para que seja adicionada uma entrada na lista respetiva.

Para explicar o porquê da classificação, é extraído a partir da árvore de decisão do modelo de classificação, o caminho que levou à decisão, e quais os atributos que contribuíram positivamente e negativamente para essa decisão. Recordando o capítulo de Conhecimento Base, é claro que para cada decisão que a árvore faz existe um caminho da raiz da árvore até a um nó folha (que contém a decisão final, indicando se o URL é malicioso ou não), que consiste numa série de decisões binárias baseadas no atributo de cada nó. Ao percorrer esse caminho de decisão, cada nó da árvore tem uma pontuação de saída (um peso) e a contribuição de um atributo no caminho de decisão é o quanto a pontuação muda do nó pai para o nó filho. Os pesos de todos os atributos somam a pontuação de saída do modelo [59]. Mais detalhes serão apresentados no capítulo seguinte que é dedicado exclusivamente ao modelo de classificação.

7.4.4. Módulo de acesso a dados

O módulo de acesso a dados é uma camada demarcada, que contém um método para adicionar um URL classificado na blacklist ou na whitelist. Este método começa por verificar se o URL em questão já se encontra na blacklist ou na whitelist da base de dados. Caso se trate de um URL classificado como malicioso que não está presente em nenhuma das listas ele é inserido na blacklist. Se o URL já estiver presente na blacklist a data da sua última análise é atualizada. Se o URL se encontrar na whitelist, primeiro ele é removido da whitelist e depois é inserido na blacklist. Quando se trata de um URL classificado como seguro um processo semelhante é realizado para o caso inverso. Para um URL produzido através de serviços de encurtamento duas entradas são adicionadas na lista, uma para o URL encurtado e uma para o URL real. De forma a evitar o SQL Injection todos os parâmetros foram introduzidos nas queries através do método setParameter do conetor psycopg2 [54], de acesso à base de dados PostgreSQL. Deste modo assegura-se a segurança da camada de dados ao não permitir a introdução direta de parâmetros nas queries.

7.5. Container do agendador

Nesta secção apresenta-se os detalhes de implementação dos quatro módulos (construtor do modelo, rastreador, acesso a dados, e acesso à fila) que constituem o container do agendador, responsável por periodicamente retreinar o modelo de classificação e enviar URLs para serem reanalisados.

7.5.1. Módulo do construtor do modelo

O módulo do construtor do modelo todos os domingos às 5 horas da manhã (configurável) despoleta um processo para iniciar o processo de retreino e atualização do modelo de classificação com novos dados. Quando despoletado começa por ler os URLs da blacklist e da whitelist que não tenham sido adicionados pelo administrador com recurso ao método fornecido pelo módulo de acesso a dados, e etiqueta-os com a classe alvo correspondente. De seguida constrói uma matriz de atributos (lexicais, de palavras-chave e de reputação) extraídos

dos URLs num formato legível do modelo de classificação. De seguida treina um novo modelo, efetua uma avaliação da sua precisão recorrendo à técnica de cross-validation, e insere o novo modelo na tabela do Modelo com recurso ao método que atualiza o modelo de classificação também fornecido pelo módulo de acesso a dados. Desta forma garante-se que o modelo de classificação aprende continuamente novos padrões maliciosos nos dados e que se mantêm atualizado ao longo do tempo. Os atributos de cada uma destas categorias de atributos são apresentados em detalhe no capítulo seguinte. O leitor poderia questionar se o modelo seria na mesma atualizado caso a precisão se degrade, e a resposta é afirmativa uma vez que a questão foi discutida internamente e ficou decidido nesta fase do projeto não preocupar com esse assunto.

7.5.2. Módulo do rastreador

O módulo do rastreador todos os dias às 23 horas (configurável) despoleta um processo para iniciar o processo de envio dos URLs mais acedidos para reanálise. Quando despoletado começa por selecionar todos os URLs da whitelist que não tenham sido adicionados pelo administrador, e que tenham tido no dia atual um número de acessos pelo menos três vezes superior ao registo do dia anterior. Esses URLs são enviados para a fila com recurso ao método fornecido pelo método do módulo de acesso à fila. De seguida reinicia os contadores, ou seja, atribui ao atributo do dia de ontem o valor do dia atual para depois colocar o dia atual a zero.

7.5.3. Módulo de acesso a dados

O módulo de acesso a dados é uma camada demarcada, que contém um método para ler os URLs da blacklist e da whitelist que não tenham sido adicionados pelo administrador, e outro para atualizar o modelo de classificação na tabela do Modelo. Para aceder ao servidor de base de dados PostgreSQL recorreu-se ao conetor psycopg2 [54] do Python.

7.5.4. Módulo de acesso à fila

O módulo de acesso à fila é uma camada demarcada, que contém um método para enviar URLs para a fila com uma prioridade ligeiramente mais baixa do que a do container da API, a fim de serem reanalisados pelo classificador. Para aceder ao servidor do RabbitMQ recorreu-se ao cliente AMQP [55] do Python.

8. Modelo de Classificação

Este capítulo apresenta em detalhe todo o estudo, análise comparativa e experimentação realizada para o desenvolvimento do modelo de classificação usado pelo container do classificador, com o objetivo de desenvolver um modelo capaz de classificar novos URLs com a melhor precisão possível. O capítulo começa com uma visão geral da abordagem usada neste problema de classificação, seguido do conjunto de dados construído para treinar o modelo, seguido do processo de engenharia de atributos a partir do qual os atributos a usar no modelo foram selecionados, uma avaliação de classificadores onde foi escolhido o algoritmo de classificação, uma análise aos atributos para escolher o subconjunto ótimo de atributos mais relevantes e, por fim, a afinação do modelo final.

8.1. Abordagem

Para este problema de classificação binária recorreu-se à técnica de classificação de machine learning usando heurísticas e informações disponíveis no URL para classificar os URLs como sendo de phishing ou legítimos. Para isso foi desenvolvido um modelo de classificação usando a metodologia iterativa apresentada na Figura 4 do capítulo de Conhecimento Base, e teve-se como ponto de partida as abordagens e as diversas categorias de atributos estudadas e apresentadas no capítulo de Estado da Arte.

Para o desenvolvimento de um bom modelo é fundamental extrair atributos informativos que descrevam suficientemente o URL e, ao mesmo tempo, possam ser interpretados matematicamente pelos algoritmos machine learning de classificação. Por exemplo, usar diretamente a cadeia de carateres do URL não irá permitir aprender um bom modelo de previsão. A capacidade desses atributos em fornecer informações relevantes sobre o URL é essencial, já que a suposição subjacente aos modelos de machine learning é que as representações dos atributos entre os URLs de phishing e legítimos tenham distribuições diferentes. Como a qualidade dos atributos é crucial para a obtenção de um bom modelo de classificação, uma grande ênfase foi colocada na etapa de engenharia de atributos.

8.2. Conjunto de dados

Para treinar o modelo de classificação foi construído um conjunto de dados de 75.000 URLs, dos quais 37.500 são legítimos e 37.500 são de phishing. Os URLs de phishing foram recolhidos ao longo dos meses de Janeiro e Fevereiro de 2019 da lista negra do PhishTank [29], que é um serviço que contém URLs e informações sobre phishing na internet. Os URLs legítimos foram recolhidos do Common Crawl [60], que é um serviço que mantém um repositório aberto de petabytes de dados de rastreamento da web que têm vindo a ser recolhidos desde 2011. Na preparação do conjunto de dados de treino fez-se uma dupla verificação ao estado dos URLs usando a API do Google Safe Browsing nas 12 horas seguintes à sua recolha, uma vez que este é um serviço de lista negra que não é atualizado de imediato. Para além disso todos os URLs repetidos foram removidos e, os URLs produzidos através de serviços de encurtamento como por exemplo o bit.ly, o ow.ly, o tiny.cc, etc., foram detetados e expandidos para não criar nenhum tipo de enviesamento. Na Tabela 20 apresenta-se uma amostra de dez URLs do conjunto de dados, dos quais cinco são de phishing (classe positiva (1)) e cinco são legítimos (classe negativa (0)).

URL	Classe
https://mail.apaypal.co/Secure/PP/Paypal/	1

https://anuragjewellery.com/upd_CH_as/index.php	1
http://www.neymo.org/joomla/edu/zi.htm	1
https://www.s4rver.com/05291/verify.php	1
https://uniteddental.ca/new/trust/db/index.php	1
http://www.tutorialspoint.com/dbms/database_normalization.htm	0
https://www.dropbox.com/help/desktop-web/system-requirements	0
https://en.wikipedia.org/wiki/Typeface	0
https://www.foxnews.com/	0
http://web.mit.edu/ghudson/dev/third/emacs.aside/lisp/menu-bar.el	0

Tabela 20 - Amostra do conjunto de dados

Nesta amostra pode-se observar que os URLs de phishing e legítimos podem ser bastante semelhantes. Isto já era expectável, uma vez que o objetivo dos atacantes é ludibriar os utilizadores da web, tornando os URLs de phishing os mais genuínos possíveis.

8.3. Engenharia de atributos

Como atributos do modelo de classificação foram utilizados atributos lexicais, atributos de palavras-chave e atributos de reputação. Em cada uma destas categorias, não só se aproveitou os atributos mais relevantes propostos na literatura para o problema de deteção de URLs de phishing com machine learning apresentados no estado da arte, como também foram propostos novos atributos inferidos a partir da análise exploratória dos dados e usando o conhecimento no domínio. Importa referir que o classificador foi sempre pensado como um produto, e como tal evitou-se por exemplo a utilização de atributos baseados em motores de busca que exigem o scraping dos resultados de pesquisa, que atualmente é uma prática proibida com direito a bloqueio do IP. Durante o processo de engenharia de atributos procurou-se obter um número relativamente pequeno de atributos, não só para tornar os limites de decisão menos complexos e, portanto, menos propensos ao overfitting, como também para o processo de treino do modelo ser mais eficiente. Nas subsecções seguintes descreve-se brevemente cada uma destas três categorias de atributos que irão fornecer informações úteis sobre os URLs, para que os modelos treinados com algoritmos de classificação sejam capazes de detetar anomalias e determinados padrões maliciosos com uma grande probabilidade.

8.3.1. Lexicais

Foram usados inicialmente no modelo de classificação 30 atributos lexicais, sendo que 12 são de valor real e 18 são de valor binário que se apresenta na Tabela 21 e Tabela 22 respetivamente. Estes atributos lexicais têm como principal objetivo detetar padrões de ofuscação dos URLs usualmente empregues pelos atacantes, como por exemplo a ofuscação do host com um endereço IP, com um segundo domínio, com nomes grandes, etc. Alguns dos atributos foram retirados do capítulo de Estado da Arte, no entanto também foram inferidos novos atributos a partir da análise exploratória dos dados:

- Comprimento do subdomínio: os URLs de phishing por vezes tentam imitar o URL legítimo usando o seu domínio como subdomínio, logo tendem a ter um subdomínio mais longo.
- Entropia do URL: quanto maior a entropia de um URL, mais complexo ele é. Como os URLs de phishing tendem a ter texto aleatório, pode-se detetar pela entropia.
- Proporção de comprimento entre URL e path: os URLs de phishing tendem a ter uma proporção entre o comprimento do URL e do path maior do que os URLs legítimos.
- Contagem dos 'w': os URLs de phishing por vezes podem ter mais de 3 ‘w’.

- Contagem de arrobas e hífens: nos URLs tudo o que está à esquerda do @ é ignorado, por isso os URLs de phishing costumam usar isso para enganar os utilizadores.
- Contagem de carateres de pontuação: a contagem de '.', '!', '#', '$', '%', '&', ',' ';' '' ' '' no URL. Os URLs de phishing geralmente apresentam uma maior ocorrência de carateres de pontuação.
- TLD conhecido: o top 5 de TLDs (.com, .org, .ru, .net, .de, .gov) mais usados no mundo, extraído de [61]. Os URLs de phishing tendem a usar TLDs estranhos.
- Redirecionamento de barra dupla: a existência de '//' no path do URL faz com que o utilizador seja redirecionado para outro URL.

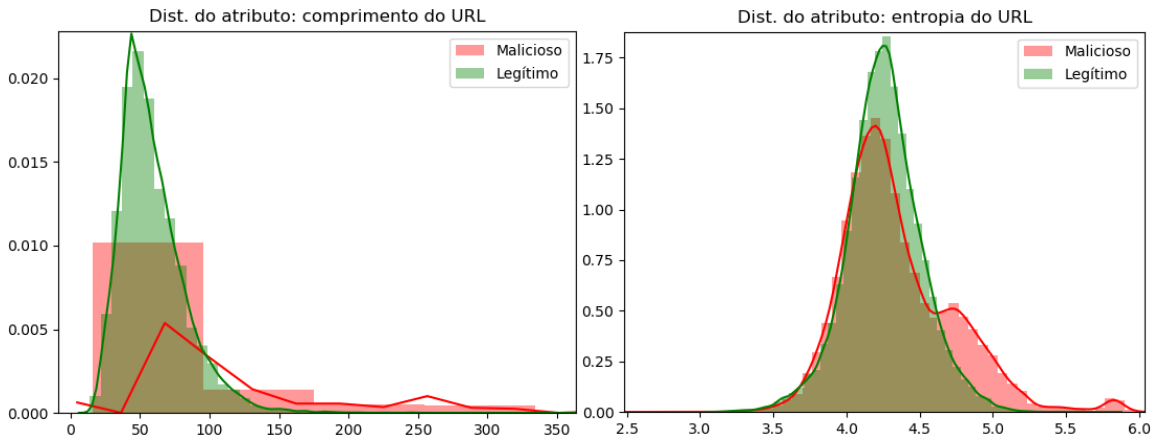
Descrição do atributo		
Comprimento do URL		Comprimento do subdomínio (novo)
Comprimento do host		Entropia do URL (novo)
Comprimento do path		Proporção de comprimento entre URL e path (novo)
Número de '.' no URL		Contagem dos 'w' (novo)
Número de '.' no host		Contagem de arrobas e hífens (novo)
Número de '.' no path		Contagem de carateres de pontuação (novo)

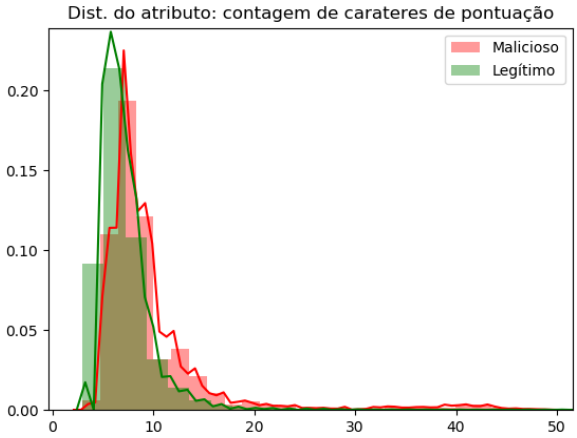
Tabela 21 - Resumo dos atributos lexicais de valor real

Descrição do atributo		
'-' no host		';' no path
Dígito [0-9] no host		Parte do parâmetro
Host baseado em IP		Parte da query
Host baseado em hex		'=' na parte da query
'-' no path		Parte do fragmento
'_' no path		'@' no URL
'/' no path		Protocolo 'https'
'=' no path		TLD conhecido (novo)
',' no path		Redirecionamento de barra dupla (novo)

Tabela 22 - Resumo dos atributos lexicais de valor binário

Nas próximas figuras apresenta-se uma comparação das distribuições na classe positiva e negativa de alguns dos atributos lexicais mais significativos, nomeadamente o atributo do comprimento do URL, da entropia do URL, e da contagem de carateres de pontuação. Estes gráficos de distribuição foram usados durante a análise exploratória para perceber os dados e inferir novos atributos.





No primeiro gráfico pode-se constatar que a distribuição do comprimento dos URLs é bastante semelhante em ambas as classes, no entanto existem alguns URLs de phishing que são bastante longos. No segundo a distribuição também é semelhante, no entanto existem alguns URLs de phishing com entropia elevada. No terceiro pode-se verificar que os URLs de phishing geralmente apresentam uma maior ocorrência de carateres de pontuação. Os gráficos de distribuição para os restantes atributos podem ser consultados no anexo C.1.

8.3.2. Palavras-chave

No modelo de classificação foi usado um atributo booleano baseado em palavras-chave que se apresenta na Tabela 23, que têm como objetivo detetar a presença de determinadas palavras suspeitas nos URLs que os atacantes empregam para atrair a atenção dos utilizadores.

Descrição do atributo
Presença de palavras suspeitas

Tabela 23 - Resumo dos atributos de palavras-chave

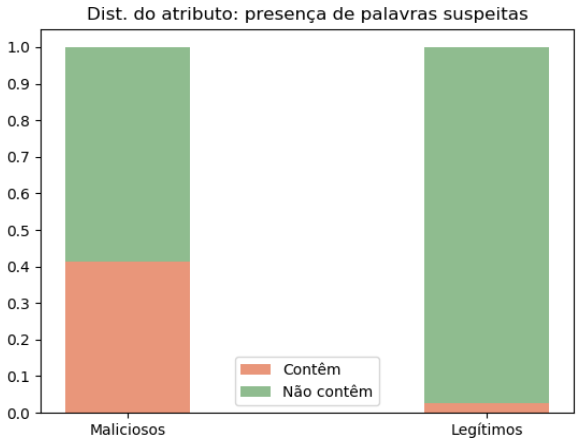
Para selecionar o conjunto de palavras mais discriminativo a incluir na lista de palavras suspeitas usou-se uma técnica semelhante à apresentada no capítulo de Estado da Arte, ainda que com a representação do atributo mais otimizada e eficiente. Começou-se por dividir cada URL de phishing do conjunto de dados pelos carateres não alfanuméricos e ao mesmo tempo calculou-se a frequência de cada uma das palavras, tendo-se ficado com um total de 25.261 palavras e suas frequências. Depois descartou-se todos as palavras com comprimento menor que três, uma vez que a análise experimental demonstrou que as palavras de um carater não oferecem qualquer significado, enquanto que as palavras de dois carateres são principalmente os TLDs com o código do país. Depois descartou-se algumas partes comuns dos URLs (http, https, com, org, etc.) e as extensões dos ficheiros das páginas web (html, htm, php, etc.). Finalmente descartou-se todas as palavras com frequências menores que 25, tendo-se com esta primeira seleção ficado com um total de 966 palavras. Para cada uma das restantes palavras calculou-se a sua informação mútua na classe de phishing, que mede a quantidade de informação que a presença ou ausência da palavra contribui para tomar a decisão correta. Na Tabela 24 apresenta-se o top 20 de palavras obtidas com base na sua informação mútua.

Palavra	Informação mútua
confirm	0.0944
paypal	0.0544
logon	0.0445
cmd	0.0379

signin	0.0333
login	0.0242
submit	0.0221
webscr	0.0217
secure	0.0152
wp	0.0148
update	0.0122
account	0.0104
verify	0.0101
admin	0.0097
email	0.0093
dropbox	0.0087
verification	0.0083
auth	0.0080
free	0.0075
rand	0.0073

Tabela 24 - Top 20 de palavras baseado na informação mútua

As palavras com valores mais altos de informação mútua são as mais discriminativas em relação à classe alvo. Para selecionar o número de palavras ideal a incluir na lista de palavras suspeitas, treinou-se e testou-se o modelo começando com uma palavra na lista, depois com duas palavras, e assim sucessivamente. A partir da 15ª palavra na lista deixou de haver um ganho de desempenho e parou-se o processo. Utilizou-se, portanto, na lista de palavras suspeitas do modelo de classificação o top 15 de palavras com valor mais alto de informação mútua: 'confirm', 'paypal', 'logon', 'cmd', 'signin', 'login', 'submit', 'webscr', 'secure', 'wp', 'update', 'account', 'verify', 'admin' e 'email'. Na próxima figura apresenta-se uma comparação da distribuição na classe positiva e negativa do atributo da presença de palavras suspeitas.



No gráfico de distribuição pode-se constatar que a presença de palavras suspeitas nos URLs é bastante superior nos URLs de phishing.

8.3.3. Reputação

Foram usados inicialmente no modelo de classificação 4 atributos de reputação, sendo que 1 é de valor real e 3 são de valor binário que se apresenta na Tabela 25 e Tabela 26 respetivamente. Estes atributos de reputação têm como principal objetivo aumentar o grau de confiança das classificações. Alguns dos atributos foram retirados do capítulo de Estado da Arte, no entanto também foram inferidos novos atributos a partir da análise exploratória dos dados e usando o conhecimento no domínio:

- Page rank do domínio: page rank dos domínios que fornece uma classificação para o domínio no intervalo de 0 a 10, obtido do Open Page Rank [26]. Se o page rank do domínio for inexistente significa que o domínio foi criado recentemente e como tal o URL em questão tem uma maior probabilidade de ser de phishing, se o page rank for baixo é um indicativo que o domínio é pouco fiável. Este atributo é equivalente ao atributo da idade do domínio.
- Domínio no top do Alexa: o ranking do Alexa [62] é uma lista de domínios organizados pela sua popularidade na internet. A maioria dos URLs de phishing está alojado ou em domínios comprometidos que poderão já não fazer parte dos principais domínios do Alexa, ou em novos domínios que também não irão aparecer de imediato na classificação do Alexa.
- Domínio na lista do PhishTank: se o domínio é frequente de phishing na lista negra do PhishTank.
- Domínio na lista do OpenPhish: se o domínio é frequente de phishing na lista negra do OpenPhish.

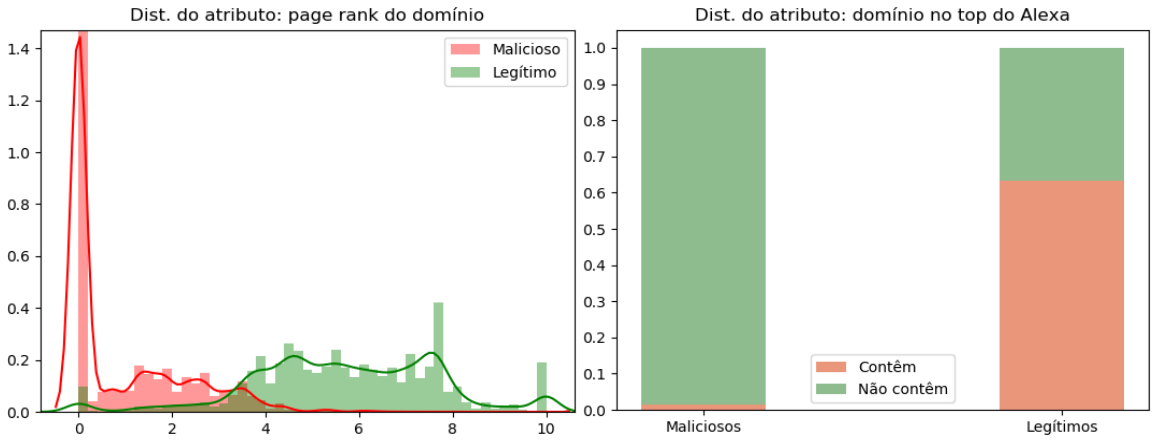
Descrição do atributo
Page rank do domínio

Tabela 25 - Resumo dos atributos de reputação de valor real

Descrição do atributo
Domínio no top do Alexa
Domínio na lista do PhishTank
Domínio na lista do OpenPhish

Tabela 26 - Resumo dos atributos de reputação de valor binário

Nas próximas figuras apresenta-se uma comparação das distribuições na classe positiva e negativa de alguns dos atributos de reputação mais significativos, nomeadamente o atributo do page rank do domínio, e do domínio no top do Alexa.



No primeiro gráfico pode-se constatar que a distribuição do page rank dos URLs apresenta um padrão bastante divergente em ambas as classes, uma vez que nos URLs de phishing o page rank é baixo ou inexistente enquanto que nos legítimos apresenta regra geral valores mais elevados. No segundo a distribuição também é significativa uma vez que muitos dos URLs legítimos aparecem no top de domínios do Alexa. Os gráficos de distribuição para os restantes atributos podem ser consultados no anexo C.1.

8.4. Avaliação de classificadores

Terminada a engenharia de atributos extraiu-se os atributos apresentados e foi-se perceber qual o melhor classificador para o problema de deteção de URLs de phishing com machine learning. Para tal treinou-se, avaliou-se e comparou-se empiricamente os resultados do modelo para os sete algoritmos de classificação apresentados no capítulo de Conhecimento Base: o Naive Bayes, a Regressão Logística, as Support Vector Machines, as Árvore de Decisão, as Florestas Aleatórias, o Gradient Boosting, e o Multilayer Perceptron. A eficácia dos modelos de classificação foi avaliada recorrendo à técnica de cross-validation com 10 folds apresentada no capítulo de Conhecimento Base que permite avaliar melhor a sua capacidade de generalização. Relembrando rapidamente, esta é uma técnica que divide aleatoriamente os dados iniciais em k subconjuntos mutuamente exclusivos (ou folds) de igual tamanho. Depois o processo de treino e teste do modelo é realizado k vezes. Com base nos resultados das classificações das k iterações é obtida a matriz de confusão que descreve o desempenho completo do modelo, em termos de true positives (TP), true negatives (TN), false positives (FP) e false negatives (FN) e, por fim, calculou-se accuracy a partir destas quatro métricas que avalia o desempenho global do modelo. Todas estas métricas podem ser consultadas em maior detalhe no capítulo de Conhecimento Base. Na Tabela 27 apresenta-se os resultados do modelo para cada algoritmo na sua configuração default, que podem ser consultados em [63], [64], [65], [66], [67], [68] e [69] respetivamente.

Algoritmo	Accuracy (%)	TP (%)	TN (%)	FP (%)	FN (%)
Naive Bayes	72.45	46.08	98.82	1.18	53.92
Regressão Logística	96.45	97.07	95.83	4.17	2.93
Support Vector Machines	97.21	98.11	96.31	3.69	1.89
Árvore de Decisão	97.71	97.94	97.49	2.51	2.06
Florestas Aleatórias	97.88	98.29	97.46	2.54	1.71
Gradient Boosting	97.92	98.21	97.62	2.38	1.79
Multilayer Perceptron	97.75	98.10	97.39	2.61	1.90

Tabela 27 - Avaliação dos algoritmos de classificação

A partir da análise da tabela pode-se concluir a seguinte informação:

- Os modelos baseados em métodos de ensemble que combinam uma série de modelos de classificação individuais (Florestas Aleatórias, Gradient Boosting) apresentam um desempenho ligeiramente superior aos modelos baseados em métodos básicos (Naive Bayes, Regressão Logística, Support Vector Machines e Árvore de Decisão).
- O modelo do Gradient Boosting foi o que mais se destacou, tendo obtido uma accuracy de 97.92%. Para além disso também foi o mais equilibrado, uma vez que apresentou uma taxa de falsos positivos e de falsos negativos com valores baixos e aproximados, o que indica uma previsão equilibrada e imparcial. Como tal este será o algoritmo de classificação escolhido para realizar as experiências daqui para a frente.

8.5. Análise de atributos

Esta secção apresenta a avaliação da importância dos atributos para o classificador, e o estudo realizado para selecionar o subconjunto final de atributos mais relevantes para usar no modelo de classificação.

8.5.1. Avaliação de importância

A importância dos atributos foi avaliada a dois níveis, ao nível da importância dos atributos para o classificador e ao nível da importância de cada uma das categorias de atributos. Para avaliar a importância dos atributos para o classificador recorreu-se ao método feature_importances_ da biblioteca do Gradient Boosting do Python [68] para devolver a importância dos atributos. Internamente isso é obtido contando o número de vezes que cada atributo foi selecionado nas diferentes árvores de decisão do Gradient Boosting durante o seu processo de treino. Na Figura 17 apresenta-se graficamente a importância de cada atributo para o classificador.

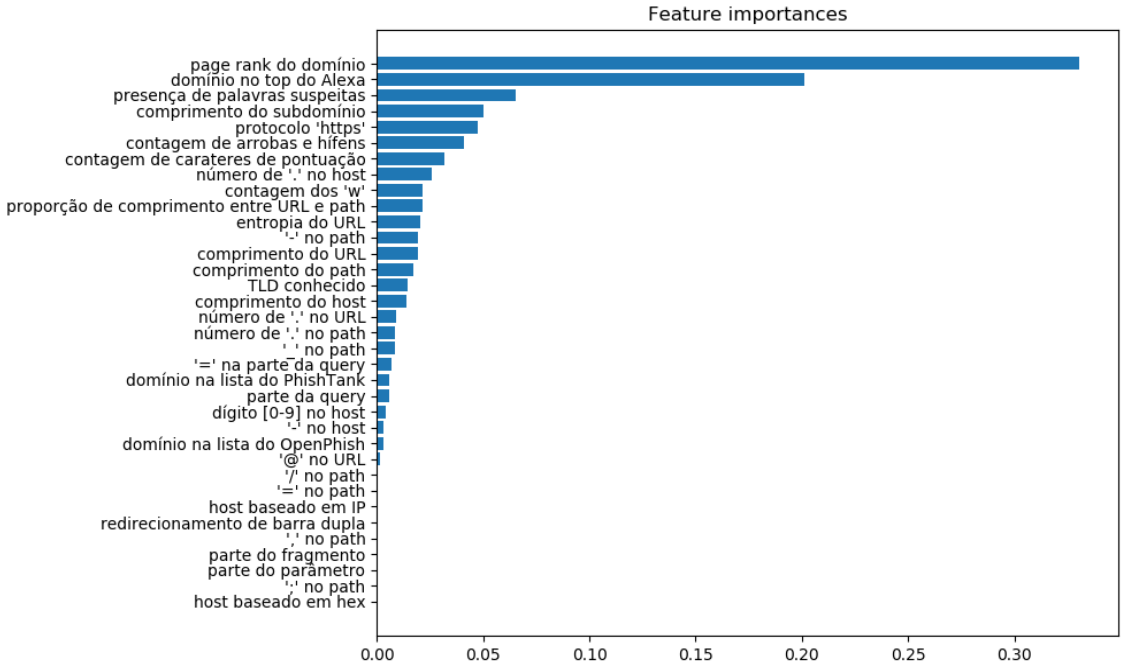


Figura 17 - Importância dos atributos para o classificador

O gráfico da importância dos atributos mostra que o atributo mais importante é o page rank do domínio, o que não surpreende já que os domínios de phishing recém-criados possuem um page rank baixo ou inexistente. Em segundo lugar aparece o top do Alexa que também não surpreende já que os domínios dos URLs legítimos costumam estar bem classificados no top do Alexa. Em terceiro aparece a presença de palavras suspeitas que também não surpreende já que os atacantes tentem enganar os utilizadores empegando palavras suspeitas nos URLs para tentar atrair a sua atenção. Seguem-se outros atributos que também possuem um peso significativo como o comprimento do subdomínio, o protocolo 'https', a contagem de arrobas e hífens, a contagem de carateres de pontuação, o número de '.' no host, etc.

Para avaliar a importância de cada uma das categorias de atributos, usou-se o algoritmo Gradient Boosting para treinar e avaliar o modelo com as diversas combinações de categorias de atributos a fim de avaliar a sua eficácia na discriminação dos URLs. Os resultados desta experiência são apresentados na Tabela 28.

Categoria de atributos	Accuracy (%)
Lexicais	91.49
Palavras-chave	69.44
Reputação	94.05
Lexicais + Palavras-chave	92.65

Lexicais + Reputação	97.63
Palavras-chave + Reputação	94.86
Todas	97.92

Tabela 28 - Importância das categorias de atributos

8.5.2. Seleção de relevantes

A seleção de atributos relevantes consiste em encontrar o subconjunto final de atributos mais discriminativo em relação à classe alvo para usar no modelo de classificação. Esta é uma etapa importante porque a seleção destes atributos permite aumentar a precisão da classificação, tornar o processo de treino e aplicação do modelo mais eficiente, uma vez que o número de atributos e ruído nos dados diminui. Para realizar esta seleção final vai-se recorrer a três métodos distintos de seleção de atributos, o modelo de filtro através de um threshold calculado (que define o ponto de corte no ranking dos atributos), o modelo de wrapper através da eliminação recursiva dos atributos, que foram apresentados no capítulo de Conhecimento Base, e ainda um método de procura sequencial exaustiva, que consiste em testar pela ordem de importância dos atributos subconjuntos de atributos sequenciais, o primeiro, os dois primeiros, os três primeiros, e por aí em diante até todos os 35 atributos dos dados serem utilizados. Na Figura 18 apresenta-se o gráfico com a accuracy do modelo usando o algoritmo Gradient Boosting para cada método de seleção de atributos experimentado. A primeira coluna referente à utilização de todos os atributos serve apenas como valor de referência, uma vez que o valor foi obtido nas mesmas condições da avaliação de classificadores efetuada na secção 7.2.4.

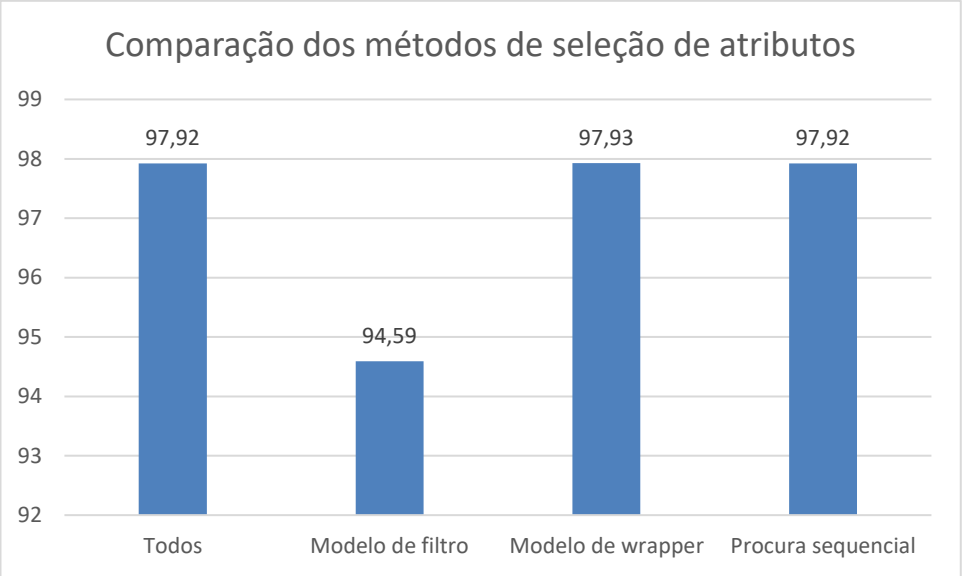


Figura 18 - Comparação dos métodos de seleção de atributos

O gráfico da comparação dos métodos de seleção de atributos mostra que o modelo de wrapper permitiu obter mais um pequeno ganho de desempenho, em comparação com a utilização de todos os atributos. Este ganho foi pequeno porque durante o processo de engenharia de atributos, muitos atributos de ruído que não trouxeram desempenho foram sendo removidos. O modelo de wrapper obteve a precisão mais elevada com 27 atributos, tendo este removido os seguintes 8 atributos: host baseado em IP, host baseado em hex, '/' no path, ',' no path, ';' no path, parte do parâmetro, parte do fragmento, redirecionamento de barra dupla. A título de curiosidade o modelo de filtro selecionou os três primeiros atributos da hierarquia, enquanto que o método de procura sequencial obteve o valor máximo ao 26º atributo da hierarquia de atributos, ou seja, removeu os últimos 9 atributos da hierarquia.

8.6. Afinação do modelo final

Terminado o desenvolvimento do modelo o último passo foi fazer a sua afinação final, ou seja, tirou-se partido dos parâmetros do algoritmo Gradient Boosting (que está explicado no capítulo de Conhecimento Base) para fazer uma última otimização do modelo. Para isso foram experimentadas várias combinações dos principais parâmetros do algoritmo, através do treino e avaliação do modelo, para escolher a parametrização que minimiza o erro do modelo. Neste processo foram usados os seguintes parâmetros e conjuntos de valores:

- learning_rate: a taxa de aprendizagem que define a contribuição de cada estimador (cada estimador é uma árvore de decisão). Testado para valores de 0.1, 0.5 e 1.
- n_estimators: o número de estágios de reforço a serem executados. Testado para valores de 300, 500, 700, 900 e 1200.
- subsample: a fração de amostras a ser usada para ajustar os estimadores. Testado para valores de 0.7, 0.85 e 1.
- min_samples_split: o número mínimo de amostras necessárias para dividir um nó interno. Testado para valores de 2, 5 e 10.
- min_samples_leaf: o número mínimo de amostras necessárias para estar em um nó folha. Testado para valores de 1, 3 e 5.

Após a experimentação a melhor parametrização obtida para o modelo foi com a learning_rate a 0.1, o n_estimators a 1200, o subsample a 1, o min_samples_split a 2, e o min_samples_leaf a 1. Na Tabela 29 apresenta-se a comparação de desempenho do modelo na sua configuração default e do modelo final obtido, já com o subconjunto de atributos mais relevantes e com a sua parametrização otimizada.

Modelo	Accuracy (%)	TP (%)	TN (%)	FP (%)	FN (%)
Default	97.92	98.21	97.62	2.38	1.79
Otimizado	98.28	98.60	97.96	2.04	1.40

Tabela 29 - Avaliação do modelo final

A afinação do modelo permitiu obter mais um pequeno ganho de desempenho. De destacar que após afinado o modelo continuou bastante equilibrado, uma vez que apresentou uma taxa de falsos positivos e de falsos negativos com valores baixos e aproximados, o que indica uma previsão equilibrada e imparcial.

8.7. Sumário

Os resultados experimentais mostraram que o modelo de classificação proposto foi capaz de detetar URLs de phishing com uma accuracy de 98.28%, uma taxa de falsos positivos de 2.04% e uma taxa de falsos negativos de 1.40% com o algoritmo Gradient Boosting que superou todos os outros algoritmos de classificação testados. Em relação aos trabalhos relacionados, foi usado um conjunto de dados relativamente grande de URLs, explorou-se e inclui-se no modelo de classificação novos atributos, como por exemplo as métricas de page rank dos domínios que revelou ser um atributo bastante significativo. Durante o desenvolvimento do modelo foi também feita uma avaliação dos vários algoritmos de classificação, uma avaliação de importância dos atributos, uma avaliação de três métodos de seleção do subconjunto final de atributos e, por fim, afinou-se o modelo e demonstrou-se utilizando outros conjuntos de dados que o modelo produzido é generalizável (os resultados podem ser consultados no anexo C.2).

9. Testes

Este capítulo apresenta o plano e os resultados da execução dos testes aos requisitos funcionais e não funcionais do sistema desenvolvido, e os resultados dos testes de aceitação. Uma vez que a qualidade do modelo de classificação desenvolvido já foi analisada em detalhe no capítulo anterior, optou-se por não o incluir neste capítulo.

9.1. Requisitos funcionais

O plano de testes aos requisitos funcionais é constituído por testes de black-box [70] ao sistema, que abrange tanto os testes à API que expõe o serviço de back-end como os testes ao backoffice de administração. Estes testes de black-box como requerem uma perspetiva do utilizador final, tiveram como ponto de referência a especificação funcional para a definição dos casos de teste (inputs e outputs) para testar as funcionalidades.

9.1.1. Sistema de back-end

Os testes ao sistema de back-end incidiram sobre o seu ponto de entrada no sistema, ou seja, a API que expõe o serviço de análise de mensagens. Para testar foram definidos um conjunto de casos de teste para os seus dois URIs (/login e /message) com entradas bem definidas e o resultado previsto para a sua execução, para garantir que a API está protegida contra parâmetros de entrada inválidos, acessos não autorizados, que se mantêm em funcionamento na presença de falhas, assim como validar os resultados dos pedidos nos mais variados cenários. Como o universo de todos os casos de teste possíveis é potencialmente infinito e nunca será possível executar todos os casos possíveis, optou-se por testar os aspetos mais comuns. Para automatizar a execução dos testes recorreu-se à framework JUnit de testes do Spring Boot [71], validando as respostas devolvidas pelos pedidos com recurso a asserções.

Na Tabela 30 e na Tabela 31 apresentam-se respetivamente os testes realizados ao URI /login (que solicita um token de acesso) e ao URI /message (que verifica se uma mensagem é maliciosa) da API que expõe o serviço de back-end. De notar o significado dos códigos de estado HTTP que podem ser devolvidos nas respostas dos pedidos: 200 (sucesso), 401 (acesso não autorizado), 400 (pedido inválido), 500 (erro interno no servidor).

Teste	Descrição do teste	Resposta esperada	Resultado
1	Pedido com parâmetros de entrada inválidos (para o username e password)	Código 400, indicação do erro	Passou
2	Pedido com o username e/ou a password inválidos	Código 401, indicação do erro	Passou
3	Pedido com o username e a password válidos	Código 200, token de acesso	Passou

Tabela 30 - Testes ao URI /login

Teste	Descrição do teste	Resposta esperada	Resultado
1	Pedido com parâmetros de entrada inválidos (para o token, type e content da mensagem)	Código 400, indicação do erro	Passou
2	Pedido com o token de acesso inválido	Código 401, indicação do erro	Passou
3	Pedido com a base de dados e/ou com a fila indisponíveis	Código 500, indicação do erro	Passou
4	Pedido com uma mensagem sem URLs	Código 200, indicação que a mensagem não é maliciosa	Passou

5	Pedido com uma mensagem com um URL presente na blacklist	Código 200, indicação que a mensagem é maliciosa	Passou
6	Pedido com uma mensagem com dois URLs presentes na blacklist	Código 200, indicação que a mensagem é maliciosa	Passou
7	Pedido com uma mensagem com um URL presente na whitelist	Código 200, indicação que a mensagem não é maliciosa	Passou
8	Pedido com uma mensagem com dois URLs presentes na whitelist	Código 200, indicação que a mensagem não é maliciosa	Passou
9	Pedido com uma mensagem com um URL presente na blacklist e outro presente na whitelist	Código 200, indicação que a mensagem é maliciosa	Passou
10	Pedido com uma mensagem com um URL presente na whitelist e outro presente na blacklist	Código 200, indicação que a mensagem é maliciosa	Passou
11	Duplo pedido (intervalado por 0.3s) com uma mensagem com um novo URL malicioso	Código 200, indicação que a mensagem é maliciosa no segundo	Passou
12	Duplo pedido (intervalado por 0.3s) com uma mensagem com um novo URL seguro	Código 200, indicação que a mensagem não é maliciosa em ambos	Passou
13	Pedido com uma mensagem com um URL imitado presente na blacklist, modificando o ruído de um diretório do path do URL	Código 200, indicação que a mensagem é maliciosa	Passou
14	Pedido com uma mensagem com um URL imitado presente na blacklist, adicionando a ancora #1 no fim do URL	Código 200, indicação que a mensagem é maliciosa	Passou
15	Pedido com uma mensagem com um URL imitado presente na whitelist, adicionando a ancora #1 no fim do URL	Código 200, indicação que a mensagem não é maliciosa em ambos	Passou

Tabela 31 - Testes ao URI /message

Para garantir que o sistema também está a detetar URLs em texto corretamente, optou-se por nas mensagens com um URL posicioná-lo no meio do conteúdo da mensagem, e nas mensagens com dois URLs por posicionar um no início e o outro no fim do conteúdo da mensagem. Importa referir que os testes não passaram todos na primeira tentativa, tendo sido corrigidos alguns defeitos para conseguir que os testes passassem todos os casos de teste.

9.1.2. Backoffice de administração

Os testes ao backoffice de administração incidiram sobre a sua interface a web. Para testar foram definidos um conjunto de casos de teste para as suas quatro páginas web (de login, da blacklist, da whitelist e do classificador) para garantir que as funcionalidades dos requisitos referentes ao backoffice de administração têm o comportamento esperado. Á semelhança do anterior, optou-se por testar também os aspetos principais e mais recorrentes porque como o universo de todos os casos de teste possíveis é potencialmente infinito nunca será possível testar todos os casos possíveis.

Na Tabela 32, na Tabela 33, na Tabela 34 e na Tabela 35 apresentam-se respetivamente os testes realizados à página de login, à página da blacklist, à página da whitelist e à página do classificador do backoffice de administração.

Teste	Descrição do teste	Resposta esperada	Resultado
1	Autenticação com o username e/ou a password inválidos	O utilizador mantém-se na página de login, mensagem de notificação adequada	Passou

2	Autenticação com o username e a password válidos	O utilizador é levado para a página da blacklist	Passou

Tabela 32 - Testes à página de login

Teste	Descrição do teste	Resposta esperada	Resultado
1	Verificação se a lista de URLs bloqueados é apresentada corretamente	A lista de URLs é exibida corretamente, o número de itens por página é ajustável e os botões de previous e next relativos à paginação nunca indexam fora	Passou
2	Adição de URLs com parâmetros inválidos (para o URL, data de entrada e data da última análise)	O formulário continua a ser apresentado, mensagem de notificação adequada	Passou
3	Adição de URLs com parâmetros válidos	Visualização do novo URL no topo da lista com a informação inserida, mensagem de notificação adequada	Passou
4	Remoção de URLs da lista	O URL desaparece da lista, mensagem de notificação adequada	Passou
5	Edição de URLs com parâmetros inválidos (para o URL, data de entrada e data da última análise)	O formulário continua a ser apresentado, mensagem de notificação adequada	Passou
6	Edição de URLs com parâmetros válidos	Visualização do URL com a informação atualizada, mensagem de notificação adequada	Passou
7	Edição de URLs trocando-os de lista de forma a corrigir um falso positivo	Visualização do URL no topo da whitelist, mensagem de notificação adequada	Passou
8	Explicação da classificação de URLs	A explicação da classificação é apresenta-da com os atributos, os seus valores e a sua contribuição para a decisão	Passou
9	Ordenamento e filtragem da lista de URLs por qualquer um dos atributos	A lista de URLs é exibida corretamente de acordo com a opção de ordenamento e/ou filtragem selecionados	Passou

Tabela 33 - Testes à página da blacklist de URLs

Teste	Descrição do teste	Resposta esperada	Resultado
1	Verificação se a lista de URLs permitidos é apresentada corretamente	A lista de URLs é exibida corretamente, o número de itens por página é ajustável e os botões de previous e next relativos à paginação nunca indexam fora	Passou
2	Adição de URLs com parâmetros inválidos (para o URL, data de entrada e data da última análise)	O formulário continua a ser apresentado, mensagem de notificação adequada	Passou
3	Adição de URLs com parâmetros válidos	Visualização do novo URL no topo da lista com a informação inserida, mensagem de notificação adequada	Passou
4	Remoção de URLs da lista	O URL desaparece da lista, mensagem de notificação adequada	Passou
5	Edição de URLs com parâmetros inválidos (para o URL, data de entrada e data da última análise)	O formulário continua a ser apresentado, mensagem de notificação adequada	Passou
6	Edição de URLs com parâmetros válidos	Visualização do URL com a informação atualizada, mensagem de notificação adequada	Passou

7	Edição de URLs trocando-os de lista de forma a corrigir um falso negativo	Visualização do URL no topo da blacklist, mensagem de notificação adequada	Passou
8	Explicação da classificação de URLs	A explicação da classificação é apresentada com os atributos, os seus valores e a sua contribuição para a decisão	Passou
9	Ordenamento e filtragem da lista de URLs por qualquer um dos atributos	A lista de URLs é exibida corretamente de acordo com a opção de ordenamento e/ou filtragem selecionados	Passou

Tabela 34 - Testes à página da whitelist de URLs

Teste	Descrição do teste	Resposta esperada	Resultado
1	Alteração do grau de confiança do classificador c/ valores inválidos	O valor não é alterado, mensagem de notificação adequada	Passou
2	Alteração do grau de confiança do classificador c/ valores válidos	Visualização do novo valor na página, mensagem de notificação adequada, e verificação do log no classificador	Passou
3	Alteração da API de verificação a usar quando a confiança da classificação for baixa	Visualização da nova API na página, mensagem de notificação adequada, e verificação do log no classificador	Passou

Tabela 35 - Testes à página do classificador

Importa referir que nos casos de testes relativos ao teste de entradas inválidas teve-se o cuidado de incluir testes tanto aos valores centrais como aos valores limite (caso do grau de confiança), duplicados (caso dos URLs), carateres no lugar de números, entradas vazias, etc. À semelhança do anterior os testes também não passaram todos na primeira tentativa, tendo sido corrigidos alguns defeitos para conseguir que os testes passassem todos os casos de teste.

9.2. Requisitos não funcionais

Esta secção apresenta os testes realizados para validar os requisitos não funcionais do sistema, desempenho, escalabilidade e interoperabilidade, tendo como ponto de partida os cenários descritos na secção 5.2.

9.2.1. Desempenho

O desempenho do sistema desenvolvido foi avaliado pelo tempo de execução dos pedidos de análise de mensagens que chegam ao container da API que expõe o serviço de back-end. Para esta avaliação fez-se chegar pedidos à API e mediu-se os tempos de processamento dos pedidos em cinco tipos de mensagens de 100 carateres: sem URLs, com um URL presente na blacklist, com dois URLs presentes na blacklist, com um URL não presente na blacklist, e com um URL não presente na blacklist e outro presente. Para realizar o teste optou-se por ter 37.500 URLs distintos na blacklist e na whitelist da base de dados, e utilizou-se uma máquina de desenvolvimento com as seguintes caraterísticas:

- Sistema operativo Windows 10 de 64 bits;
- Processador i7 de 2.40 GHz;
- Memória RAM de 8.00 Gb;
- Disco SSD com capacidade de fazer leituras e escritas de informação sequencial a 540 Mb/s e 500 Mb/s respetivamente.

Na Tabela 36 apresenta-se o tempo médio e o tempo máximo dos testes de desempenho realizados a cada um destes casos após 100 execuções sequenciais.

Caso	Tempo (ms)		Resultado
	Médio	Máximo	
Mensagem sem URLs	0.05 ± 0.01	0.17	Passou
Mensagem com um URL presente na blacklist	8.04 ± 1.84	13.90	Passou
Mensagem com dois URLs presentes na blacklist	9.89 ± 1.51	15.26	Passou
Mensagem com um URL não presente na blacklist	9.63 ± 1.40	14.32	Passou
Mensagem com um URL não presente na blacklist e outro presente	18.40 ± 2.88	23.25	Passou

Tabela 36 - Testes de desempenho

Analisando os resultados obtidos pode-se concluir que o desempenho da API de integração está dentro do limite de tempo de 100 milissegundos definido, sendo que o caso que mais tempo demorou a executar foi o último, onde uma das execuções demorou 23.25 milissegundos, ainda assim longe do limite de 100 milissegundos. O leitor poderia questionar o porquê da análise de uma mensagem com dois URLs presentes na blacklist não ser ligeiramente mais lenta que a análise de uma mensagem com um URL ou equivalente ao último caso que também continha dois URLs, e a resposta é que neste caso foi realizada apenas uma querie à blacklist, porque basta o primeiro URL estar presente na blacklist para a resposta ao pedido ser retornada de imediato.

9.2.2. Escalabilidade

A escalabilidade do sistema desenvolvido foi avaliada pela sua capacidade de lidar com quantidades crescentes de trabalho. Para esta avaliação fez-se chegar pedidos à API que expõe o serviço de back-end e mediu-se os tempos médios de processamento dos pedidos para dez tipos diferentes de carga: 200 pedidos, 400 pedidos, 600 pedidos, e assim sucessivamente até aos 2000 pedidos, e, para dois tipos de mensagens de 100 carateres: sem URLs e com um URL (uma vez que em tempo real antes de ser devolvida uma resposta, é sempre efetuada uma querie à blacklist para verificar se o URL está presente).

Para realizar este teste manteve-se o mesmo setup experimental utilizado durante os testes de desempenho, e configurou-se a pool das conexões da base de dados, da fila e das threads da API para um limite máximo de 100 conexões/threads (o valor também pré-definido para o número máximo de conexões do servidor de base de dados PostgreSQL [72]). Para enviar os pedidos em paralelo recorreu-se à ferramenta JMeter do Apache [73], e antes de fazer as medições teve-se o cuidado de efetuar um warn-up inicial para abrir todos os recursos. Na Tabela 37 e na Tabela 38 apresenta-se o tempo médio e o tempo máximo dos testes de escalabilidade realizados para as diferentes cargas de pedidos nos dois tipos de mensagens.

Número de pedidos	Respondidos com sucesso (%)	Tempo (ms)		Resultado
		Médio	Máximo	
200	100.00	0.06 ± 0.02	0.18	Passou
400	100.00	0.08 ± 0.07	0.91	Passou
600	100.00	0.11 ± 0.19	1.69	Passou
800	100.00	0.12 ± 0.22	2.66	Passou
1000	100.00	0.14 ± 0.33	3.53	Passou
1200	100.00	0.15 ± 0.44	4.05	Passou
1400	100.00	0.15 ± 0.35	4.17	Passou
1600	100.00	0.19 ± 0.29	3.24	Passou
1800	100.00	0.19 ± 0.32	3.34	Passou
2000	100.00	0.21 ± 0.51	2.53	Passou

Tabela 37 - Testes de escalabilidade para mensagens sem URLs

Número de pedidos	Respondidos com sucesso (%)	Tempo (ms)		Resultado
		Médio	Máximo	
200	100.00	11.92 ± 2.55	29.17	Passou
400	100.00	22.12 ± 21.32	88.39	Passou
600	100.00	38.81 ± 52.41	147.29	Passou
800	100.00	41.35 ± 62.10	295.84	Passou
1000	100.00	44.85 ± 53.12	369.45	Passou
1200	100.00	61.63 ± 74.69	365.96	Passou
1400	100.00	70.32 ± 89.43	454.56	Passou
1600	100.00	91.78 ± 120.73	695.92	Passou
1800	98.60	110.10 ± 214.14	1128.64	Falhou
2000	97.49	113.01 ± 202.92	1095.92	Falhou

Tabela 38 - Testes de escalabilidade para mensagens com um URL

Analisando os resultados obtidos pode-se concluir que a API de integração consegue superar o cenário de escalabilidade descrito na secção 5.2 (processar os pedidos com latências até 100 milissegundos quando sobrecarregado com 300 pedidos em simultâneo) em ambos os tipos de mensagens. No entanto, em produção, é expetável que apenas uma pequena percentagem das mensagens contenha URLs.

De qualquer forma, é interessante verificar para o setup experimental descrito que, enquanto no primeiro cenário o tempo de processamento das mensagens se manteve relativamente baixo com o aumento do número de pedidos, no segundo cenário verificou-se que o tempo de processamento das mensagens se degradou com o aumento do número de pedidos. Para além disso, as cargas finais foram tão elevadas que ocorreram alguns timeouts na obtenção das conexões (configurado em 1s), o que explica o facto de alguns pedidos não terem sido processados e tendo sido devolvidos alguns códigos 500. Estes testes também foram importantes para garantir que a API que expõe o serviço de back-end está corretamente implementada ao nível da gestão dos vários recursos, assegurando que estes nunca ficam presos.

9.2.3. Interoperabilidade

A interoperabilidade do sistema foi garantida ao nível da arquitetura através da criação do container da API para expor o serviço de back-end, que fornece o conjunto de endpoints que permitem analisar e bloquear mensagens maliciosas. O formato de dados para comunicar e trocar informações é o JSON, e também ele foi escolhido criteriosamente por ser um formato de dados leve, interoperável e universal na indústria para devolver informação. Desta forma, garantiu-se que o sistema desenvolvido pode ser facilmente integrado com qualquer outro sistema desde que o formato da comunicação seja respeitado.

9.3. Aceitação

Os testes de aceitação tiveram como objetivo verificar se os requisitos funcionais do sistema cumpriram os objetivos propostos para o estágio. Para fazer esta validação apresentou-se o sistema e os resultados obtidos aos orientadores da empresa e validou-se se estes cumpriram os objetivos. Na Tabela 39 apresenta-se o resultado dos testes de aceitação realizados.

Requisito	Implementado	Aprovado
Mecanismo de autenticação	Sim	Sim
Detetor de mensagens maliciosas	Sim	Sim
Classificador de URLs	Sim	Sim
Retreinar periodicamente o modelo de classificação	Sim	Sim
Reanalisar periodicamente os URLs	Sim	Sim
Autenticação de um administrador	Sim	Sim
Visualização da lista de URLs bloqueados e permitidos	Sim	Sim
Adição e remoção de URLs	Sim	Sim
Edição e correção de falsos positivos e de falsos negativos	Sim	Sim
Explicação da classificação dos URLs	Sim	Sim
Ordenamento e filtragem de URLs	Sim	Sim
Definição das configurações do classificador	Sim	Sim
Monitorização do modelo de classificação	Não	Não aplicável

Tabela 39 - Testes de aceitação

A tabela mostra que todos os requisitos e suas funcionalidades foram aprovados, assim como todo o sistema em geral, tendo os objetivos da empresa para este estágio sido atingidos com sucesso. De referir que o último requisito referente à monitorização do modelo de classificação não foi implementado de forma a alocar mais tempo para outras tarefas, no entanto, este era um requisito com prioridade baixa e que não era necessário nesta fase do projeto.

10. Conclusão

Estando o estágio terminado é altura de tirar conclusões e apresentar uma visão geral de como o ano letivo decorreu, do trabalho realizado e de sugestões para trabalho futuro. Por fim, e em jeito de balanço, é dada uma opinião pessoal do autor sobre o estágio.

10.1. Trabalho realizado

Este relatório apresentou todas as fases do desenvolvimento do projeto ao longo do estágio. O primeiro semestre teve como principal objetivo o estudo dos conceitos relacionados com o projeto e a elaboração da proposta de estágio. O semestre iniciou-se com o estudo dos conceitos gerais relacionados com o tema do projeto (RCS, phishing, componentes do URL), e o estudo da técnica de classificação de machine learning (pré-processamento de dados, algoritmos de classificação, métricas de avaliação de desempenho). Seguiu-se o estado da arte com o levantamento dos sistemas concorrentes, o estudo de abordagens analíticas para o classificador com base em trabalhos relacionados, e o levantamento e análise de APIs de deteção de URLs maliciosos e de ferramentas de machine learning. De seguida e com base na informação recolhida durante o estudo do estado da arte fez-se a especificação dos requisitos para o sistema a desenvolver. Finalizada a especificação de requisitos, fez-se o desenho de uma arquitetura para o sistema que suportasse todos os requisitos definidos. Por fim, começou-se a implementação do sistema, com a criação do modelo de dados para a base de dados, e com a implementação da API que expõe o serviço de back-end já com a fila integrada. O segundo semestre teve como principal objetivo a continuação da implementação do sistema proposto. As tarefas de desenvolvimento passaram pelo desenvolvimento do modelo de classificação, seguido do classificador e do agendador e, por fim, do backoffice de administração. Terminado o desenvolvimento, foram realizados testes aos requisitos funcionais e não funcionais para validar o sistema desenvolvido.

Deste estágio resultou o protótipo de um sistema desenhado para a interoperabilidade capaz de identificar e prevenir o envio de URLs maliciosos através da plataforma de messaging da WIT, com o intuito de evitar phishing e execução de código nos aparelhos dos clientes. Como principais fatores diferenciadores, destaca-se o facto de o protótipo desenvolvido utilizar machine learning, fornecer informação sobre os motivos que levaram à classificação de cada URL, possibilitar a configuração consoante o cliente final e expandir os URLs encurtados para classificar (o seu uso para ataques de phishing tem vindo a crescer segundo os últimos dados estatísticos [74]). Para a empresa, além de todo o conhecimento desta área de investigação que este documento sintetiza, ficou também criado todo o ambiente para que no futuro este protótipo possa ser produtizado de forma a poder ser integrado na plataforma de messaging do RCS. Do ponto de vista científico, além de novos atributos lexicais e de reputação que foram explorados, ficou demonstrado que um URL malicioso pode ser detetado usando as informações disponíveis no URL, foi feita uma avaliação da importância dos atributos e das suas diversas categorias em relação à sua capacidade de discriminação, e também uma avaliação dos vários algoritmos de classificação para determinar o mais eficaz para o problema de deteção de URLs maliciosos com machine learning.

10.2. Trabalho futuro

Como qualquer outro software, este também poderá ser sempre melhorado. Um dos aspetos que pode ser melhorado é a qualidade do modelo de classificação, continuando o processo de pesquisa e desenvolvimento, e continuando a explorar os dados analiticamente para detetar

novos padrões e criar novos atributos lexicais, de palavras-chave, e de reputação. Para além disso seria também interessante explorar dois novos tipos de atributos, os baseados em conteúdo e os baseados em páginas que só poderão ser obtidos a partir de serviços pagos. Os atributos baseados em conteúdo estão diretamente relacionados à presença de determinadas palavras no código fonte da página. Os atributos baseados em páginas fornecem informações sobre o seu nível de fiabilidade, e estão relacionados com informações sobre as páginas calculadas por serviços de classificação de reputação. Exemplos destes atributos são: o page rank global, o page rank do país, o número estimado de visitas para o domínio numa base diária, semanal ou mensal, a média de visualizações de páginas por visita, a duração média da visita, a existência de websites semelhantes, etc.

Numa perspetiva de produtização do protótipo também seria importante implementar mecanismos de monitorização e de auto-recuperação para alguns dos seus containers mais críticos, nomeadamente o da API e o do Classificador. Por exemplo, implementado componentes whatchdogs capazes de monitorizar as suas instâncias em tempo real e, em caso de falha, estabelecer planos de autocorreção sem que seja necessária a intervenção humana.

10.3. Balanço

A experiência de desenvolver um projeto num contexto profissional foi bastante gratificante porque permitiu passar por todas as etapas de engenharia de software no desenvolvimento de um protótipo, começando pela investigação dos conceitos envolvidos, seguido do levantamento de requisitos, do desenho da arquitetura, da implementação, e dos testes de validação. Para além disso permitiu também aprender novas tecnologias emergentes, e ainda adquirir conhecimento detalhado sobre técnicas de desenvolvimento de soluções de messaging, desenvolvimento de soluções de machine learning, conhecimento enriquecido em questões de segurança informática, novas arquiteturas de software e estratégias de desenvolvimento.

Referências

[1] Mobile phishing 2018: Myths and facts facing every modern enterprise today, disponível em https://info.lookout.com/rs/051-ESQ-475/images/Lookout-Phishing-wp-us.pdf, consultado em 17/09/2018.

[2] Phishing attacks are moving to messaging and social apps at an alarming rate, disponível em https://www.wandera.com/mobile-phishing-attacks/, consultado em 17/09/2018.

[3] Zero Day Phishing Attacks, disponível em https://inky.com/phishing-threats/zero-day-attack/, consultado em 17/09/2018.

[4] GanttProject, disponível em https://www.ganttproject.biz/, consultado em 08/10/2018.

[5] Slides das aulas de Gestão de Projetos de Marco Vieira, consultado em 19/10/2018.

[6] What is RCS messaging?, disponível em https://www.digitaltrends.com/mobile/what-is-rcs-messaging/, consultado em 20/09/2018.

[7] A2P and P2P Messaging - What's the difference?, disponível em https://www.sinch.com/learn/a2p-messaging-p2p-messaging/, consultado em 22/09/2018.

[8] Safe Browsing: malware and phishing, disponível em https://transparencyreport.google.com/safe-browsing/overview?hl=en, consultado em 23/09/2018.

[9] 11. The Web at a Glance: Anatomy of a URL, disponível em https://websitebuilders.com/how-to/web-at-a-glance/url-anatomy/, consultado em 30/11/2018.

[10] Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

[11] Feedzai Unveils AutoML, disponível em https://feedzai.com/press-releases/feedzai-unveils-automl/, consultado em 07/01/2019.

[12] Domingos, P. M. (2012). A few useful things to know about machine learning. Commun. acm, 55(10), 78-87.

[13] Tang, J., Alelyani, S., & Liu, H. (2014). Feature selection for classification: A review. Data classification: Algorithms and applications, 37.

[14] Malicious Detection for OEMs, disponível em https://zvelo.com/solutions/malicious-detection/, consultado em 13/10/2018.

[15] Kaspersky Anti-Phishing Feeds, disponível em https://www.kaspersky.com/phishing-url-data-feed, consultado em 13/10/2018.

[16] Cloudmark Security Platform for Mobile Messaging, disponível em https://www.cloudmark.com/fr/s/products/cloudmark-security-platform-for-mobile, consultado em 13/10/2018.

[17] Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., & Zhang, C. (2009, July). An empirical analysis of phishing blacklists. In Sixth conference on email and anti-spam (CEAS).

[18] Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007, November). A framework for detection and measurement of phishing attacks. In Proceedings of the 2007 ACM workshop on Recurring malcode (pp. 1-8). ACM.

[19] Ma, J., Saul, L. K., Savage, S., & Voelker, G. M. (2011). Learning to detect malicious urls. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 30.

[20] Basnet, R. B., Sung, A. H., & Liu, Q. (2014). Learning to detect phishing URLs. International Journal of Research in Engineering and Technology, 3(6), 11-24.

[21] Ranganayakulu, D., & Chellappan, C. (2013). Detecting malicious URLs in E-mail – An implementation. AASRI Procedia, 4, 125-131.

[22] Bahnsen, A. C., Bohorquez, E. C., Villegas, S., Vargas, J., & González, F. A. (2017, April). Classifying phishing URLs using recurrent neural networks. In 2017 APWG Symposium on Electronic Crime Research (eCrime) (pp. 1-8). IEEE.

[23] Le, A., Markopoulou, A., & Faloutsos, M. (2011, April). Phishdef: Url names say it all. In 2011 Proceedings IEEE INFOCOM (pp. 191-195). IEEE.

[24] Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179.

[25] Marchal, S., François, J., State, R., & Engel, T. (2014). Phishstorm: Detecting phishing with streaming analytics. IEEE Transactions on Network and Service Management, 11(4), 458-471.

[26] Open Page Rank, disponível em https://www.domcop.com/openpagerank/, consultado em 02/01/2019.

[27] Google Safe Browsing, disponível em https://developers.google.com/safe-browsing/, consultado em 28/09/2018.

[28] VirusTotal, disponível em https://www.virustotal.com/, consultado em 28/09/2018.

[29] PhishTank, disponível em http://www.phishtank.com/, consultado em 28/09/2018.

[30] OpenPhish, disponível em https://openphish.com/, consultado em 30/09/2018.

[31] Urlscan, disponível em https://urlscan.io/, consultado em 30/09/2018.

[32] URLhaus, disponível em https://urlhaus.abuse.ch/, consultado em 29/09/2018.

[33] Scikit-learn, disponível em https://scikit-learn.org/, consultado em 24/09/2018.

[34] TensorFlow, disponível em https://www.tensorflow.org/, consultado em 24/09/2018.

[35] H2O, disponível em https://www.h2o.ai/, consultado em 24/09/2018.

[36] Weka, disponível em https://www.cs.waikato.ac.nz/ml/weka/, consultado em 24/09/2018.

[37] MoSCoW Method, disponível em https://www.projectsmart.co.uk/moscow-method.php, consultado em 26/09/2018.

[38] Slides das aulas de Arquitetura de Software de Bruno Cabral, consultado em 26/09/2018.

[39] The C4 model for software architecture, disponível em https://c4model.com/, consultado em 18/10/2018.

[40] Spring Framework, disponível em https://spring.io/projects/spring-framework, consultado em 08/11/2018.

[41] Flask (A Python Microframework), disponível em http://flask.pocoo.org/, consultado em 08/11/2018.

[42] React - A JavaScript library for building user interfaces, disponível em https://reactjs.org/, consultado em 08/11/2018.

[43] PostgreSQL: The World's Most Advanced Open Source Relational Database, disponível em https://www.postgresql.org/, consultado em 08/11/2018.

[44] RabbitMQ, disponível em https://www.rabbitmq.com/, consultado em 08/11/2018.

[45] Slides das aulas de Integração de Sistemas de Filipe Araújo, consultado em 08/11/2018.

[46] JWT, disponível em https://jwt.io/, consultado em 15/04/2019.

[47] Class URL, disponível em https://docs.oracle.com/javase/10/docs/api/java/net/URL.html, consultado em 04/01/2019.

[48] Introduction to Thread Pools in Java, disponível em https://www.baeldung.com/thread-pool-java-and-guava, consultado em 11/01/2019.

[49] PostgreSQL JDBC Driver, disponível em https://jdbc.postgresql.org/, consultado em 28/12/2018.

[50] RabbitMQ Java Client Library, disponível em https://www.rabbitmq.com/java-client.html, consultado em 28/12/2018.

[51] Bootstrap, disponível em https://getbootstrap.com/, consultado em 19/12/2018.

[52] Bootbox.js, disponível em http://bootboxjs.com/, consultado em 19/12/2018.

[53] jQuery, disponível em https://jquery.com/, consultado em 19/12/2018.

[54] Psycopg2, disponível em https://pypi.org/project/psycopg2/, consultado em 19/12/2018.

[55] Pika, disponível em https://pypi.org/project/pika/, consultado em 01/04/2019.

[56] Bit.ly, disponível em https://bitly.com/, consultado em 11/03/2019.

[57] Ow.ly, disponível em https://hootsuite.com/pages/owly, consultado em 11/03/2019.

[58] Tiny.cc, disponível em https://tiny.cc/, consultado em 11/03/2019.

[59] Interpreting random forests, disponível em https://blog.datadive.net/interpreting-random-forests/, consultado em 15/04/2019.

[60] Common Crawl, disponível em http://commoncrawl.org/, consultado em 08/02/2019.

[61] Most popular top-level domains worldwide as of November 2018, disponível em https://www.statista.com/statistics/265677/number-of-internet-top-level-domains-worldwide/, consultado em 11/03/2019.

[62] The top 500 sites on the web, disponível em https://www.alexa.com/topsites/, consultado em 22/01/2019.

[63] sklearn.naive_bayes.GaussianN, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html, consultado em 30/03/2019.

[64] sklearn.linear_model.LogisticRegression, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, consultado em 30/03/2019.

[65] sklearn.svm.SVC, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html, consultado em 30/03/2019.

[66] sklearn.tree.DecisionTreeClassifier, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, consultado em 30/03/2019.

[67] sklearn.ensemble.RandomForestClassifier, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, consultado em 30/03/2019.

[68] sklearn.ensemble.GradientBoostingClassifier, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html, consultado em 30/03/2019.

[69] sklearn.neural_network.MLPClassifier, disponível em https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, consultado em 30/03/2019.

[70] Slides das aulas de Qualidade e Confiabilidade de Software de Raul Barbosa e Henrique Madeira, consultado em 06/05/2019.

[71] Testing the Web Layer, disponível em https://spring.io/guides/gs/testing-web/, consultado em 06/05/2019.

[72] Connections and Authentication, disponível em https://www.postgresql.org/docs/current/runtime-config-connection.html, consultado em 17/05/2019.

[73] Apache JMeter, disponível em https://jmeter.apache.org/, consultado em 07/05/2019.

[74] Recent Uptick in Phishing Messages Using URL Shorteners, https://wp.nyu.edu/itsecurity/2018/02/08/recent-uptick-in-phishing-messages-using-url-shorteners/, consultado em 03/06/2019.

Anexo A. Arquitetura

A.1. Diagrama de atividade da API

Para melhor se compreender o funcionamento da API que expõe o serviço de back-end, na Figura 19 apresenta-se um diagrama de atividade que ilustra o seu funcionamento geral, desde o momento que recebe um pedido do sistema RCS com uma mensagem a ser analisada, até à devolução da resposta ao pedido que indica se a mensagem deve ou não ser bloqueada.

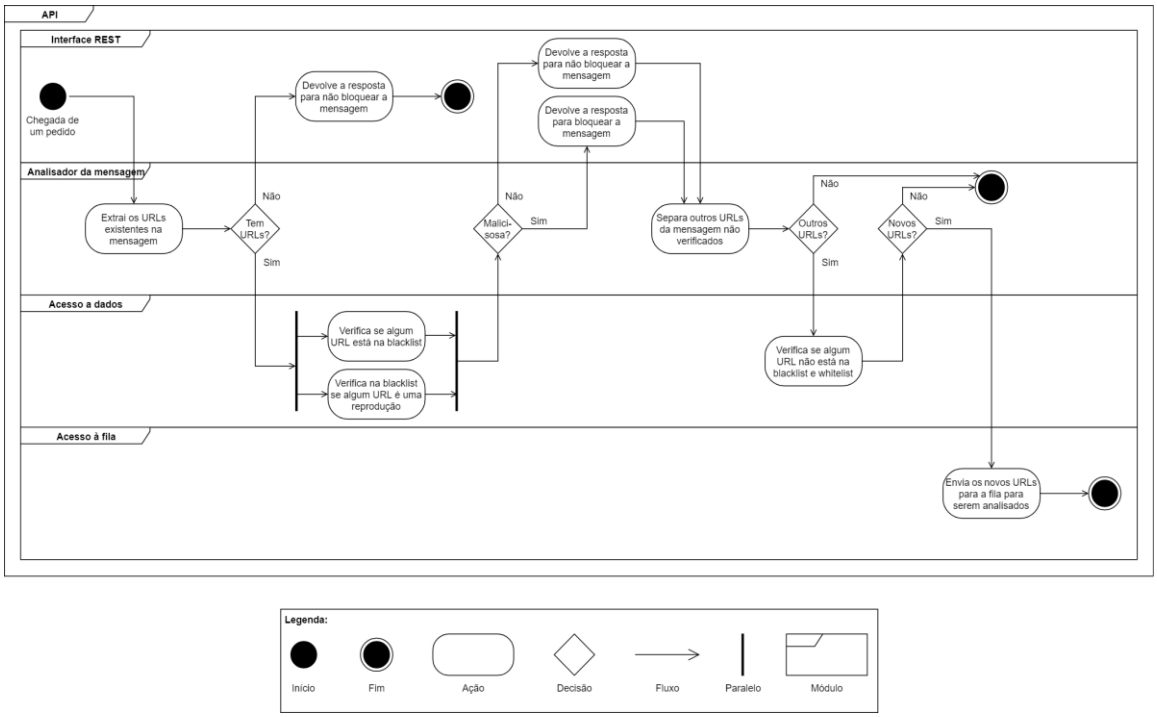


Figura 19 - Diagrama de atividade da API

Anexo B. Implementação

B.1. API REST da API

Este anexo documenta detalhadamente todos os URIs da API REST da API que expõe o sistema de back-end, que inclui para cada URI, os parâmetros recebidos e as respostas com os tipos de códigos HTTP que podem ser devolvidos.

Método

URI	Método HTTP	Descrição
/login	POST	Solicita um token de acesso à API

Parâmetros do pedido

Nome	Tipo	Descrição
username	String	O username de administrador
password	String	A password

Respostas

Código	Descrição
200	Sucesso (devolve um token de acesso à API)
401	Acesso não autorizado
400	Pedido inválido

Método

URI	Método HTTP	Descrição
/message	POST	Verifica se uma mensagem é maliciosa

Parâmetros do pedido

Nome	Tipo	Descrição
token	String	O token de acesso
type	String	O tipo de mensagem
content	String	O conteúdo da mensagem

Respostas

Código	Descrição
200	Sucesso (e indica se a mensagem deve ou não ser bloqueada)
401	Acesso não autorizado
400	Pedido inválido
500	Erro interno no servidor

B.2. API REST do backoffice

Este anexo documenta detalhadamente todos os URIs da API REST do backoffice, que inclui para cada URI, os parâmetros recebidos e as respostas com os tipos de códigos HTTP que podem ser devolvidos.

B.2.1. URIs da blacklist

Método

URI	Método HTTP	Descrição

/blacklist	GET	Obtém os URLs da blacklist

Respostas

Código	Descrição
200	Sucesso (exemplo do JSON com os URLs devolvidos em baixo)
401	Acesso não autorizado
500	Erro interno no servidor


```
{
  "blacklist": [
    {
      "id": 1,
      "url": "https://www.sapo.pt/",
      "dateEntry": "2018-12-01",
      "dateLastAnalysis": "2018-12-01",
      "addedAdmin": "True"
    },
    {
      "id": 2,
      "url": "https://www.record.pt/",
      "dateEntry": "2018-12-01",
      "dateLastAnalysis": "2018-12-03",
      "addedAdmin": "False"
    }
  ]
}
```

Método

URI	Método HTTP	Descrição
/blacklist	POST	Adiciona um novo URL à blacklist

Parâmetros do pedido

Nome	Tipo	Descrição
url	String	O URL
dateEntry	Date	A data de entrada
dateLastAnalysis	Date	A data da última análise

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
400	Pedido inválido
500	Erro interno no servidor

Método

URI	Método HTTP	Descrição
/blacklist/{id}	PUT	Edita um URL da blacklist

Parâmetros do pedido

Nome	Tipo	Descrição
id (path)	Integer	O id do URL da blacklist a ser editado
url	String	O URL
dateEntry	Date	A data de entrada
dateLastAnalysis	Date	A data da última análise
list	String	A lista (blacklist ou whitelist)

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
400	Pedido inválido
500	Erro interno no servidor

Método

URI	Método HTTP	Descrição
/blacklist/{id}	DELETE	Remove um URL da blacklist

Parâmetros do pedido

Nome	Tipo	Descrição
id (path)	Integer	O id do URL da blacklist a ser removido

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
500	Erro interno no servidor

B.2.2. URIs da whitelist

Método

URI	Método HTTP	Descrição
/whitelist	GET	Obtém os URLs da whitelist

Respostas

Código	Descrição
200	Sucesso (exemplo do JSON com os URLs devolvidos em baixo)
401	Acesso não autorizado
500	Erro interno no servidor


```
{
  "whitelist": [
    {
      "id": 1,
      "url": "https://www.formula1.com/",
      "dateEntry": "2018-12-01",
      "dateLastAnalysis": "2018-12-01",
      "addedAdmin": "True"
    },
    {
      "id": 2,
      "url": "https://www.google.com/",
      "dateEntry": "2018-12-01",
      "dateLastAnalysis": "2018-12-01",
      "addedAdmin": "False"
    }
  ]
}
```

Método

URI	Método HTTP	Descrição
/whitelist	POST	Adiciona um novo URL à whitelist

Parâmetros do pedido

Nome	Tipo	Descrição
url	String	O URL
dateEntry	Date	A data de entrada
dateLastAnalysis	Date	A data da última análise

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
400	Pedido inválido
500	Erro interno no servidor

Método

URI	Método HTTP	Descrição
/whitelist/{id}	PUT	Edita um URL da whitelist

Parâmetros do pedido

Nome	Tipo	Descrição
id (path)	Integer	O id do URL da whitelist a ser editado
url	String	O URL
dateEntry	Date	A data de entrada
dateLastAnalysis	Date	A data da última análise
list	String	A lista (whitelist ou blacklist)

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
400	Pedido inválido
500	Erro interno no servidor

Método

URI	Método HTTP	Descrição
/whitelist/{id}	DELETE	Remove um URL da whitelist

Parâmetros do pedido

Nome	Tipo	Descrição
id (path)	Integer	O id do URL da whitelist a ser removido

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
500	Erro interno no servidor

B.2.3. URIs do classificador

Método

URI	Método HTTP	Descrição
/classifier	GET	Obtém as configurações do classificador

Respostas

Código	Descrição
200	Sucesso (exemplo do JSON com as config. devolvidas em baixo)
401	Acesso não autorizado
500	Erro interno no servidor

```
{
  "confidence": 85,
  "api": "Virus Total"
}
```

Método

URI	Método HTTP	Descrição
/classifier	POST	Edita as configurações do classificador

Parâmetros do pedido

Nome	Tipo	Descrição
confidence	Integer	O grau de confiança
api	String	A API de verificação

Respostas

Código	Descrição
200	Sucesso
401	Acesso não autorizado
400	Pedido inválido
500	Erro interno no servidor

B.2.4. URIs gerais

Método

URI	Método HTTP	Descrição
/	GET	Obtém a página de login

Respostas

Código	Descrição
200	Devolve a página de login

Método

URI	Método HTTP	Descrição
/login	POST	Faz o login do utilizador, iniciando a sessão

Parâmetros do pedido

Nome	Tipo	Descrição
username	String	O nome do utilizador
password	String	A password

Respostas

Código	Descrição
200	Devolve a página da blacklist
401	Devolve a página de login

Método

URI	Método HTTP	Descrição
/logout	GET	Faz o logout do utilizador, removendo a sessão

Respostas

Código	Descrição
200	Devolve a página de login

Método

URI	Método HTTP	Descrição
/explain	GET	Obtém a explicação da classificação dum URL

Parâmetros do pedido

Nome	Tipo	Descrição
url	String	O URL

Respostas

Código	Descrição
200	Sucesso (devolve a explicação da classificação)
401	Acesso não autorizado
500	Erro interno no servidor

Método

URI	Método HTTP	Descrição
/html/{path}	GET	Obtém as páginas HTML do servidor

Parâmetros do pedido

Nome	Tipo	Descrição
path (path)	Path	O ficheiro HTML

Respostas

Código	Descrição
200	Devolve a página solicitada
401	Devolve a página de login

Método

URI	Método HTTP	Descrição
/js/{path}	GET	Obtém os ficheiros JavaScript do servidor

Parâmetros do pedido

Nome	Tipo	Descrição
path (path)	Path	O ficheiro JavaScript

Respostas

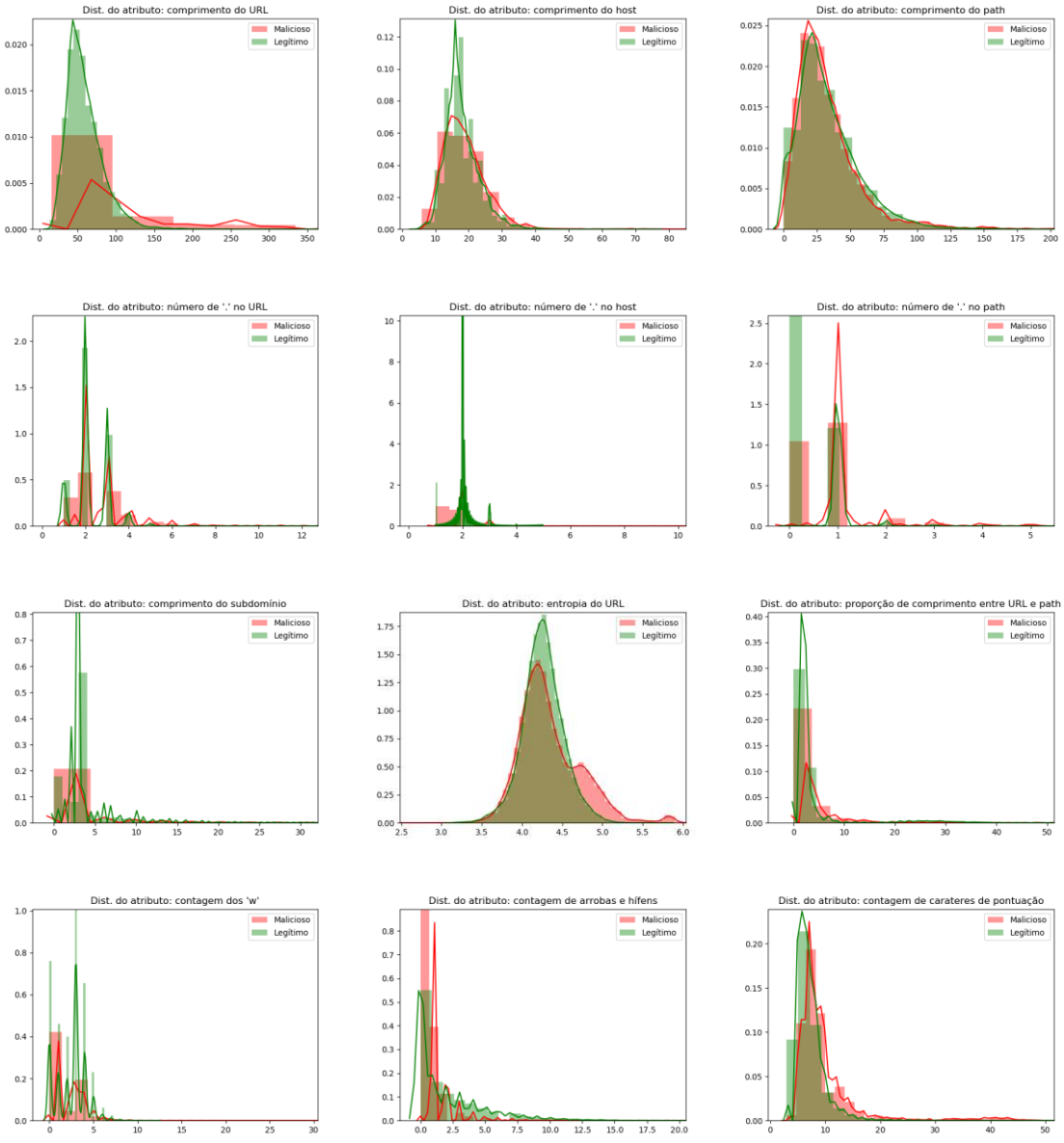
Código	Descrição
200	Devolve o ficheiro JavaScript solicitado
401	Acesso não autorizado

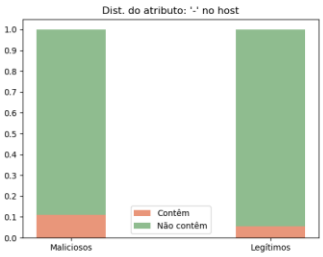
Anexo C. Modelo de Classificação

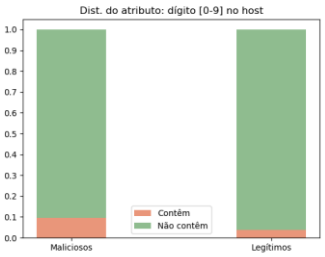
C.1. Distribuições dos atributos do modelo

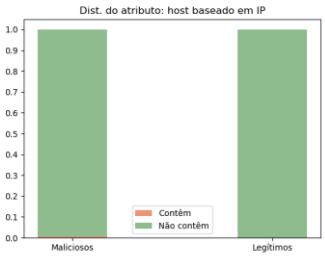
Este anexo apresenta a comparação das distribuições na classe positiva e negativa de todos os atributos lexicais, de palavras-chave e de reputação, usados inicialmente no modelo de classificação.

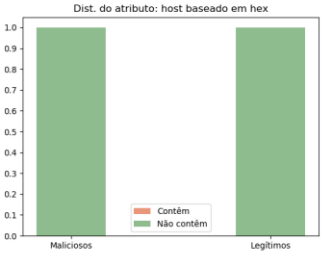
C.1.1. Atributos lexicais

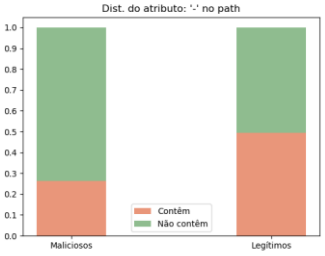


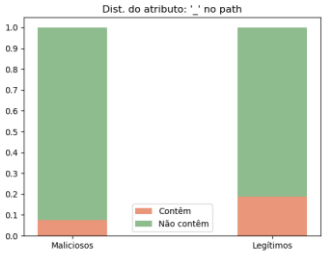

Dist. do atributo: '-' no host
Contêm
Não contêm
Maliciosos
Legítimos

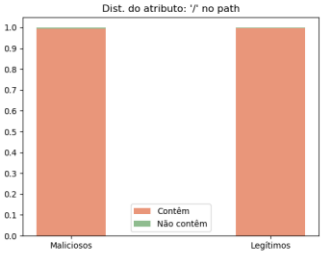

Dist. do atributo: dígito [0-9] no host
Contêm
Não contêm
Maliciosos
Legítimos

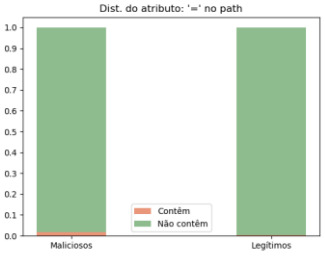

Dist. do atributo: host baseado em IP
Contêm
Não contêm
Maliciosos
Legítimos

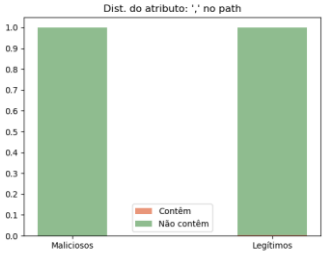

Dist. do atributo: host baseado em hex
Contêm
Não contêm
Maliciosos
Legítimos

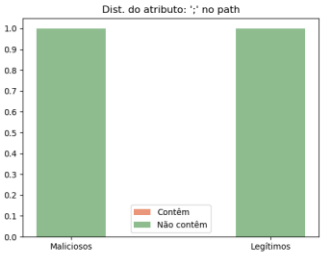

Dist. do atributo: '-' no path
Contêm
Não contêm
Maliciosos
Legítimos

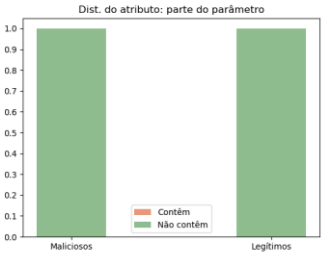

Dist. do atributo: '_' no path
Contêm
Não contêm
Maliciosos
Legítimos

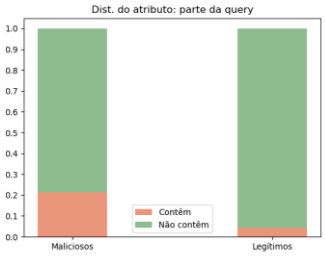

Dist. do atributo: '/' no path
Contêm
Não contêm
Maliciosos
Legítimos

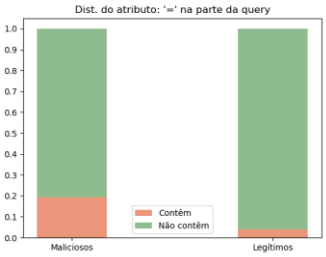

Dist. do atributo: '=' no path
Contêm
Não contêm
Maliciosos
Legítimos

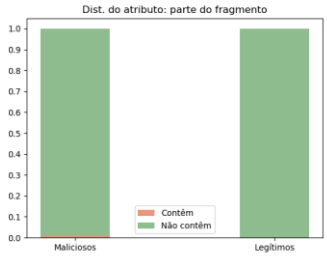

Dist. do atributo: ',' no path
Contêm
Não contêm
Maliciosos
Legítimos

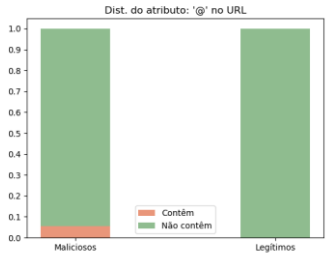

Dist. do atributo: ';' no path
Contêm
Não contêm
Maliciosos
Legítimos

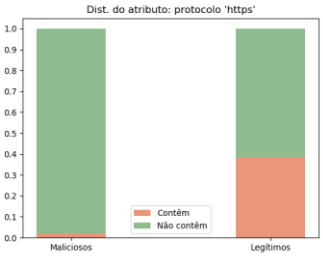

Dist. do atributo: parte do parâmetro
Contêm
Não contêm
Maliciosos
Legítimos

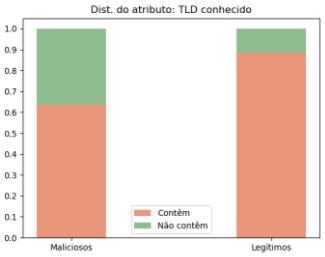

Dist. do atributo: parte da query
Contêm
Não contêm
Maliciosos
Legítimos

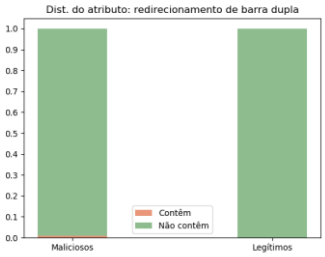

Dist. do atributo: '=' na parte da query
Contêm
Não contêm
Maliciosos
Legítimos


Dist. do atributo: parte do fragmento
Contêm
Não contêm
Maliciosos
Legítimos

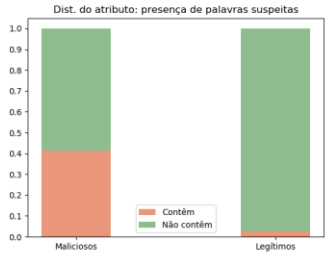

Dist. do atributo: '@' no URL
Contêm
Não contêm
Maliciosos
Legítimos


Dist. do atributo: protocolo 'https'
Contêm
Não contêm
Maliciosos
Legítimos

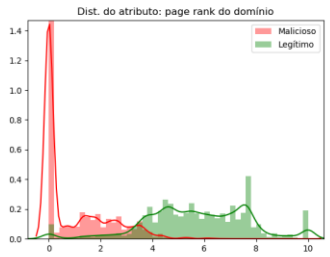

Dist. do atributo: TLD conhecido
Contêm
Não contêm
Maliciosos
Legítimos

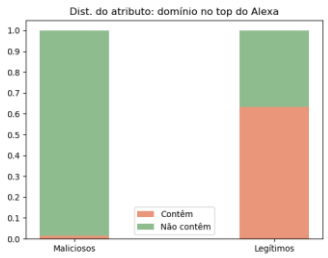

Dist. do atributo: redirecionamento de barra dupla
Contêm
Não contêm
Maliciosos
Legítimos

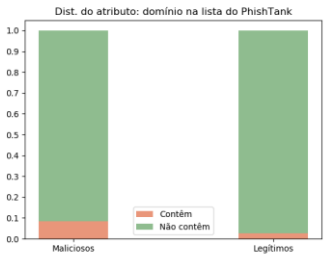
C.1.2. Atributos de palavras-chave

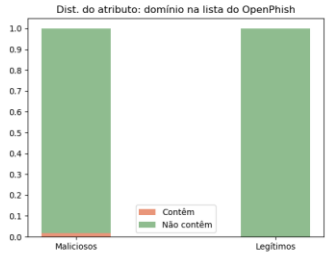


C.1.3. Atributos de reputação









C.2. Validação do modelo final

Para validar que o modelo de classificação produzido é generalizável recorreu-se a outros conjuntos de dados de diferentes fontes para verificar a sua taxa de deteção. Para fazer esta avaliação utilizaram-se os seguintes conjuntos de dados que foram sendo recolhidos semanalmente entre os meses de Fevereiro e Abril de 2019 (semanas de 19/02, de 07/03, de 18/03, de 25/03, de 29/03 e de 12/04):

- Feeds de URLs de phishing do OpenPhish [30] (é um serviço de identificação e partilha de URLs de phishing recentes), que contêm para as semanas indicadas 1643 URLs, 1953 URLs, 1765 URLs, 1567 URLs, 2445 URLs e 2476 URLs respetivamente.
- Feeds de URLs de malware do URLhaus [32] (é um serviço que recolhe e partilha URLs de malware recentes), que contém para as semanas indicadas os 5000 primeiros URLs.

Na Tabela 40 apresenta-se a taxa de deteção do modelo em percentagem para estes conjuntos de dados.

Conjunto de dados	19/02	07/03	18/03	25/03	29/03	12/04	Média
OpenPhish (phishing)	98.66	98.67	96.49	98.15	96.97	97.13	97.68
URLhaus (malware)	95.91	96.82	96.48	92.46	92.08	91.14	94.15

Tabela 40 - Validação do modelo final

Analisando os resultados obtidos pode-se concluir que o modelo de classificação produzido está bastante eficaz, tendo obtido uma taxa de deteção de URLs de phishing de 97.68% e uma taxa de deteção de URLs de malware de 94.15%, embora o modelo tenha sido desenvolvido e otimizado especialmente para a deteção de URLs de phishing. Estes resultados também demonstram que o conjunto de dados de treino usado no desenvolvimento do modelo foi bastante representativo do mundo real. No entanto, acredita-se que se o modelo fosse retreinado ao longo das semanas aprenderia grande parte dos novos padrões maliciosos que foram surgindo, o que aumentaria ligeiramente a sua taxa de deteção. Esta situação foi particularmente notória nos conjuntos de dados do URLhaus, onde a taxa de deteção se foi degradando ao longo das semanas. Por esta razão é que existe o container do Agendador para retreinar e atualizar periodicamente o modelo de classificação com novos dados, para que este aprenda continuamente novos padrões maliciosos ao longo do tempo, a fim de se manter atualizado para obter um melhor desempenho.