LLMs e Engenharia de *Prompt* para Classificação Automatizada de Incidentes em SOCs

Alex Sandre Pinheiro Severo[1], **Douglas Paim Lautert**[1], **Gefté Alcantara de Almeida**[1]
Diego Kreutz[1], **Godinho Rodrigo**[2], **Lourenco A. Pereira Jr**[3], **Leandro M. Bertholdo**[4]

[1] Universidade Federal do Pampa (UNIPAMPA)

[2]VALE

[3]Instituto Tecnológico de Aeronáutica (ITA)

[4] Universidade Federal do Rio Grande do Sul (UFRGS)

{alexsevero,douglaslautert,geftealmeida}.aluno@unipampa.edu.br

diegokreutz@unipampa.edu.br

godinho.rodrigo@vale.com

ljr@ita.br

leandro.bertholdo@ufrgs.br

Resumo. *Neste trabalho, avaliamos o uso de LLMs e engenharias de prompt para automatizar a classificação de incidentes de segurança em SOCs, buscando acelerar a resposta e aprimorar a qualidade das ações (e.g., seleção de playbooks), o que pode reduzir o tempo de resposta das empresas. Testamos três estratégias de prompting (PHP, SHP, HTP) em quatro LLMs (Gemini 2, GPT-4, LLaMA 4 e Grok 3) usando dados reais de CSIRTs e SOCs de empresas brasileiras. Nossos resultados indicam que o GEMINI alcançou 92,27% de acurácia em relação às classificações humanas ao usar PHP, enquanto as outras combinações de técnicas e LLMs demonstraram alguma variabilidade, o que pode afetar sua confiabilidade em cenários sensíveis.*

1. Introdução

Segundo dados do CERT.br, em 2024 o Brasil registrou 516.556 incidentes de segurança reportados por *Security Operations Centers* (SOCs) e *Computer Security Incident Response Teams* (CSIRTs) de diversas organizações, incluindo o setor industrial, o que representa uma média de 9.933 ocorrências por semana. No primeiro quadrimestre de 2025, já foram contabilizados 169.820 incidentes, evidenciando uma tendência de crescimento [CERT.br 2025]. Esse cenário coloca os SOCs das grandes corporações frente a um desafio operacional constante, isto é, o alto volume diário de alertas exige capacidades avançadas de monitoramento e processos eficientes de triagem e resposta.

Para mitigar o problema, consideramos que um primeiro passo é usar soluções automatizadas para filtrar e analisar dados de eventos adversos. Isso se mostra uma estratégia com potencial para acelerar a categorização de incidentes e aumentar a eficiência operacional dos processos de resposta nos SOCs. Entre essas soluções, nós podemos destacar o uso de *Large Language Models* (LLMs), que demonstram capacidade de adaptação a tarefas especializadas por meio de engenharia de *prompt* (*Prompt Engineering*)

[Zhou et al. 2022], o que permite sua aplicação em domínios especializados, como no contexto de segurança cibernética [Ogundairo and Broklyn 2024, Nasution et al. 2025].

A escassez de dados rotulados, a ambiguidade semântica dos relatos de incidentes, a complexidade na definição de categorias e a variabilidade nos formatos e padrões de ataques representam desafios significativos para a automação da classificação de incidentes de segurança [Ibrishimova 2019]. A eficácia dos LLMs em classificações tão complexas depende criticamente da qualidade do método de instrução (engenharia de *prompt*) utilizado. Neste trabalho, damos um passo inicial na resolução desse problema por meio de uma avaliação sistemática de três engenharias de *prompt* (PHP, HTP, SHP) aplicadas a quatro arquiteturas de LLMs (Gemini 2, GPT-4, LLaMA 4 e Grok 3), visando à classificação automatizada de incidentes de segurança.

A engenharia de *prompts* é uma disciplina emergente focada na formulação estratégica de instruções para LLMs, visando guiar a geração de respostas mais relevantes, contextuais e consistentes. Essa técnica é especialmente útil em tarefas dependentes do contexto, como a categorização de incidentes de segurança, nas quais a ambiguidade dos relatos e a possibilidade de múltiplas interpretações tornam o processo propenso a erros. As diversas técnicas de engenharia de *prompting* promovem um refinamento iterativo das respostas do LLM, tornando-o mais sensível ao contexto e melhor alinhado às necessidades operacionais de ambientes como SOCs e CSIRTs.

Este trabalho busca investigar a integração entre diferentes engenharias de *prompt* com taxonomias estruturadas para classificação de incidentes de segurança. Desta forma, é possível avaliar as diferentes estratégias para categorização automatizada. O restante deste artigo está organizado em 4 seções: Técnicas de Engenharia de *Prompt* Seção 2 *Pipeline* de Classificação Automatizada (Seção 3), Resultados e Discussão (Seção 4) e Considerações Finais e Trabalhos Futuros (Seção 5).

2. Técnicas de Engenharia de *Prompt*

Com o avanço dos modelos de linguagem de grande porte (LLMs), novas técnicas de engenharia de *prompt* têm sido desenvolvidas com o objetivo de adaptar esses modelos a tarefas específicas. Este trabalho concentra-se na aplicação de três estratégias distintas: *Progressive-Hint Prompting* (PHP), *Hypothesis Testing Prompting* (HTP) e *Self-Hint Prompting* (SHP). As três técnicas foram selecionadas por apresentarem abordagens metodológicas complementares e independentes, favorecendo a interpretação semântica em contextos marcados por ambiguidade textual e relatos não estruturados, como aqueles fornecidos por CSIRTs e SOCs. A categorização baseada em taxonomias pré-definidas assegura o alinhamento com padrões internacionalmente reconhecidos, conferindo consistência e robustez às respostas esperadas.

Cada técnica selecionada apresenta características específicas: o PHP realiza um refinamento iterativo por meio de dicas progressivas (*hints*) geradas a partir das próprias respostas intermediárias do modelo, favorecendo a convergência semântica ao longo das interações. O HTP, por sua vez, estrutura o processo de classificação como uma testagem sistemática de hipóteses verdadeiras e falsas para cada categoria da taxonomia, simulando um raciocínio dedutivo formal. Já o SHP induz ciclos de autorreflexão no modelo, nos quais ele analisa, valida e corrige suas próprias inferências, buscando coerência interna e minimizando erros.

Como complemento ao artigo principal, foi elaborado um documento técnico auxiliar contendo detalhes dessas três abordagens, incluindo pseudocódigos, explicações conceituais, exemplos de *prompts* e orientações para a reprodução dos experimentos. Esse material está disponível no repositório oficial do projeto, no GitHub[1].

O repositório também disponibiliza um conjunto de cinco incidentes anonimizados como exemplo, além de instruções completas para reprodução experimental. Cabe ressaltar que os 194 incidentes utilizados no estudo, embora tenham sido anonimizados para fins de análise controlada e interação com LLMs, ainda não estão aptos para publicação aberta. A liberação irrestrita desses registros, em sua forma atual, pode representar riscos de identificação ou exposição indireta de padrões operacionais sensíveis, especialmente se explorados por adversários com técnicas de inferência avançadas. Por essa razão, a disponibilização completa do conjunto de dados dependerá de etapas adicionais de revisão e validação ética quanto à privacidade e à segurança da informação. Os leitores interessados nos 194 incidentes podem entrar em contato com os autores.

3. *Pipeline* de Classificação Automatizada

Conforme ilustrado na Figura 1, a solução foi estruturada em um *pipeline* de cinco etapas (Dados de Entrada, Pré-processamento, Processamento, Análise dos resultados e Saída de dados) com o objetivo de avaliar de forma automática e sistemática a acurácia e o desempenho das estratégias de *prompting* PHP, SHP e HTP aplicadas a LLMs como Gemini 2, Grok 3, LLaMA 4 e GPT-4 para a categorização automatizada de incidentes de segurança.

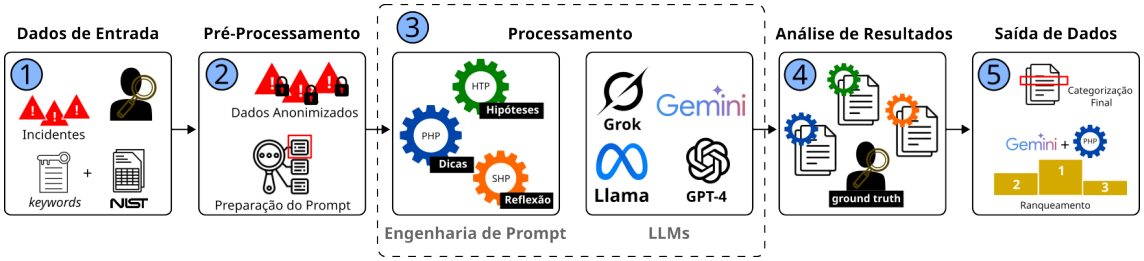

Figura 1. Fluxograma das etapas do *pipeline* da solução proposta

Dados de Entrada

Os relatos de incidentes utilizados neste estudo foram enviados por CSIRTs e SOCs por meio de *e-mails* institucionais, como parte das rotinas de compartilhamento e coordenação. Cada *e-mail* contém a descrição de um evento adverso, redigido por analistas humanos com base em registros coletados em suas respectivas infraestruturas. Embora o conteúdo seja claramente derivado de ferramentas de segurança (como EDRs, *firewalls* ou SIEMs), os *e-mails* recebidos não trazem metadados que identifiquem explicitamente a origem técnica de cada *log*. Nesse sentido, os incidentes analisados refletem a interpretação dos analistas de segurança, sendo o foco deste trabalho a categorização das informações relatadas, e não dos artefatos técnicos originais.

[1] https://github.com/AI-Horizon-Labs/SecLINC

Ao todo, 327 *reports* foram inicialmente recebidos por *e-mail* por um CSIRT de integração. Após triagem manual, 194 relatos foram identificados como incidentes de segurança e selecionados para análise. Esses registros passaram por um processo de anonimização utilizando a ferramenta AnonLFI[2], garantindo a proteção de informações sensíveis.

Posteriormente, os incidentes foram categorizados com base na taxonomia e recomendações da NIST SP 800-61r3 [Nelson et al. 2025]. Para a geração dos *prompts*, além do relato do incidente, foram utilizadas palavras-chave extraídas da tabela da NIST. Para cada categoria, buscou-se identificar, em diferentes LLMs, termos relevantes para auxiliar na análise e categorização. Além disso, a categorização foi realizada previamente por dois especialistas em cibersegurança, que analisaram os incidentes de forma independente, estabelecendo uma linha de base (*ground truth*) para comparação com os resultados automatizados.

Pré-processamento: compreende duas ações, a primeira os dados selecionados são anonimizados por uma ferramenta específica, denominada AnonLFI, para garantir a segurança da informação sobre as informações sensíveis. Em seguida, ocorre a formulação do *prompt* conforme a técnica selecionada (PHP, SHP ou HTP), estruturando os dados como entrada inicial a ser processada pela LLM. Para garantir uma avaliação comparativa rigorosa, a estrutura básica dos *prompts* contendo os incidentes foi sistematicamente aplicada a todas as LLMs avaliadas.

Processamento: implementa as diferentes técnicas de engenharia de *prompt* (PHP, SHP ou HTP) em um *pipeline* unificado. Este *pipeline* é responsável por gerar as entradas na LLM e controlar o fluxo de processamento para análise e categorização dos incidentes. O PHP [Zheng et al. 2023] é implementado por dicas (*hints*) incrementais usando a resposta anterior obtida do *prompt* para a obtenção da resposta final. Enquanto o SHP [Chen et al. 2024] gera planos intermediários que induzem o modelo a refletir sobre como resolver o problema de classificação, o HTP [Li et al. 2024] realiza uma iteração para cada categoria da taxonomia NIST, combinada com suas palavras-chave, formulando hipóteses (verdadeira e falsa) e, ao final, comparando-as em um *prompt* final que determina a categoria mais adequada. Cada técnica de engenharia de *prompt* é então aplicada às quatro LLMs (e versões) avaliadas: Gemini 2 (gemini-2.0-flash), GPT-4 (GPT-4o Mini), LLaMA 4 (llama4-scout) e Grok 3 (grok-3-mini-beta). Os modelos foram acessados por meio de suas respectivas *Application Programming Interfaces* (APIs), utilizando os parâmetros padrão recomendados pelos provedores. Para aplicações relacionadas a LLMs ou SLMs foram utilizados as dependências em *python* relacionadas ao Huggingface e OpenAI.

Análise dos resultados: nesta etapa, os resultados das classificações advindas das LLMs foram comparados com as análises realizadas por especialistas humanos. É importante ressaltarmos que o *ground truth* utilizado na avaliação foi estabelecido por dois especialistas humanos, profissionais com experiência prática em classificação de incidentes de segurança, garantindo um referencial confiável para comparação.

Saída de dados: cada incidente recebe uma classificação final, determinando o melhor resultado dentro das categorias estabelecidas pela tabela NIST. Adicionalmente, é

[2]`https://github.com/gt-rnp-lfi/anon`

realizado também um ranqueamento das LLMs e das engenharias de *prompt* com base no desempenho observado. Para isso, utilizamos uma abordagem binária, permitindo medir o grau de correspondência entre as categorias atribuídas pelos modelos e as classificações humanas. Ao final da categorização, o ciclo retorna à origem com o incidente rotulado, onde a organização, CSIRT ou SOC tem condições de iniciar os protocolos de mitigação que estejam definidos.

4. Resultados e Discussão

Nesta seção, apresentamos os resultados da avaliação experimental das três estratégias de *prompting* aplicadas a Modelos de Linguagem (LLMs), com o objetivo de identificar padrões de desempenho e analisar como as estratégias de *prompt* e as taxonomias estruturadas afetam a qualidade da classificação.

A Figura 2 ilustra o mapa de calor com os resultados da classificação de 194 incidentes de segurança, destacando os acertos absolutos e as taxas de acurácia. Nossos resultados indicam que as estratégias de *prompting* estruturadas (PHP e SHP), especialmente combinadas com os modelos Gemini 2 e GPT-4, alcançaram as maiores taxas de acurácia, o que demonstra sua eficácia na classificação de incidentes.

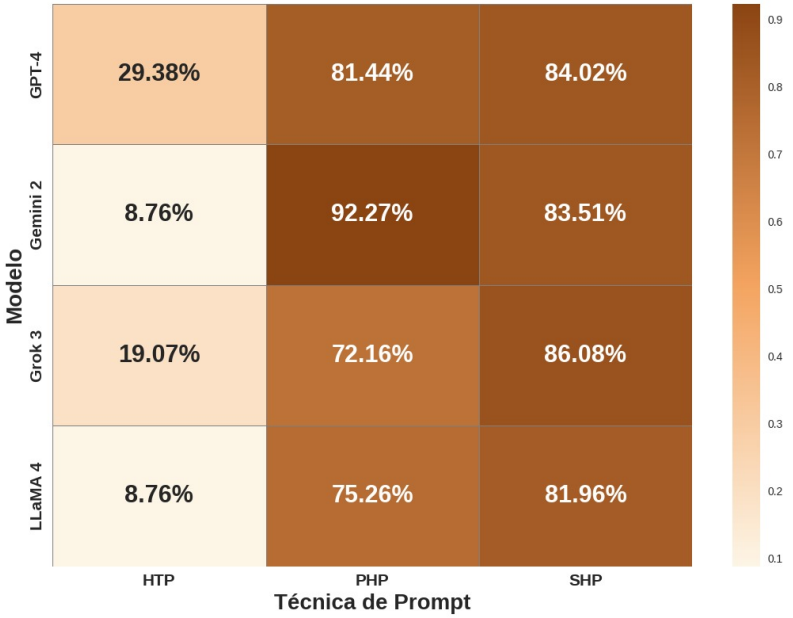

Figura 2. Resultados percentuais da classificação automatizada versus humana

A técnica HTP apresentou acurácia criticamente baixa (8,76%–29,38%) e elevada variabilidade entre modelos, evidenciando sua incompatibilidade com tarefas de classificação de incidentes de segurança. Originalmente desenvolvida para domínios lógicos estruturados, essa abordagem exige que o modelo teste explicitamente hipóteses binárias (verdadeira e falsa) para cada categoria. No entanto, os relatos de incidentes de CSIRTs são predominantemente narrativos, sem premissas lógicas claras, o que gera sobrecarga cognitiva: os modelos desperdiçam *tokens* com inferências irrelevantes e acabam truncando partes críticas do contexto. A variação de acurácia entre modelos — 29,38% no GPT-4 vs. 8,76% no LLaMA — reflete disparidades na capacidade de retenção de contexto e raciocínio lógico, com o GPT-4 relativamente menos impactado por sua arquitetura

mais avançada.

O SHP manteve desempenho estável nos modelos LLaMA 4 e Grok 3 (82% e 86%), em função de sua abordagem iterativa de auto-reflexão. Ao substituir testes hipotéticos formais por ciclos de auto-crítica, o SHP reduz a complexidade lógica da tarefa. Esses modelos, embora limitados em raciocínio abstrato, conseguem identificar contradições textuais básicas e corrigir erros grosseiros em poucas iterações. Trata-se de um processo menos dependente da capacidade de estabelecer conexões complexas ou prever consequências, sendo mais ancorado no contexto imediato da descrição do incidente.

O PHP alcançou acurácia excepcional no Gemini 2 (92,27%), mas oscilou significativamente em outros modelos como Grok 3 e GPT-4 (72,16%–81,44%). Essa variação está relacionada à própria natureza da engenharia de *prompt*, que utiliza *hints* sequenciais para refinar progressivamente a resposta, exigindo processamento eficiente de instruções encadeadas. Modelos mais consolidados no mercado tendem a ter uma maior acurácia em casos incrementais que envolvam incidentes de segurança, pois têm maior inteligência para tratar de *embeddings* textuais e treinamento.

O Gemini 2, otimizado para tarefas com instruções aninhadas, lida bem com essa estrutura. Em contraste, modelos como Grok 3 e LLaMA 4, com menor capacidade de contexto e atenção seletiva, tendem a confundir instruções adicionais com evidências previamente fornecidas ou a negligenciar comandos essenciais. Já o GPT-4, embora mais robusto, enfrenta perda de eficiência ao equilibrar múltiplas dicas com o contexto original, o que explica sua acurácia inferior ao Gemini 2.

Para implementações práticas, recomenda-se o SHP como base padrão para uso geral, enquanto PHP para sistemas com acesso ao Gemini, e evitar o uso do HTP em contextos narrativos ou não estruturados. A escolha ótima depende do equilíbrio entre acurácia máxima (PHP mais Gemini 2) e estabilidade entre modelos (SHP).

4.1. Tempo de Processamento

Para cada combinação de modelo e técnica, realizamos 10 testes, totalizando 120 execuções, todas em um processador Intel Core i7 de sétima geração. Como *baseline*, utilizamos o tempo de 240 minutos.

Essa estimativa de tempo para a classificação dos incidentes por analistas humanos baseia-se na experiência declarada pelos próprios profissionais envolvidos. Dois analistas especializados realizaram a categorização dos mesmos 194 incidentes de forma independente, ou seja, sem troca de informações ou discussão prévia. Embora a atividade não tenha sido cronometrada formalmente, ambos relataram ter despendido entre 3 e 4 horas para concluir o processo, que consistiu na leitura individual dos relatos e na classificação segundo a tabela de categorias da NIST, com base em julgamento técnico e conhecimento prévio.

Ambos os analistas possuem formação na área de Tecnologia da Informação, com especializações voltadas à estruturação e operação de CSIRTs e SOCs. São profissionais atuantes na área de segurança da informação, com experiência prática em análise, configuração e implementação de soluções em ambientes de monitoramento e resposta a incidentes. Apesar de atuarem no mesmo setor, apresentam níveis distintos de senioridade: um possui cerca de 10 anos de experiência direta na função, enquanto o outro atua

há aproximadamente 5 anos. A atividade de categorização foi conduzida de forma semelhante àquela realizada em seus ambientes profissionais, nos quais a análise de incidentes é conduzida individualmente, com base na expertise do analista.

Utilizamos ainda a mediana dos tempos como métrica central de comparação. Os resultados mostram que a técnica HTP apresentou o tempo médio mais elevado, especialmente com o modelo Grok 3 (cerca de 190 minutos), devido à sua abordagem hierárquica que exige múltiplas requisições à API. A técnica PHP, por sua vez, apresentou o menor tempo médio, com destaque para sua aplicação com o Gemini 2, que alcançou um tempo inferior a 25 minutos devido ao seu método eficiente e iterativo.

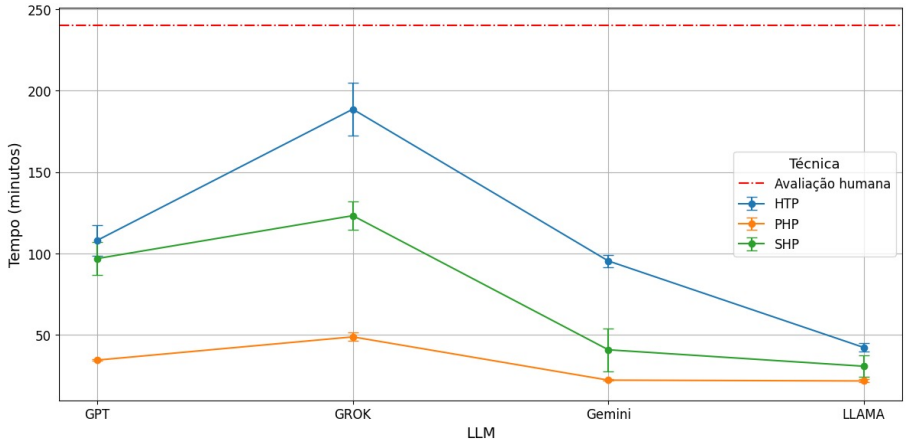

Figura 3. Mediana e desvio médio padrão de tempo para classificação automatizada

As variações entre os modelos também são significativas. Grok 3, embora robusto, teve o pior desempenho temporal com HTP, enquanto Gemini 2 demonstrou o melhor desempenho geral com PHP. O SHP, por sua vez, apresentou resultados intermediários em termos de tempo. Podemos concluir que técnicas como PHP e SHP são mais indicadas para cenários em que o tempo é um fator crítico, enquanto a técnica HTP deve ser desconsiderada em tarefas de classificação de SOCs, devido à sua baixa precisão e ao elevado tempo de execução. A análise reforça a importância de alinhar corretamente a escolha da LLM e da técnica de *prompting* ao objetivo do sistema.

4.2. *Tokens* Consumidos

Os LLMs apresentam variações significativas em suas janelas de contexto, impactando diretamente a capacidade de processar informações extensas e a eficiência na formulação de *prompts*, a Tabela 1 especifica a janela de execução com a definição de *tokens* para cada execução da LLM.

Tabela 1. Limites de *tokens* dos modelos de linguagem

Referência	Modelo	*Tokens* de Entrada	*Tokens* de Saída
[xAI 2024]	Grok 3	128.000	16.384
[Meta 2024]	LLaMA 4	10.000.000	8.192
[Google 2024]	Gemini 2	1.048.576	8.192
[OpenAI 2024]	GPT-4o	128.000 (combinados)	100.000

A Tabela 2 ilustra como a engenharia de *prompt* e a escolha da LLM influenciam diretamente o volume médio de *tokens* por interação, afetando custo e eficiência. LLMs

utilizando a PHP apresentaram volumes médios de *tokens* por interação mais baixos (entre 1.718 e 1.856), enquanto engenharias de *prompt* como SHP e HTP resultaram em volumes mais altos, chegando a 2.802 *tokens* por interação no caso do Grok 3 com SHP. Essa variação impacta diretamente no custo, visto que quanto maior for o volume de *tokens*, implica em maior processamento e, consequentemente, maior despesa. Por exemplo, o Gemini com SHP resultou em um custo de US$ 2,17, comparado a US$ 1,08 com PHP. Observamos que a diferença de acurácia é considerável. O PHP, além do custo menor, resultou em uma acurácia de 92%, quase 10% superior ao SHP.

Tabela 2. Comparativo de desempenho entre LLM, técnicas, volume de *tokens*, custo e acurácia

LLM	Eng. Prompt	Volume Tokens	Interações	Prompts	Custo (USD)	Acurácia
GEMINI	PHP	721.900	389	1.856	$1,08	92%
GROK 3	SHP	2.903.037	1.036	2.802	$2,90	86%
GPT-4	SHP	1.425.210	620	2.299	$2,85	84%
GEMINI	SHP	1.444.549	720	2.006	$2,17	84%
LLAMA 4	SHP	1.593.586	734	2.171	$0,80	82%
GPT-4	PHP	666.661	388	1.718	$1,33	81%
LLAMA 4	PHP	716.043	389	1.841	$0,36	75%
GROK 3	PHP	671.521	388	1.731	$0,67	72%
GPT-4	HTP	1.157.607	621	1.864	$2,32	29%
GROK 3	HTP	1.456.244	673	2.164	$1,46	19%
GEMINI	HTP	3.891.740	1.974	1.971	$5,84	9%
LLAMA 4	HTP	1.147.509	553	2.075	$0,57	9%

5. Considerações finais e Trabalhos Futuros

Com base em 194 relatos reais de incidentes e na taxonomia NIST SP 800-61r3, observamos que LLMs e estratégias de *prompting* podem acelerar a categorização, aumentar a precisão e reduzir ambiguidades. Das abordagens testadas, o PHP mostrou-se eficaz com modelos bem ajustados, enquanto o SHP demonstrou maior versatilidade. O HTP, por sua vez, apresentou limitações com dados narrativos não estruturados, levando a um desempenhos menos favoráveis. Isso indica que a seleção da técnica de *prompting* deve considerar a acurácia, o tipo de dado, o modelo e os custos operacionais. Nosso estudo sugere que a engenharia de *prompt* pode alcançar altos níveis de automação e acurácia na categorização de incidentes, o que pode reduzir o esforço manual e aumentar a resiliência cibernética, mesmo considerando desafios como a explicabilidade das decisões.

Como direções futuras, destacam-se: o treinamento de um modelo de linguagem compacto (SLM) com dados reais de incidentes; a integração de classificadores baseados em LLMs a *frameworks* SOAR; e a análise quantitativa do impacto dessa integração na redução do tempo médio de resposta e recuperação (MTTR) em ambientes industriais.

Agradecimentos. Esta pesquisa contou com apoio parcial da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), código de financiamento 001; da Rede Nacional de Ensino e Pesquisa (RNP), por meio do Programa Hackers do Bem e do GT LFI – *Learn From Incidents*; e da Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS), por meio dos editais 08/2023 e 09/2023, e dos termos de outorga 24/2551-0001368-7 e 24/2551-0000726-1.

Referências

CERT.br (2025). Incidentes notificados ao cert.br. https://stats.cert.br/incidentes/.

Chen, J., Tian, J., and Jin, Y. (2024). Self-hint prompting improves zero-shot reasoning in large language models via reflective cycle. In *Proceedings of the 46th Annual CCSS*.

Google (2024). Modelos gemini na vertex ai. https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-0-flash.

Ibrishimova, M. D. (2019). Cyber incident classification: Issues and challenges. In Xhafa, F., Leu, F.-Y., Ficco, M., and Yang, C.-T., editors, *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 469–477. Springer International Publishing.

Li, Y., Tian, J., He, H., and Jin, Y. (2024). Hypothesis testing prompting improves deductive reasoning in large language models. *arXiv preprint arXiv:2405.06707*.

Meta (2024). Llama 4: Advancing multimodal intelligence. https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Nasution, A. H., Monika, W., Onan, A., and Murakami, Y. (2025). Benchmarking 21 open-source large language models for phishing link detection with prompt engineering. *Information*, 16(5):366.

Nelson, A., Rekhi, S., Souppaya, M., and Scarfone, K. (2025). Incident response recommendations and considerations for cybersecurity risk management: A csf 2.0 community profile. Technical Report NIST SP 800-61r3, NIST.

Ogundairo, O. and Broklyn, P. (2024). Natural language processing for cybersecurity incident analysis. *Journal of Cyber Security*.

OpenAI (2024). Gpt-4o mini. https://platform.openai.com/docs/models/gpt-4o-mini.

xAI (2024). Grok-3: Next-generation model. https://x.ai/news/grok-3.

Zheng, Liu, X. et al. (2023). Progressive-hint prompting improves reasoning in large language models.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. (2022). Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*.

A. Apêndice: Tabela de Categorias de Incidentes adotada segundo orientações do NIST SP 800-61r3

Tabela 3. Categorização de Incidentes de Segurança (com prioridade)

Código	Categoria	Descrição	Prioridade
CAT1	Comprometimento de Conta	Acesso não autorizado a contas de usuários ou administradores.	5
CAT2	Malware	Infecção por código malicioso que compromete dispositivos ou dados.	5
CAT3	Ataque de Negação de Serviço (DoS/DDoS)	Tornar sistemas ou redes indisponíveis.	4
CAT4	Exfiltração ou Vazamento de Dados	Acesso, cópia ou divulgação não autorizada de dados sensíveis.	5
CAT5	Exploração de Vulnerabilidade	Uso de falhas conhecidas ou desconhecidas para comprometer ativos.	5
CAT6	Abuso Interno	Ações intencionais ou negligentes de usuários internos.	5
CAT7	Engenharia Social	Engano de pessoas para obter acesso ou informações.	3
CAT8	Incidente Físico ou de Infraestrutura	Violação física que impacta ativos computacionais.	4
CAT9	Alteração Não Autorizada	Modificação não autorizada em sistemas, dados ou configurações.	3
CAT10	Uso Indevido de Recursos	Uso não autorizado de sistemas para outros fins.	2
CAT11	Problema de Fornecedor/Terceiro	Incidente originado por falha de segurança de terceiros.	4
CAT12	Tentativa de Intrusão	Tentativas hostis de invasão ainda não confirmadas como bem-sucedidas.	3

Fonte: NIST (National Institute of Standards and Technology)