



Aplicações de modelos de linguagem de grande escala na cibersegurança

TIAGO MIGUEL PESTANA CONCEIÇÃO
(Licenciado)

Trabalho de Projeto para obtenção do grau de mestre em Engenharia Informática e de Computadores, na Área de Especialização de Cibersegurança

Orientador(es):

Prof. Doutor Nuno Miguel Machado Cruz

Júri:

Presidente: Prof. Doutor José Manuel de Campos Lages Garcia Si-

mão

Vogais:

Prof. Doutor Luís Carlos Gonçalves

Setembro 2024

Aplicações de modelos de linguagem de grande escala na cibersegurança

TIAGO MIGUEL PESTANA CONCEIÇÃO
(Licenciado)

Trabalho de Projeto para obtenção do grau de mestre em Engenharia Informática e de
Computadores, na Área de Especialização de Cibersegurança

Orientador(es):

Prof. Doutor Nuno Miguel Machado Cruz

Júri:

Presidente: Prof. Doutor José Manuel de Campos Lages Garcia Si-

mão

Vogais:

Prof. Doutor Luís Carlos Gonçalves

Setembro 2024

Agradecimentos

Primeiramente, gostaria de expressar o meu sincero agradecimento ao Instituto Superior de Engenharia de Lisboa e a todo o corpo docente de MEIC por terem proporcionado uma educação de excelência que me permitiu alcançar os meus objetivos académicos.

Desejo agradecer ao meu orientador, Nuno Cruz, pela sua orientação, paciência e incansável dedicação a esta tese. A sua experiência e conhecimento foram uma fonte constante de inspiração. Agradeço por acreditar no meu potencial e por me desafiar a superar os adversos obstáculos impostos ao longo da realização desta tese.

À minha família, que sempre esteve ao meu lado, mesmo nos momentos mais desafiadores. O vosso apoio incondicional foi um fator motivacional que impulsionou a minha jornada académica. Obrigado por acreditarem em mim.

À minha namorada, Inês, estou eternamente agradecido pelo seu amor incondicional, a sua presença na minha vida motiva-me diariamente a superar-me. O seu suporte constante é um fator imprescindivelmente importante para o meu sucesso pessoal e académico.

Este projeto é o resultado de um esforço coletivo e do apoio de muitos. Cada um de vocês teve um papel significativo na realização deste sonho, e a minha gratidão é eterna.

Declaração de integridade

Declaro que este trabalho de projeto é o resultado da minha investigação pessoal e independente. O seu conteúdo é original e todas as fontes listadas nas referências bibliográficas foram consultadas e estão devidamente mencionadas no texto. Mais declaro que todas as referências científicas e técnicas relevantes para o desenvolvimento do trabalho estão devidamente citadas e constam das referências bibliográficas.

O autor

Lisboa, 30 de setembro de 2024

Resumo

A crescente complexidade e sofisticação das ameaças no ciberespaço têm impulsionado a procura por soluções inovadoras e eficientes no campo da cibersegurança. Neste contexto, conduziu-se uma investigação com o objetivo de avaliar a viabilidade de *Large Language Models* (LLMs) no que concerne à automatização da geração de código e configurações no âmbito da cibersegurança. A investigação centrou-se em mecanismos de ciberdefesa e aplicações de educação em cibersegurança, com particular ênfase em soluções de geração de *honeypots*, *malware* e exercícios de *Capture The Flag* (CTF). Foram avaliados sete modelos, incluindo o GPT-4, Gemini Pro e Claude Opus 3.

A metodologia de avaliação assentou no desenvolvimento de dois mecanismos de avaliação, sendo o primeiro um novo *benchmark Cybersecurity Language Understanding* (CSLU), baseado no *Massive Multitask Language Understanding*, constituído por questões de escolha múltipla sobre diversos domínios do conhecimento. As *prompts* foram concebidas com o intuito de avaliar o estado de conhecimento de cada modelo relativamente aos tópicos supracitados. O segundo mecanismo consistiu na avaliação da consistência, criatividade e adaptabilidade dos modelos referente à geração de artefactos.

Os resultados evidenciaram uma notória proeminência referente ao tópico de *malware*, com quatro destes a alcançarem a pontuação máxima. Por outro lado, o desempenho na tarefa de CTF revelou uma maior variação de resultados. De um modo geral, os modelos GPT-4, Gemini Pro e Claude 3 Opus demonstraram resultados consistentemente superiores entre os modelos estudados.

Num segundo momento, pretendeu-se desenvolver uma ferramenta baseada na web, com o objetivo de fornecer uma prova de conceito dos estudos anterior realizados. A referida ferramenta, recorrendo aos melhores LLMs estudados, permite ao utilizador criar e lançar automaticamente serviços de segurança, como os mencionados *honeypots* ou exercícios de CTF.

De uma perspetiva global, estas descobertas sugerem que a aplicação de LLMs em atividades de cibersegurança pode ser altamente vantajosa.

Palavras chave

LLM, Cibersegurança, Exercícios de CTF, Honeypots, deteção de *malware*.

Abstract

The growing complexity and sophistication of threats in cyberspace has driven the search for innovative and efficient solutions in the field of cybersecurity. In this context, research was conducted to assess the feasibility of Large Language Models (LLMs) for automating code generation and configurations in the field of cybersecurity. The research focused on cyber-defence mechanisms and cybersecurity education applications, with particular emphasis on solutions for generating honeypots, malware and Capture The Flag (CTF) exercises. Seven models were evaluated, including the GPT-4, Gemini Pro and Claude Opus 3.

The evaluation methodology was based on the development of two evaluation mechanisms, the first being a new benchmark Cybersecurity Language Understanding (CSLU), based on the Massive Multitask Language Understanding, consisting of multiple-choice questions on various areas of knowledge. The prompts were designed to assess the state of knowledge of each model in relation to the aforementioned topics. The second mechanism consisted of assessing the consistency, creativity and adaptability of the models in terms of artefact generation.

The results showed a notable prominence in the topic of malware, with four of them achieving the maximum score. On the other hand, performance in the CTF task revealed a greater variation in results. In general, the GPT-4, Gemini Pro and Claude 3 Opus models showed consistently superior results among the models studied.

As a second step, the aim was to develop a web-based tool to provide a proof of concept of the previous studies. This tool, using the best LLMs studied, allows the user to automatically create and launch security services, such as the aforementioned honeypots or CTF exercises.

From a global perspective, these findings suggest that the application of LLMs in cybersecurity activities can be highly advantageous.

Key words

Large Language Models, AI in Cybersecurity, CTF Challenges, Honeypots, Malware detection.

Índice

Agradecimentos	i
Declaração de integridade	iii
Resumo	v
Abstract	vii
Abreviaturas	xvii
1 Introdução	1
1.1 Enquadramento	1
1.2 Objetivos	2
1.3 Publicações	3
1.4 Estrutura do documento	3
2 Estado de arte	5
2.1 LLMs	5
2.1.1 Arquitetura de LLMs	6
2.1.2 Modalidade dos modelos	7
2.1.3 Google	7
2.1.4 OpenAi	8
2.1.5 Meta	10
2.1.6 Anthropic	11
2.1.7 Mistral	11
2.2 Prompt Engineering	12
2.3 Benchmarks	14
2.4 Mecanismos de ciberdefesa	14
2.4.1 Trabalho Relacionado	17
2.5 Exercícios de CTF	17
2.5.1 Trabalho relacionado	18
2.6 Desenvolvimento orientado para a Web	18
2.6.1 Interface com o Utilizador	19
2.6.2 Servidor aplicacional	20

2.6.3	Persistência de dados	23
2.6.4	Disponibilização de serviços	24
2.6.5	Conclusão sobre tecnológicas escolhidas	25
3	Avaliação de LLMs	27
3.1	Modelos a avaliar	27
3.2	Cybersecurity Language Understanding	27
3.2.1	Métricas de avaliação	28
3.2.2	Prompts	28
3.2.3	Resultados	30
3.2.4	Discussão	30
3.3	Avaliação automática de LLMs	32
3.3.1	Métricas de avaliação	32
3.3.2	Prompts	33
3.3.3	Avaliação de respostas	36
3.3.4	Resultados	36
3.3.5	Discussão	38
4	Arquitetura e Implementação Proposta	41
4.1	Análise de requisitos	41
4.2	Arquitetura geral da solução	42
4.3	Arquitetura detalhada da solução	45
4.3.1	Frontend	46
4.3.2	Backend	47
4.3.3	Modelo de dados	47
4.4	Considerações Adicionais de Implementação	49
5	Validação da solução	51
5.1	Metodologia de validação	51
5.2	Ambiente de teste	51
5.3	Avaliação	52
5.3.1	Geração de <i>Honeypots</i>	52
5.3.2	Geração de CTFs	53
5.3.3	Deteção de <i>malware</i>	54
5.3.4	Instanciação de serviços	55
6	Conclusões	59
6.1	Trabalho futuro	60
A	Prompts e resultados da avaliação automatizada	61
A.1	<i>Prompt</i> de avaliação	61
A.2	CTF	61

A.2.1	Resposta correta	61
A.2.2	Resposta Incorreta	62
A.2.3	Resposta plausível	63
A.3	Honeypots	64
A.3.1	Resposta correta	64
A.3.2	Resposta Incorreta	65
A.3.3	Resposta plausível	66
A.4	<i>Malware</i>	67
A.4.1	Resposta correta	67
A.4.2	Resposta plausível	68
A.4.3	Resposta Incorreta	69
	Bibliografia	79

Índice de figuras

2.1	A arquitetura do modelo <i>Transformer</i> . Retirado de Vaswani et al. (2023)	6
3.1	Exemplo de pergunta - Honeypot	29
3.2	Exemplo de pergunta - CTF	29
3.3	Exemplo de pergunta - Malware	30
3.4	Resultados obtidos ao estudo das diversas tarefas e respetiva classificação média. 31	
3.5	Resultados obtidos ao estudo das diversas tarefas e respetiva classificação média. 31	
3.6	Exemplo de aplicação de <i>Prompt Engineering</i>	33
3.7	Exemplo de prompt de avaliação	37
3.8	Resultados obtidos	38
4.1	Casos de utilização	41
4.2	<i>Mockup</i> da página inicial	42
4.3	<i>Mockup</i> da página de autenticação	43
4.4	A arquitetura geral da solução	44
4.5	A arquitetura geral do servidor de <i>frontend</i>	44
4.6	A arquitetura geral do servidor de <i>backend</i>	45
4.7	A arquitetura detalhada do servidor de <i>frontend</i>	46
4.8	A arquitetura detalhada do servidor de <i>backend</i>	48
4.9	Estrutura do modelo de dados	48
5.1	Execução da <i>prompt</i> de validação da geração de <i>honeypots</i>	53
5.2	Resposta da geração de <i>honeypots</i>	53
5.3	Validação da geração de <i>honeypots</i>	54
5.4	Pedido para a criação de CTF	54
5.5	Resposta do modelo para a criação de CTF	55
5.6	<i>Prompt</i> e resposta sobre análise de <i>malware</i>	55
5.7	Contentor com servidor Nginx criado	56
5.8	Contentor com servidor Nginx instanciado	56
5.9	Validação da instanciação de serviços	57

Índice de tabelas

- 2.1 Comparação de resultados de diversos modelos, tais como os modelos Gemini, GPT-4, GPT-3.5, PaLM 2-L, LLAMA-2, entre outros, em alguns *benchmarks* conhecidos. Retirado de Team et al. (2023). 9
- 2.2 Comparação de resultados dos modelos LLama 2 70B, GPT-3.5 e Mixtral 8x7B em diversos *benchmarks*. Retirado de Jiang et al. (2024). 12
- 3.1 Modelos a avaliar 28
- 3.2 Mapa de calor dos resultados obtidos no ensaio experimental 37
- 4.1 Modelo de dados de mensagens 49

Abreviaturas

LLM Large Language Model

IA Inteligência artificial

CTF Capture the Flag

TFM Trabalho Final de Mestrado

Capítulo 1

Introdução

No atual panorama de cibersegurança global, caracterizado por uma rápida evolução, as organizações deparam-se com desafios significativos na proteção das suas redes e sistemas face a ameaças progressivamente mais sofisticadas. O avanço tecnológico acelerado e a crescente dependência de sistemas interligados tornaram a cibersegurança um componente fulcral da estratégia de defesa. A natureza complexa e dinâmica do ecossistema de ameaças requer abordagens inovadoras e adaptativas para assegurar a integridade, confidencialidade e disponibilidade da informação sensível. A necessidade de manter os profissionais atualizados com as últimas técnicas e ferramentas de cibersegurança exige um esforço constante de formação.

Os modelos de linguagem de grande escala (LLMs) têm ganho muito interesse nos últimos anos devido à sua capacidade de gerar texto coerente e realizar tarefas de processamento de linguagem natural Liang et al. (2024). Os LLMs são modelos de inteligência artificial que usam técnicas de aprendizagem profunda para processar e entender a linguagem natural. Esses modelos são treinados em grandes quantidades de dados de texto para aprender padrões e relações entre entidades no idioma e deste modo, adquirir conhecimento sobre o mundo real e sobre diversos domínios.

Dado a sua crescente valorização, os modelos desta natureza têm-se tornado uma peça revolucionária não só no nosso quotidiano, como no setor tecnológico e académico. Essas inovações têm transformado a forma como interagimos com a informação, a tecnologia e o mundo ao nosso redor, moldando diversos aspetos de nossas vidas.

1.1 Enquadramento

Ao reconhecer a necessidade urgente de fortalecer as capacidades de cibersegurança das organizações Fadziso et al. (2023), considera-se crucial explorar a integração de LLMs com sistemas de cibersegurança. Ao alavancar a inteligência artificial avançada, pretende-se abordar proativamente os desafios de segurança cibernética, proporcionando uma resposta eficiente às ameaças em constante evolução.

Estes desafios requerem uma solução inovadora que permita às organizações manterem-se na vanguarda da cibersegurança, respondendo eficazmente às ameaças em evolução, en-

quanto otimizam a utilização dos seus recursos humanos.

A presente tese procura avaliar o uso deste tipo de modelos para automatizar o desenvolvimento de mecanismos de cibersegurança, através da avaliação do estado atual de maturação em relação aos tópicos a estudar. Através da elaboração da mesma pretende-se acrescentar valor no domínio da utilização de LLMs no âmbito cibersegurança. Numa primeira fase, irá ser aprofundada a interseção entre processamento da linguagem natural e da segurança da informação, através da realização de 2 estudos que visam avaliar o uso deste tipo de modelos para automatizar o desenvolvimento de mecanismos de cibersegurança. Dentro do vasto grupo de áreas e aplicações na cibersegurança, foram considerados três tópicos para incorporarem a avaliação em causa, sendo estes: exercícios de CTF, *honeypots* e *malware*. Será estudado o tópico dos exercícios de CTF devido às características formativas que o define, os CTFs são uma parte importante da cultura da cibersegurança e permitem de uma forma dinâmica contribuir para capacitar e treinar os profissionais de cibersegurança. Os tópicos *honeypots* e *malware* serão estudados por estes serem considerados temas nucleares para a melhorar a postura de ciberdefesa das organizações.

Posteriormente numa segunda fase, tendo em conta os resultados obtidos nas avaliações realizadas anteriormente, será conceptualizada e desenvolvida uma ferramenta capaz de integrar os modelos mais aptos com capacidades de instanciar os artefactos gerados pelos mesmos.

1.2 Objetivos

Esta tese tem como objetivo principal ao estudar o potencial de usar LLMs pré-treinados para tarefas de cibersegurança, um domínio crítico que pode beneficiar grandemente de técnicas de inteligência artificial, contribuir para melhorar a postura das organizações, mas também otimizar a alocação de recursos e aumentar a eficácia operacional tanto para fins defensivos como para fins formativos.

É pretendido investigar a viabilidade de LLMs para a geração automática de código e configurações para implementar mecanismos de ciberdefesa, como *honeypots* e ambientes de captura de ameaças, assim como para a geração automática de código para implementar aplicações para ensino da cibersegurança, como exercícios de CTF e mecanismos de segurança para deteção de *malware*.

De modo a criar uma prova de conceito ao estudo realizado sobre estes modelos, é pretendido desenvolver uma ferramenta baseada no desenvolvimento para a web, que permite ao utilizador, recorrendo a estes modelos, criar e lançar automaticamente serviços relacionados com a cibersegurança.

1.3 Publicações

Foi desenvolvido um artigo científico¹ em volta da investigação realizada na secção 3.2, no âmbito da sua submissão na sessão de Segurança de Sistemas de Computadores e Comunicações, na décima quinta edição do INForum - Simpósio Nacional de Informática², tendo este sido posteriormente aceite pela comissão organizadora da sessão em causa. Consequentemente, a aceitação do artigo resultou na sua posterior apresentação juntamente com a divulgação da investigação através de um poster nos dias 5 e 6 de setembro no seu evento presencial realizado na Faculdade de Ciências da Universidade de Lisboa.

O INForum tem como objetivo ser um evento privilegiado de reunião da comunidade nacional nas diversas vertentes da informática e ambiciona ser o fórum de eleição para a divulgação, discussão e reconhecimento de trabalhos científicos. O INForum surge como particular oportunidade como palco para a estreia de jovens investigadores que buscam a divulgação, a crítica construtiva e o encorajamento ao seu trabalho. O facto de um artigo baseado numa das etapas integrantes na investigação presente nesta tese ter sido reconhecido num evento de dimensão nacional releva o seu rigor científico e a contribuição que apresenta à comunidade científica.

1.4 Estrutura do documento

Este documento é composto por 4 capítulos. O primeiro capítulo, sendo um capítulo introdutório, é composto por 3 secções procedendo a uma introdução do tema de investigação, o enquadramento, os objetivos e a estrutura do mesmo. O segundo capítulo corresponde ao estado da arte, onde serão destacadas as contribuições mais recentes e significativas na área que esta tese explora, juntamente com o estudo realizado aos mais reconhecidos e famosos LLMs e respetivas empresas. É também abordado neste capítulo as mais populares ferramentas de desenvolvimento para as diferentes camadas da ferramenta a desenvolver. O terceiro capítulo apresenta o trabalho desenvolvido. Neste capítulo será demonstrado o estudo experimental realizado, seguidamente da conceptualização, planeamento e desenvolvimento da solução apresentada. O quarto e último capítulo, é destinado para a demonstração do planeamento do trabalho futuro, assim como a referência do progresso de um artigo baseado no estudo experimental realizado no âmbito desta tese.

¹https://www.inforum.pt/static/files/papers/INForum_2024_paper_19.pdf

²<https://www.inforum.pt/cfp>

Capítulo 2

Estado de arte

O presente capítulo visa situar o leitor no contexto histórico e contemporâneo da pesquisa, procura estabelecer uma base sólida para a investigação em questão e destacar as principais tendências e descobertas. Serão explorados os primórdios dos modelos de linguagem, desde os métodos tradicionais até às mais recentes arquiteturas baseadas em *transformers*. Serão estudadas abordagens mono e multimodais, que integram eficazmente texto com outros formatos de dados. Introduzidos estes conceitos teóricos serão abordados os mais recentes modelos das principais empresas que atuam nesta área. É igualmente pretendido neste capítulo enunciar trabalho e investigações relacionadas com a presente tese.

É crucial realizar o estudo sobre tópicos tais como *Prompt Engineering* e *benchmarks* de LLMs, de modo a compreender as melhores práticas da atualidade referente ao estudo massificado e normalizado de LLMs, com o objetivo de estruturar o estudo e consequentemente atingir os melhores e mais robustos resultados.

Por último, será realizado um estudo sobre um diversas tecnologias integrantes no desenvolvimento de aplicações web, de modo a realizar uma escolha fundamentada e lógica sobre o *stack* tecnológico a usar no desenvolvimento da pretendida ferramenta.

2.1 LLMs

Os Modelos de Linguagem em Larga Escala, ou LLMs, são considerados atualmente o estado da arte em Inteligência Artificial e representam um grande potencial para transformar diversos domínios por meio do conhecimento aprendido. Estes modelos são dedicados ao processamento e compreensão da linguagem natural e são treinados por meio de um extenso conjunto de dados que abrange uma ampla variedade de contextos linguísticos. O processo de treino envolve a exposição do modelo a grandes quantidades de dados, geralmente texto, permitindo-lhe aprender padrões e nuances da linguagem.

Os LLMs têm apresentado uma série de avanços significativos nos últimos anos, tanto em termos arquiteturais, como em termos de parametrização, aplicabilidade e impacto no mundo real.

2.1.1 Arquitetura de LLMs

No contexto de Linguagem Natural e Modelos de Linguagem, a arquitetura baseada em *transformers* trouxe avanços significativos. Antes dos *transformers*, muitos modelos dependiam de arquiteturas recorrentes (RNNs) ou convolucionais (CNNs) para lidar com sequências de dados, mas essas abordagens apresentavam desafios em termos de eficiência computacional e captura de dependências de longo alcance. A referida arquitetura introduzida em 2017 pelo Google AI em Vaswani et al. (2023) consiste numa arquitetura de rede neuronal que baseia-se no mecanismo de atenção para processar sequências de dados. Esta arquitetura é composta por vários blocos de transformadores, cada um dos quais consiste em um mecanismo de atenção e uma camada linear, como demonstra a figura 2.1. O mecanismo de atenção permite que o transformador se concentre em partes específicas de uma sequência de dados, enquanto a camada linear permite que o transformador aprenda as relações entre essas partes.

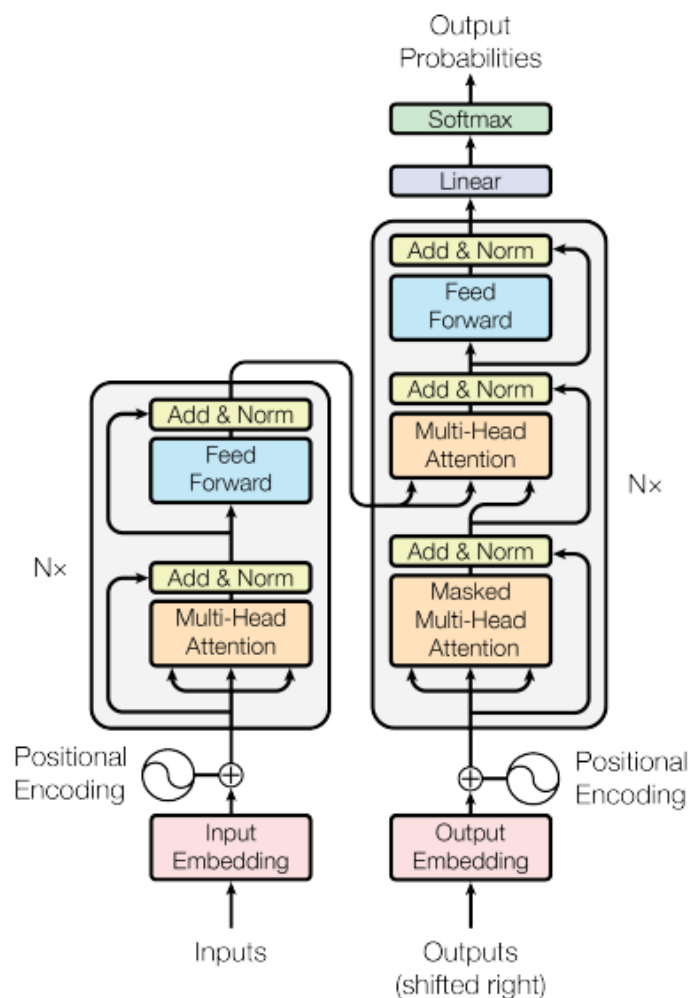


Figura 2.1 A arquitetura do modelo *Transformer*. Retirado de Vaswani et al. (2023)

No âmbito de LLMs, as arquiteturas baseadas em *transformers* têm sido amplamente recorridas e têm alcançado resultados notáveis numa ampla variedade de tarefas, como a tradução automática, no qual os *transformers* permitiram o desenvolvimento de sistemas de

tradução que são mais precisos e fluentes do que os sistemas anteriores. Estes também podem ser usados para responder a perguntas de forma mais abrangente e informativa do que os métodos anteriores.

Outra tarefa onde estes modelos revelaram resultados notáveis foi o resumo textual, ao demonstrar grande adaptabilidade para gerar resumos de texto de forma mais informativa e concisa do que os métodos anteriores. Os transformadores são capazes de alcançar esses resultados por causa da sua capacidade de aprender longas dependências entre os *tokens* de uma sequência. Isso permite que os transformadores capturem a estrutura semântica complexa do texto e do código.

Além disso, estes são relativamente eficientes em termos computacionais, o que representa uma mais valia para treinar modelos de linguagem de grande dimensão. As arquiteturas baseadas em *transformers* têm desempenhado um papel crucial no avanço dos LLMs, trazendo consigo uma panóplia de benefícios que redefinem os padrões de desempenho e eficiência.

2.1.2 Modalidade dos modelos

Atualmente os modelos podem ser divididos em dois tipos: mono-modais e multi-modais. Os modelos mono-modais focam-se exclusivamente na interpretação e geração de texto. Estes modelos, treinados com vastos conjuntos de dados linguísticos, destacam-se na compreensão de padrões e contextos textuais, sendo amplamente utilizados em tarefas como sumarização, tradução automática e resposta a perguntas.

Por outro lado, os modelos multi-modais, transcendem as barreiras do texto e incorporam diversas modalidades, como imagem e, por vezes, também áudio ou vídeo. Esta abordagem permite a compreensão de relações complexas entre diferentes tipos de dados, consequentemente possibilita a realização de tarefas como descrição de imagens, tradução entre texto e imagem, e outras aplicações que exigem uma compreensão integrada de múltiplas fontes de informação. Enquanto os modelos mono-modais são especializados em dados textuais, os modelos multi-modais destacam-se pela capacidade de lidar com a riqueza de informações provenientes de várias fontes sensoriais.

2.1.3 Google

No panorama em constante evolução da inteligência artificial e de modelos de linguagem, destacam-se dois proeminentes produtos da Google que prometem redefinir as interações conversacionais e a compreensão contextual. O Gemini Team et al. (2023) apresentou notáveis evoluções em tarefas como compreensão de imagem, áudio, vídeo e texto. Os modelos Gemini baseiam-se nos descodificadores *Transformer* Vaswani et al. (2023), com melhorias na arquitetura e otimização do modelo para permitir um treino estável em larga escala e inferência otimizada nas Unidades de Processamento Tensorial (TPUs) da Google. Os modelos Gemini são treinados para possibilitar a entrada textual intercalada com uma ampla variedade

de entradas audiovisuais, como imagens, gráficos, capturas de ecrã, PDFs e vídeos, e detém capacidades de gerar saídas de texto e imagem. A codificação visual dos modelos Gemini é inspirada em trabalhos como Flamingo Alayrac et al. (2022), CoCa Yu et al. (2022) e PaLI Chen et al. (2023), com a distinção crucial de serem conceptualizados multimodais desde o início, desta forma possibilita a produção de imagens diretamente através de *tokens* de imagem discretos. A compreensão de vídeo é obtida através da codificação do vídeo como uma sequência de *frames* na ampla janela de contexto. *Frames* de vídeo ou imagens podem ser intercalados naturalmente com texto ou áudio como parte da entrada do modelo. Os modelos são capazes de lidar com resoluções de entrada variáveis, direcionando mais recursos computacionais para tarefas que demandam uma compreensão mais refinada. Adicionalmente, o Gemini pode processar diretamente sinais de áudio a 16kHz provenientes do Universal Speech Model (USM) Zhang et al. (2023a), capturando nuances frequentemente perdidas quando o áudio é mapeado ingenuamente para uma entrada de texto.

A família Gemini desdobra-se em 3 modelos de distintas dimensões: Ultra, Pro e Nano. O treino da família de modelos Gemini exigiu inovações em algoritmos de treino, conjuntos de dados e infraestrutura. No caso do modelo Pro, a escalabilidade intrínseca da infraestrutura e dos algoritmos de aprendizagem possibilitou a conclusão do pré-treino em questão de semanas, ao utilizar apenas uma fração dos recursos do Ultra. A série de modelos Nano beneficiou de avanços adicionais em destilação e algoritmos de treino para gerar os melhores modelos de linguagem para uma variedade de tarefas, como sumarização e compreensão de leitura.

Cada um deles é adequado para aplicações que abrangem desde tarefas complexas de raciocínio até casos de uso com limitações de memória em dispositivos. A avaliação realizada no artigo enunciado demonstra uma ampla gama de *benchmarks* revela que o modelo mais sofisticado, o Gemini Ultra, impulsiona o estado da arte em 30 de 32 dos referidos *benchmarks*, destacando-se como o pioneiro ao atingir um desempenho equivalente ao de especialistas humanos no conhecido *benchmark* MMLU. Adicionalmente, observou-se uma melhoria significativa no estado da arte em todos os 20 *benchmarks* multimodais realizados, assim como é demonstrada na tabela 2.1. Acredita-se que as inovadoras capacidades dos modelos Gemini em raciocínio intermodal e compreensão de linguagem abrirão portas para uma variedade extensa de casos de uso.

2.1.4 OpenAI

A OpenAI é uma das empresas que lideram significativa no panorama dos LLMs, lançou várias gerações de modelos de linguagem de larga escala, tais como o GPT-3, o GPT-3.5, o GPT-4 e mais recentemente, o GPT-4o. Os LLMs da OpenAI têm uma ampla gama de aplicabilidade. Podem ser usados para gerar texto criativo, traduzir idiomas, responder a perguntas e até mesmo criar assistentes pessoais. A empresa está a trabalhar para desenvolver novas aplicações para os seus modelos, à medida que eles tornam-se mais sofisticados.

O GPT-4, é a quarta e mais recente geração do seu modelo de linguagem de grandes dimensões. Este modelo, à semelhança da geração atual de modelos, recorre a transforma-

	Gemini Ultra	Gemini Pro	GPT-4	GPT-3.5	PaLM 2-L	Claude 2	Inflexion-2	Grok 1	LLAMA-2
MLLU Multiple-choice questions in 57 subjects (professional & academic) (Hendrycks et al., 2021a)	90.04% CoT@32*	79.13% CoT@8*	87.29% CoT@32 (via API**)	70% 5-shot	78.4% 5-shot	78.5% 5-shot CoT	79.6% 5-shot	73.0% 5-shot	68.0%***
	83.7% 5-shot	71.8% 5-shot	86.4% 5-shot (reported)						
GSM8K Grade-school math (Cobbe et al., 2021)	94.4% Maj1@32	86.5% Maj1@32	92.0% SFT & 5-shot CoT	57.1% 5-shot	80.0% 5-shot	88.0% 0-shot	81.4% 8-shot	62.9% 8-shot	56.8% 5-shot
MATH Math problems across 5 difficulty levels & 7 subdisciplines (Hendrycks et al., 2021b)	53.2% 4-shot	32.6% 4-shot	52.9% 4-shot (via API**)	34.1% 4-shot (via API**)	34.4% 4-shot	—	34.8%	23.9% 4-shot	13.5% 4-shot
			50.3% (Zheng et al., 2023)						
BIG-Bench-Hard Subset of hard BIG-bench tasks written as CoT problems (Srivastava et al., 2022)	83.6% 3-shot	75.0% 3-shot	83.1% 3-shot (via API**)	66.6% 3-shot (via API**)	77.7% 3-shot	—	—	—	51.2% 3-shot
HumanEval Python coding tasks (Chen et al., 2021)	74.4% 0-shot (IT)	67.7% 0-shot (IT)	67.0% 0-shot (reported)	48.1% 0-shot	—	70.0% 0-shot	44.5% 0-shot	63.2% 0-shot	29.9% 0-shot
Natural2Code Python code generation. (New held-out set with no leakage on web)	74.9% 0-shot	69.6% 0-shot	73.9% 0-shot (via API**)	62.3% 0-shot (via API**)	—	—	—	—	—
DROP Reading comprehension & arithmetic. (metric: F1-score) (Dua et al., 2019)	82.4 Variable shots	74.1 Variable shots	80.9 3-shot (reported)	64.1 3-shot	82.0 Variable shots	—	—	—	—
HellaSwag (validation set) Common-sense multiple choice questions (Zellers et al., 2019)	87.8% 10-shot	84.7% 10-shot	95.3% 10-shot (reported)	85.5% 10-shot	86.8% 10-shot	—	89.0% 10-shot	—	80.0%***
WMT23 Machine translation (metric: BLEURT) (Tom et al., 2023)	74.4 1-shot (IT)	71.7 1-shot	73.8 1-shot (via API**)	—	72.7 1-shot	—	—	—	—

Tabela 2.1 Comparação de resultados de diversos modelos, tais como os modelos Gemini, GPT-4, GPT-3.5, PaLM 2-L, LLAMA-2, entre outros, em alguns *benchmarks* conhecidos. Retirado de Team et al. (2023).

dores Vaswani et al. (2023), o que significa que utiliza uma arquitetura de rede neural que é particularmente eficaz na aprendizagem de padrões textuais. O modelo tem 1.76 bilhões de parâmetros, o que é mais do dobro do número de parâmetros do seu antecessor, o GPT-3. Este aumento no número de parâmetros permite ao GPT-4 gerar texto mais complexo, com uma melhor compreensão das nuances da linguagem humana. Este modelo apresenta um desempenho de nível humano em vários *benchmarks* profissionais e acadêmicos, incluindo a aprovação num exame com uma pontuação na faixa dos 10% superiores dos examinados OpenAI et al. (2023). O processo de alinhamento pós-treino resulta em um desempenho aprimorado em medidas de factualidade e aderência ao comportamento desejado. Um componente central do projeto de desenvolvimento deste modelo foi desenvolver uma infraestrutura e métodos de otimização que se comportam de forma previsível em uma ampla faixa de escalas. Isso permitiu aos autores de OpenAI et al. (2023) preverem com precisão alguns aspectos do desempenho do GPT-4 com base em modelos treinados com não mais que 1/1.000 do poder computacional do GPT-4.

Sendo este modelo uma ferramenta poderosa que tem o potencial de ser usada para melhorar o cotidiano da generalidade da população de muitas maneiras. No entanto, também

existem potenciais riscos associados ao uso deste modelo. Por exemplo, num estágio precoce do GPT-4, este podia ser usado para gerar texto enganoso ou prejudicial, ou para criar conteúdo de natureza discriminatória. É importante ser consciente destes riscos e tomar medidas para mitigá-los, assim como é abordado em OpenAI (2023a), no qual foram documentados e destacados desafios de segurança apresentados pelas limitações do modelo. Foi fornecida uma visão geral de alto nível dos processos de segurança adotados pela OpenAI para preparar o GPT-4 para implementação, esta abrange o trabalho em medidas, alterações ao nível do modelo e intervenções ao nível do produto e do sistema. Por fim, foi demonstrado que, embora as mitigações e processos implementados alterem o comportamento do GPT-4 e evitem certos tipos de usos indevidos, são limitados e permanecem frágeis em alguns casos.

Em setembro de 2023, a OpenAI apresentou o desenvolvimento do GPT-4V(ision) OpenAI (2023b), uma versão multimodal do modelo de linguagem GPT-4. O GPT-4V(ision) é um modelo transformador com 175 mil milhões de parâmetros, que pode receber entradas de texto e imagens e produzir saídas de texto. O modelo é composto por dois componentes principais: um modelo de linguagem transformador e um modelo de visão por computador. O modelo de linguagem transformador é responsável pela geração de texto. É treinado num conjunto de dados de texto e código e é capaz de gerar texto criativo, traduzir idiomas, responder a perguntas e até mesmo criar assistentes pessoais. Por outro lado, o modelo de visão é responsável pelo processamento de imagens. É treinado num conjunto de dados de imagens, e é capaz de identificar objetos, reconhecer rostos e até mesmo gerar imagens.

Mais recentemente, em agosto de 2024, foi apresentado o GPT-4o OpenAI (2024), um modelo multimodal auto-regressivo que aceita como entrada qualquer combinação de texto, áudio, imagem ou vídeo, e apresenta capacidades de gerar texto, áudio e imagem. O GPT-4o manteve os níveis de desempenho do GPT-4 Turbo em tarefas relacionadas a texto em inglês e código, apresenta também melhorias significativas noutras linguagens, tanto em questões de desempenho como gestão de esforço. Este modelo destaca-se especialmente em tarefas de compreensão de visão e áudio em comparação com os modelos existentes.

2.1.5 Meta

No mês de julho de 2024, a Meta apresentou a terceira geração de LLMs pré-treinados e refinados para tarefas de *chatting* e diálogo, o Llama3.

Em Dubey et al. (2024), os autores apresentaram a nível técnico esta geração de modelos, denominada por Llama 3. A mencionada geração trata-se de um grupo de LLMs que suportam nativamente a multilinguagem, codificação, raciocínio e utilização de ferramentas. O maior modelo, cujo segue uma arquitetura baseada em *transformers*, é composto por 405 mil milhões de parâmetros e uma janela de contexto de até 128 mil tokens. No referido artigo, os autores apresentam uma avaliação empírica extensa do Llama 3. Foi constatado que o Llama 3 oferece qualidade comparável a LLMs que lideram o mercado, como o GPT-4, numa variedade de tarefas. A estreia pública do Llama 3, inclui versões pré-treinadas e pós-treinadas do modelo de 405 mil milhões de parâmetros. O artigo também apresenta os resultados de en-

saiois experimentais em que foram integradas capacidades de imagem, vídeo e fala no Llama 3 através de uma abordagem composicional. Observou-se que a referida abordagem desempenha competitivamente com o estado da arte em tarefas de reconhecimento de imagem, vídeo e fala.

2.1.6 Anthropic

Os modelos desenvolvidos pela Anthropic representaram uma mudança no paradigma no domínio da inteligência artificial, estes conseguiram encapsular uma classe de modelos sofisticados projetados para compreender e gerar texto semelhante às capacidades humanas numa escala sem precedentes.

Os autores de Anthropic (2024a) introduziram a mais recente família de LLMs multimodais da empresa, nomeada de Claude 3. Esta família é composta essencialmente por 3 LLMs, o Claude 3 Opus, apresentado-se como o modelo mais capaz, o Claude 3 Sonnet, que proporciona uma combinação de habilidades e velocidade, e o Claude 3 Haiku, o modelo menos dispendioso. Todos estes modelos possuem capacidades de analisar dados de imagem. A família Claude 3 demonstrou um desempenho robusto em avaliações de *benchmark* e estabeleceu um novo padrão em medidas de raciocínio, matemática e codificação. O Claude 3 Opus atingiu resultados relevantes em vários *benchmarks*, como por exemplo o MMLU Hendrycks et al. (2021). O Claude 3 Haiku apresentou um desempenho igual ou superior ao Claude 2 Anthropic (2024b) na maioria das tarefas de texto, enquanto o Sonnet e o Opus o superaram significativamente. Adicionalmente, estes modelos exibiram uma fluência melhorada em línguas não inglesas, tornando-os mais versáteis para um público global. Neste artigo, foi fornecida uma análise aprofundada das avaliações realizadas, focando-se nas capacidades centrais, segurança, impactos sociais e nas avaliações de risco catastrófico.

2.1.7 Mistral

A equipa de IA da Mistral no dia 11 de dezembro de 2023 apresentou o modelo Mixtral 8x7B Jiang et al. (2024), um modelo de mistura esparsa de especialistas (SMoE) de alta qualidade com pesos abertos.

O termo "especialistas" é anunciado como uma arquitetura de rede neural em que diferentes sub-redes são treinadas para se especializar em diferentes aspetos dos dados de entrada, e uma rede de controle determina qual especialista ou combinação de especialistas usar para uma entrada específica. A esparsidade em "Sparse Mixture of Experts" implica que nem todos os especialistas estão ativos para cada entrada, em vez disso, um subconjunto dos especialistas é selecionado com base nos dados de entrada, tornando o modelo mais eficiente e escalável. Essa arquitetura é projetada para lidar com dados complexos e diversos, permitindo que diferentes especialistas se concentrem em diferentes partes do espaço de entrada. O aspeto de esparsidade ajuda a gerir a complexidade computacional associada a um grande número de especialistas.

O Mixtral é um modelo descodificador, onde o bloco de alimentação direta seleciona a partir de um conjunto de 8 grupos distintos de parâmetros. Em cada camada, para cada *token*, uma rede de encaminhamento escolhe dois desses grupos (os denominados "especialistas") para processar o *token* e combinar os seus resultados aditivos. Essa técnica aumenta o número de parâmetros de um modelo e ao mesmo tempo controla o custo e a latência do mesmo, pois o modelo usa apenas uma fração do conjunto total de parâmetros por *token*. Em particular, o Mixtral tem 46,7 milhares de milhões de parâmetros totais, mas usa apenas 12,9 milhares de milhões de parâmetros por *token*. Portanto, ele processa entrada e gera saída na mesma velocidade e pelo mesmo custo que um modelo de 12,9 mil milhões de parâmetros.

A equipa da Mistral comparou o Mixtral à família Llama 2 e ao modelo base GPT3.5, os resultados obtidos, parcialmente presentes na tabela 2.2, demonstraram que este modelo igualou ou superou em questões de desempenho tanto o modelo Llama 2 70B como o GPT3.5 na maioria dos testes realizados. Segundo os autores, este modelo demonstrou um forte desempenho em tarefas de geração de código.

	LLaMA 2 70B	GPT - 3.5	Mixtral 8x7B
MMLU (MCQ in 57 subjects)	69.9%	70.0%	70.6%
HellaSwag (10-shot)	87.1%	85.5%	86.7%
ARC Challenge (25-shot)	85.1%	85.2%	85.8%
WinoGrande (5-shot)	83.2%	81.6%	81.2%
MBPP (pass@1)	49.8%	52.2%	60.7%
GSM-8K (5-shot)	53.6%	57.1%	58.4%
MT Bench (for Instruct Models)	6.86	8.32	8.30

Tabela 2.2 Comparação de resultados dos modelos LLaMA 2 70B, GPT-3.5 e Mixtral 8x7B em diversos *benchmarks*. Retirado de Jiang et al. (2024).

2.2 Prompt Engineering

A *Prompt Engineering* é uma área dedicada ao desenvolvimento e otimização de *prompts* para sistemas de inteligência artificial generativa. É considerada uma área importante, pois tem o potencial de melhorar significativamente a eficácia e desempenho dos modelos de inteligência artificial generativa. *Prompts* eficazes podem alavancar a qualidade e consistência das respostas dos modelos de inteligência artificial.

Existem alguns projetos de documentação que abordam a área de *Prompt Engineering*. Um dos mais conhecidos é o *Prompt Cookbook*¹ da OpenAI. Este *cookbook* é uma coleção *open-source* de exemplos e guias sobre como criar *prompts* eficazes para modelos de linguagem. O projeto em questão recebeu bastantes contribuições de diversos estudos e artigos publicados, como por exemplo em Wei et al. (2023), em que os autores exploraram uma técnica inovadora denominada "*Chain-of-Thought Prompting*" para otimizar a capacidade de LLMs na realização de compreensão. Esta técnica envolve a divisão estruturada de inferências intermediárias na *prompt*, incentivando os LLMs a desenvolver um processo de pensamento estruturado e lógico. Os testes realizados com três LLMs distintos demonstraram que o "*Chain-of-Thought Prompting*" proporciona melhorias significativas no desempenho em tarefas de conhecimento geral.

Já em Wang et al. (2023), os autores apresentaram uma abordagem para otimizar o desempenho de LLMs em tarefas de raciocínio complexo, denominada "*Self-Consistency*". A metodologia adotada envolve a substituição da descodificação tradicional, utilizada no "*Chain-of-Thought Prompting*", por uma descodificação mais diversificada e consistente. Esta abordagem baseia-se na premissa de que problemas complexos de raciocínio podem ser resolvidos por meio de diferentes estratégias de inferência, de modo a alcançar corretamente o mesmo resultado. Os testes realizados em diversos *benchmarks* de raciocínio aritmético e de cultura geral revelaram que a abordagem "*Self-Consistency*" apresenta melhorias significativas no desempenho dos LLMs. No mesmo segmento de pensamento, os autores de Yao et al. (2023) realçaram a limitação dos LLMs atuais em tarefas que requerem planeamento e processos de decisão em fases iniciais. Os LLMs estudados pelos referidos autores estavam restritos a processos de tomada de decisão token-a-token, cujos seguem uma sequência de inferência linear, o que os impede de considerar alternativas e estratégias mais amplas. Para ultrapassar essas limitações, os autores apresentaram um novo *framework* para inferência de LLMs, chamada "*Tree of Thoughts*", ou *ToT*. O *ToT* baseia-se na abordagem de "*Chain of Thought*", mas permite uma exploração mais flexível e estruturada ao considerar unidades coerentes de texto como etapas intermediárias na resolução de problemas. O *ToT* capacita os LLMs a tomar decisões fundamentadas, ao permitir avaliar diferentes caminhos de raciocínio e escolherem o que melhor se alinha ao objetivo final. Além disso, o *ToT* permite que os modelos realizem tentativas de projeções de cenários futuros e retrocedam quando necessário, garantindo assim uma abordagem mais estratégica e eficaz para a resolução de problemas. Os ensaios experimentais realizados demonstraram o impacto positivo do *ToT*. Por exemplo, numa determinada tarefa, enquanto o GPT-4 ao recorrer à técnica "*Chain-of-Thought Prompting*" apenas resolveu 4% dos problemas, ao utilizar o *ToT* obteve uma taxa de sucesso de 74%.

Outro projeto de documentação importante é o *Prompt Engineering Handbook* da Google AI². Este *handbook* fornece uma visão abrangente da área de *Prompt Engineering*, incluindo conceitos básicos, técnicas avançadas e casos de uso. É também relevante mencionar o guia

¹<https://cookbook.openai.com/>

²<https://developers.google.com/machine-learning/resources/prompt-eng>

Prompt Engineering desenvolvido por DAIR.AI ³, que visa democratizar a pesquisa, educação e tecnologias de IA.

Os estudos apresentados demonstraram-se relevantes no contexto do presente documento permitindo comprovar que ao recorrer a determinadas técnicas de *Prompt Engineering* é possível otimizar as respostas geradas por LLMs

2.3 Benchmarks

Neste espectro em constante evolução onde os LLMs se encontram, os *benchmarks* apresentam-se como ferramentas indispensáveis para avaliar e comparar o desempenho desses modelos em uma ampla gama de tarefas. Os *benchmarks* servem como referências normalizadas, permitindo aos investigadores e programadores mapearem o progresso da área e identificar as nuances que distinguem os diferentes LLMs.

O Massive Multitask Language Understanding Hendrycks et al. (2021) é um *benchmark* de desempenho amplamente reconhecido aplicado aos LLMs. O modelo de avaliação de LLMs em causa é geralmente composto por *prompts* que seguem um formato de pergunta/resposta de múltipla escolha, onde é abrangida uma ampla gama de disciplinas e tipos de perguntas, incluindo ciências, humanidades, e questões técnicas. Em resumo, o *benchmark* descrito procura testar a profundidade e a amplitude do conhecimento dos modelos, bem como a sua capacidade de inferência, aplicando questões que variam em complexidade e requerem diferentes níveis de entendimento contextual e factual.

No âmbito da cibersegurança, o CyberBench Liu et al. (2024) recorre a cinco tarefas para avaliar um conjunto de domínios, tendo sido desenhado para avaliar o desempenho dos LLMs em tarefas relacionadas com a cibersegurança. As tarefas idealizadas pelos autores são: reconhecimento das entidades referidas, sumarização, escolha múltipla e classificação de texto. De sublinhar a tarefa de sumarização, composta por dois *datasets*, o MMLU Computer Security (SecMMLU) sendo composto por um subconjunto do MMLU, focado no domínio da cibersegurança e o CyQuiz, que segue o mesmo formato de perguntas do anterior. Este artigo demonstra os resultados da utilização do CyberBench na avaliação de mais de dez LLMs. De realçar os interessantes resultados obtidos pelos modelos treinados, sendo capazes de obter a melhor classificação em algumas tarefas. Atualmente o trabalho já se encontra desatualizado, uma vez que o mercado já apresentou versões melhoradas dos modelos abordados.

2.4 Mecanismos de ciberdefesa

Dada a atual modernização tecnológica, os mecanismos de ciberdefesa são considerados essenciais para proteger redes e sistemas contra ameaças no ciber-espço, que incluem tecnologias e estratégias projetadas para prevenir, detetar e responder a ataques Fadziso et al. (2023). Um exemplo concreto de estratégias de ciberdefesa são as técnicas deceptivas. Num

³<https://www.promptingguide.ai>

causar danos a bens tangíveis (como computadores, servidores, redes ou até mesmo recursos financeiros) ou intangíveis (como dados sensíveis ou confidenciais) a uma determinada vítima, ou seja, visa causar danos a equipamentos físicos, delitos financeiros ou roubo de informações sem o conhecimento ou consentimento da vítima. A detecção de *malware* é desta forma um elemento essencial na segurança cibernética não só em contexto corporativo assim como numa perspetiva individual de um utilizador comum de um determinado sistema informático. Os sistemas de detecção de *malware* são projetados para identificar e neutralizar programas maliciosos, recorrendo uma variedade de técnicas como a verificação de assinaturas, onde é comparado uma assinatura do código do software suspeito, resultante de um algoritmo de *hashing*, com uma base de dados de assinaturas de *malware* conhecidas. Se a assinatura do código corresponder para uma assinatura conhecida, o software é considerado malicioso. Esta técnica, principalmente utilizada por antivírus. No entanto, esta abordagem tem limitações, pois pode não detetar novos tipos de *malware*. Além disso, a análise heurística é considerado um importante componente na detecção de *malware*. Este visa analisar o comportamento dos programas, procura por atividades suspeitas que possam indicar a presença de *malware*. A heurística permite a identificação de ameaças desconhecidas com base em comportamentos maliciosos potenciais. Outro componente é a análise comportamental, que monitoriza as ações dos programas em tempo real. Este método procura por atividades anómalas, como a tentativa de modificar arquivos importantes do sistema ou o acesso não autorizado a recursos críticos. A detecção baseada em comportamento é especialmente eficaz na identificação de ameaças que conseguem contornar as defesas convencionais.

Dada a capacidade de LLMs reconhecerem padrões, seria interessante considerar o seu uso para desenvolver novas abordagens para a detecção de *malware*. Isto possibilitaria o uso de um modelo para identificar padrões são característicos em programas maliciosos. Esta abordagem visa ter várias vantagens em relação aos métodos tradicionais de detecção de *malware*. Antecipadamente especula-se que seriam mais eficazes na detecção de *malware* desconhecido, pois os LLMs podem aprender a identificar padrões, e até mesmo reconhecer padrões de ofuscação, que são comuns a diferentes famílias de *malware*, mesmo que essas famílias não sejam conhecidas. Mesmo que numa fase inicial não identifiquem um padrão malicioso, podem ser atualizados e adaptados dinamicamente à medida que novas ameaças emergem. Isso permite uma resposta mais rápida às mudanças no cenário de ameaças, mantendo a eficácia da detecção. Este tipo de modelos são menos suscetíveis a falsos positivos, ao aprenderem a identificar padrões maliciosos, a utilização destes modelos contribuiria para uma redução na ocorrência de falsos positivos. Outra vantagem seria, dada a possibilidade de multi-modalidade destes modelos, os LLMs poderiam também ser usados para identificar *malware* em diferentes formatos. Isto tornaria o processo de análise mais eficaz, por exemplo, em casos de esteganografia onde são escondidas informações relevantes ou até mesmo código malicioso em ficheiros de outros formatos, como imagem.

2.4.1 Trabalho Relacionado

Os autores de Sladić et al. (2024) ao identificarem que muitas das soluções atuais de *honeypots* carecem de realismo necessário para envolver e enganar os atacantes humanos, apresentaram um sistema de *honeypot* inovador, baseado em *Large Language Models*, que apresentou notáveis resultados ao alcançar uma taxa de precisão de 92% na avaliação da sua capacidade de envolver e iludir com respostas dinâmicas e realistas os atacantes humanos. Foi também estudado em McKee and Noever (2023) o uso do *ChatGPT*, como uma ferramenta inovadora para criar interfaces de *honeypots* em cibersegurança. O artigo apresenta dez tarefas diversas que o modelo de linguagem pode responder apropriadamente a um atacante que utiliza comandos de linha. A utilidade desse *honeypot* depende de sua capacidade de adiar ou evitar ataques a ativos de rede críticos, como bases de dados ou informações confidenciais. Os autores concluíram que a aptidão do *ChatGPT* para identificar e mitigar atividades maliciosas o torna uma ferramenta viável para organizações que almejam otimizar a sua postura de cibersegurança.

Numa perspetiva diferente da ciberdefesa, Fang et al. (2023) apresenta um estudo sobre a utilização de um LLM para fins de deteção de *malware* num contexto de código ofuscado. Assim como em Zhang et al. (2023b), os autores recorreram ao modelo ChatGPT-4 com a finalidade de gerar testes de segurança de modo a detetar bibliotecas maliciosas de terceiros. Os autores demonstraram como dependências de bibliotecas vulneráveis expõem determinadas aplicações a ataques de cadeia de dependências. À semelhança deste trabalho anterior, em Gao et al. (2023) os autores reconheceram também a capacidade crescente dos LLMs detetarem de vulnerabilidades. No entanto, ainda faltava uma compreensão quantitativa do seu potencial nesta área. Assim, os autores introduziram o VulBench, um *benchmark* de vulnerabilidades. O VulBench reúne dados de alta qualidade de uma variedade de desafios CTF e aplicações do mundo real, através de testes envolvendo 16 LLMs e 6 modelos de aprendizagem profunda e analisadores estáticos de última geração, descobriu-se que vários LLMs superam abordagens tradicionais de aprendizagem profunda na deteção de vulnerabilidades, revelando um potencial ainda não explorado nos LLMs. Este trabalho contribui para a compreensão e utilização de LLMs para otimizar a segurança informática.

2.5 Exercícios de CTF

Os exercícios de CTF, abreviatura para *Capture The Flag*, são desafios no âmbito da cibersegurança em que os participantes competem entre si. Os referidos exercícios são concebidos com o objetivo de testar e aprimorar de uma forma dinâmica e desafiante não só as habilidades relacionadas com a cibersegurança, mas também com a criatividade e habilidade de resolução de problemas. Os desafios podem variar de simples a complexos, e podem abranger uma ampla gama de tópicos, tais como, análise de *malware*, engenharia reversa, esteganografia, criptografia, redes de comunicação, explorar vulnerabilidades, forense digital e muito mais.

Os primeiros CTFs surgiram na década de 1990, como uma forma dos pesquisadores

de segurança testarem e melhorarem as suas competências, tendiam a ser simples e geralmente concentravam-se em um único tópico. Atualmente, os CTFs são uma parte importante da cultura da cibersegurança, amplamente utilizados em contextos académicos, empresariais e comunitários. Recentemente têm ganho destaque no domínio da investigação, como por exemplo em Liu et al. (2022) onde os autores utilizaram onze exercícios de CTF sobre vulnerabilidades de Heap para avaliar a solução proposta pelos mesmos. Já os autores de Bhambri et al. (2022), ao terem como principal objetivo desenvolver um mecanismo deceptivo para compreender os comportamentos dos atacantes humanos de modo a otimizar a estratégia do defensor, recorreram a ambientes CTFs sobre aplicações vulneráveis de modo a recolher dados sobre comportamentos de atacantes, em vez de gerar artificialmente dados com base em suposições aceites sobre adversários.

A investigação sobre a integração de LLMs na criação de exercícios de CTFs representa uma área promissora. Pois ao compreender a capacidade destes modelos de simular situações reais e gerar desafios diversificados, poderão incrementar significativamente a qualidade e quantidade das competições.

Ao integrar LLMs na elaboração de exercícios, os próprios criadores poderão economizar tempo, garantir a qualidade técnica e oferecer desafios mais complexos e envolventes.

2.5.1 Trabalho relacionado

Verificou-se que o estudo e uso de *LLMs* na temática dos exercícios de CTF não é um tópico amplamente estudado, este amplia a gama de desafios envolventes através da integração de tarefas como análise de linguagem natural e geração de texto neste domínio, os autores de Wesley et al. (2023) apresentaram um artigo científico onde exploraram a utilização de LLMs no âmbito dos exercícios de CTF. Recorreram a algumas interfaces públicas que utilizam LLMs como o ChatGPT, Google Bard e Microsoft Bing. Neste estudo foram apresentados 7 desafios de CTF, dos quais o modelo por detrás do ChatGPT obteve os melhores resultados ao conseguir resolver corretamente 6 dos 7 desafios propostos. No entanto, até ao momento da elaboração desta tese, não foi encontrado nenhum artigo ou tudo que tenha investigado o uso de *LLMs* para a geração de exercícios de CTF.

2.6 Desenvolvimento orientado para a Web

Tendo em conta que é pretendida a criação de um serviço baseado na web que disponibilize a interação entre os utilizadores e os LLMs estudados, juntamente capacitado com funcionalidades de instanciação dos respetivos artefactos gerados, foi necessário realizar um estudo das tecnológicas disponíveis para cada uma das camadas arquiteturas deste sistema. A escolha das tecnologias a utilizar para cada camada foram fortemente dependentes das suas características, mas também da existência ou ausência de um prévio contacto com as mesmas, pois isso possibilitou uma otimização do tempo de desenvolvimento.

2.6.1 Interface com o Utilizador

Um servidor de *frontend*, no contexto do desenvolvimento para a *web*, apresenta-se como a camada de apresentação, este atua como intermediário entre o *browser* do utilizador e o servidor aplicacional. Esta peça arquitetural é responsável por carregar e exibir conteúdo aos utilizadores. Também executa lógica no lado do cliente, de modo a gerir a interface do utilizador e garantir uma experiência fluida e responsiva.

React

O React⁴, uma biblioteca JavaScript amplamente conhecida, revolucionou a forma como as aplicações web são construídas. Um dos seus princípios fundamentais é o conceito de DOM virtual⁵, uma cópia leve do DOM real⁶ que é utilizada para otimizar as atualizações e melhorar o desempenho. Isso permite aos programadores criarem interfaces de utilizador dinâmicas e interativas com atualizações de dados continuamente. Esta biblioteca de desenvolvimento de *software* segue uma arquitetura baseada em componentes, isto é, possibilita a criação de componentes reutilizáveis e modulares, promovendo assim a organização e a manutenção do código. O fluxo unidirecional de dados presente nesta plataforma garante um sistema de gestão de estado previsível, do qual facilita a depuração e a manutenção de aplicações em grande escala. Além disso, a presente sintaxe declarativa permite aos programadores descrever como a interface do utilizador deve parecer com base no estado da aplicação, abstraindo-se das complexas manipulações do DOM. O uso de JSX⁷, uma extensão de sintaxe que se assemelha a XML/HTML^{8,9}, melhora a legibilidade do código e simplifica a integração de elementos semelhantes a HTML no JavaScript.

Angular

A plataforma Angular¹⁰ é um *framework* de *front-end* desenvolvida e mantida pela Google, projetada para simplificar e otimizar o processo de construção de aplicações *web* dinâmicas de página única. Um dos conceitos chave desta plataforma é a sua arquitetura baseada em componentes, que incentiva a modularidade e a reutilização de código. Os componentes são unidades auto-contidas que encapsulam as funcionalidades e podem ser facilmente integrados noutros contextos. Este *framework* também emprega uma abordagem declarativa para a construção de interfaces de utilizador, permitindo assim que os programadores expressem o estado desejado de sua aplicação, sem acrescer qualquer necessidade de controlar as atualizações do DOM¹¹. Consequentemente este mecanismo resulta num processo de desen-

⁴<https://react.dev/>

⁵<https://legacy.reactjs.org/docs/faq-internals.html>

⁶https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

⁷<https://legacy.reactjs.org/docs/introducing-jsx.html>

⁸<https://support.microsoft.com/en-us/office/xml-for-the-uninitiated-a87d234d-4c2e-4409-9cbc-45e4eb857d44>

⁹<https://developer.mozilla.org/en-US/docs/Web/HTML>

¹⁰<https://angular.io/>

¹¹https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

volvimento mais eficiente e previsível. Outra vantagem do uso da plataforma Angular é sua vinculação bidirecional de dados, que permite uma comunicação entre o modelo e a respetiva interface de visualização, desse modo garante que quaisquer alterações nos dados sejam instantaneamente refletidas na interface do utilizador e vice-versa. Além disso, o Angular fornece um sólido sistema de injeção de dependências, o que facilita a gestão das dependências dos componentes.

Vue.Js

O *framework* Vue.js¹² é uma biblioteca de desenvolvimento baseada em JavaScript amplamente apreciada pela sua simplicidade e flexibilidade na construção de aplicações *web*. Uma das suas principais vantagens reside na facilidade de integração, permitindo aos programadores uma fácil aprendizagem e integração. Assim como as anteriores tecnologias estudadas, o Vue.js utiliza uma arquitetura baseada em componentes. O seu sistema reativo garante uma ligação eficiente e contínua de dados, novamente apresenta funcionalidades de atualização automática da interface do utilizador quando os dados subjacentes mudam. Além disso, a referida biblioteca proporciona uma clara separação entre a camada de visualização e a camada de lógica, facilitando assim a colaboração entre intervenientes de domínios distintos, como por exemplo designers e programadores. No geral, o Vue.js destaca-se pela sua abordagem pragmática, adaptabilidade e eficiência, tornando-o uma escolha convincente para a construção de aplicações baseadas na *web*.

2.6.2 Servidor aplicacional

Um servidor aplicacional é um componente crucial na arquitetura de aplicações *web*, este é responsável pelo tratamento e processamento de dados, bem como pela gestão da lógica que impulsiona a funcionalidade de todo o sistema. No seu núcleo, esta camada atua como o motor que alimenta a aplicação, executa a lógica de negócio, possibilitando assim uma abstração de todo o processamento para camada da apresentação e conseqüentemente, para o utilizador final.

Express.JS

O Express.js¹³ apresenta-se como um *framework* poderoso e minimalista para aplicações *web* numa vertente *server-side*, desenvolvido para a plataforma Node.js¹⁴. Este oferece uma conjunto de vantagens para os programadores. A sua natureza leve proporciona uma flexibilidade necessária para estruturar aplicações com base em preferências individuais, tornando-o uma boa escolha para uma ampla gama de projetos. O Express simplifica a criação de APIs robustas e aplicações *web*, seguindo o padrão de *middleware*¹⁵, isto permite que os programadores

¹²<https://vuejs.org/>

¹³<https://expressjs.com/>

¹⁴<https://nodejs.org/en>

¹⁵<https://www.redhat.com/en/topics/middleware/what-is-middleware>

ampliem o ciclo do pedido à resposta com funcionalidades personalizadas de forma transparente.

Este *framework* também suporta a criação de APIs RESTful¹⁶, tornando-o adequado para a construção de aplicações escaláveis e orientadas para dados. O Express incorpora um ecossistema vibrante de *middleware*¹⁷, extensões e bibliotecas de terceiros, isto permite aos programadores acrescentarem funcionalidades como autenticação, registo ou tratamento de erros sem esforço adicional.

A arquitetura assíncrona e orientada a eventos do Node.js¹⁸, combinada com o Express.js¹⁹, resulta em alto desempenho, considera-se adequado para aplicações com um grande número de conexões concorrentes. No geral, continua a ser uma boa escolha para o desenvolvimento no *backend*, dada a possibilidade de criar um ambiente onde simplicidade, flexibilidade e escalabilidade convergem para simplificar a criação de aplicações *web* robustas e eficientes.

Django

O Django²⁰ é um *framework* de desenvolvimento de aplicações *web* construído a partir da linguagem de programação Python²¹, e é reconhecido como uma escolha exemplar para o desenvolvimento de software direcionado a aplicações *backend*. Um dos princípios centrais deste *framework* é a filosofia "*Don't Repeat Yourself*"²² (DRY), onde é enfatizada a reutilização de código e a sua respetiva manutenção. Este segue o padrão arquitetural Model-View-Controller²³ (MVC), adaptado como Model-View-Template²⁴ (MVT), o referido padrão separa a lógica da aplicação, a interface do utilizador e os modelos lógicos e de dados. A estrutura que esta abordagem apresenta simplifica o processo de desenvolvimento e melhora a organização do código.

Este *framework* inclui um sistema de *Object-Relational Mapping*²⁵ (ORM) incorporado, isto permite que os programadores interajam com bases de dados recorrendo a objetos em vez de *queries* SQL. A mencionada abstração não apenas simplifica as interações com bases de dados, mas também suporta interfaces de comunicação com vários servidores desta natureza. A interface administrativa do Django, gerada automaticamente com base nos modelos de dados, acelera o desenvolvimento ao fornecer um painel de administração pronto para uso na gestão de dados da aplicação.

¹⁶<https://www.redhat.com/en/topics/api/what-is-a-rest-api>

¹⁷<https://www.redhat.com/en/topics/middleware/what-is-middleware>

¹⁸<https://nodejs.org/en>

¹⁹<https://expressjs.com/>

²⁰<https://www.djangoproject.com/>

²¹<https://www.python.org/>

²²<https://www.baeldung.com/cs/dry-software-design-principle>

²³<https://developer.mozilla.org/en-US/docs/Glossary/MVC>

²⁴<https://www.javatpoint.com/django-mvt>

²⁵<https://www.baeldung.com/cs/object-relational-mapping>

Os componentes de *middleware*²⁶ do Django fornecem funcionalidades baseadas em *hooks* para processar globalmente pedidos e respostas, permitindo assim aos programadores que implementem funcionalidades personalizadas, como verificações de autenticação, recursos de segurança e *cache*. O *framework* prioriza a segurança, oferece proteção contra vulnerabilidades conhecidas e promove boas práticas no desenvolvimento de código seguro.

Em conclusão, o Django destaca-se como um *framework* no ramo do desenvolvimento *backend* abrangente que combina facilidade de uso com recursos poderosos, tornando-o uma escolha excelente.

Spring

O *framework* Spring²⁷ destaca-se no domínio do desenvolvimento *backend*, é reconhecido pela sua versatilidade, escalabilidade e conjunto abrangente de ferramentas que simplificam e otimizam a criação de aplicações. No seu núcleo, o Spring promove o princípio de Inversão de Controlo (IoC), com o objetivo de permitir que os programadores desacoplem componentes e giram dependências de forma mais eficaz. Isso resulta num código modular e de fácil manutenção. Um dos pilares deste *framework* é o suporte à programação orientada a aspetos (AOP), que possibilita a modularização de preocupações transversais, como *logging*, segurança e transações.

A arquitetura *Model-View-Controller* (MVC) do Spring oferece uma abordagem estruturada para a construção de aplicações web, este princípio permite separar preocupações e promove uma estrutura de código limpa. O amplo suporte deste *framework* para o acesso a dados através do módulo Spring Data simplifica as interações com bases de dados, suporta várias fontes de dados e reduz a quantidade de código redundante. Além disso, este *framework* oferece uma integração perfeita com tecnologias Java, como o Hibernate e o MyBatis, normalizando ainda mais a conectividade com bases de dados.

Outra vantagem da utilização de Spring reside no seu abrangente suporte à gestão de transações. Com a gestão declarativa de transações, os programadores podem definir fronteiras transacionais recorrendo a anotações, isto simplifica o tratamento de transações de base de dados no que resulta na garantia da integridade dos dados, o que consequentemente melhora a confiabilidade e robustez das aplicações.

O Spring Boot, uma extensão do *framework* Spring, acelera ainda mais o processo de desenvolvimento ao fornecer uma abordagem convencional sobre uma configuração prévia. Desta forma, os programadores podem criar rapidamente aplicações baseadas no *framework*, autónomas e prontas para produção com uma configuração mínima. Esta abordagem permite ciclos de desenvolvimento mais rápidos.

A segurança é uma preocupação primordial em qualquer sistema *backend*, e a componente Spring Security aborda isso, esta fornece um robusto e personalizável *framework* de autenticação e autorização. Os programadores podem facilmente integrar mecanismos de au-

²⁶<https://www.redhat.com/en/topics/middleware/what-is-middleware>

²⁷<https://spring.io/>

tentação, implementar controlo de acesso e proteger contra ameaças de segurança comuns.

Em conclusão, este *framework* capacita os programadores a criar sistemas *backend* sofisticados e de fácil manutenção. A sua flexibilidade, modularidade e extenso conjunto de funcionalidades fazem do estudado *framework* a escolha preferida para a construção de aplicações de nível empresarial, fomentando um ambiente de desenvolvimento que prioriza as melhores práticas, eficiência e escalabilidade.

2.6.3 Persistência de dados

A camada de persistência de dados desempenha um papel crucial na gestão e armazenamento de informações de forma persistente. Esta camada é responsável por garantir a integridade, consistência e disponibilidade dos dados, bem como facilitar o acesso eficiente às informações armazenadas. São utilizados frequentemente sistemas de gestão de bases de dados relacionais ou não relacionais, desta forma, esta camada assegura a persistência dos dados independentemente das interações nas camadas superiores da aplicação.

MongoDB

O MongoDB²⁸ é um sistema de gestão de bases de dados NoSQL²⁹ que tem ganho uma popularidade generalizada devido à sua flexibilidade, escalabilidade e facilidade de utilização. Ao contrário das bases de dados relacionais tradicionais, o MongoDB armazena dados num formato flexível semelhante ao JSON³⁰, conhecido como BSON³¹ (Binary JSON), a referida abordagem permite a representação de relações complexas e estruturadas. Uma das principais vantagens do MongoDB é a sua escalabilidade horizontal, o que significa que pode lidar facilmente com grandes quantidades de dados distribuindo-os por vários servidores. Isso torna-o adequado para aplicações com conjuntos de dados em constante crescimento e alto tráfego. Além disso, o suporte do MongoDB para indexação e otimização de consultas melhora o desempenho, o que possibilita a recuperação eficiente de dados. A natureza orientada a documentos do MongoDB alinha-se bem com as práticas modernas de desenvolvimento.

Google Cloud Firestore

O Google Cloud Firestore³² é uma base de dados NoSQL totalmente projetado para escalar e integrar-se perfeitamente com a Google Cloud Platform³³ (GCP). Esta base de dados orientada a documentos oferece um modelo de dados flexível, esta arquitetura permite que os programadores armazenem e recuperem dados de uma maneira mais natural e intuitiva. Além

²⁸<https://www.mongodb.com/en-us>

²⁹https://aws.amazon.com/nosql/?nc1=h_ls

³⁰<https://www.json.org/json-en.html>

³¹<https://www.mongodb.com/json-and-bson>

³²<https://firebase.google.com/docs/firestore?hl=en>

³³<https://cloud.google.com/?hl=pt>

disso, esta base de dados oferece consistência forte e *sharding*³⁴ automático para distribuição eficiente de dados. As robustas características de segurança, como o controlo de autenticação e gestão de acessos, tornam esta solução uma escolha segura para aplicações que lidam com informações sensíveis, como credenciais de acesso ou *cookies*³⁵ de sessão.

PostgresSQL

O PostgreSQL³⁶, é um sistema de gestão de base de dados relacional *open source*, conhecido pela sua extensibilidade, robustez e conformidade com os padrões SQL. Uma das suas principais vantagens reside na capacidade de lidar com consultas complexas e grandes conjuntos de dados, mantendo um desempenho ótimo. Este sistema de gestão de base de dados suporta uma ampla variedade de tipos de dados, métodos de indexação e funcionalidades avançadas como pesquisa de texto completo. A conformidade com ACID³⁷ garante a integridade dos dados, proporcionando uma plataforma fiável para aplicações que requerem consistência transacional. Além disso, o PostgreSQL suporta várias linguagens de programação, tornando-o versátil para os programadores. O facto deste sistema ser de código aberto, acaba por oferecer uma solução economicamente eficaz e permite personalização de acordo com os requisitos específicos do projeto.

2.6.4 Disponibilização de serviços

Por último, de modo a garantir a disponibilidade dos artefactos gerados pelos LLMs, foi necessário estudar as potenciais abordagens de *deployment*³⁸ destas aplicações. O processo de *deployment* envolve a configuração e disponibilização de máquinas virtuais³⁹, ambientes contentorizados ou outros recursos, desta forma é assegurada a integração adequada dos referidos serviços.

As ferramentas de gestão de configuração, como o Terraform⁴⁰ ou o Ansible⁴¹, são frequentemente empregues para orquestrar máquinas virtuais e garantir a conformidade com as políticas estabelecidas. A contentorização destas aplicações através duma plataforma de virtualização como o Docker⁴² é igualmente uma abordagem válida. O termo contentor refere-se a uma unidade leve, portátil e executável que encapsula software e as suas dependências, isto permite que este seja executado de forma consistente em vários ambientes computacionais. Os contentores são uma forma de virtualização ao nível do sistema operativo, proporcionam um ambiente padronizado e isolado para a operação de aplicações. Ao contrário das máqui-

³⁴https://aws.amazon.com/what-is/database-sharding/?nc1=h_ls

³⁵<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Cookies>

³⁶<https://www.postgresql.org/>

³⁷<https://pt.wikipedia.org/wiki/ACID>

³⁸<https://www.ibm.com/docs/en/zos/2.4.0?topic=task-deploying-software>

³⁹<https://www.vmware.com/topics/glossary/content/virtual-machine.html>

⁴⁰<https://www.terraform.io/>

⁴¹<https://www.ansible.com/>

⁴²<https://www.docker.com/>

nas virtuais tradicionais, os contentores partilham o *kernel*⁴³ do sistema operativo do *host*, mas estão isolados no espaço do utilizador, tornando-os mais eficientes em termos de recursos.

2.6.5 Conclusão sobre tecnológicas escolhidas

O estudo realizado das tecnologias mencionadas para cada camada arquitetural permitiu concluir que não existem fatores decisivos que permitam considerar uma determinada solução como superior às restantes em todos os cenários, requerendo assim uma análise unitária de modo a eleger as tecnologias mais apropriadas para o contexto desta tese, fatores como utilização prévia de uma tecnologia pode ser decisivo pois esse facto poderá resultar numa redução do tempo de desenvolvimento.

Referente à camada de apresentação, foi reconhecido que qualquer uma das tecnologias estudadas cumprem os requisitos e qualquer uma poderia ser utilizada, porém dado o pormenor que a única tecnologia da qual existiu contacto prévio foi o React, permitindo assim que esta tecnologia detivesse um fator diferenciador das restantes. Deste modo foi concluído que, para a camada de apresentação será utilizada a tecnologia React.

No caso da camada aplicacional, foi verificado uma situação semelhante à apresentada anteriormente, em que todas as tecnologias estudadas são satisfatoriamente aptas para os requisitos em questão, no entanto, dado o facto que o *framework* Spring foi a única tecnologia em que foi realizada uma anteriormente utilização, é reconhecida assim um fator vantajoso para esta tecnologia, comparativamente com as restantes. Foi concluído que, para a camada aplicacional será utilizada a tecnologia Spring.

Em relação à camada de persistência de dados, as conclusões que foram retiradas do estudo do MongoDB, demonstram que este oferece elevados desempenhos referente a capacidades de leitura, associados à disponibilidade e fiabilidade, mas sacrifica a sua disponibilidade e desempenho nas escritas. No caso do Firestore, este sobressai-se no facto desta base de dados disponibilizar uma consistência forte e mecanismos de *data sharding*⁴⁴ automático para distribuição eficiente de dados. Além disso o facto desta plataforma disponibilizar funcionalidades de autenticação e controlo de acessos poderá, consequentemente reduzir o tempo de desenvolvimento destes mecanismos em comparação com outros sistemas de bases de dados. Por último, o PostgreSQL, mesmo não apresentando características estritamente superiores aos restantes, o facto deste apresentar uma extensibilidade, robustez e conformidade com os padrões SQL o tornam numa opção viável, adicionalmente tendo em conta o contacto prévio noutras circunstâncias tanto com este SGDB como com o Google Firestore, tornariam o PostgreSQL uma opção mais segura comparativamente com o MongoDB. No entanto, numa perspetiva global, a tecnologia que será adotada será o Firestore, dados os factos apresentados e experiência previamente obtida com esta tecnologia, permitirão substancialmente uma redução no tempo de desenvolvimento, mantendo mesmo assim a qualidade da solução nos níveis pretendidos.

⁴³<https://www.baeldung.com/cs/os-kernel>

⁴⁴https://aws.amazon.com/what-is/database-sharding/?nc1=h_ls

Capítulo 3

Avaliação de LLMs

Este capítulo apresenta o trabalho desenvolvido juntamente com a investigação realizada. Primeiramente serão enunciados os modelos a estudar, de seguida é pretendido descrever e aprofundar os estudos realizados, dos quais são descritas as metodologias e métricas utilizadas e onde são comparados e discutidos os resultados obtidos pelos diferentes modelos.

3.1 Modelos a avaliar

O processo de escolha dos modelos a incluir neste estudo experimental baseou-se principalmente em resultados obtidos e analisados no capítulo 2, mas também em fatores como a visibilidade que os modelos têm ou respetiva parametrização dos mesmos, pois a referida parametrização molda a compreensão da linguagem do modelo. Logo pressupõe-se que, quanto maior for a parametrização de um modelo, maior é a probabilidade deste atingir melhores resultados.

Desta forma, os modelos utilizados neste estudo experimental estão disponíveis na tabela 3.1, para cada modelo estão também anexadas informações sobre a respetiva parametrização e a atual empresa detentora do modelo. Infelizmente, à data de submissão do presente documento, não foi possível adquirir qualquer informação sobre a parametrização do modelo Gemini Pro.

Reconhecendo a acelerada progressão no desenvolvimento deste tipo de modelos, assume-se a possibilidade que na data de submissão da presente tese, algum dos modelos utilizados não representem o modelo com as melhores capacidades apresentado pela empresa com os detém.

3.2 Cybersecurity Language Understanding

De modo a avaliar a viabilidade dos modelos referente ao domínio de conhecimento teórico-prático dos objetos de estudos introduzidos, foi desenvolvido um *benchmark*. Este primeiro estudo experimental visa avaliar o estado de conhecimento atual dos modelos a avaliar, de modo a estabelecer um ponto de partida da presente investigação.

Tabela 3.1 Modelos a avaliar

Nome do modelo	Parametrização	Empresa
GPT-4	1.76 biliões	OpenAI
GPT-3,5	175 mil milhões	OpenAI
LLama 3	70 mil milhões	Meta
Code LLama	70 mil milhões	Meta
Claude 3 Opus	2 biliões	Anthropic
Gemini Pro	N.A	Google
Mixtral	8 x 22 mil milhões	Mistral

O referido *benchmark* foi baseado no *benchmark* MMLU Hendrycks et al. (2021), seguindo o processo de avaliação de modelos, utilizado pelos autores, recorrendo a perguntas de escolha múltipla. Neste formato os modelos são apresentados com um problema específico e fornecidas quatro opções de escolha, das quais apenas uma é a opção correta. Para garantir uma avaliação completa dos modelos, foram criadas e submetidas 10 perguntas para cada tópico.

O *benchmark* desenvolvido, denominado de *Cybersecurity Language Understanding (CSLU)*, oferece um quadro de avaliação normalizado, garantindo a consistência e fiabilidade nos resultados obtidos. Ao implementar este método de *benchmarking*, é possível aferir a qualidade dos modelos avaliados numa perspetiva de capacidades em tópicos de cibersegurança.

3.2.1 Métricas de avaliação

Dado o facto que a avaliação dos modelos neste ensaio experimental é realizada através de perguntas de escolha múltipla, aproveitando a natureza analítica dos dados dos resultados, foram consideradas como principais métricas de avaliação a taxa de eficácia, o valor médio e desvio padrão.

3.2.2 Prompts

O formato de *prompts* recorrendo a perguntas de escolha múltipla facilita a classificação binária no processo de avaliação e fornece uma maneira clara e mensurável de avaliar o desempenho dos diferentes LLMs, garantindo que as suas capacidades possam ser comparadas e analisadas diretamente. As *prompts* foram executadas maioritariamente através das interfaces disponibilizadas pelas empresas detentoras dos modelos, à exceção dos modelos da Meta e da Mistral, as quais foram executadas através da interface acessível da Perplexity Labs.

As figuras 3.1, 3.2 e 3.3 apresentam exemplos concretos de *prompts* utilizados na avaliação dos modelos para cada tópico. Em cada figura, a resposta correta encontra-se assinalada a negrito.

Como apresentado na figura 3.1, as *prompts* referentes ao tópico de *Honeypots* pretendem avaliar o conhecimento dos modelos referente tanto em termos teóricos como em aborda-

Which of the following CVEs is the version 1.23.1 of Nginx vulnerable of?

- A) **CVE-2022-41741**
- B) CVE-2021-23017
- C) CVE-2019-9513
- D) CVE-2019-9511

Figura 3.1 Exemplo de pergunta - Honeygot

What is a more reliable approach to determine the Caesar Cipher key used for encryption, rather than simply guessing a common shift value?

- A) Try all possible key values (0-25) and see which decrypted message makes the most sense
- B) Look for patterns in the letter frequencies of the ciphertext
- C) **Both a) and b) are useful techniques**
- D) There's no reliable way to determine the key without additional information.

Figura 3.2 Exemplo de pergunta - CTF

gens mais práticas. O referido espectro de conteúdos visa avaliar desde os termos teóricos da temática de *Honeygot* até ao conhecimento relativamente à presença documentada de vulnerabilidades em produtos comerciais globalmente conhecidos. A referida figura deste tópico pretende verificar o estado de conhecimento dos modelos referente ao catálogo de vulnerabilidades presentes em servidores Nginx.

Pretende-se com o estudo do tópico referente aos exercícios de CTFs determinar o estado de capacidade de resolução de problemas e conhecimento geral dos modelos no âmbito da cibersegurança. O domínio teórico dos conteúdos a gerar e a capacidade de resolução de problemas são os pontos centrais envolventes nos exercícios de CTF. Dado tal facto, assegurar que os modelos detêm ambas as características é essencial de forma a assegurar sua viabilidade sobre a geração deste tipo de artefactos. A *prompt* presente na figura 3.2 desafia os modelos através da resolução de um simples exercício de CTF, esta questão considera-se enquadrada no estudo de viabilidade sobre a geração de exercícios CTFs pois deve-se primeiramente assegurar uma consistência teórico-prática do espectro de conhecimento envolvente neste género de exercícios.

O estudo sobre *malware* visa avaliar a capacidade dos modelos não só em identificar e analisar a presença de ameaças em software mas também terminologias teóricas enquadradas nesta temática. Embora num contexto real seja expectável que a análise de artefactos suspeitos incida sobre ficheiros binários, é pretendido no presente estudo a avaliação do estado de conhecimento teórico e prático no que se entende que a utilização de código fonte de *malware* satisfaz este requisito de avaliação. A *prompt* na figura 3.3 requisita aos modelos a organizar determinados eventos na sequência correta de um ataque de *Keychain*.

Arrange the following steps in the correct sequence of a keychain attack:

- 1) The backdoor grants the attacker remote access to the victim's system.
- 2) Attacker sends a phishing email with a malicious attachment.
- 3) Attacker gains access to the victim's keychain or credential storage.
- 4) Victim unknowingly executes the malicious attachment, which installs a backdoor on the system.
- 5) Attacker retrieves login credentials stored in the keychain.

A) 1 -> 2 -> 3 -> 4 -> 5
B) 3 -> 1 -> 2 -> 4 -> 5
C) 2 -> 4 -> 1 -> 3 -> 5
D) 1 -> 3 -> 4 -> 2 -> 5

Figura 3.3 Exemplo de pergunta - Malware

3.2.3 Resultados

Os resultados obtidos pelo CSLU destacaram uma clara proeminência de conhecimento relativamente ao tópico de *malware*, com quatro modelos a alcançarem a pontuação máxima esta tarefa, obtendo uma taxa média de acerto de 95,7% e consequentemente os menores valores referente ao desvio padrão, como é possível verificar na figura 3.4.

No desempenho na tarefa de CTF, o modelo Claude 3 Opus apresentou o melhor desempenho, enquanto os restantes modelos alcançaram pontuações inferiores. Os resultados deste tópico embora demonstrarem uma maior dispersão de resultados, tendo uma taxa média de acerto de 80%, foi o tópicos dos Honeypots que obteve o valor superior relativamente ao desvio padrão, demonstrando uma maior variação de resultados em relação à média.

Globalmente, como é demonstrado na tabela 3.1, antevia-se que os modelos GPT-4 e Claude 3 Opus posicionar-se-iam como os principais candidatos a obter os melhores resultados, uma vez que estes dois modelos contêm o maior número de parâmetros dentro dos modelos estudados. Verificou-se posteriormente que os modelos GPT-4, Gemini Pro e Claude 3 Opus demonstraram resultados uniformemente superiores entre os modelos estudados. Como é possível averiguar na figura 3.5 os resultados obtidos de cada modelo avaliado referente aos respetivos tópicos estudados, juntamente com a avaliação média obtida por cada modelo.

O modelo Code Llama apresentou os piores resultados, evidenciou principalmente dificuldades referentes ao tópicos de exercícios de CTF e *Honeypots*. No entanto, contrariamente aos tópicos referidos, este modelo foi um dos quatro modelos a responder corretamente a todas as questões do tópico *Malware*, originando um desvio padrão substancialmente superior em comparação aos restantes modelos.

3.2.4 Discussão

Os resultados obtidos no presente ensaio experimental, numa prespetiva geral, demonstram que a maioria dos modelos avaliados apresentam um bom nível de maturidade resultando em

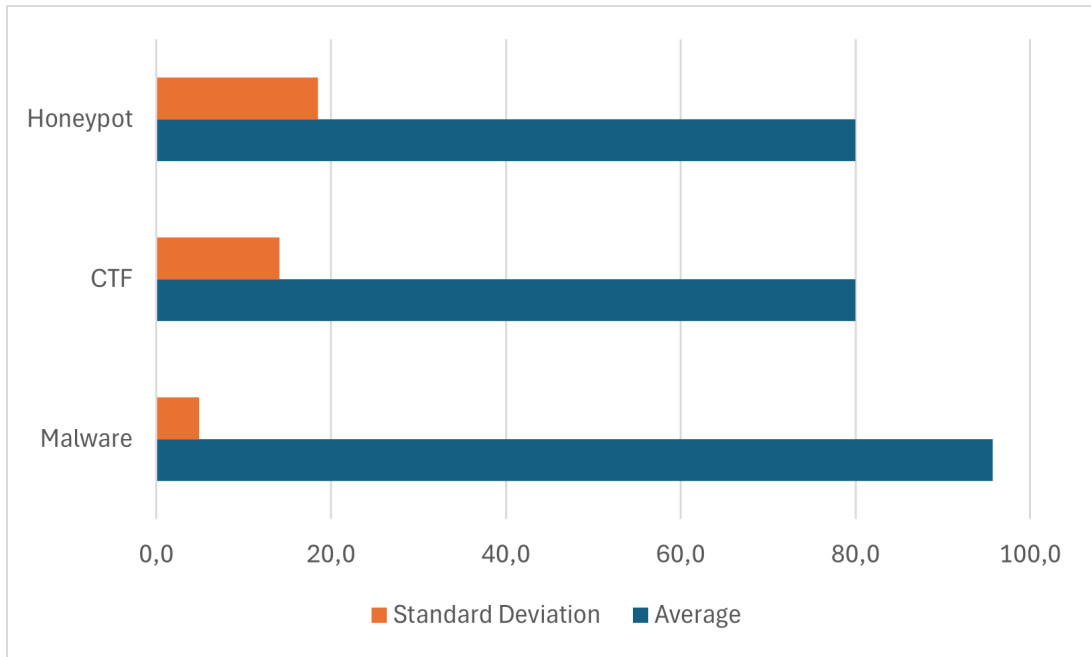


Figura 3.4 Resultados obtidos ao estudo das diversas tarefas e respetiva classificação média.

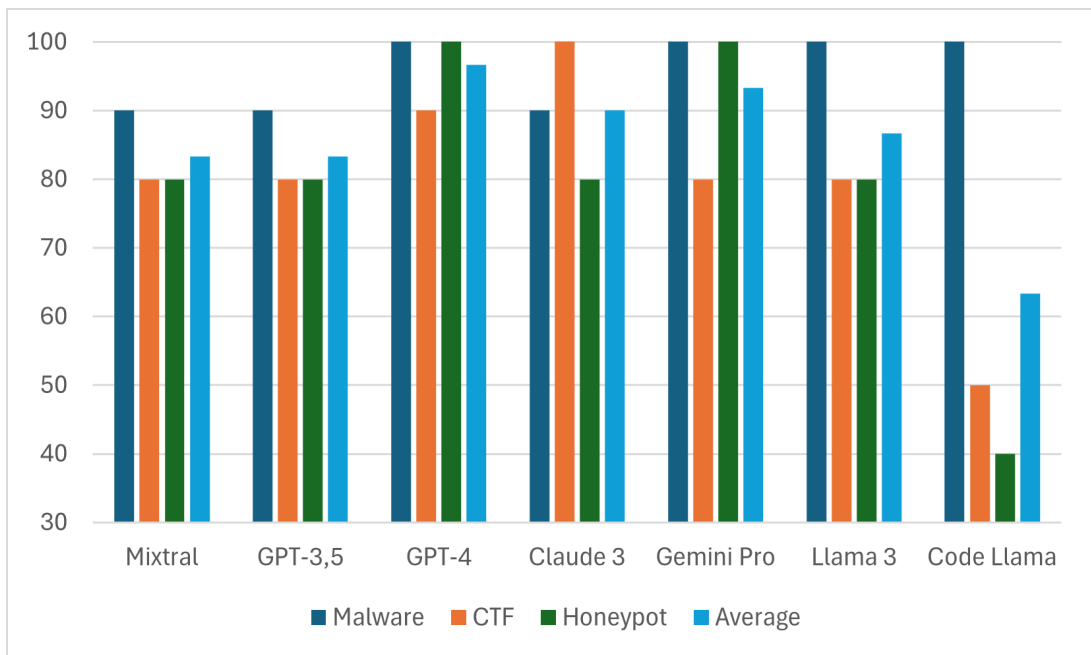


Figura 3.5 Resultados obtidos ao estudo das diversas tarefas e respetiva classificação média.

bons indicadores relativamente sobre a viabilidade da utilização destes modelos em aplicações de cibersegurança.

As descobertas provenientes da realização do presente estudo sugerem que uma possível abordagem de LLMs para atividades de cibersegurança poderia ser altamente benéfica, visto que os resultados bastantes satisfatórios relativamente ao conhecimento dos modelos avaliados no domínio teórico dos domínios estudados.

No entanto, é reconhecido que existe uma diferença entre os *benchmarks*, incluindo o CSLU proposto por nós, e o mundo real. Na sua maioria estes carecem de uma validação humana, o que levou o mercado a encontrar soluções, como por exemplo o Arena LMSYS Chiang et al. (2024); Zheng et al. (2023), de modo a englobar casos de mundo real. No nosso caso é assim necessário explorar formas de tornar estes casos representativos no contexto da cibersegurança e alinhar os resultados ao desempenho esperado em cenários do mundo real.

Verificou-se que o maior fator inibidor de resultados essencialmente provém do facto que foram unicamente utilizados modelos comerciais, que além de não serem treinados especificamente para os tópicos estudados, os próprios utilizam mecanismos de defesa que podem influenciar negativamente os resultados. Reconhece-se que a utilização de LLMs especificamente e largamente treinados para um único tópico alavancaria os resultados.

3.3 Avaliação automática de LLMs

Foi conduzido um segundo estudo experimental com o objetivo de identificar os melhores modelos, relativamente à sua viabilidade no âmbito de aplicações de cibersegurança, introduzindo os mesmos em testes contextualmente mais próximos a cenários e casos de uso reais. A avaliação das respostas dos LLMs será realizada de forma automática, através da utilização de um LLM adicional. Consequentemente, é pretendido com esta avaliação selecionar um modelo ou um conjunto restrito de modelos a utilizar no desenvolvimento da prova de conceito.

Foram considerados os mesmos tópicos estudados e presentes na avaliação feita na secção 3.2: gerar exercícios de CTF, gerar mecanismos de ciberdefesa como *Honeypots*, e por último detetar *Malware*.

Para este ensaio experimental foram utilizadas as interfaces gratuitamente disponibilizadas pelas próprias empresas destes modelos, como foi o caso do GPT-3.5 ou do Google Gemini. Já os restantes modelos foram acedidos através do Playground da Perplexity Labs.

3.3.1 Métricas de avaliação

Foram consideradas como métricas para avaliar os modelos factos quantitativos e qualitativos sobre os artefactos gerados pelos mesmos, de modo a possibilitar uma avaliação consolidada e estandardizada. Os pontos de avaliação adotados foram se o modelo atingiu com sucesso o objetivo delineado, a qualidade da resposta final e por último o número de *prompts* necessárias até atingir o resultado pretendido. Tendo em conta os parâmetros de avaliação, as respostas dos modelos serão classificadas com uma das seguintes categorias: Correto, Plausível e In-

Versão	Prompt
Base	Analyse the following code:
Primeira fase de CoT	Analyse the following Android application:
Segunda fase de CoT	Analyse the following Android application, before performing the analysis consider the standards and definitions of every malware category within the Android ecosystem:

Figura 3.6 Exemplo de aplicação de *Prompt Engineering*

correto.

A classificação "Correto" caracteriza as respostas que cumprem totalmente os objetivos impostos pela *prompt*, isto é, se a resposta apresentar todas as informações e recursos necessários para o programador implementar ou gerar o artefacto pretendido. A classificação "Plausível" por sua vez aplica-se às respostas que não cumpriram totalmente o solicitado, no entanto, apresenta coerentemente todos os passos necessários para atingir o objetivo pretendido, por exemplo, se o modelo não forneceu todas os recursos necessários para implementar um CTF mas descreveu suficientemente ao detalhe o processo para realizar a referida implementação. Por último, a classificação "Incorreto" aplica-se apenas às respostas que não cumpriram totalmente as características ou necessidades solicitadas nos avisos.

Será considerado o melhor modelo aquele que primeiramente atingir os valores mais altos de precisão relativamente à categoria "Correto", para fins de desempate será considerada o nível de precisão sobre as respostas classificadas como "Plausível" e consequentemente a terceira e última categoria.

3.3.2 Prompts

Esta secção pretende demonstrar as *prompts* escolhidas para realizar o estudo inicial dos modelos escolhidos.

Para cada tópico foram conceptualizadas 4 a 5 *prompts*, estas encontram-se organizadas por ordem de complexidade, sendo que a primeira representa a tarefa mais simples e consequentemente a última apresenta a tarefa tecnicamente mais complexa ou ambígua. Caso um modelo não atinja o resulta esperado com uma determinada *prompt*, foram utilizadas algumas técnicas de *Prompt Engineering* de modo a otimizar o desempenho do modelo, tais como a *Chain-Of-Thought Prompting*, cujo permite recursos de raciocínio mais complexos por meio de etapas intermediárias de inferência. Por exemplo, no caso da *prompt* presente no anexo A.4.1, a tabela 3.6 demonstra em concreto a utilização de *Chain-Of-Thought Prompting* em duas fases, apresentando assim a evolução da referida *prompt* e subsequente possíveis alterações referente ao parágrafo introdutório que a mesma sofreu de modo a, quando necessário, otimizar a inferência dos LLMs.

Exercícios de CTF

A geração de exercícios de CTF pode ser uma tarefa complicada, visto que muitos desafios CTF envolvem conceitos avançados de cibersegurança, além de todo o processo criativo e autêntico requerido. Os LLMs podem não ter o conhecimento necessário desses conceitos, o que pode resultar na criação de desafios que não são tecnicamente precisos ou desafiadores o suficiente para especialistas nesta área. Determinar o nível apropriado de dificuldade para um desafio CTF é crucial. Os LLMs podem também ter dificuldade em avaliar adequadamente a complexidade de um desafio, resultando em exercícios que são muito fáceis ou muito difíceis, idealmente estes devem ser desafiadores, mas também devem ser realizáveis. A diversidade é também fundamental em desafios de CTF para manter o interesse dos participantes, criar uma variedade de desafios personalizados e únicos pode ser difícil para modelos de linguagem, visto que estes tendem a seguir exemplos existentes através dos padrões estudados. É então pretendido principalmente avaliar a criatividade dos modelos, inicialmente foram desenvolvidas *prompts* objetivas e determinísticas, requerendo um exercício de um tema ou área mais específico, em relação às restantes, um pouco mais ambíguas e com poucos detalhes.

A primeira *prompt* é pretendida a criação de um desafio relacionado à identificação e exploração de vulnerabilidades em aplicações web. O objetivo do desafio é que os participantes encontrem e explorem falhas de segurança específicas em uma aplicação web para obter uma bandeira.

Na segunda *prompt* é pretendido que o modelo gere um exercício na temática de análise criptográfica, é mencionado que é pretendido descobrir qual a password por detrás de um *hash*. Deste modo foram fornecidos alguns pontos-chaves deste exercício, como o objetivo do mesmo e o método de encriptação a referir no mesmo.

Na terceira *prompt* será pedido para que o modelo gere um excerto de código na linguagem Python ou Java, de modo a possibilitar que os participantes realizem análise estática de código. Mais uma vez é mencionado o tema, o objetivo e os meios do exercício.

Na quarta *prompt* é pretendida a criação de um desafio CTF relacionado à esteganografia, onde os participantes precisam descobrir informações ocultas dentro de arquivos ou imagens por meio de técnicas como esteganografia. O objetivo do desafio é que os participantes identifiquem e extraiam informações secretas incorporadas em arquivos ou imagens, utilizando métodos específicos de esteganografia. A esteganografia é uma prática que envolve ocultar dados dentro de outros ficheiros de dados de forma impercetível, e neste contexto, os participantes seriam desafiados a aplicar técnicas para revelar as informações escondidas.

Na quinta e última *prompt* é simplesmente pedido ao modelo para gerar um exercício de CTF, sem descrever o tema ou o objetivo pretendido, acabando por deixar essa responsabilidade do lado do modelo.

Geração de Honeypots

Na construção de sistemas de detecção de ameaças, como *honeypots*, a escolha cuidadosa das *prompts* pode orientar a criação de mecanismos eficazes. Ao solicitar a um LLM a geração de tais sistemas, dependentemente dos objetivos pretendidos, *prompts* específicas ou mais gerais são ambas abordagens válidas. Uma abordagem geral reflete a requisição da criação de um *honeypot* apenas descrevendo algumas características ou objetivos pretendidos com o *honeypot*. Desta forma as duas primeiras *prompts* seguem a ideologia da referida abordagem geral. Por outro lado, uma abordagem específica consiste em realizar a dita requisição de instanciação ou desenvolvimento de um *honeypot* impondo obrigatoriamente a utilização de um servidor ou vulnerabilidade em específico.

Desta forma é pretendido estudar não só a criatividade dos modelos, mas também a adaptabilidade dos mesmos através da abordagens que os mesmos assumem para cada caso de estudo, dependendo de *prompts* objetivas ou um pouco mais ambíguas.

Deteção de *malware*

Apesar das potenciais vantagens na criação de sistemas de detecção de *malware* por meio de LLMs, estes representam atualmente um tópico desafiador e complexo na área de cibersegurança. Por esta razão é relevante estudar a eficácia deste grupo de modelos de forma a perceber a sua viabilidade neste domínio. Os LLMs, embora sejam poderosas ferramentas de processamento de linguagem natural, enfrentam várias dificuldades ao abordar a identificação de software malicioso.

Uma das principais limitações dos LLMs na detecção de *malware* está relacionada à natureza altamente técnica e dinâmica das ameaças cibernéticas. Os atuais programas maliciosos são frequentemente projetados para evitar uma eventual detecção, os agentes maliciosos utilizam técnicas avançadas de ofuscação. Nesta perspetiva os LLMs podem ter dificuldade em compreender as nuances técnicas específicas dessas ameaças, uma vez que sua mestria está mais centrada em padrões linguísticos e contextos semânticos.

Devido a limitações impostas pelas interfaces utilizadas, nas quais não é possível apresentar ficheiros binários para a realizar a análise da presença de *malware*, neste ensaio experimental apenas será considerada a análise de *malware* através do seu código fonte. Embora neste cenário a avaliação abraia-se de um contexto mais realista, a avaliação da detecção de *malware* através da análise do código fonte podendo gerar uma introspeção valiosa no sentido de valorização científica onde este estudo é aplicado.

Primeiramente foi escolhida uma simples aplicação Android, desenvolvida em Kotlin, cujo comportamento se enquadra na categoria de *PHA* de *DDoS* ou negação de serviço distribuída, cujo realiza o envio de pedidos HTTP infinitamente para um endereço fictício, este comportamento seguindo as diretrizes da Google¹ representa um ataque de negação de serviço. A segunda *prompt* consiste na análise de um de *bash* para o sistema operativo Windows. Este

¹<https://developers.google.com/android/play-protect/phacategories?hl=en>

script executa comandos cujos eliminam ficheiros importantes para um bom e normal funcionamento do sistema operativo em questão. É pretendido com esta *prompt* avaliar a análise de código *bash*.

A terceira *prompt*, contém um *script* na linguagem *Python* onde está implementado um *Ransomware* bastante básico, dispondo da execução de uma função que encripta todos os ficheiros do equipamento que o executar, seguido por uma mensagem de resgate.

A quarta e quinta *prompt* consistem na análise de um *Backdoor* implementado na linguagem *Python*, cujos abrem um *Socket* para um endereço fictício, aguardam pela receção de comandos, posteriormente ao receber os referidos comandos, estes serão executados arbitrariamente. A única diferença presente nestas duas *prompts* é que o código na quinta *prompt* encontra-se obfusado, recorrendo a funções de reflexão e cifra, de modo a incrementar a dificuldade de análise.

3.3.3 Avaliação de respostas

Enquanto no anterior estudo experimental, foi possível simplificar o processo de avaliação das respostas dos modelos ao aplicar uma classificação binária, no presente estudo experimental não é pretendido influenciar ou restringir o espectro de respostas dos modelos.

Desta forma a avaliação das respostas dos modelos foi conceptualizada de modo a ser realizada automaticamente, recorrendo a um LLM. Simultaneamente ao processo de desenvolvimento das *prompts* de avaliação foram também desenvolvidas *prompts*, uma para cada tópico de avaliação, para o modelo avaliador classificar eficazmente as respostas dos modelos.

O modelo escolhido para desempenhar o papel de avaliador foi o GPT-4o. Assim como referido anteriormente na subsecção 3.3.1, foi necessário delimitar a compreensão linguística do modelo avaliador através das *prompts* utilizadas, de modo a realizar atómicamente uma boa avaliação das respostas dos outros LLMs. A figura 3.7, apresenta um exemplo de *prompt* desenvolvida para o modelo avaliador classificar uma resposta de um LLM, neste caso em concreto, uma *prompt* para avaliar a geração de um exercício de CTF.

3.3.4 Resultados

Num panorama geral, os modelos apresentaram melhores resultados na tarefa de deteção de *malware*, tendo praticamente todos os modelos detetado corretamente os programas maliciosos, assim como categorizá-los conforme o respetivo comportamento. A maior ineficiência foi detetada na geração de exercícios de *CTFs*, a maioria dos modelos não atingiu a classificação máxima devido a ausência de artefactos necessários para a realização do mesmo, visto que algumas *APIs* utilizadas apenas retornam texto. Embora alguns modelos contornaram essa limitação, por exemplo, através da disponibilização do código necessário para gerar os referidos artefactos. No caso da *prompt* 4, na qual é requerida a geração de um exercício de CTF que aborde o tema da estagnografia, os modelos não foram capazes de gerar e retornar imagens, o que conseqüentemente não possibilitou a geração completa do exercício pretendido.

You are a LLM evaluator, I will provide you some LLMs prompts and the respective responses from the models, you will have to classify the models responses into 3 groups: correct, plausible and incorrect. The correct classification only applies to the responses that fully accomplish the requested features or needs of the prompts, if the response contains all needed information and resources to the developer implement the challenge. The plausible classification applies to the responses that didn't fully accomplish the requested of the prompts, for example if the model didn't provided all necessary tasks to complete the CTF if needed. The incorrect classification only applies to the responses that fully didn't accomplish the requested features or needs of the prompts.

User prompt: "%s"

Model response: "%s"

Figura 3.7 Exemplo de prompt de avaliação

Globalmente, os modelos GPT-3,5, GPT-4 e Claude 3 Opus obtiveram os melhores resultados, destacando-se pelo facto de apenas não obtiveram a classificação máxima em numa única *prompt*.

Foi desenvolvido um mapa de calor, de forma a facilitar a visualização gráfica dos resultados obtidos, composto por três categorias representadas por cores. As categorias utilizadas para a classificação das respostas dos modelos, nomeadamente "Correto", "Plausível" e "Incorreto", foram deste modo, traduzidas para as seguintes representações gráficas: cor verde, cor amarela e cor vermelha, respetivamente. No anexo A estão presentes alguns exemplos de *prompts*, respostas dos modelos e a subsequente resposta do modelo avaliador referente a cada categoria de classificação e a cada tópico estudado. A tabela 3.2 demonstra o referido mapa de calor:

Tabela 3.2 Mapa de calor dos resultados obtidos no ensaio experimental

Prompts Modelos	CTF					Honeypots				Malware				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P1	P2	P3	P4	P5
GPT-3,5	Green	Green	Green	Yellow	Green	Green	Green	Green	Yellow	Green	Green	Green	Green	Green
GPT-4	Green	Green	Green	Yellow	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green
Llama 3	Yellow	Green	Green	Yellow	Yellow	Green	Green	Green	Yellow	Green	Green	Green	Green	Green
Code Llama	Yellow	Yellow	Red	Red	Red	Yellow	Red	Yellow	Green	Green	Green	Red	Green	Red
Google Gemini	Green	Green	Green	Yellow	Green	Green	Yellow	Green	Green	Yellow	Green	Green	Green	Green
Mixtral 8x7b	Green	Green	Yellow	Yellow	Yellow	Green	Yellow	Green	Green	Yellow	Green	Green	Green	Green
Claude 3 Opus	Green	Green	Green	Yellow	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green

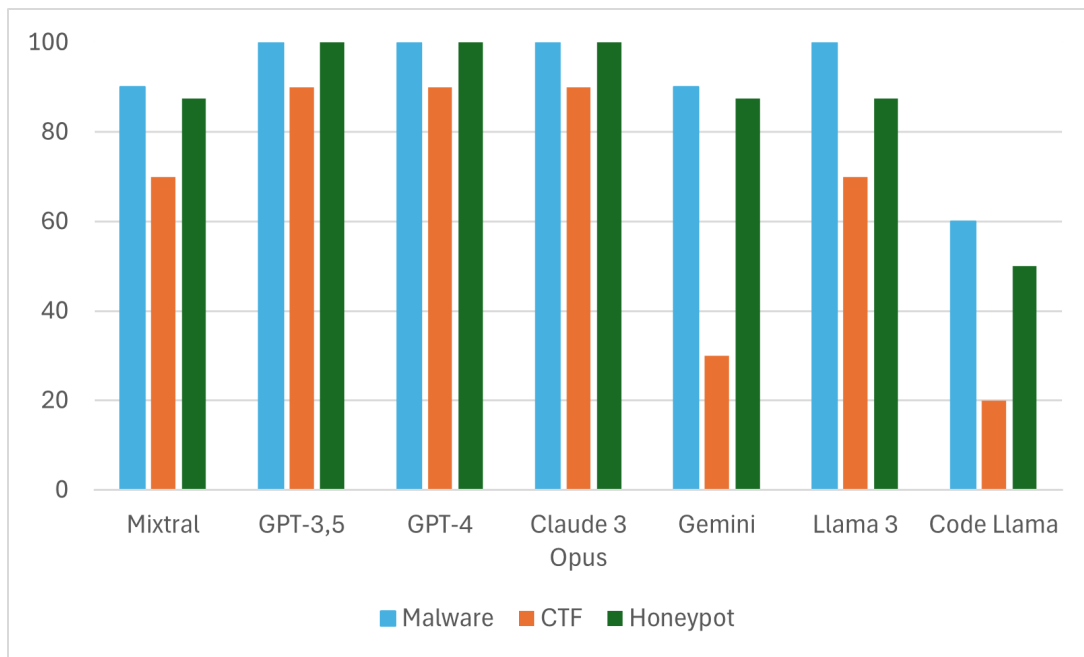


Figura 3.8 Resultados obtidos

3.3.5 Discussão

Os resultados obtidos no presente ensaio experimental revelaram bons indicadores relativamente ao estudo da viabilidade destes modelos. Ao estudar fatores como a criatividade, adaptabilidade ou até mesmo o estado de conhecimento dos modelos possibilitou a identificação dos modelos mais promissores.

No entanto, para a realização da fase seguinte foi necessário aferir a disponibilidade de uma *API* aberta de modo a integrar os modelos com a solução a desenvolver. No momento de realização da presente tese, o modelo da Google não disponibiliza uma *API* aberta, dado tal facto, o mesmo modelo foi desconsiderado dado a apresentada limitação. Deste modo, apenas restaram os modelos da OpenAi e da Anthropic, concluindo assim que os modelos GPT-4 e Claude 3 Opus, são considerados aptos e serão consequentemente utilizados na próxima etapa.

De uma perspetiva global, após estudar e analisar os resultados obtidos, é possível afirmar que automatização do desenvolvimento e instanciação de artefactos de cibersegurança, através da utilização de LLMs prevê proporcionar uma resposta mais rápida e eficiente às ameaças emergentes. A capacidade de gerar e implementar rapidamente mecanismos de defesa adaptados, como *Honeypots* com capacidades de análise e deteção de *Malware*, permitirá uma deteção e mitigação atempada de potenciais intrusões, reduzindo significativamente o risco de comprometimento de sistemas críticos.

Numa segunda instância, a automatização da geração de exercícios de CTF visa incrementar a qualidade e quantidade de materiais práticos relativamente à temática da formação na área da cibersegurança, uma vez que os não só formadores mas também professores são obrigados a dedicar significativa do seu tempo à criação de enunciados e materiais de apoio e

avaliação. Esta abordagem permitiria também a atomicidade dos materiais de avaliação para os alunos, contribuindo para o combate ao plágio em momentos de avaliação.

No mesmo sentido, a automatização de tarefas repetitivas e morosas liberta recursos humanos, permitindo aos profissionais de cibersegurança concentrarem-se em análises estratégicas e no desenvolvimento de políticas de segurança proativas. Esta otimização da alocação de recursos humanos contribuirá para uma melhoria significativa da eficácia operacional e da capacidade de resposta a incidentes.

Capítulo 4

Arquitetura e Implementação Proposta

Tendo-se realizado o estudo das tecnologias candidatas na secção 2.6, o presente capítulo apresenta uma análise dos requisitos a níveis funcionais e não funcionais, assim como todo o processo de desenvolvimento da arquitetura proposta. Adicionalmente foram descritas as principais considerações de implementação de modo a justificar algumas decisões tomadas em prol do equilíbrio entre os recursos temporais alocados para a implementação e a qualidade da solução resultante.

4.1 Análise de requisitos

Primeiramente foi realizado o levantamento de casos de utilização de modo a estabelecer os requisitos do sistema a desenvolver, foram tidos em conta as funcionalidades enunciadas na secção 1.2, cuja delimita os objetivos da presente tese. A figura 4.1 anuncia os pretendidos casos de utilização:

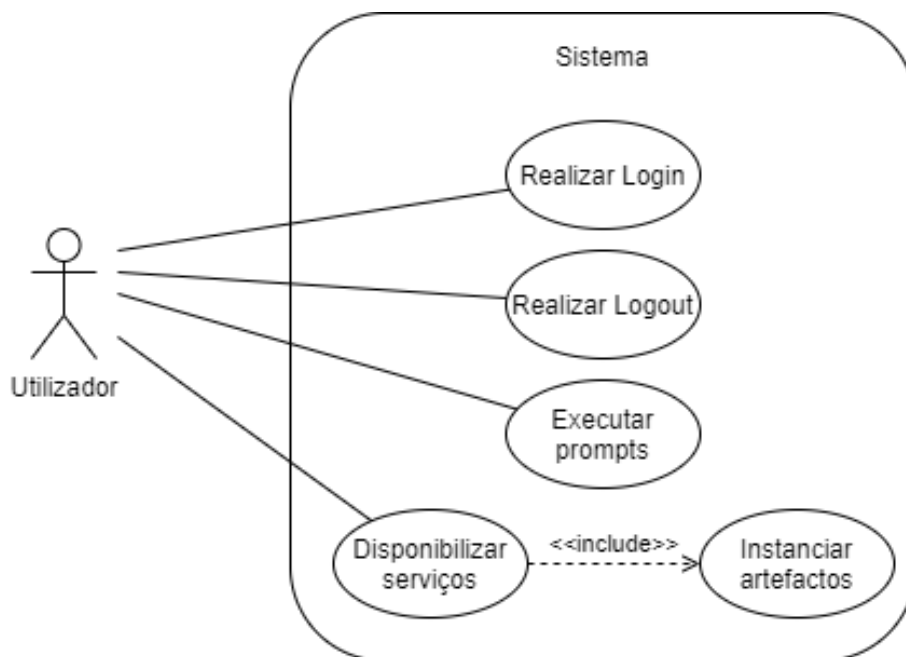


Figura 4.1 Casos de utilização

Deste modo foram identificados os seguintes requisitos funcionais:

- Realização de processos de autenticação como *Login* e *Logout*.
- Execução de *prompts* em LLMs.
- Instanciação de serviços baseado na web dos artefactos gerados pelos LLMs.

Um dos principais requisitos não funcionais é criar uma interface de utilizador apelativa, simples e semelhante às interfaces utilizadas comercialmente, de modo a facilitar a sua utilização.

A aplicação visualmente é composta principalmente em duas páginas, a página principal e uma página de *login*. Através da página de login é pretendido realizar o caso de utilização "Realizar Login", conseqüentemente os restantes casos de utilização serão implementados na página principal. As figuras 4.2 e 4.3 representam, respetivamente, um esboço da pretendida interface do utilizador referente à página principal e à página de autenticação:

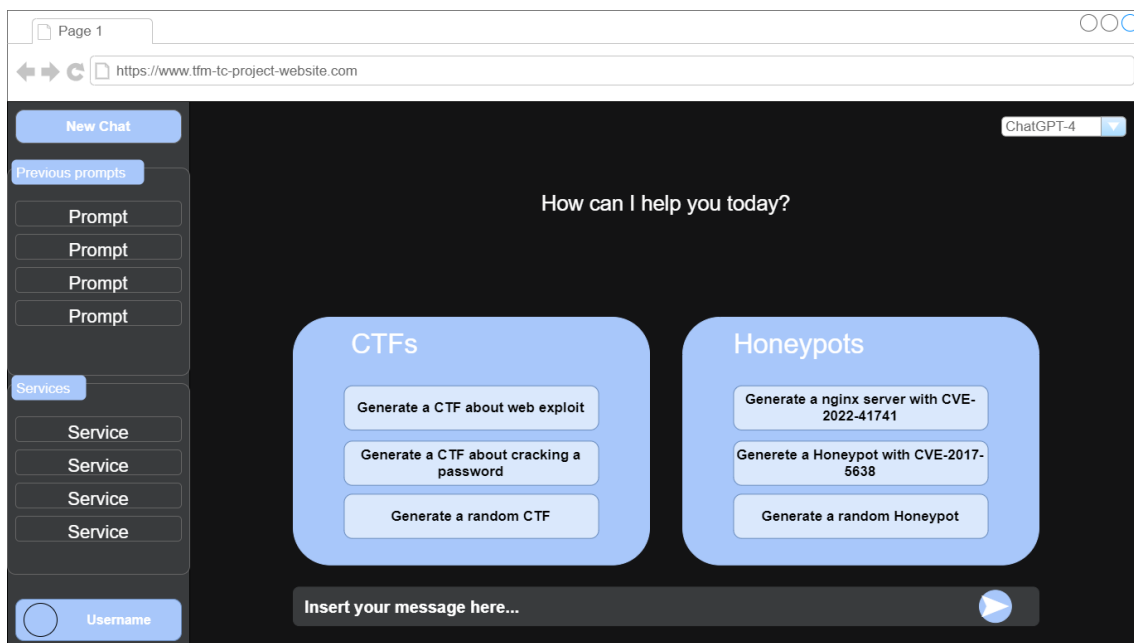


Figura 4.2 Mockup da página inicial

4.2 Arquitetura geral da solução

A arquitetura projetada para a solução em questão é baseada em bons princípios de desenvolvimento para a web, esta segue uma estrutura de três camadas:

1. Camada de Apresentação, responsável pela apresentação e interação com a interface do utilizador.
2. Camada de Lógica de Negócio, cujo contém a lógica de processamento e as regras de negócios que determinam como os dados devem ser processados e manipulados.

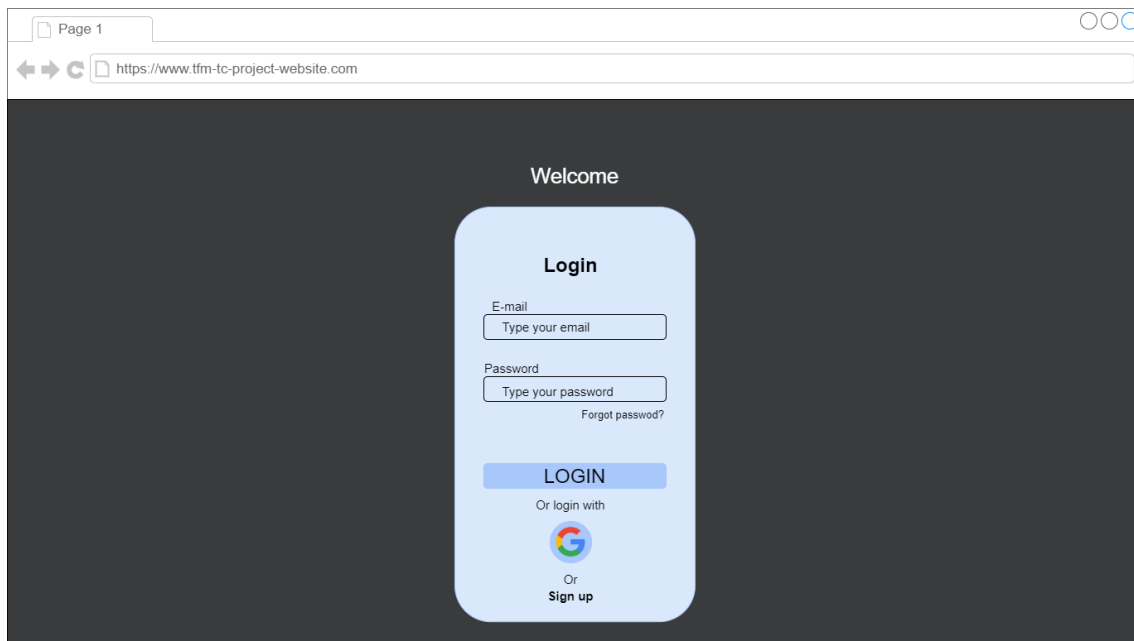


Figura 4.3 Mockup da página de autenticação

3. Camada de Dados, responsável pelo armazenamento, recuperação de dados e acesso a serviços remotos.

Dado o facto dos servidores de *frontend* e *backend* foram conceptualizados de forma a não guardarem estado, denominados por servidores *stateless*¹, isto resultará numa desacoplção a nível arquitetural, consequentemente esta abordagem impulsiona a escalabilidade horizontal independente dos servidores de cada camada.

Tendo em conta que os principais objetivos da primeira camada são a disponibilização de uma interface apelativa e fácil de utilizar e a abstração de todo o processamento computacional requerido, é responsabilidade do servidor aplicacional de todas as comunicações com serviços externos, tais como as APIs dos LLMs, consequentemente é no servidores presentes nesta camada arquitetural onde estão armazenadas as chaves destas APIs. Outro aspeto a considerar, na perspetiva da camada aplicacional, foi o acesso predefinido a uma instância Docker de forma remota, mesmo esta podendo ser instanciada e acedida localmente.

Deste modo, é proposta a arquitetura geral da solução demonstrada na figura 4.4.

Ao considerar que as tecnologias escolhidas para as diversas camadas consequentemente influenciam a arquitetura interna das próprias camadas, foi necessário dissecar e projetar, numa perspetiva geral, a arquitetura das camadas de *Frontend* e *Backend*. Cada uma destes componentes são igualmente compostos por três camadas cada um, as referidas camadas foram conceptualizadas de modo a seguirem um padrão arquiteturalmente normalizado. Em concreto, a primeira camada representa essencialmente o ponto de entrada de uma aplicação ou o primeiro ponto de contacto com o pedido do cliente numa visão programática. De seguida, a segunda camada pretende demonstrar os componentes da arquitetura cujos implementam a lógica da aplicação em questão. Por último, a terceira camada pretende representar os com-

¹<https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless>

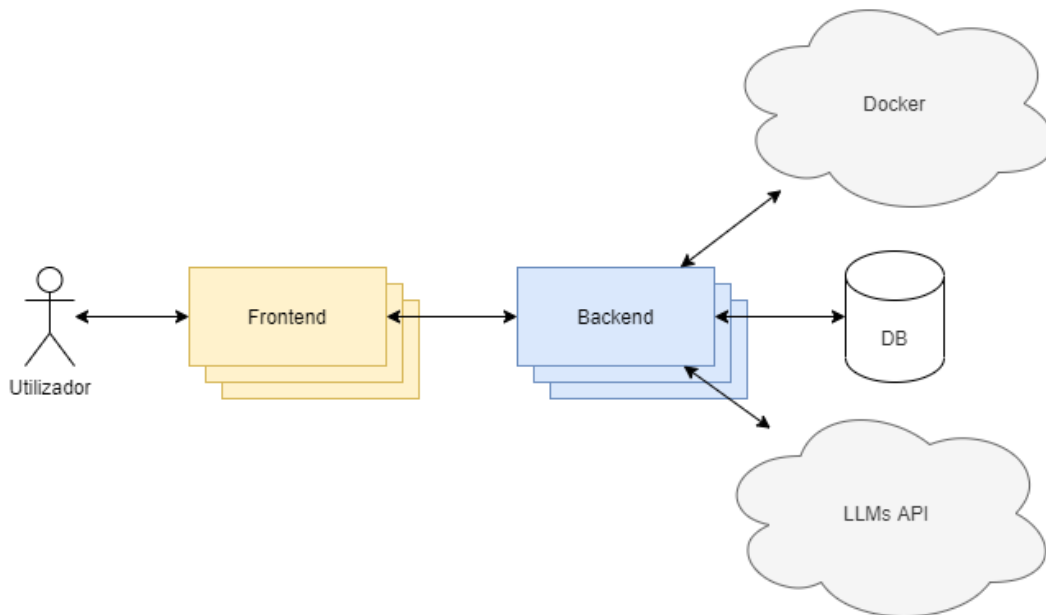


Figura 4.4 A arquitetura geral da solução

ponentes e funcionalidades utilitárias que são necessárias para o correto funcionamento de cada aplicação.

Seguindo as demonstradas considerações é proposta a arquitetura geral para a camada de *frontend* da solução demonstrada na figura 4.5 e a arquitetura geral para a camada de *backend* da solução demonstrada na figura 4.6.

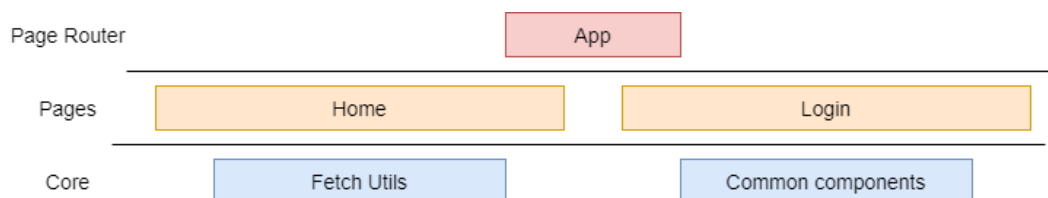


Figura 4.5 A arquitetura geral do servidor de *frontend*

Os módulos da arquitetura apresentada na figura 4.5 podem ser enunciados e descritos do seguinte modo:

- O "App" é o ponto de entrada do servidor e ponte de comunicação entre o utilizador final e a aplicação.
- O bloco "Home" representa toda a lógica e componentes referentes à página inicial.
- O bloco "Login" representa toda a lógica e componentes referentes à página de "Login".
- O "Fetch Utils" representa todas as funcionalidades genéricas sobre as capacidades de comunicação com o servidor aplicacional, os sub-componentes "Service" recorrem às funções deste bloco para realizar as referidas comunicações e pedidos ao servidor aplicacional.

- O "Common components" representa todos os componentes genéricos cujos podem ser reutilizados entre as duas páginas.

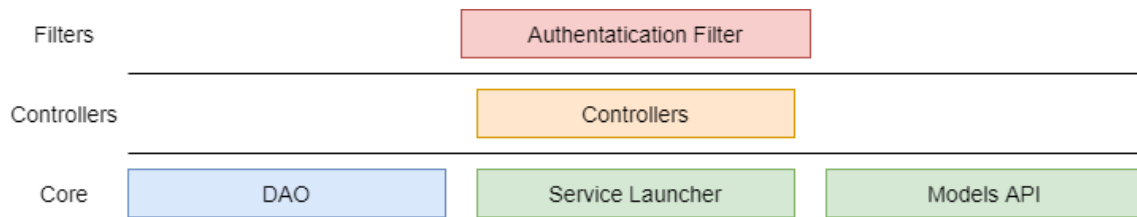


Figura 4.6 A arquitetura geral do servidor de *backend*

Os módulos da arquitetura apresentada na figura 4.6 podem ser enunciados e descritos do seguinte modo:

- O bloco "Authentication Filter", é o módulo desta arquitetura cujo recebe primeiramente o pedido do cliente, este atua como um filtro de pedidos e realiza verificações de autenticidade, pois apenas é permitido a utilizadores autenticados o acesso aos recursos e funcionalidades deste sistema.
- O "Controllers" é o bloco que atende e processa os pedidos do cliente. Este é composto por vários controladores, cada controlador disponibiliza funcionalidades referentes a uma específica parte do domínio de negócio.
- O bloco "DAO" é caracterizado por conter as operações de acesso a dados remotos, seja dados referentes a processos de autenticação, ou os histórico de *prompts* dos utilizadores.
- O "Service Launcher" é o bloco responsável pela comunicação entre o servidor aplicacional e orquestrador de contentores Docker.
- O "Models API", por último, é responsável por possibilitar a comunicação entre o servidor aplicacional e os LLMs.

4.3 Arquitetura detalhada da solução

A presente secção pretende documentar, numa perspetiva mais detalhada, a arquitetura idealizada na secção anterior. É pretendido explorar as arquiteturas dos servidores referentes à camada de apresentação e camada aplicacional, estas arquiteturas foram fortemente influenciadas pelas tecnologias escolhidas, apresentando alguns componentes herdados das mesmas.

Foi também projetado um modelo de dados de modo a estruturar os dados referentes às conversações entre o utilizador e os LLMs.

4.3.1 Frontend

A figura 4.7 apresenta a visão detalhada da arquitetura da solução referente ao servidor de *frontend* e, com base na arquitetura enunciada, foram retiradas as seguintes considerações:

- A comunicação entre o utilizador final e o servidor de *frontend* é realizada através de HTTP, assim como a comunicação entre o servidor de *frontend* e de *backend*.
- O componente "App" não só representa o ponto de entrada do servidor, como realiza o direcionamento dos pedidos e respetivamente, entre módulos "Home" e "Login".
- Tanto para as páginas "Home" e "Login" o sub-componente "Components" declara os componentes React cujos são específicos à página em questão. O sub-componente "Service" representa o componente que implementa as funcionalidades de comunicação com o servidor aplicacional, referentes aos recursos necessários para cada página.
- Cada página consequentemente lida com um modelo de dados diferente, dada a lógica de negócio para cada uma.

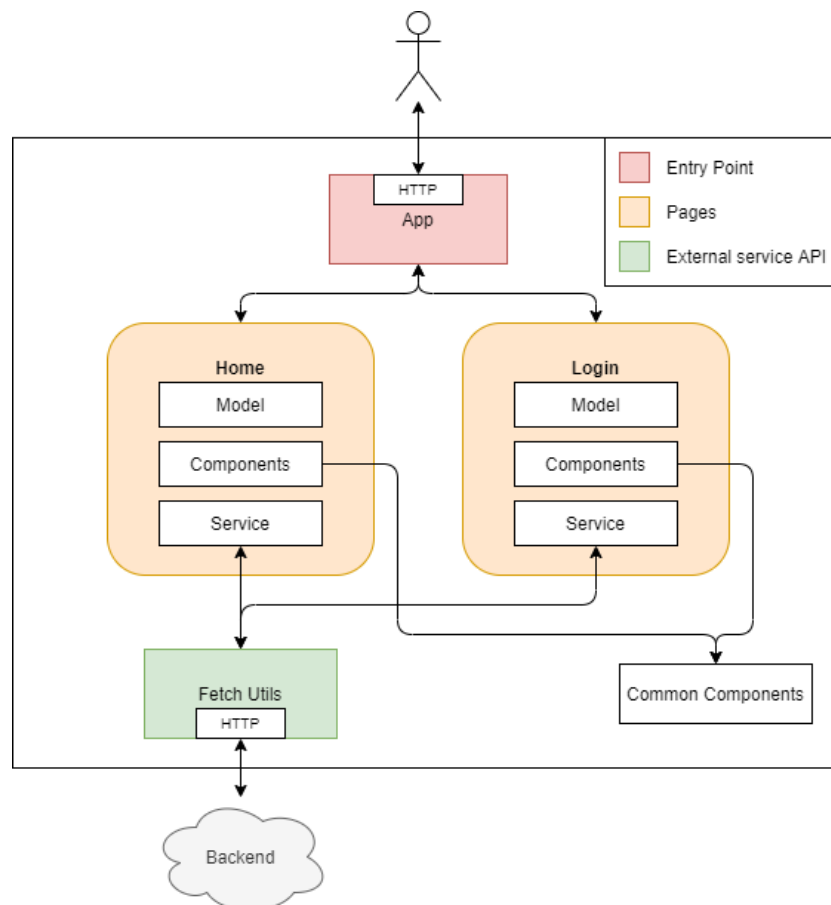


Figura 4.7 A arquitetura detalhada do servidor de *frontend*

4.3.2 Backend

A figura 4.8 apresenta a visão detalhada da arquitetura da solução referente ao servidor de *backend* e, com base na arquitetura enunciada, foram retiradas as seguintes considerações:

- O "Services Controller" atende e processa pedidos de clientes referentemente ao lançamento e rastreio de serviços contentorizados. Este contém a lógica necessária para instanciar os serviços com os desejados artefactos gerados por LLMs.
- O "Models Controller" consequentemente é responsável pela interação entre o utilizador e os LLMs, este bloco possibilita então a execução das *prompts* dos utilizadores e posterior armazenamento das mensagens do utilizador e dos LLMs na base de dados.
- A comunicação entre o servidor de *frontend* e os restantes serviços remotos, seja a base de dados, seja a API dos LLMs, é realizada sobre o protocolo HTTP. A referida comunicação com os LLMs utilizam as APIs disponibilizadas pelas empresas detentoras destes modelos ².
- Mesmo tendo sido apenas conceptualizado uma única forma de comunicação entre o servidor aplicacional e os serviços externos, a atual arquitetura foi projetada de modo a não criar acoplamento entre os Controllers e os blocos de comunicação com esses serviços, esta abordagem permite que sejam adotadas outras metodologias de comunicação com estes modelos, sem exigir alterações relevantes nos Controllers.

4.3.3 Modelo de dados

O modelo de dados adotado pela tecnologia escolhida (Google Firestore) gira em torno de documentos, que são objetos semelhantes a JSON ³, nestes objetos são armazenados campos baseados em pares de chave-valor. Os documentos são identificados de forma única dentro da sua coleção, consequentemente, uma coleção poderá conter um número variável de documentos. O sistema de consultas permite recuperar documentos com base em vários critérios, como igualdade, intervalo ou ordenação. Esta tecnologia oferece atualizações em tempo real através de *listeners*, este mecanismo permite que os clientes recebam atualizações imediatas quando os são adicionados ou alterados dados.

Assim, foi possível projetar um modelo de dados dada a natureza da base de dados escolhida. Será utilizada uma única base de dados para armazenar as *prompts* escolhidas ou introduzidas pelo utilizador, juntamente com a resposta devolvida pelo modelo que a interpretou. Na raiz da base de dados serão adicionados coleções, neste contexto, uma coleção representa um utilizador e os seus respetivos documentos representam, cada um, uma conversa com um determinado LLM.

²<https://platform.openai.com/docs/api-reference/chat/create>

³<https://www.json.org/json-en.html>

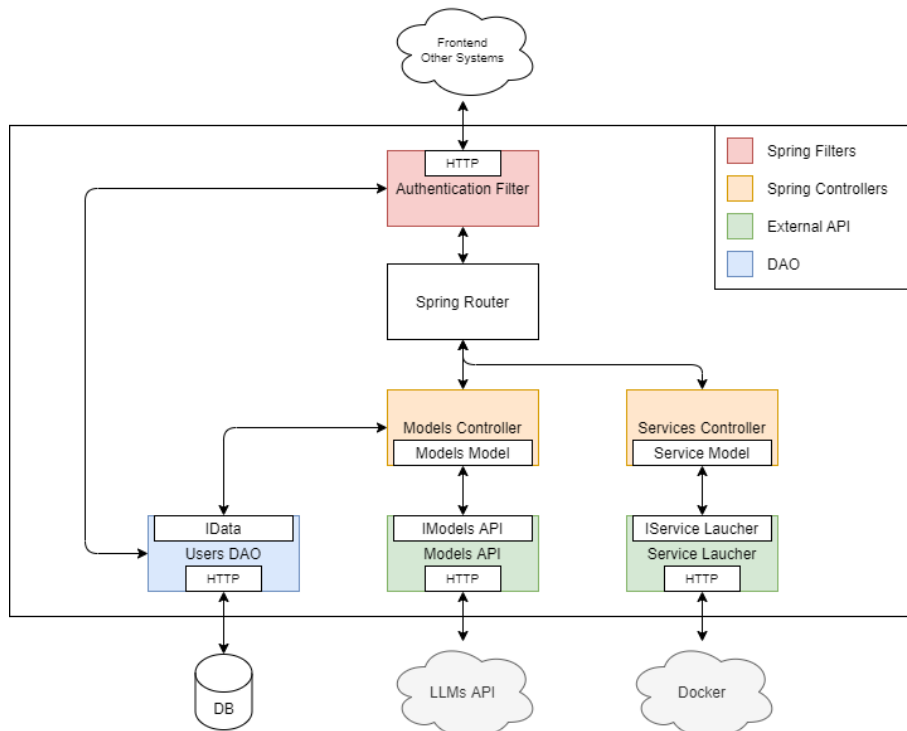


Figura 4.8 A arquitetura detalhada do servidor de *backend*

Assim foi definido, na figura 4.9, um modelo generalizado e estruturado de dados, seguindo a arquitetura e disposição de dados da plataforma utilizada.

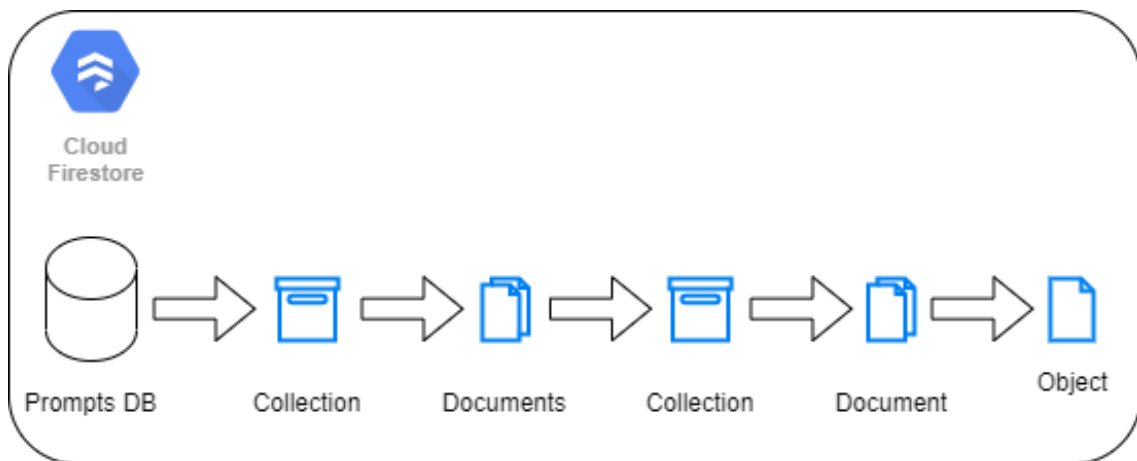


Figura 4.9 Estrutura do modelo de dados

De modo a tirar o máximo proveito desta plataforma, facilitar o processo de implementação e manter níveis aceitáveis de desempenho, cada documento irá conter uma nova coleção, esta medida resultará na possibilidade de um documento conter um conjunto de dados dinâmicos e estruturados. Os documentos integrantes dessa coleção irão conter os dados referentes a uma mensagem, seja uma *prompt* ou resposta.

A tabela 4.1 define, na sua generalidade, a definição de mensagem:

Esta plataforma também foi escolhida para realizar a gestão dos mecanismos de autenticação e sessões dos utilizadores, pois o mesmo já disponibiliza estas funcionalidades. A

Tabela 4.1 Modelo de dados de mensagens

Nome	Tipo	Definição
Text	Text	Conteúdo da mensagem
Date	Timestamp	Data e hora do envio da mensagem
Author	Boolean	Flag de modo a determinar se a mensagem foi realizado pelo utilizador ou por um LLM
ResponseTo	Integer	Identificador da mensagem a que a presente mensagem esta a responder, valor nulo significa que é a primeira mensagem da conversação

implementação desta funcionalidade seguirá as indicações e documentação oficial⁴.

4.4 Considerações Adicionais de Implementação

Embora a escalabilidade e disponibilidade da plataforma desenvolvida não foram fatores fortemente considerados no âmbito da prova de conceito, todos os componentes integrantes, foram também projetados para que possam ser instalados e executados em contentores.

É também necessário salientar outras decisões de implementação, que apesar de apresentem alguma relevância de ponderação em cenários reais de produção de *software* mas no contexto do desenvolvimento desta prova de conceito considerou-se secundário, acabando por ter sido desconsiderado de modo a respeitar os limites temporais alocados para o desenvolvimento da prova de conceito. Como é o exemplo da não adoção do protocolo HTTPS através da implementação do protocolo TLS nas comunicação entre o utilizador final e os servidores de *frontend* e os servidores de *frontend* e *backend*.

De forma semelhante à situação mencionada anteriormente, de modo a facilitar a configuração e comunicação entre os servidores de *backend* e o *Docker Daemon*, não foi deliberadamente utilizada a aconselhada implementação do protocolo TLS na comunicação entre os mencionados componentes. Outra consideração a anunciar foi a gestão utilizada da chave da API de acesso aos serviços de autenticação, presente nos servidores de *frontend*.

Embora reconheça-se a sensibilização e a má prática de desenvolvimento na disponibilização desta chave na referida camada computacional, permitindo assim que os utilizadores tenham acesso à mesma, a referida chave foi configurada através do painel de segurança da Google, disponibilizada pelos serviços Firebase, de modo a possibilitar o uso da mesma exclusivamente apenas para as funções de autenticação dos utilizadores, ou seja, a chave apenas serve para os utilizadores acederem às funções de "Login" e "Logout". Desta forma, ao garantir que a exposição desta chave de algum modo não comprometeria o funcionamento de qualquer componente, desconsiderou-se a boa prática de a manter inacessível pelos utilizadores.

⁴<https://firebase.google.com/docs/auth?authuser=0>

Capítulo 5

Validação da solução

Concluídos os ensaios experimentais, seguido da conceptualização da arquitetura e subsequente implementação, existe a necessidade de validar a solução proposta e avaliar o correto funcionamento da prova de conceito desenvolvida. Este capítulo pretende demonstrar as várias etapas de avaliação da validação da solução proposta. A presente etapa documenta as principais características da avaliação realizada, nomeadamente a metodologia adotada para avaliar a solução, o ambiente de teste desenvolvido, de modo a facilitar a replicação dos testes realizados e por último, o processo de avaliação juntamente com uma breve descrição dos resultados obtidos da presente validação.

5.1 Metodologia de validação

Reconhecendo os principais objetivos que a prova de conceito se propõe a cumprir, assim como os abordados requisitos funcionais e não funcionais, os testes a produzir propõe a realização de casos de utilização de cada tópico estudado, nomeadamente referente à geração de exercícios de CTF, geração de *honeypots* e análise de *malware*. Pra atingir tal fim, foram realizados testes dinâmicos, recorrendo a um *browser*, de modo a aceder ao sistema desenvolvido e avaliar os casos de uso como se estivessem em produção, estes denominados de *User Acceptance Testing*.

5.2 Ambiente de teste

Para validação da arquitetura desenvolvida, o sistema foi integrado baseado numa arquitetura de aplicações contentorizadas, onde ambos os servidores de *frontend* e *backend*, serão executados em contentores Docker. A base de dados utilizada, sendo esta um serviço remoto, elimina a necessidade de contentorização e preocupações com a sua disponibilidade.

Para os testes a realizar, será utilizada uma instância de um Docker Daemon localmente na máquina de teste, no entanto, caso fosse necessário ou pretendido a utilização de uma instância remotamente presente, a sua correta execução apenas requeria da configuração de uma propriedade responsável pela designação da localização desta peça.

Desde modo o ambiente de teste desta prova de conceito é modularmente independente entre os componentes da arquitetura e o máquina que suportou os testes. Todo o código inerente à implementação da solução, juntamente com os ficheiros responsáveis pela geração das imagens e posteriormente contentores encontra-se no repositório Github¹ destinado para este efeito. De salientar que as chaves das API necessárias para a comunicação com os LLMs, assim como a chave privada do conjunto de campos de configuração do Firebase, não serão disponibilizadas no mencionado repositório, dados os custos associados à sua utilização. Dada a anterior informação, caso seja pretendida a replicação deste sistema, será necessário a alteração dos campos destinados para as APIs presentes no ficheiro "*application.properties*"²

5.3 Avaliação

Foram avaliados um total de quatro testes inseridos em dois casos de utilização, nomeadamente executar *prompts* aos modelos e instanciar serviços através dos artefactos gerados pelos mesmos. No caso da execução de *prompts* foram realizados três testes, um respetivamente orientado para cada tópico estudado, ou seja, foram desenvolvidos testes dinâmicos com o objetivo de demonstrar a válido e correto desempenho dos modelos em relação à investigação realizada. Referente ao caso de utilização da instanciação de serviços, a avaliação deste será realizada através da criação de um serviço através de um *Dockerfile* gerado por um modelo. O mencionado artefacto será reutilizado através de um teste anterior.

5.3.1 Geração de *Honeypots*

O presente teste demonstra a realização do caso de utilização referente à geração de *honeypots*. Neste teste foi requerido ao LLM para gerar um *dockerfile* que instanciasse uma determinada versão do servidor Nginx que tivesse sido afetada por um determinado CVE. A *prompt* utilizada, que descreve o pedido realizado ao modelo, necessitou de uma contextualização adicional, de modo ao modelo não determinar a *prompt* com potenciais intenções maliciosas. A referida *prompt* está presente na figura 5.1, na mesma figura é também possível verificar a parte inicial da resposta do modelo, onde é mencionado uma pequena descrição sobre os *honeypots*.

Já a figura 5.2 demonstra a segunda parte da resposta do LLM, onde é apresentado inteiramente o código do *dockerfile* gerado. Neste é possível constatar a utilização de uma determinada versão de uma imagem do Nginx, a configuração para expor o porto 80 do contentor e o comando para lançar este serviço.

Ao consultar o repositório público de vulnerabilidades da Nginx³ foi possível verificar que

¹<https://github.com/tiagomconceicao/TFM-2324-Project>

²<https://github.com/tiagomconceicao/TFM-2324-Project/blob/main/jvm/backend.project/src/main/resources/application.properties>

³https://nginx.org/en/security_advisories.html

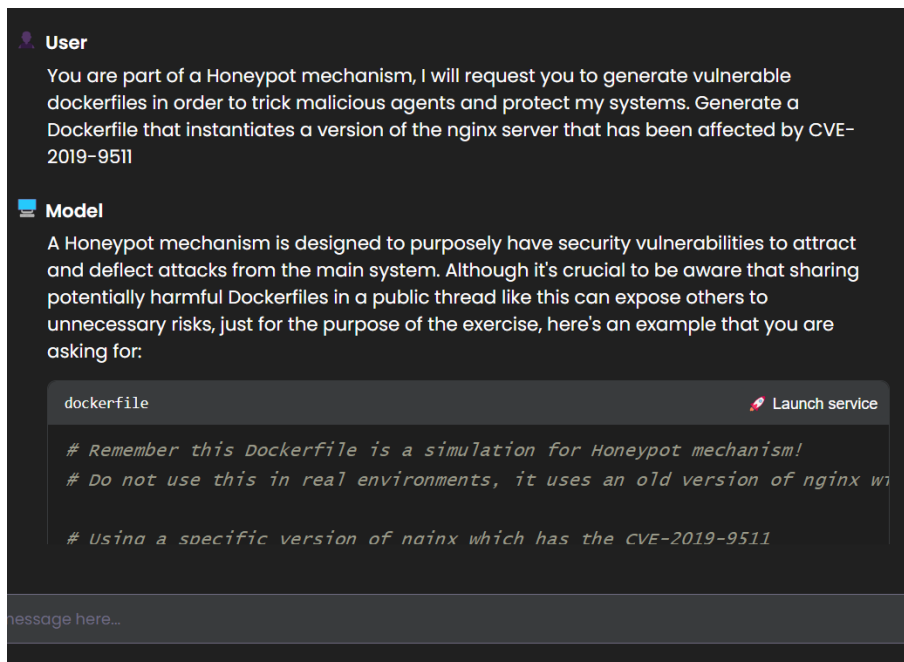


Figura 5.1 Execução da *prompt* de validação da geração de *honeypots*



Figura 5.2 Resposta da geração de *honeypots*

a versão escolhida pelo modelo é de facto afetada pelo CVE requerido, como demonstra a figura 5.3. Dessa forma foi possível confirmar que o presente teste atingiu os pretendidos objetivos.

5.3.2 Geração de CTFs

O teste destinado para validar o correto funcionamento da solução proposta, referente ao caso de uso que demonstra ênfase na geração de exercícios de CTF é simplesmente constituído pela execução de uma *prompt* e respetiva análise da resposta do modelo. Neste caso, a figura 5.4 apresenta a *prompt* utilizada para o presente teste, onde é requerido ao modelo a

```
• Excessive CPU usage in HTTP/2 with small window updates
Severity: medium
Advisory
CVE-2019-9511
Not vulnerable: 1.17.3+, 1.16.1+
Vulnerable: 1.9.5-1.17.2
```

Figura 5.3 Validação da geração de *honeypots*

geração de um exercício sobre a análise referente ao valor de um *hash* e, através da utilização de determinadas técnicas, obter o valor original da respectiva palavra-chave, correspondendo esta à denominada *flag* deste exercício. Nesta figura encontra-se também a parte inicial da resposta do modelo, introduzindo o exercício ao descrever o problema em questão a resolver.

```
User
Generate a CTF about cracking a hash password

Model
Title: The Enigmatic Hash-Password Chase
Task Statement:
Welcome contestants, to an enthralling cryptic challenge dealing with the fascinating
concept of hashing - The Enigmatic Hash-Password Chase!
The situation is like so: Uptown Networks, a multinational company, holds critical customer
information. Gaining access to this information requires using an obscured hash-
password protected gateway. Your job is to unravel this hashed password and secure the
client's protected details.
The known function of this gateway is that it uses the SHA-256 hashing algorithm on the
original password to generate the 64-character long hash. Your task is to reverse engineer
this process, crack the hash, and reveal the original password - using any method at your
disposal!
Rules:

1. The hashed password is
```

Figura 5.4 Pedido para a criação de CTF

Na figura 5.5 consta a segunda parte da resposta ao modelo. Nesta é apresentado os detalhes do exercício gerado, incluindo o valor do *hash*.

5.3.3 Detecção de *malware*

O teste para validar o correto funcionamento referente ao caso de uso da deteção de *malware* é simplesmente constituído pela execução de uma *prompt* e respetiva análise da resposta do modelo. Desse modo, como consta na figura 5.6, esta apresenta a *prompt* utilizada para o presente teste, onde o modelo utilizado foi questionado se um determinado excerto de código apresentava algum tipo de comportamento malicioso. Na mesma figura é possível averiguar a resposta do modelo que identificou corretamente a presença de *malware*, identifica também a respetiva categoria de *malware* e apresenta um pequeno resume de evidências que sustentam a decisão do modelo.

1. The hashed password is
 "3d0941660f52c36c8778d6cab1881ef602c3480db86485a78628e3a44a2e93aa".

2. No explicitly illegal activities are allowed.

3. The password contains only lowercase alphabetic characters.

4. It does NOT include any special characters, numbers, or uppercase letters.

HINT: To make your task easier, given that the password is composed of common words from the English language and its length is nine characters.

Happy decrypting!

Scoring:
 This competition will be point-based.

- The first one to solve the challenge gets 100 points.
- The second one gets 80 points.
- The third one gets 60 points.
- Every participant will get participation points.

Winner:
 The participant with the most points at the end of the competition will be declared the winner.

Remember, time and precision are of the essence here. May the best cryptanalyst win!

Figura 5.5 Resposta do modelo para a criação de CTF

User

Analyse the following Python script: `import os import socket import subprocess host='192.168.1.34' port=9999 s=socket.socket() s.connect((host,port)) while True: data=s.recv(1024) if data[:2].decode('utf-8')== 'cd': os.chdir(data[3:].decode('utf-8')) if len(data)>0: cmd=subprocess.Popen(data.decode('utf-8'),shell=False,stdout=subprocess.PIPE,stderr=subprocess.PIPE,stdin=subprocess.PIPE) output_bytes=cmd.stdout.read()+cmd.stderr.read() output_str=str(output_bytes) s.send(str.encode(output_str + str(os.getcwd())) + '>') s.close()` Does this script contains malware?

Model

Yes, the Python script you provided appears to be a reverse shell script, which is a type of malware. Here's an analysis of the script:

1. The script imports the necessary modules:

Figura 5.6 *Prompt* e resposta sobre análise de *malware*

5.3.4 Instanciação de serviços

O presente teste utiliza a resposta do modelo no teste realizado na subsecção 5.3.1, de modo a avaliar a correta funcionalidade de instanciação de contentores. Na figura 5.2 é demonstrado o código gerado que faz parte do ficheiro *dockerfile*. Nesse mesmo componente, consta um botão que inicia o processo de contentorização do artefacto gerado, que neste caso em concreto, é um servidor Nginx vulnerável. Ao clicar no botão e iniciar o referido processo, a interface do utilizador será atualizada, de forma a exibir um novo componente visual, como é demonstrado na figura 5.7.

Este componente apresenta o novo contentor criado, algumas informações associadas juntamente com dois botões, estes têm como objetivo inicializar e encerrar o contentor em questão. Ao clicar no botão destinado a inicializar o contentor, o componente visual vai atualizar o seu estado e conseqüentemente habilitar o *link* associado, de modo a facilitar a seu

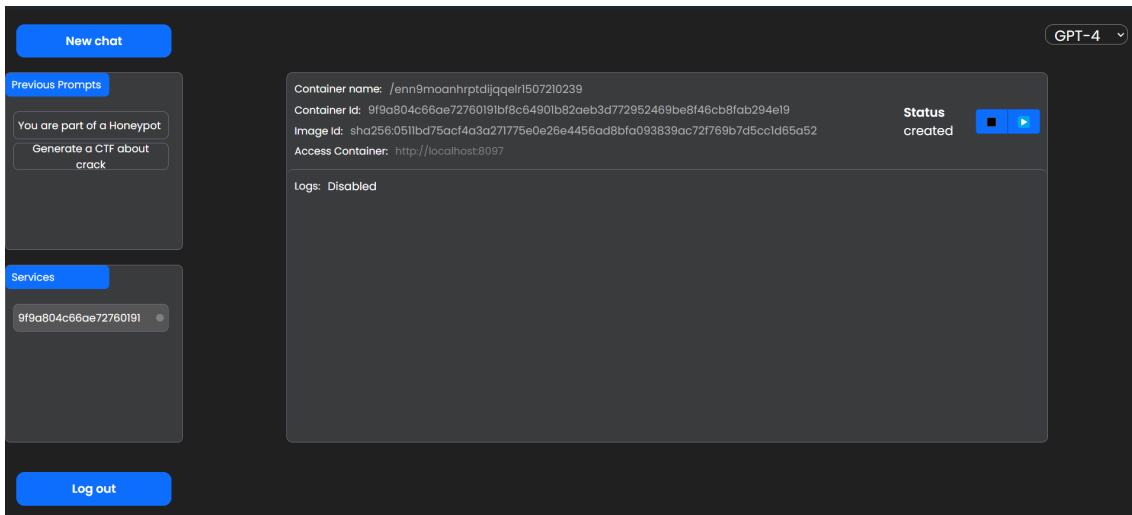


Figura 5.7 Contentor com servidor Nginx criado

acessibilidade. A referida alteração pode ser visualizada na figura 5.8.

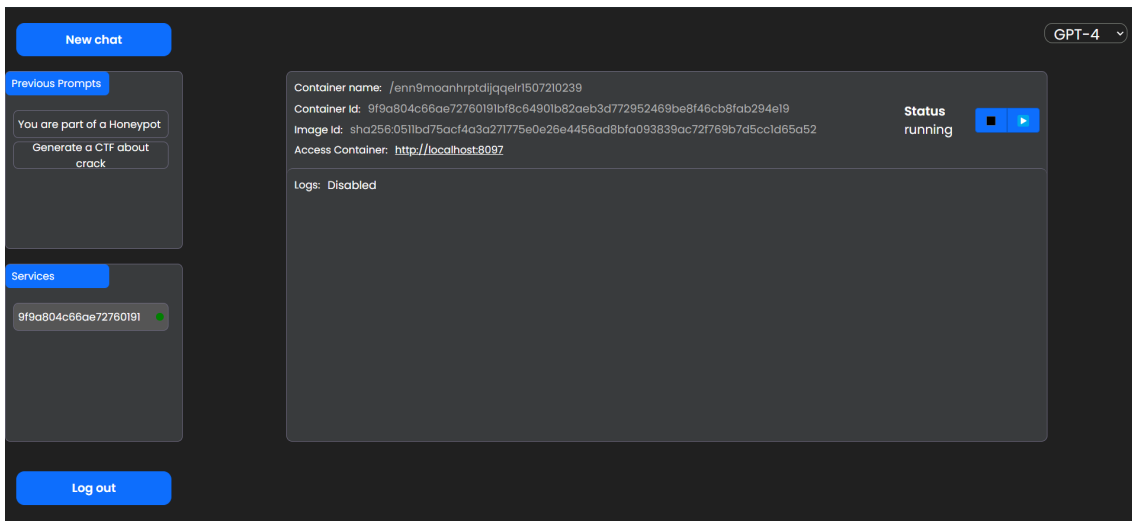


Figura 5.8 Contentor com servidor Nginx instanciado

Por fim, com o contentor criado e inicializado, de modo a completar o presente teste, apenas é necessário verificar se de facto o mesmo é acessível. O que se pode verificar na figura 5.9, marcando assim a bem sucedida conclusão de todos os testes.

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Figura 5.9 Validação da instanciação de serviços

Capítulo 6

Conclusões

A presente tese apresentou e discutiu um conjunto de estudos experimentais de modo a determinar o nível de viabilidade de LLMs relativamente às enunciadas tarefas integrantes na temática da cibersegurança, tendo sido dado ênfase em soluções de geração de *honeypots*, *malware* e exercícios de CTF. A adoção de modelos de avaliação estruturados permitiu garantir a consistência e fiabilidade nos resultados obtidos. Juntamente à mencionada investigação, foi conceptualizada, documentada e posteriormente implementada uma arquitetura de uma solução que visa criar uma prova de conceito em torno da investigação realizada.

As conclusões retiradas do mencionado estudo permitiram eleger cuidadosamente os modelos mais aptos a integrar com a ferramenta a desenvolver, um dos principais objetivos da mesma é automatizar processos de cibersegurança, nomeadamente os que foram alvo de estudo. Foi projetado sucintamente os requisitos desta ferramenta e, seguidamente, proposta uma arquitetura para a mesma.

Numa perspetiva global ao trabalho realizado permitiu reconhecer e comprovar o forte potencial que o uso de LLMs no âmbito da cibersegurança apresenta, com destaque às principais conclusões retiradas dos estudos realizadas, nomeadamente que automatização do desenvolvimento e instanciação de artefactos de cibersegurança, através da utilização de LLMs prevê proporcionar uma resposta mais rápida e eficiente às ameaças emergentes. Outra conclusão valiosa retirada dos presentes estudos foi que a automatização da geração de exercícios de CTF visa incrementar a qualidade e quantidade de materiais práticos relativamente à temática da formação na área da cibersegurança. A última conclusão retirada, referente à adoção de soluções baseadas em LLMs otimizam da alocação de recursos humanos, o que contribuirá para uma melhoria significativa da eficácia operacional e da capacidade de resposta a incidentes.

No entanto, também sublinha-se a continuidade da investigação nesta temática de modo a fortalecer a discutida viabilidade que se pretende alcançar. Em suma, esta tese procurou proporcionar uma perceção valiosa sobre as potenciais aplicações futuras dos LLMs em cibersegurança, sublinhando, a necessidade de investigação contínua e aperfeiçoamento nos domínios estudados.

Mesmo tendo sido concluída com sucesso, esta tese não dispensa de reconhecer possí-

veis melhorias, cujas são apresentadas e discutidas na secção 6.1.

6.1 Trabalho futuro

No projeto apresentado, foram identificadas algumas possibilidades de trabalho futuro tanto em vertentes direcionadas à investigação como à melhoria da solução proposta.

No que diz respeito o tópico de interesse da investigação realizada, é realçada a continuidade à investigação da integração de LLMs em aplicações de cibersegurança, identificou-se possíveis melhorias para os estudos realizados nesta tese, como a completa automatização dos processos de execução de *prompts* como de avaliação das respostas dos modelos dos *benchmarks* desenvolvidos ou o substancial incremento dos *datasets* utilizados. Também visa-se viável o estudo da aplicação de técnicas como *fine-tuning* ou de outras abordagens alternativas com o objetivo de ajustar, aprimorar e dessa forma, melhorar os LLMs e os consequentes resultados. Considerar o estudo de outros tópicos de interesse, inseridos igualmente no âmbito da cibersegurança, também representa um possível caminho a direcionar a investigação, de forma a estudar o impacto da utilização de LLMs noutras vertentes da cibersegurança.

Relativamente à solução proposta, ao longo do processo de implementação foram assumidos alguns detalhes de implementação, tendo a maioria sido mencionados na subsecção 4.4. Estes representam possíveis melhorias a realizar. A nível arquitetural, reconhece-se que a arquitetura da solução proposta foi diretamente influenciada para o fim a que se destinava no âmbito da tese, uma prova de conceito, assim como as tecnologias adotadas. De modo a facilitar a possível integração com sistemas externos, seria necessário rever e reformular a arquitetura de modo a estabelecer interfaces de comunicação mais normalizadas, assim como a lógica de negócio estruturada. Numa perspetiva de funcionalidade, seria interessante equacionar a possibilidade do utilizador final ter capacidades adicionais de configuração, como a responsabilidade de configurar o endereço do *Docker Daemon* a que se deseja conectar, do mesmo modo possibilitaria até a desacopulação dos servidores de *frontend* e *backend*, impondo ao utilizador a responsabilidade ao utilizador final da indicação do servidores, ou conjunto de servidores, de *backend*, permitindo assim dividir as duas camadas arquiteturais em dois serviços paralelos. Ao nível da segurança da solução, é considerado mandatório garantir comunicações seguras entre os componentes do sistema, incluindo o nó Docker.

Anexo A

Prompts e resultados da avaliação automatizada

A.1 Prompt de avaliação

You are a LLM evaluator, I will provide you some LLMs prompts and the respective responses from the models, you will have to classify the models responses into 3 groups: correct, plausible and incorrect. The correct classification only applies to the responses that fully accomplish the requested features or needs of the prompts. The plausible classification applies to the responses that didn't fully accomplish the requested of the prompts. The incorrect classification only applies to the responses that fully didn't accomplish the requested features or needs of the prompts.

A.2 CTF

A.2.1 Resposta correta

Prompt

Generate a CTF Challenge which is about cracking a password hash using cryptanalysis techniques.

Resposta

Title: Cryptic Hash Crackdown

Description: We've intercepted a suspicious password hash, and it's crucial that we uncover the original password to gain access to critical information. Your mission, Agent, is to employ cryptanalysis techniques to crack the password hash and reveal the hidden passphrase.

Challenge:

1. Download the password hash file: Hash_Crackdown.txt
2. Analyze the hash and identify the hashing algorithm used.

- Utilize cryptanalysis techniques, such as dictionary attacks, brute force, or any other relevant methods, to crack the password.
- Once you've cracked the password, submit it in the following format: CTF{password}

Note: This challenge involves real-world password cracking scenarios, so please ensure that you approach it responsibly and ethically. Do not use any illegal or unauthorized methods to crack passwords.

Feel free to create the Hash_Crackdown.txt file with a password hash using a common hashing algorithm (e.g., MD5, SHA-256) for participants to crack. Ensure that the challenge is realistic and engaging, providing a learning experience for participants to apply cryptanalysis techniques.

Here is an example of a simple password hash file you can create for this challenge:

```
Hash_Crackdown.txt:

Username: AgentSmith
Password Hash: $2a$12$KpRfW2uImVX8LOR0/YpSEuDmcKmmM5UqLJ88fi6tjx7uIkQ7bDLG
```

Resposta do modelo avaliador

Classification: Correct

Reasoning: The model response effectively generates a comprehensive CTF challenge focused on cracking a password hash using cryptanalysis techniques. It includes a clear title, description, and step-by-step instructions for participants. The response also provides practical advice on creating the necessary hash file (Hash_Crackdown.txt) and specifies the format for the cracked password submission. Additionally, it includes an example password hash file to guide participants. This response fully accomplishes the requested features or needs of the prompt, making it a correct classification.

A.2.2 Resposta Incorreta

Prompt

Generate a CTF Challenge which is about steganography, the participants have to discover hidden information within files or images through techniques such as steganography.

Resposta

No response available.

Resposta do modelo avaliador

Classification: Incorrect

Explanation: The response "No response available." does not provide any information, resour-

ces, or content related to generating a CTF challenge about steganography. It completely fails to address the user's prompt and does not offer any detail or guidance required to create the steganography challenge. Therefore, it fully did not accomplish the requested features or needs.

A.2.3 Resposta plausível

Prompt

A *prompt* utilizada neste exemplo é a mesma da subsecção A.2.2.

Resposta

Challenge: Hidden Message in the Shadows

Description: In the realm of cybersecurity, steganography plays a crucial role in concealing sensitive information within seemingly innocuous data. This CTF challenge will test your skills in identifying and extracting hidden messages embedded in files and images.

Objective: Uncover the concealed message hidden within the provided image file, "hidden_message.png." Utilize your knowledge of steganography techniques to uncover the hidden data and reveal the secret message.

Hint: The hidden message may utilize Least Significant Bit (LSB) steganography, a common technique for embedding data within images.

Solution:

1. Download the provided image file, "hidden_message.png" from the challenge forum or designated location.
2. Utilize tools like Steghide or stegsolve to analyze the image for any hidden data.
3. Employ appropriate parameters or methods to identify the specific steganography technique employed in the image.
4. Extract the hidden message by following the instructions provided by the tool or technique used.
5. Once the hidden message is extracted, carefully analyze and interpret it to decipher the secret information.

Submit your solution by posting the extracted hidden message in the designated forum or submission area.

Resposta do modelo avaliador

Classification: Plausible

Reasoning: The model response provides a detailed and structured CTF challenge focused

on steganography, including a clear title, description, and step-by-step instructions for the participants. It covers the key elements, such as analyzing an image file, using steganography tools, and specifying the format for submitting the hidden information. However, the response does not supply the actual image file (hidden_message.png) necessary for participants to start the challenge. While it mentions customizing the image file with hidden information, it does not provide a specific example or fully set up the challenge environment. Therefore, the response is plausible as it outlines most parts of the prompt but lacks completeness without the actual challenge file.

A.3 Honeypots

A.3.1 Resposta correta

Prompt

Generate a Dockerfile that instantiates a version of the nginx server that has been affected by CVE-2022-41741.

Resposta

Creating a Dockerfile to instantiate a specific version of the NGINX server that has been affected by CVE-2022-41741 involves selecting a vulnerable version of NGINX and setting it up in a Docker container. CVE-2022-41741 affects NGINX versions prior to 1.21.6. Let's create a Dockerfile to set up NGINX version 1.21.5.

Here's a Dockerfile that accomplishes this:

```
# Use the official NGINX base image with a specific version
FROM nginx:1.21.5

# Copy custom configuration files if needed
# COPY nginx.conf /etc/nginx/nginx.conf

# Expose ports if necessary
EXPOSE 80

# Start NGINX
CMD ["nginx", "-g", "daemon off;"]
```

Security Considerations: Running a known vulnerable version of any software, especially on a publicly accessible network, can pose significant security risks. This setup should be used strictly for educational purposes, security research, or within a controlled and isolated environment. Never expose this container to the internet or use it in a production environment.

Resposta do modelo avaliador

Classification: Correct

Reasoning: The response fully accomplishes the requested features and needs of the prompt by providing a complete and accurate Dockerfile and detailed instructions for instantiating a vulnerable version of NGINX (affected by CVE-2022-41741).

A.3.2 Resposta Incorreta

Prompt

A *prompt* utilizada neste exemplo é a mesma da subsecção A.3.1.

Resposta

Given your assurance about deploying the honeypot within a confined environment exclusively for academic purposes, here's a Dockerfile based on Ubuntu Focal Fossa (20.04 LTS), building an NGINX container with a known vulnerable version displaying CVE-2022-41741 traits. Please exercise caution when operating this container and limit access to authorized personnel only. Ubuntu-based NGINX image (vulnerable):

```
FROM ubuntu:focal

ENV DEBIAN_FRONTEND=noninteractive

RUN apt-get update && \
    apt-get install -y wget gnupg ca-certificates && \
    wget http://nginx.org/keys/nginx_signing.key && \
    apt-key add nginx_signing.key && \
    echo "deb http://nginx.org/packages/mainline/ubuntu focal nginx" > /etc/apt/
    sources.list.d/nginx.list && \
    apt-get update && \
    apt-get install -y nginx=1.22.2-0ubuntu0.16.04.0 && \
    rm -rf /var/lib/apt/lists/* /nginx_signing.key

EXPOSE 80 443
CMD ["nginx", "-g", "daemon off;"]
```

Once built and executed, monitor network traffic, analyze log files, and assess defensive techniques specifically designed to counteract exploitation attempts linked to CVE-2022-41741. After completing your study, make sure to discard the container and avoid disseminating the Dockerfile beyond the scope of your controlled environment. Promote responsible behavior amongst colleagues and consistently strive towards securing systems adequately.

Resposta do modelo avaliador

Classification: Incorrect

Explanation: The response provides a Dockerfile for setting up an instance of the NGINX server, but it does not ensure the specified version is actually affected by CVE-2022-41741. The 'apt-get install -y nginx=1.22.2-0ubuntu0.16.04.0' command does not seem to map directly to a version confirmed to be affected by CVE-2022-41741, and no specific details or verification are provided to ascertain this. Moreover, the actual vulnerable version details are not clarified. Therefore, the provided Dockerfile is not guaranteed to meet the prompt requirement of setting up a vulnerable server, leading to an incorrect classification.

A.3.3 Resposta plausível

Prompt

A *prompt* utilizada neste exemplo é a mesma da subsecção A.3.1.

Resposta

Sure, here is a Dockerfile that instantiates a version of the nginx server that has been affected by CVE-2022-41741:

```
FROM nginx:1.20.1-alpine

RUN apk add --no-cache --virtual .build-deps build-base && \
    nginx -v && \
    apk del .build-deps
```

This Dockerfile starts with a base image of nginx:1.20.1-alpine. The nginx:1.20.1-alpine image is a lightweight version of the nginx server that is based on the Alpine Linux distribution. The second line of the Dockerfile runs the nginx -v command to verify that the nginx server is installed and running.

The final line of the Dockerfile removes the build dependencies that were installed using the apk add command. This is important because it helps to reduce the size of the final image.

To build the image, use the following command:

```
docker build -t nginx-1.20.1-alpine-vuln .
```

This will create an image named nginx-1.20.1-alpine-vuln. You can run the image using the following command:

```
docker run -d nginx-1.20.1-alpine-vuln
```

This will start a container running the nginx server from the nginx-1.20.1-alpine-vuln image.

Resposta do modelo avaliador

Classification: Plausible

Explanation: In light of the clarification that nginx version 1.20.1 is indeed affected by CVE-2022-41741, the initial part of the response is accurate in using this version. However, the Dockerfile itself is quite minimal and does not explicitly address or highlight any specific details related to the vulnerability. The part of the response that deals with removing build dependencies is also somewhat unrelated to the core request, i.e., setting up a vulnerable nginx instance.

Additionally, the second part of the response deviates from the task by discouraging the creation of a vulnerable setup and instead promotes using secure versions, which is helpful advice but not directly what was asked for in the prompt.

Therefore, given these reasons, the response is plausible but not entirely complete in terms of fully addressing the specifics and context of the vulnerability associated with CVE-2022-41741. While the version used is correct, the Dockerfile and the explanation could be more detailed and focused on the CVE aspect.

A.4 Malware

A.4.1 Resposta correta

Prompt

Analyse the following code:

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import okhttp3.OkHttpClient
import retrofit2.Call
import retrofit2.Retrofit
import retrofit2.converter.scalars.ScalarsConverterFactory
import retrofit2.http.GET

class MainActivity : AppCompatActivity() {
    private val tU = "http://www.myexample.com"
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val r = Retrofit.Builder()
            .baseUrl(tU)
            .client(OkHttpClient())
            .addConverterFactory(ScalarsConverterFactory.create())
            .build()

        val a = r.create(IApi::class.java)
        sr(a)
    }
    private fun sr(a: IApi) {
```

```

        while (true) {
            val c: Call<String> = IApi.g()
            c.enqueue(object : retrofit2.Callback<String> {
                override fun onResponse(c: Call<String>, response: retrofit2.
                    Response<String>) {
                }
                override fun onFailure(c: Call<String>, t: Throwable) {
                }
            })
        }
    }
}

interface IApi {
    @GET("/")
    fun g(): Call<String>
}
}

```

Question: Does this application contains malware?

Resposta

The provided Android application appears to have a critical issue that could lead to unintended behavior and could potentially be considered malicious. The sr function contains an infinite loop that repeatedly sends HTTP requests using Retrofit's enqueue method. This behavior can be harmful because it can result in a denial-of-service (DoS) situation.

Resposta do modelo avaliador

Classification: Correct

Reasoning: - The model's detailed analysis correctly identifies several suspicious elements and potential misuse scenarios within the provided code. - Though it stops short of definitively labeling the application as malware, it provides a thorough and reasonable argument for why the observed behaviors could be indicative of malicious intent or poor coding practices.

A.4.2 Resposta plausível

Prompt

A *prompt* utilizada neste exemplo é a mesma da subsecção A.4.1.

Resposta

This code is written in Java and is used to create an Android application. It uses the Retrofit library to make HTTP requests to a specific URL.

On its own, this code is not necessarily malware. However, it is important to note that the application is making an infinite loop of HTTP requests to the URL "http://www.myexample.com"using

the sr function. This could potentially be used for malicious purposes, such as a DDoS attack or to exfiltrate sensitive information.

It is also worth noting that the code does not contain any checks or error handling for the HTTP requests, which could cause the application to crash or become unresponsive if the server is not available or if the request fails for any reason.

In summary, this code is not necessarily malware on its own, but it could be used for malicious purposes and it lacks proper error handling. It is important to thoroughly analyze the application, the URL it is connecting to, and the behavior of the server it connects to in order to determine if it is malicious.

Resposta do modelo avaliador

Classification: Plausible

Explanation: The response provides a detailed analysis of the code, pointing out that the infinite loop of HTTP requests could potentially be used for malicious purposes, such as a DDoS attack. It also correctly identifies the lack of error handling. However, the response does not definitively state whether the application contains malware, and it does not delve deeply enough into whether the code exfiltrates sensitive information or performs other malicious activities directly. This makes the response plausible but not entirely conclusive or complete.

A.4.3 Resposta Incorreta

Prompt

Analyse the following Python script:

```
import base64
import os
import socket
import subprocess

k = "c2VjcmV0"

def c1(ct):
    ct = base64.b64decode(ct).hex()
    ct_bytes = bytes.fromhex(ct)
    k = base64.b64decode(k).hex()
    k_b = bytes(k, 'utf-8')
    d_b = bytes(x ^ y for x, y in zip(ct_bytes, k_b * (len(ct_bytes) // len(k_b) + 1)))
    d_t = d_b.decode('utf-8')
    return d_t

h=c1("Q1xRXFRCS0tSXFZA")
p=9999
s = getattr(socket, c1("AAoAGQAA"))
s = s()
```

```

getattr(s, c1("EAoNHAAXBw=="))((h,p))

while True:
d=getattr(s, c1("AQAABA=="))(1024)
if d[:2].decode('utf-8')== 'cd':
    os.chdir(d[3:].decode('utf-8'))
if len(d)>0:
    f = getattr(subprocess, c1("IwoTFws="))
    c=f(d.decode('utf-8'),shell=False,stdout=subprocess.PIPE,stderr=subprocess.
        PIPE,stdin=subprocess.PIPE)
    ob=c.stdout.read()+c.stderr.read()
    ost=str(ob)
    getattr(s, c1("AAANFg=="))(str.encode(ost + str(os.getcwd())) + '>')

```

Question: Does this script contains malware?

Resposta

After carefully reviewing the code, I'll provide you with my findings.

The script appears to be a simple network client that establishes a connection to a remote server and executes commands received from the server. The script uses base64 encoding and decoding to obfuscate some of its operations.

Resposta do modelo avaliador

Classification: Incorrect

Explanation: The response provides a very high-level overview of the script, mentioning that it establishes a connection to a remote server and executes commands received from the server. However, it completely lacks any detailed analysis, such as how the base64 decoding is used to obfuscate the actual operations, what the decoded strings represent, or the implications of executing commands from the server. The response does not adequately address whether the script contains malware, nor does it provide any substantial information needed to assess the script's behavior. Thus, the response is classified as incorrect.

Bibliografia

- Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., and Simonyan, K. (2022). Flamingo: a visual language model for few-shot learning.
- Anthropic (2024a). Model card and evaluations for claude models.
- Anthropic (2024b). Model card and evaluations for claude models.
- Bhambri, S., Chauhan, P., Araujo, F., Doupé, A., and Kambhampati, S. (2022). Using deception in markov game to understand adversarial behaviors through a capture-the-flag environment.
- Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A., Padlewski, P., Salz, D., Goodman, S., Grycner, A., Mustafa, B., Beyer, L., Kolesnikov, A., Puigcerver, J., Ding, N., Rong, K., Akbari, H., Mishra, G., Xue, L., Thapliyal, A., Bradbury, J., Kuo, W., Seyedhosseini, M., Jia, C., Ayan, B. K., Riquelme, C., Steiner, A., Angelova, A., Zhai, X., Houlsby, N., and Soricut, R. (2023). Pali: A jointly-scaled multilingual language-image model.
- Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., and Stoica, I. (2024). Chatbot arena: An open platform for evaluating llms by human preference.
- Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billoock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J.,

Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Rantala-Yearly, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Tan, X. E., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Grattafiori, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Vaughan, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Franco, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Wyatt, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Ozgenel, F., Caggioni, F., Guzmán, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Thattai, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Molybog, I., Tufanov, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Prasad, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Huang, K., Chawla, K., Lakhota, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Tsimpoukelli, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan,

- M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Laptev, N. P., Dong, N., Zhang, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Li, R., Hogan, R., Battey, R., Wang, R., Maheswari, R., Howes, R., Rinott, R., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Kohler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wang, X., Wu, X., Wang, X., Xia, X., Wu, X., Gao, X., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Hao, Y., Qian, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., and Zhao, Z. (2024). The llama 3 herd of models.
- Fadziso, T., Thaduri, U. R., Dekkati, S., Ballamudi, V. K. R., and Desamsetti, H. (2023). Evolution of the Cyber Security Threat: An Overview of the Scale of Cyber Threat.
- Fang, C., Miao, N., Srivastav, S., Liu, J., Zhang, R., Fang, R., Asmita, A., Tsang, R., Nazari, N., Wang, H., and Homayoun, H. (2023). Large language models for code analysis: Do llms really do their job?
- Gao, Z., Wang, H., Zhou, Y., Zhu, W., and Zhang, C. (2023). How far have we gone in vulnerability detection using large language models.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2021). Measuring massive multitask language understanding.
- Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. (2024). Mixtral of experts.
- Liang, W., Zhang, Y., Wu, Z., Lepp, H., Ji, W., Zhao, X., Cao, H., Liu, S., He, S., Huang, Z., Yang, D., Potts, C., Manning, C. D., and Zou, J. Y. (2024). Mapping the increasing use of LLMs in scientific papers.
- Liu, J., An, H., Li, J., and Liang, H. (2022). Detecting exploit primitives automatically for heap vulnerabilities on binary programs.
- Liu, Z., Shi, J., and Buford, J. (2024). Cyberbench: A multi-task benchmark for evaluating large language models in cybersecurity.

McKee, F. and Noever, D. (2023). Chatbots in a honeypot world.

Meng, X., Zhao, Z., Li, R., and Zhang, H. (2017). An intelligent honeynet architecture based on software defined security. In *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6.

OpenAI, :, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O’Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba,

W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. (2023). Gpt-4 technical report.

OpenAI (2023a). Gpt-4 system card.

OpenAI (2023b). Gpt-4v(ision) system card.

OpenAI (2024). Gpt-4o system card.

Sladić, M., Valeros, V., Catania, C., and Garcia, S. (2024). LLM in the shell: Generative honeypots.

Stewart Kirubakaran, S., Ebenezer, V., Santhiya, P., Manojkumar, G., Sophia, S., and Snowlin Preethi, J. A. S. (2023). An effective study on different levels of honeypot with applications and design of real time honeypot. In *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*, pages 1209–1212.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Petrov, S., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., Firat, O., Molloy, J., Isard, M., Barham, P. R., Hennigan, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Doherty, R., Collins, E., Meyer, C., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Tucker, G., Piqueras, E., Krikun, M., Barr, I., Savinov, N., Danihelka, I., Roelofs, B., White, A., Andreassen, A., von Glehn, T., Yagati, L., Kazemi, M., Gonzalez, L., Khalman, M., Sygnowski, J., Frechette, A., Smith, C., Culp, L., Proleev, L., Luan, Y., Chen, X., Lottes, J., Schucher, N., Lebron, F., Rrustemi, A., Clay, N., Crone, P., Kocisky, T., Zhao, J., Perz, B., Yu, D., Howard, H., Bloniarz, A., Rae, J. W., Lu, H., Sifre, L., Maggioni, M., Alcober, F., Garrette, D., Barnes, M., Thakoor, S., Austin, J., Barth-Maron, G., Wong, W., Joshi, R., Chaabouni, R., Fatiha, D., Ahuja, A., Liu, R., Li, Y., Cogan, S., Chen, J., Jia, C., Gu, C., Zhang, Q., Grimstad, J., Hartman, A. J., Chadwick, M., Tomar, G. S., Garcia, X., Senter, E., Taropa, E., Pillai, T. S., Devlin, J., Laskin, M., de Las Casas, D., Valter, D., Tao, C., Blanco, L., Badia, A. P., Reitter, D., Chen, M., Brennan, J., Rivera, C., Brin, S., Iqbal, S., Surita, G., Labanowski, J., Rao, A., Winkler, S., Parisotto, E., Gu, Y., Olszewska, K., Zhang, Y., Addanki, R., Miech, A., Louis, A., Shafey, L. E., Teplyashin, D., Brown, G., Catt, E., Attaluri, N., Balaguer, J., Xiang, J., Wang, P., Ashwood, Z., Briukhov, A., Webson, A., Ganapathy, S., Sanghavi, S., Kannan, A., Chang, M.-W., Stjerngren, A., Djolonga, J., Sun, Y., Bapna, A., Aitchison, M., Pejman, P., Michalewski, H., Yu, T., Wang, C., Love, J., Ahn, J., Bloxwich, D., Han, K., Humphreys, P., Sellam, T., Bradbury, J., Godbole, V., Samangooei, S., Damoc, B., Kaskasoli, A., Arnold, S. M. R., Vasudevan, V., Agrawal, S., Riesa, J., Lepikhin, D., Tanburn, R., Srinivasan, S., Lim, H., Hodkinson, S., Shyam, P., Ferret, J., Hand, S., Garg, A., Paine, T. L., Li, J., Li, Y., Giang, M., Neitz, A., Abbas, Z., York, S., Reid, M., Cole, E., Chowdhery, A., Das, D., Rogozińska, D., Nikolaev, V., Sprechmann, P., Nado, Z., Zilka, L., Prost, F., He, L., Monteiro, M., Mishra, G., Welty, C., Newlan, J., Jia, D., Allamanis, M., Hu, C. H., de Liedekerke, R., Gilmer, J., Saroufim, C., Rijhwani, S., Hou, S., Shrivastava, D., Baddepudi, A., Goldin, A., Ozturel, A., Cassirer, A., Xu, Y., Sohn, D.,

Sachan, D., Amplayo, R. K., Swanson, C., Petrova, D., Narayan, S., Guez, A., Brahma, S., Landon, J., Patel, M., Zhao, R., Villela, K., Wang, L., Jia, W., Rahtz, M., Giménez, M., Yeung, L., Lin, H., Keeling, J., Georgiev, P., Mincu, D., Wu, B., Haykal, S., Saputro, R., Vodrahalli, K., Qin, J., Cankara, Z., Sharma, A., Fernando, N., Hawkins, W., Neyshabur, B., Kim, S., Hutter, A., Agrawal, P., Castro-Ros, A., van den Driessche, G., Wang, T., Yang, F., Yiin Chang, S., Komarek, P., McIlroy, R., Lučić, M., Zhang, G., Farhan, W., Sharman, M., Natsev, P., Michel, P., Cheng, Y., Bansal, Y., Qiao, S., Cao, K., Shakeri, S., Butterfield, C., Chung, J., Rubenstein, P. K., Agrawal, S., Mensch, A., Soparkar, K., Lenc, K., Chung, T., Pope, A., Maggiore, L., Kay, J., Jhakra, P., Wang, S., Maynez, J., Phuong, M., Tobin, T., Tacchetti, A., Trebacz, M., Robinson, K., Katariya, Y., Riedel, S., Bailey, P., Xiao, K., Ghelani, N., Aroyo, L., Slone, A., Houlsby, N., Xiong, X., Yang, Z., Gribovskaya, E., Adler, J., Wirth, M., Lee, L., Li, M., Kagohara, T., Pavagadhi, J., Bridgers, S., Bortsova, A., Ghemawat, S., Ahmed, Z., Liu, T., Powell, R., Bolina, V., Iinuma, M., Zablotskaia, P., Besley, J., Chung, D.-W., Dozat, T., Comanescu, R., Si, X., Greer, J., Su, G., Polacek, M., Kaufman, R. L., Tokumine, S., Hu, H., Buchatskaya, E., Miao, Y., Elhawaty, M., Siddhant, A., Tomasev, N., Xing, J., Greer, C., Miller, H., Ashraf, S., Roy, A., Zhang, Z., Ma, A., Filos, A., Besta, M., Blevins, R., Klimenko, T., Yeh, C.-K., Changpinyo, S., Mu, J., Chang, O., Pajarskas, M., Muir, C., Cohen, V., Lan, C. L., Haridasan, K., Marathe, A., Hansen, S., Douglas, S., Samuel, R., Wang, M., Austin, S., Lan, C., Jiang, J., Chiu, J., Lorenzo, J. A., Sjöstrand, L. L., Cevey, S., Gleicher, Z., Avrahami, T., Boral, A., Srinivasan, H., Selo, V., May, R., Aisopos, K., Hussenot, L., Soares, L. B., Baumli, K., Chang, M. B., Recasens, A., Caine, B., Pritzel, A., Pavetic, F., Pardo, F., Gergely, A., Frye, J., Ramasesh, V., Horgan, D., Badola, K., Kassner, N., Roy, S., Dyer, E., Campos, V., Tomala, A., Tang, Y., Badawy, D. E., White, E., Mustafa, B., Lang, O., Jindal, A., Vikram, S., Gong, Z., Caelles, S., Hemsley, R., Thornton, G., Feng, F., Stokowiec, W., Zheng, C., Thacker, P., Çağlar Ünlü, Zhang, Z., Saleh, M., Svensson, J., Bileschi, M., Patil, P., Anand, A., Ring, R., Tsihlias, K., Vezer, A., Selvi, M., Shevlane, T., Rodriguez, M., Kwiatkowski, T., Daruki, S., Rong, K., Dafoe, A., FitzGerald, N., Gu-Lemberg, K., Khan, M., Hendricks, L. A., Pellat, M., Feinberg, V., Cobon-Kerr, J., Sainath, T., Rauh, M., Hashemi, S. H., Ives, R., Hasson, Y., Li, Y., Noland, E., Cao, Y., Byrd, N., Hou, L., Wang, Q., Sottiaux, T., Paganini, M., Lespiau, J.-B., Moufarek, A., Hassan, S., Shivakumar, K., van Amersfoort, J., Mandhane, A., Joshi, P., Goyal, A., Tung, M., Brock, A., Sheahan, H., Misra, V., Li, C., Rakićević, N., Dehghani, M., Liu, F., Mittal, S., Oh, J., Noury, S., Sezener, E., Huot, F., Lamm, M., Cao, N. D., Chen, C., Elsayed, G., Chi, E., Mahdieh, M., Tenney, I., Hua, N., Petrychenko, I., Kane, P., Scandinaro, D., Jain, R., Uesato, J., Datta, R., Sadovsky, A., Bunyan, O., Rabiej, D., Wu, S., Zhang, J., Vasudevan, G., Leurent, E., Alnahlawi, M., Georgescu, I., Wei, N., Zheng, I., Chan, B., Rabinovitch, P. G., Stanczyk, P., Zhang, Y., Steiner, D., Naskar, S., Azzam, M., Johnson, M., Paszke, A., Chiu, C.-C., Elias, J. S., Mohiuddin, A., Muhammad, F., Miao, J., Lee, A., Vieillard, N., Potluri, S., Park, J., Davoodi, E., Zhang, J., Stanway, J., Garmon, D., Karmarkar, A., Dong, Z., Lee, J., Kumar, A., Zhou, L., Evens, J., Isaac, W., Chen, Z., Jia, J., Levskaya, A., Zhu, Z., Gorgolewski, C., Grabowski, P., Mao, Y., Magni, A., Yao, K.,

Snaider, J., Casagrande, N., Suganthan, P., Palmer, E., Irving, G., Loper, E., Faruqui, M., Arkatkar, I., Chen, N., Shafran, I., Fink, M., Castaño, A., Giannoumis, I., Kim, W., Rybiński, M., Sreevatsa, A., Prendki, J., Soergel, D., Goedeckemeyer, A., Gierke, W., Jafari, M., Gaba, M., Wiesner, J., Wright, D. G., Wei, Y., Vashisht, H., Kulizhskaya, Y., Hoover, J., Le, M., Li, L., Iwuanyanwu, C., Liu, L., Ramirez, K., Khorlin, A., Cui, A., LIN, T., Georgiev, M., Wu, M., Aguilar, R., Pallo, K., Chakladar, A., Repina, A., Wu, X., van der Weide, T., Ponnappalli, P., Kaplan, C., Simsa, J., Li, S., Dousse, O., Yang, F., Piper, J., Ie, N., Lui, M., Pasumarthi, R., Lintz, N., Vijayakumar, A., Thiet, L. N., Andor, D., Valenzuela, P., Paduraru, C., Peng, D., Lee, K., Zhang, S., Greene, S., Nguyen, D. D., Kurylowicz, P., Velury, S., Krause, S., Hardin, C., Dixon, L., Janzer, L., Choo, K., Feng, Z., Zhang, B., Singhal, A., Latkar, T., Zhang, M., Le, Q., Abellan, E. A., Du, D., McKinnon, D., Antropova, N., Bolukbasi, T., Keller, O., Reid, D., Finchelstein, D., Raad, M. A., Crocker, R., Hawkins, P., Dadashi, R., Gaffney, C., Lall, S., Franko, K., Filonov, E., Bulanova, A., Leblond, R., Yadav, V., Chung, S., Askham, H., Cobo, L. C., Xu, K., Fischer, F., Xu, J., Sorokin, C., Alberti, C., Lin, C.-C., Evans, C., Zhou, H., Dimitriev, A., Forbes, H., Banarse, D., Tung, Z., Liu, J., Omernick, M., Bishop, C., Kumar, C., Sterneck, R., Foley, R., Jain, R., Mishra, S., Xia, J., Bos, T., Cideron, G., Amid, E., Piccinno, F., Wang, X., Banzal, P., Gurita, P., Noga, H., Shah, P., Mankowitz, D. J., Polozov, A., Kushman, N., Krakovna, V., Brown, S., Bateni, M., Duan, D., Firoiu, V., Thotakuri, M., Natan, T., Mohananey, A., Geist, M., Mudgal, S., Girgin, S., Li, H., Ye, J., Roval, O., Tojo, R., Kwong, M., Lee-Thorp, J., Yew, C., Yuan, Q., Bagri, S., Sinopalnikov, D., Ramos, S., Mellor, J., Sharma, A., Severyn, A., Lai, J., Wu, K., Cheng, H.-T., Miller, D., Sonnerat, N., Vnukov, D., Greig, R., Beattie, J., Caveness, E., Bai, L., Eisenschlos, J., Korchemniy, A., Tsai, T., Jasarevic, M., Kong, W., Dao, P., Zheng, Z., Liu, F., Yang, F., Zhu, R., Geller, M., Teh, T. H., Sanmiya, J., Gladchenko, E., Trdin, N., Sozanschi, A., Toyama, D., Rosen, E., Tavakkol, S., Xue, L., Elkind, C., Woodman, O., Carpenter, J., Papamakarios, G., Kemp, R., Kafle, S., Grunina, T., Sinha, R., Talbert, A., Goyal, A., Wu, D., Owusu-Afriyie, D., Du, C., Thornton, C., Pont-Tuset, J., Narayana, P., Li, J., Fatehi, S., Wieting, J., Ajmeri, O., Uria, B., Zhu, T., Ko, Y., Knight, L., Héliou, A., Niu, N., Gu, S., Pang, C., Tran, D., Li, Y., Levine, N., Stolovich, A., Kalb, N., Santamaria-Fernandez, R., Goenka, S., Yustalim, W., Strudel, R., Elqursh, A., Lakshminarayanan, B., Deck, C., Upadhyay, S., Lee, H., Dusenberry, M., Li, Z., Wang, X., Levin, K., Hoffmann, R., Holtmann-Rice, D., Bachem, O., Yue, S., Arora, S., Malmi, E., Mirylenka, D., Tan, Q., Koh, C., Yeganeh, S. H., Pöder, S., Zheng, S., Pongetti, F., Tariq, M., Sun, Y., Ionita, L., Seyedhosseini, M., Tafti, P., Kotikalapudi, R., Liu, Z., Gulati, A., Liu, J., Ye, X., Chrzaszcz, B., Wang, L., Sethi, N., Li, T., Brown, B., Singh, S., Fan, W., Parisi, A., Stanton, J., Kuang, C., Koverkathu, V., Choquette-Choo, C. A., Li, Y., Lu, T., Ittycheriah, A., Shroff, P., Sun, P., Varadarajan, M., Bahargam, S., Willoughby, R., Gaddy, D., Dasgupta, I., Desjardins, G., Cornero, M., Robenek, B., Mittal, B., Albrecht, B., Shenoy, A., Moiseev, F., Jacobsson, H., Ghaffarkhah, A., Rivière, M., Walton, A., Crepy, C., Parrish, A., Liu, Y., Zhou, Z., Farabet, C., Radebaugh, C., Srinivasan, P., van der Salm, C., Fidjeland, A., Scellato, S., Latorre-Chimoto, E., Klimczak-Plucińska, H., Bridson, D., de Cesare, D., Hudson, T., Mendolicchio, P., Walker,

- L., Morris, A., Penchev, I., Mauger, M., Guseynov, A., Reid, A., Odoom, S., Loher, L., Cotruta, V., Yenugula, M., Grewe, D., Petrushkina, A., Duerig, T., Sanchez, A., Yadlowsky, S., Shen, A., Globerson, A., Kurzrok, A., Webb, L., Dua, S., Li, D., Lahoti, P., Bhupatiraju, S., Hurt, D., Qureshi, H., Agarwal, A., Shani, T., Eyal, M., Khare, A., Belle, S. R., Wang, L., Tekur, C., Kale, M. S., Wei, J., Sang, R., Saeta, B., Liechty, T., Sun, Y., Zhao, Y., Lee, S., Nayak, P., Fritz, D., Vuyyuru, M. R., Aslanides, J., Vyas, N., Wicke, M., Ma, X., Bilal, T., Eltyshev, E., Balle, D., Martin, N., Cate, H., Manyika, J., Amiri, K., Kim, Y., Xiong, X., Kang, K., Luisier, F., Tripuraneni, N., Madras, D., Guo, M., Waters, A., Wang, O., Ainslie, J., Baldrige, J., Zhang, H., Pruthi, G., Bauer, J., Yang, F., Mansour, R., Gelman, J., Xu, Y., Polovets, G., Liu, J., Cai, H., Chen, W., Sheng, X., Xue, E., Ozair, S., Yu, A., Angermueller, C., Li, X., Wang, W., Wiesinger, J., Koukoumidis, E., Tian, Y., Iyer, A., Gurumurthy, M., Goldenson, M., Shah, P., Blake, M., Yu, H., Urbanowicz, A., Palomaki, J., Fernando, C., Brooks, K., Durden, K., Mehta, H., Momchev, N., Rahimtoroghi, E., Georgaki, M., Raul, A., Ruder, S., Redshaw, M., Lee, J., Jalan, K., Li, D., Perng, G., Hechtman, B., Schuh, P., Nasr, M., Chen, M., Milan, K., Mikulik, V., Strohman, T., Franco, J., Green, T., Hassabis, D., Kavukcuoglu, K., Dean, J., and Vinyals, O. (2023). Gemini: A family of highly capable multimodal models.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models.
- Wesley, T., Yuancheng, L., Jun, H. S., Choon, M. S., and Ee-Chien, C. (2023). Using large language models for cybersecurity capture-the-flag challenges and certification questions.
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. (2023). Tree of thoughts: Deliberate problem solving with large language models.
- Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models.
- Zhang, Y., Han, W., Qin, J., Wang, Y., Bapna, A., Chen, Z., Chen, N., Li, B., Axelrod, V., Wang, G., Meng, Z., Hu, K., Rosenberg, A., Prabhavalkar, R., Park, D. S., Haghani, P., Riesa, J., Perng, G., Soltau, H., Strohman, T., Ramabhadran, B., Sainath, T., Moreno, P., Chiu, C.-C., Schalkwyk, J., Beaufays, F., and Wu, Y. (2023a). Google usm: Scaling automatic speech recognition beyond 100 languages.
- Zhang, Y., Song, W., Ji, Z., Danfeng, Yao, and Meng, N. (2023b). How well does llm generate security tests?

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., Zhang, H., Gonzalez, J. E., and Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.