UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

# Detecção de Phishing sob Ataques Adversariais

## Avaliação de Robustez de Modelos Textuais e Visuais

Enzo Rodrigues Novais Dias

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

ENZO RODRIGUES NOVAIS DIAS

# Detecção de Phishing sob Ataques Adversariais

Avaliação de Robustez de Modelos Textuais e Visuais

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

**1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)**

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): ENZO RODRIGUES NOVAIS DIAS

Título do trabalho: Detecção de Phishing sob Ataques Adversariais

Avaliação de Robustez de Modelos Textuais e Visuais

**2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento** [ X ] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

**Casos de embargo:**

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

**Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.**

Documento assinado eletronicamente por **Enzo Rodrigues Novais Dias**, **Discente**, em 04/02/2026, às 16:38, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:30, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

ENZO RODRIGUES NOVAIS DIAS

# Detecção de Phishing sob Ataques Adversariais

Avaliação de Robustez de Modelos Textuais e Visuais

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

DIAS, ENZO RODRIGUES NOVAIS
Detecção de Phishing sob Ataques Adversariais [manuscrito]: Avaliação de Robustez de Modelos Textuais e Visuais / ENZO RODRIGUES NOVAIS DIAS. - 2025.
94 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação ) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Cibersegurança. 3. Phishing.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

ENZO RODRIGUES NOVAIS DIAS

**Detecção de Phishing sob Ataques Adversariais**

Avaliação de Robustez de Modelos Textuais e Visuais

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Antonio Carlos de Oliveira Júnior
(INF-UFG)

ENZO RODRIGUES NOVAIS DIAS

# Detecção de Phishing sob Ataques Adversariais

Avaliação de Robustez de Modelos Textuais e Visuais


## RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Análise Adversarial de Sistemas de Detecção de Phishing**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.


Palavras-chave: Inteligência artificial; Cibersegurança; Phishing .


## ABSTRACT

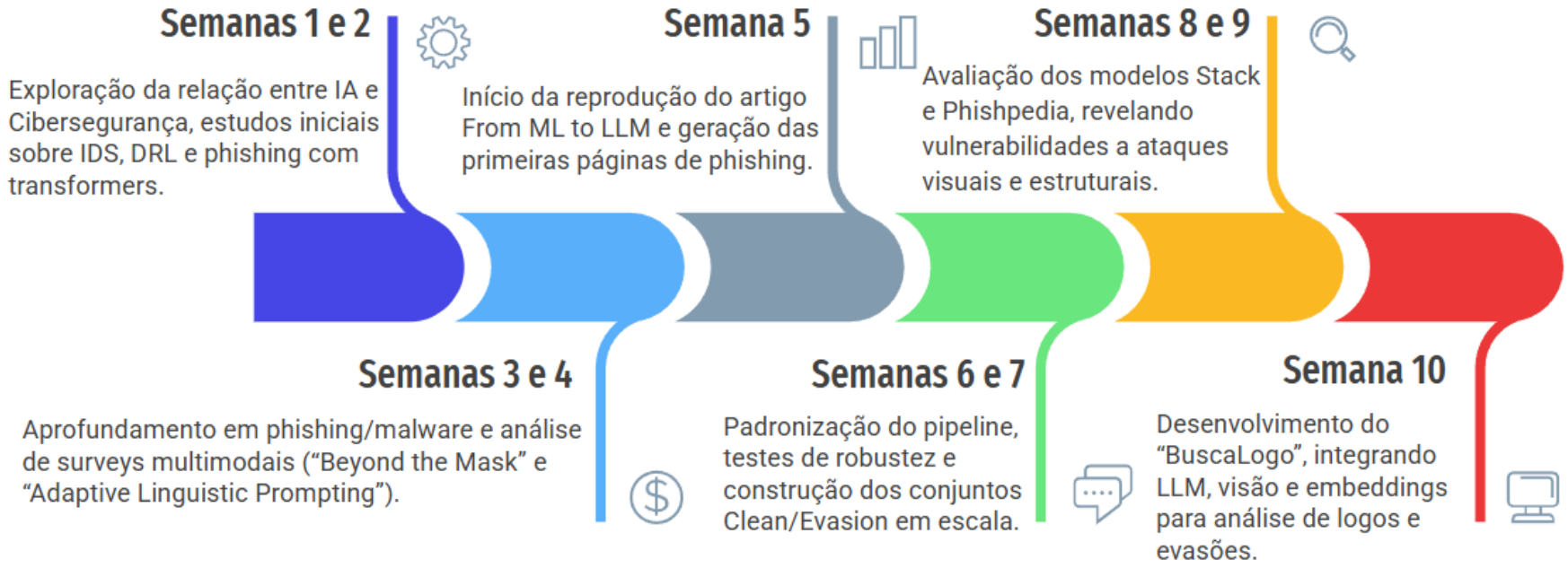
This Course Completion Report aims to bring together the results of my journey to become an expert in **Adversarial Analysis of Phishing Detection Systems**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.


Keywords: Artificial intelligence; Cybersecurity; Phishing.

Goiânia

2025

# Minha Jornada




Enzo Rodrigues Novais Dias

Especialista em: Análise Adversarial de Sistemas de Detecção de Phishing

# MINHA JORNADA

**Nome:** **Enzo Rodrigues Novais Dias**

**Especialidade:** **Análise Adversarial de Sistemas de Detecção de Phishing**

## Objetivo deste documento

Durante o processo da disciplina Residência em IA, foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

## Minha Jornada

Minha jornada teve início na **Semana 1**, quando comecei a explorar de forma sistemática as interseções entre Inteligência Artificial (IA) e Cibersegurança. A partir da análise das chamadas dos congressos do CSCI'2025 e, especialmente, do SAM'25, percebi a amplitude dos temas possíveis e a complexidade da escolha. Esse processo, somado ao resgate dos conteúdos já estudados ao longo do Bacharelado, me permitiu reconhecer que as minhas inclinações estavam relacionadas a sistemas inteligentes submetidos a pressões adversariais, ou seja, modelos que precisam operar em ambientes onde atacantes exploram fragilidades sutis. A partir dessas reflexões, ficou claro para mim claro que gostaria de seguir o caminho da detecção de phishing, um campo em que teoria e prática se cruzam de maneira intensa. Os primeiros materiais consultados, juntamente com minhas anotações e observações iniciais, foram reunidos no **Apêndice 1.** Na **Semana 2**, aprofundei esse direcionamento ao estudar o conceito de Intrusion Detection Systems baseados em Deep Reinforcement Learning, além de analisar o modelo URLTran, fundamentado na arquitetura transformers para detecção de URLs maliciosas. Durante esse período, foi possível observar que o problema vai muito além de simplesmente classificar páginas: trata-se de resistir a

técnicas de evasão cada vez mais sofisticadas. Pequenas alterações em caracteres, parâmetros ou estruturas HTML podem comprometer profundamente a performance de modelos tradicionais de detecção. Esses insights formaram a base de discussões posteriores e encontram-se descritos, com exemplos e comentários adicionais também no **Apêndice 1**.

A **Semana 3** foi dedicada para ampliar meu entendimento sobre o panorama atual da área. Ao estudar um survey sobre detecção de malware com Deep Learning e levantar tendências modernas em phishing detection, como multimodalidade, uso de grafos, LLMs semânticos e metodologias adversariais, tornou-se possível identificar melhor onde estão as grandes lacunas do campo de estudo. Já na **Semana 4**, com leituras de surveys, como *Beyond the Mask e Adaptive Linguistic Prompting (ALP)*, percebi de forma mais "concreta" a sensibilidade de modelos multimodais a pequenas mudanças na entrada, especialmente relacionadas à análise de URLs e HTML. Esses materiais, assim como as relações que estabeleci entre esses conhecimentos, estão organizados no **Apêndice 2**.

A **Semana 5** marcou o início da etapa experimental da minha jornada. Passei a reproduzir o artigo *From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks*, iniciando o estudo completo do PhishOracle, ferramenta de detecção de phishing baseada em IA. Durante esse período, minhas principais atividades envolveram compreender as estruturas HTML manipuladas pelo *pipeline*, replicar as técnicas de injeção adversarial e gerar minhas primeiras páginas Clean e Evasion. Essa etapa inaugurou uma transição importante: deixei o campo exclusivamente conceitual para trabalhar com dados reais, erros reais e decisões experimentais concretas. Os registros dessa fase, incluindo as primeiras versões de páginas, dificuldades encontradas e ajustes feitos no *pipeline*, foram compilados no **Apêndice 3**. Na **Semana 6**, dediquei-me a melhorar o *pipeline* experimental. Padronizei diretórios, implementei tratamentos para imagens RGBA, avaliei tempos de geração por meio de CDFs e identifiquei casos de overfitting, corrigidos por GroupShuffleSplit para evitar vazamentos. Foi uma **Semana** importante, pois tornou claro que a reprodutibilidade experimental é tão essencial quanto os próprios modelos. Já na **Semana 7**, ampliei o volume de dados ao realizar scraping da lista

Tranco Top 1000, enfrentando bloqueios automatizados, erros de parsing e timeouts que são típicos de coletas em ambiente real. Nesse período, percebi como pequenas decisões, como reintentar requisições, padronizar salvamentos e lidar com ativos indisponíveis, influenciam diretamente a integridade dos experimentos. Todos esses materiais e resultados práticos, acompanhados de suas justificativas metodológicas, encontram-se também no **Apêndice 3.**

A **Semana 8** marcou uma mudança significativa: a fase de avaliação de modelos. Ao testar o Stack Model, inicialmente observei performances surpreendentemente altas, que não condiziam com o comportamento esperado. Enquanto na **Semana 6** eu buscava apenas estruturar o *pipeline*, na **Semana 8** percebi que algumas funções de ataque não estavam ativadas, causando resultados artificiais. Depois da correção, os resultados passaram a refletir o que a literatura reporta, evidenciando a vulnerabilidade do modelo a ataques estruturais. Essa **Semana** também deu início aos experimentos envolvendo adulterações visuais no EvasionSet², revelando que modelos orientados apenas a conteúdo textual têm dificuldade em lidar com variantes visuais. Esses resultados e suas análises estão reunidos no **Apêndice 4**. Durante o período da **Semana 9**, reimplementei completamente o Phishpedia, um detector visual baseado na identificação de logos. Treinei um modelo ResNet50 com o dataset Logo2K+, construí minha própria brand gallery e enfrentei um bug crítico relacionado a ataques recursivos, que gerou centenas de milhares de versões duplicadas de logos adulterados. Foi neste período que percebi, mais claramente, como modelos visuais podem ser frágeis quando confrontados com transformações sutis, e como o *pipeline* precisa ser cuidadosamente projetado para evitar distorções experimentais. A documentação dessa reconstrução completa, incluindo capturas de tela, scripts e resultados também pode ser encontrada também no **Apêndice 4**.

A **Semana 10** encerrou a jornada com um resultado que sintetiza tudo o que foi construído: o desenvolvimento do BuscaLogo, um *pipeline* multimodal voltado para analisar genericidade, similaridade e detecção visual de logos, um dos componentes mais sensíveis na detecção multimodal de phishing. Esse sistema integra três módulos independentes: (1) um classificador de genericidade via LLM, (2) um comparador visual por embeddings e (3) um *pipeline* de detecção por modelos YOLO e OpenVINO conectados a buscas externas.

Durante essa **Semana**, pude observar de forma muito clara como todo o conhecimento acumulado, desde multimodalidade até ataques adversariais, convergiu para um sistema funcional. O conjunto dessa etapa, incluindo exemplos e scripts finais, está documentado no **Apêndice 5**.

Ao observar todo esse percurso, percebo que minha jornada não foi linear, mas ainda sim evolutiva. Cada **Semana** trouxe compreensões novas e reposicionou minhas expectativas sobre o campo. Compreendi que phishing detection é um problema essencialmente multimodal e adversarial, que exige rigor metodológico, sensibilidade às nuances e atenção às pequenas transformações que podem alterar completamente um resultado. Finalizo estas dez **Semanas** com uma compreensão sólida da área, diversos pipelines implementados e o desejo de continuar explorando os desafios que emergem quando IA e Segurança Digital se encontram de maneira tão profunda.

# APÊNDICE 1

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 3 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

**Enzo Rodrigues Novais Dias**

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Em minha primeira Semana, foi feito um estudo sobre os tópicos de interesses segundo o CSCI'2025, em especial a The 24th International Conference on Security & Management (SAM'25).

Após análises, optei pela **Cibersegurança**, uma área de grande relevância, mas assim como a Inteligência Artificial, extremamente ampla. Por isso, busquei compreender as intersecções entre essas duas áreas, entendendo mais sobre:

- Referências e trabalhos clássicos relacionados ao tema
- Autores e grupos relevantes
- Principais áreas de intersecção:
  - IA como ferramenta de defesa cibernética
  - IA como vetor de ameaça
  - Segurança das IAs

A documentação detalhada desses tópicos pode ser encontrada aqui: Documento da Semana 01

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Aprofundar o estudo sobre IA como ferramenta de defesa cibernética, com foco em estudar e produzir um fichamento dos seguintes materiais:

- Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey - aprendizado por reforço profundo aplicado para detectar intrusões em redes IoT
- A Survey of Malware Detection Using Deep Learning - deep learning para identificar malwares
- URLTran: Improving Phishing URL Detection Using Transformers - Transformers para identificar URLs maliciosas

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

[Documento da Semana 1 - citado no Termo de Aceite de Entrega de 3 de Setembro]

Na primeira semana, o objetivo foi realizar um estudo sobre as áreas de intersecção entre **Inteligência Artificial (IA)** e **Cibersegurança**, identificando aplicações, riscos, trabalhos relacionados e autores de referência. Esse estudo demonstrou que essas duas disciplinas estão bastante conectadas: a IA pode tanto fortalecer mecanismos de defesa quanto criar novas superfícies de ataque, exigindo governança, ética e práticas seguras de implementação.

**IA como ferramenta de defesa**

A literatura mostra que técnicas de aprendizado de máquina vêm sendo aplicadas à detecção de intrusões, à identificação de malwares e à análise de tentativas de fraude. Pesquisas apontam que esses métodos oferecem ganhos significativos de precisão, mas ainda sofrem com desafios como falsos positivos e falta de padronização em datasets.

- Buczak & Guven (2016) — A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection
- Zhang et al. (2025) — Deep Learning Applications in IDS

**IA como vetor de ameaça**

O mesmo potencial da IA também é explorado por atacantes. Modelos generativos são usados para criar *deepfakes* realistas, automatizar campanhas de phishing e desenvolver malwares polimórficos, ampliando a sofisticação das ameaças. Autoridades como o FBI têm emitido alertas sobre golpes que utilizam vozes e imagens sintéticas em fraudes corporativas.

- FBI IC3 Alert (2024) — *Deepfake and Voice Cloning Fraud:* Internet Crime Complaint Center (IC3) | Criminals Use Generative Artificial Intelligence to Facilitate Financial Fraud

**Segurança das próprias IAs**

Modelos de IA são vulneráveis a ataques como *data poisoning*, *model extraction* e *prompt injection*. Nicolas Papernot e Patrick McDaniel são referências nesse campo, tendo demonstrado como exemplos adversariais podem manipular classificadores de redes neurais.

- Papernot et al. (2016) — The Limitations of Deep Learning in Adversarial Settings.

**Governança, ética e conformidade**

Diante dos riscos, órgãos internacionais têm proposto frameworks e legislações. O **NIST AI RMF** (2023) e o **Generative AI Profile (2024)** estabelecem diretrizes de governança e mitigação de riscos técnicos e sociais.

- NIST AI RMF 1.0: DOI 10.6028/NIST.AI.100-1. Disponível em: Artificial Intelligence Risk Management Framework (AI RMF 1.0)
- NIST Generative AI Profile (2024): DOI 10.6028/NIST.AI.600-1. Disponível em: Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile

**Privacidade e resiliência**

Com a evolução dos ataques, cresce a preocupação com violações de privacidade (ex.: *membership inference attacks*) e com a continuidade de operações críticas. Técnicas de *privacy-preserving ML* (como *federated learning*) são propostas para proteger dados sensíveis, mas ainda enfrentam desafios de escalabilidade e confiabilidade.

**Trabalhos relacionados**

- Sommer & Paxson (2010) — *Outside the Closed World: On Using Machine Learning for Network Intrusion Detection*
- Buczak & Guven (2016) — A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection
- Berman et al. (2019) — *A Survey of Deep Learning Methods for Cyber Security*
- Zhang et al. (2025) — Deep Learning Applications in IDS
- Papernot et al. (2016) — The Limitations of Deep Learning in Adversarial Settings.

**Autores e grupos relevantes**

- **James P. Anderson** — pioneiro em auditoria e detecção de intrusões, considerado um dos pais da segurança de computadores.
- **Whitfield Diffie & Martin Hellman** — inventores da criptografia de chave pública.
- **Richard Sommer & Vern Paxson** — críticos iniciais sobre ML em segurança de redes.
- **Andrew Buczak & Erhan Guven** — autores de survey clássico em ML para IDS.
- **Patrick McDaniel & Nicolas Papernot** — principais nomes em adversarial machine learning.
- **MITRE ATLAS Team** — criadores da base de conhecimento sobre ataques a IA.

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 10 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante a segunda Semana, o objetivo foi ler e produzir um fichamento de um um survey que revisa o uso de DRL em IDS para IoT e de um trabalho que propõe o modelo URLTran para detecção de phishing com transformers.

- **Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey**

Os IDS funcionam como uma segunda linha de proteção além dos firewalls. Enquanto os firewalls estabelecem barreiras de acesso, os IDS têm a função de identificar comportamentos anômalos ou tráfego malicioso que consiga atravessar essas barreira
Possui três linhas de aplicação principais:

* **IDS em Wireless Sensor Networks (WSN). -** detecção de ataques como blackhole e flooding, que exploram vulnerabilidades no roteamento de dados entre sensores.
*** IDS baseados em Deep Q-Networks (DQN/DDQN). -** aplica arquiteturas populares para classificar tráfego entre benigno e malicioso
*** IDS aplicados à Saúde (Healthcare IoT). -** IoMT (Internet of Medical Things). Aplicado para detectar tráfego anômalo em dispositivos médicos.

- **URLTran: Improving Phishing URL Detection Using Transformers**

URLTran: um modelo de detecção de URLs de phishing baseado em transformers.
Uso de um grande dataset real, extraído da telemetria de navegação dos navegadores Microsoft Edge e Internet Explorer em 2019, contendo mais de 1 milhão de URLs para treino, cerca de 260 mil para validação e um conjunto de testes com quase 2 milhões de URLs, tendo apenas 9 mil de phishing.
A arquitetura URLTran é composta por três etapas principais:

- Tokenização - URLs segmentadas em subpalavras usando algoritmos como Byte Pair Encoding (BPE)
- Transformers - tokens são processados por modelos baseados em BERT e RoBERTa, que aprendem representações contextuais
- Classificador - embeddings produzidas pelos transformers são passadas por camadas que realizam a classificação binária entre phishing e benigno.

Também investiga a robustez do modelo frente a três tipos de ataques adversariais comuns em phishing:
Homoglyph attack – substituição de caracteres por similares em Unicode ( "apple.com" com um "a" cirílico).
Compound attack – inserção de hífens para imitar domínios compostos ( "bankofamerica.com" → "bank-of-america.com").

Parameter reordering – mudança na ordem dos parâmetros em strings de consulta (query strings).

Documento da Semana 02

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Consolidar o estudo em duas áreas complementares, avançando na compreensão de malware detection por DL, ao mesmo tempo em que aprofundo o tema de phishing detection, que se mostra promissor para futuras investigações e aplicações práticas.

- Leitura e fichamento do survey - A Survey of Malware Detection Using Deep Learning
- Levantamento de possíveis direções futuras em phishing detection, como uso de multimodalidade (URLs + páginas), integração com LLMs para Security Operations Center (SOCs) e defesas baseadas em adversarial training.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Documento da Semana 2 - citado no Termo de Aceite de Entrega de 10 de Setembro]

Na segunda Semana, o objetivo foi Aprofundar o estudo sobre IA como ferramenta de defesa cibernética, com foco em estudar e produzir um fichamento dos seguintes materiais:

- Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey - aprendizado por reforço profundo aplicado para detectar intrusões em redes IoT
- URLTran: Improving Phishing URL Detection Using Transformers - Transformers para identificar URLs maliciosas

**Deep Reinforcement Learning for Intrusion Detection in IoT: A Survey**

(Gueriani et al., 2024)

O artigo tem como propósito apresentar uma revisão abrangente sobre o uso de Deep Reinforcement Learning (DRL) em Intrusion Detection Systems (IDS) direcionados a redes de Internet das Coisas (IoT). Mais do que listar aplicações, o estudo organiza a literatura existente em categorias de métodos, datasets e métricas de avaliação, além de propor direções futuras de pesquisa que busquem preencher as lacunas identificadas no estado da arte.

Os Intrusion Detection Systems (IDS) são mecanismos essenciais de defesa em segurança da informação. Eles monitoram redes e sistemas em busca de atividades suspeitas ou não autorizadas, funcionando como uma segunda linha de proteção além dos firewalls. Enquanto os firewalls estabelecem barreiras de acesso, os IDS têm a função de identificar comportamentos anômalos ou tráfego malicioso que consiga atravessar essas barreiras.

No contexto da Internet das Coisas, essa necessidade é ainda mais crítica. O ambiente IoT é marcado por uma enorme heterogeneidade de dispositivos (sensores, câmeras, dispositivos médicos, drones, entre outros), pela limitação de recursos computacionais e pela constante expansão da superfície de ataque. Esses fatores tornam as soluções tradicionais insuficientes. IDS baseados em assinaturas, por exemplo, falham ao lidar com ataques de dia zero, já que dependem de padrões previamente conhecidos. IDS baseados em heurísticas, por sua vez, têm dificuldade em acompanhar o dinamismo e a diversidade do tráfego em IoT.

O avanço do machine learning e do deep learning trouxe progressos relevantes para a detecção de intrusões, mas essas técnicas ainda sofrem com a falta de adaptação em tempo real e com a dificuldade de generalizar para cenários mais realistas. Nesse cenário, o

Deep Reinforcement Learning surge como uma alternativa promissora. Ao combinar a capacidade do Reinforcement Learning de aprender sequencialmente por interação com o ambiente com o poder de representação do Deep Learning, o DRL consegue se ajustar dinamicamente em contextos complexos e mutáveis, como os encontrados em IoT.

O survey organiza os trabalhos revisados em cinco linhas de aplicação, sendo a “a”, “b” e “d” as principais:

**a) IDS em Wireless Sensor Networks (WSN).**
Esses IDS são projetados especificamente para redes de sensores sem fio, bastante comuns em soluções IoT. Seu foco está na detecção de ataques como sinkhole, blackhole, selective forwarding e flooding, que exploram vulnerabilidades no roteamento de dados entre sensores. A principal vantagem do uso de DRL nesse contexto é a capacidade de aprender políticas adaptativas que acompanham mudanças na topologia da rede, reduzindo falsos positivos e aumentando a resiliência.

**b) IDS baseados em Deep Q-Networks (DQN/DDQN).**
Trata-se da linha mais consolidada, que aplica arquiteturas populares como o DQN e o Double DQN para classificar tráfego entre benigno e malicioso. Os resultados mostram que essas abordagens superam técnicas tradicionais (como SVM, KNN ou CNNs aplicadas isoladamente) em métricas de acurácia e recall. Porém, apresentam limitações importantes, como o alto custo computacional para treinamento e o risco de overfitting quando aplicados a datasets reduzidos ou artificiais.

**c) IDS Híbridos.**
Uma tendência recente é integrar o DRL a outras técnicas de detecção para criar soluções mais robustas. Exemplos incluem modelos que combinam DRL com assinaturas para aumentar a eficácia contra ataques já conhecidos, ou DRL com detecção estatística de anomalias para melhorar a resposta a ataques de dia zero. Esses modelos híbridos buscam um equilíbrio entre precisão e generalização, embora introduzam maior complexidade de implementação.

**d) IDS aplicados à Saúde (Healthcare IoT).**
Uma categoria particularmente sensível é o uso de IDS em sistemas médicos conectados, conhecidos como IoMT (Internet of Medical Things). Nesse caso, o DRL tem sido aplicado para detectar tráfego anômalo em dispositivos médicos, com foco especial em manter privacidade e baixo consumo energético, algo crucial em wearables e dispositivos de monitoramento contínuo. A confiabilidade é vital, pois falhas de segurança podem afetar diretamente a integridade e a vida dos pacientes.

**e) Outros contextos (NFV, SDN, Green IoT).**
O artigo também menciona aplicações emergentes de IDS com DRL em ambientes de

Network Function Virtualization (NFV), onde funções de rede são executadas de forma virtualizada; em Software Defined Networking (SDN), com a integração de IDS nos controladores centrais para resposta em tempo real; e no campo do Green IoT, que busca otimizar o balanço entre segurança e eficiência energética em dispositivos de baixo consumo.

Um ponto recorrente na literatura é a dependência de datasets públicos, entre os mais usados: NSL-KDD, UNSW-NB15, CICIDS2017, CIC-DDoS2019, BoT-IoT e N-BaIoT. Embora úteis para testes, muitos deles são considerados artificiais ou pouco representativos da complexidade do tráfego real em IoT, o que limita a capacidade de generalização dos modelos.

Quanto às métricas, os estudos normalmente utilizam Accuracy, Precision, Recall, F1-score, False Positive Rate (FPR) e False Negative Rate (FNR). Os resultados revisados indicam que IDS baseados em DRL conseguem reduzir significativamente a taxa de falsos positivos quando comparados a métodos clássicos, além de alcançarem níveis de acurácia superiores.

De maneira geral, os trabalhos revisados indicam que os IDS baseados em DRL apresentam benefícios claros:

- Adaptação contínua, com aprendizado incremental que dispensa re-treinamento completo.
- Maior precisão na identificação de ataques novos e complexos.
- Redução da taxa de falsos positivos, um dos maiores desafios em detecção de intrusões.

Por outro lado, ainda persistem limitações importantes:

- Alto custo de treinamento e execução, que dificulta sua aplicação em dispositivos IoT com poucos recursos.
- Dependência de datasets artificiais, que reduz a aplicabilidade em cenários reais.
- Vulnerabilidade a ataques adversariais, já que os próprios modelos de DRL podem ser explorados por inputs maliciosos.

O survey aponta várias frentes promissoras para pesquisas futuras:

- Construção de datasets mais realistas, que representem de fato o tráfego e os ataques em ambientes IoT.
- Integração de DRL com Graph Neural Networks (GNNs) e com aprendizado federado (Federated Learning), visando maior capacidade de generalização e preservação de privacidade.

- Desenvolvimento de modelos mais leves, adequados para dispositivos IoT com restrições energéticas e computacionais.
- Aumento da resiliência contra ataques adversariais, tornando os IDS não apenas mais inteligentes, mas também mais confiáveis frente a manipulações maliciosas.

**URLTran: Improving Phishing URL Detection Using Transformers**

(Maneriker et al., 2021)

O artigo tem como objetivo propor e avaliar o URLTran, um modelo de detecção de URLs de phishing baseado em transformers. A motivação central é superar as limitações dos modelos anteriores, como CNNs e LSTMs aplicados a sequências de caracteres, e também de arquiteturas mais específicas como o URLNet e o Texception. O URLTran é projetado para atender a um requisito crítico em ambientes de produção, especialmente em navegadores e serviços de segurança: alcançar alta taxa de detecção mesmo quando a taxa de falsos positivos precisa ser extremamente baixa.

O phishing permanece entre as ameaças cibernéticas mais prevalentes e custosas. Apenas em 2019, esse tipo de ataque resultou em prejuízos estimados em US$ 57 milhões nos Estados Unidos. Um dos fatores que tornam o phishing particularmente desafiador é o curto ciclo de vida das URLs maliciosas, que podem existir apenas por algumas horas antes de serem removidas ou substituídas. Isso exige sistemas de detecção capazes de identificar novos ataques rapidamente, sem depender exclusivamente de listas negras ou heurísticas fixas, que são facilmente burladas.

Métodos baseados em machine learning e deep learning surgiram como alternativas para identificar padrões em URLs, mas arquiteturas tradicionais como CNNs e LSTMs ainda apresentavam deficiências: dificuldade em capturar padrões sublexicais (como combinações incomuns de caracteres), falta de robustez frente a ataques de homografia (ex.: substituição de caracteres por semelhantes em Unicode) e pouca generalização em cenários desbalanceados. Nesse contexto, o sucesso dos transformers em processamento de linguagem natural inspirou sua aplicação ao domínio das URLs, que embora não sejam linguagem natural no sentido clássico, possuem uma estrutura rígida e regular que pode ser decomposta em tokens significativos.

O estudo se destaca pelo uso de um grande dataset real, extraído da telemetria de navegação dos navegadores Microsoft Edge e Internet Explorer em 2019. O conjunto incluiu:

- Mais de 1 milhão de URLs para treino;

- Cerca de 260 mil para validação;
  E um conjunto de teste com 1,78 milhão de URLs, no qual apenas 8.742 eram de phishing.

Esse cenário evidencia o desbalanceamento severo característico da detecção de phishing: apenas uma URL maliciosa para cada 50 mil benignas. Para lidar com isso, os autores aplicaram técnicas de downsampling e garantiram que as URLs fossem rotuladas manualmente por analistas de segurança, com apoio de serviços como o Phishtank, assegurando a qualidade dos rótulos.

A arquitetura URLTran é composta por três etapas principais:

1. Tokenização – URLs são segmentadas em subpalavras usando algoritmos como Byte Pair Encoding (BPE) e WordPiece, permitindo capturar padrões dentro de nomes de domínio e caminhos. Por exemplo, “bankofamerica” pode ser decomposto em “bank + of + america”, o que facilita a detecção de variantes maliciosas.
2. Transformers – os tokens são processados por modelos baseados em BERT e RoBERTa, que aprendem representações contextuais. Também foi testada uma versão treinada do zero com vocabulário customizado de URLs (URLTran_CustVoc).
3. Classificador Denso – as embeddings produzidas pelos transformers são passadas por camadas densas que realizam a classificação binária entre phishing e benigno.

O treinamento foi dividido em duas fases: um pré-treinamento com Masked Language Modeling (MLM), adaptado para o domínio de URLs, e o fine-tuning para classificação binária.

A avaliação foi conduzida com foco em métricas críticas para sistemas de produção: True Positive Rate (TPR) em taxas de False Positive Rate (FPR) extremamente baixas, além de AUC e F1-score. Os principais resultados foram:

- URLTran_BERT alcançou 86,8% de detecção (TPR) em FPR=0,01%, superando amplamente os baselines.
- Para comparação: URLNet alcançou 71,2% e Texception 52,1% no mesmo cenário. O modelo também apresentou AUC de 0,9993 e F1-score de 0,91, consolidando-se como estado da arte.
- Em termos de eficiência, o tempo de inferência foi de aproximadamente 0,36 ms por URL, viabilizando o uso em navegadores e ambientes de alto tráfego.

O artigo não se limita a avaliar métricas tradicionais, mas também investiga a robustez do modelo frente a três tipos de ataques adversariais comuns em phishing:

1. Homoglyph attack – substituição de caracteres por similares em Unicode (ex.: uso de “apple.com” com um “a” cirílico).
2. Compound attack – inserção de hífens para imitar domínios compostos (ex.: “bankofamerica.com” → “bank-of-america.com”).
3. Parameter reordering – mudança na ordem dos parâmetros em strings de consulta (query strings).

Nos testes, o URLTran sofreu perdas de desempenho contra esses ataques, mas a aplicação de treinamento adversarial (AdvTraining) permitiu recuperar e até superar a performance original, mostrando que a arquitetura é capaz de se adaptar a cenários adversos quando reforçada com exemplos simulados.

# APÊNDICE 2

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 17 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta terceira Semana, o objetivo foi consolidar o estudo em duas áreas complementares, avançando na compreensão de malware detection por DL, ao mesmo tempo em que aprofundo o tema de phishing detection, que se mostra promissor para futuras investigações e aplicações práticas.

- **A Survey of Malware Detection Using Deep Learning**

Sistemas operacionais são atualizados constantemente para corrigir vulnerabilidades, mas criadores de malware também evoluem seus códigos para explorar novas falhas.

* Diversidade grande de dispositivos com diferentes arquiteturas e recursos de hardware

Diferentemente de técnicas clássicas, os modelos de DL têm maior capacidade de generalização para amostras nunca vistas, o que é crucial para lidar com variantes inéditas e ataques de dia zero.

* Uso de documentos PDF como vetor de ataque, uso da keyword /OpenAction para executar automaticamente uma ação quando o PDF é aberto

* Malware assim como outros programas, é feito de bits e bytes. "malware como imagem" > converter esses bytes em uma matriz bidimensional de pixels, criando uma imagem que pode ser analisada por modelos de aprendizado profundo

A principal vantagem dessa abordagem é transformar o problema de detecção de malware em um problema de classificação de imagens, permitindo o uso direto de redes neurais convolucionais ( CNNs) e Transfer Learning

* Ataques adversariais > inputs modificados, capazes de enganar o modelo e forçá-los a fazer classificações erradas, tornando-os inseguros em ambientes reais

- **Possíveis Direções Futuras em Phishing Detection**

Delimitei o foco em phishing detection e usei ArXiv, Google Scholar, ResearchGate e ACM/IEEE para buscar artigos recentes (2024 e 2025). Usei termos como *'phishing multimodal', 'LLM phishing detection', 'adversarial robustness'* e filtrei por papers que apresentassem propostas conceituais. Li os abstracts e conclusões para selecionar apenas os que discutiam abordagens novas. Ao final, agrupei os achados em temas principais, como multimodalidade, LLMs, defesas contra evasão, uso de grafos e explicabilidade, que são hoje os tópicos mais promissores.

**1 - Conhecimento de Marcas e Grafos -** Detectar uso indevido de logos, cores e identidades visuais, para identificar phishing que imita sites conhecidos. - KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection (2024)

**2 - LLMs e Agentes Inteligentes -** Usar modelos grandes para entender o contexto, tom de linguagem e elementos persuasivos para detectar ataques sofisticados que enganam modelos tradicionais. - Adaptive Linguistic Prompting (ALP) Enhances Phishing Webpage Detection in Multimodal Large Language Models (JULHO 2025)

**3 - Defesas contra Ataques Adversariais -** Treinar modelos com exemplos adversariais (adversarial training), para garantir que o detector não seja facilmente enganado por pequenas modificações na URL ou na página. - From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks | Digital Threats: Research and Practice (JUNHO 2025)

**4 - Multimodalidade -** Combinar análise de URLs, HTML, screenshots e elementos visuais para detectar ataques que copiam a aparência de sites legítimos, indo além da análise de texto. - Beyond the Mask: Benchmarking Multimodal Large Language Models to Expose Phishing Websites (SETEMBRO 2025)

**5 - Expansão para Novos Vetores -** Analisar phishing em redes sociais, SMS e voz, por que o phishing está migrando para outras plataformas. - Multimodal framework for phishing attack detection and mitigation through behavior analysis using EM-BERT and SPCA-BASED EAI-SC-LSTM (JULHO 2025)

Documento da Semana 03

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima Semana, pretendo iniciar o estudo das direções futuras que levantei para *phishing detection*.

Ainda não defini se abordarei todos os trabalhos de uma vez ou se dividirei em etapas.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

[Documento da Semana 3 - citado no Termo de Aceite de Entrega de 17 de Setembro]

Na terceira Semana, o objetivo foi consolidar o estudo em duas áreas complementares, avançando na compreensão de malware detection por DL, ao mesmo tempo em que aprofundo o tema de phishing detection, que se mostra promissor para futuras investigações e aplicações práticas.

- Leitura e fichamento do survey - A Survey of Malware Detection Using Deep Learning
- Levantamento de possíveis direções futuras em phishing detection, como uso de multimodalidade (URLs + páginas), integração com LLMs para Security Operations Center (SOCs) e defesas baseadas em adversarial training.

**A Survey of Malware Detection Using Deep Learning**

(Bensaoud, Kalita & Bensaoud, 2024)

O survey contextualiza o problema da detecção de malware como um dos maiores desafios da cibersegurança moderna. Sistemas operacionais como Windows, Linux, Android e MacOS são atualizados constantemente para corrigir vulnerabilidades críticas, mas criadores de malware também evoluem seus códigos de forma contínua para explorar novas falhas e contornar mecanismos de proteção.

Um dos grandes problemas atuais é a diversidade de dispositivos que precisam ser protegidos, não apenas desktops e servidores, mas também roteadores, câmeras de segurança, dispositivos IoT, drones e outros equipamentos. Cada tipo de dispositivo apresenta diferentes arquiteturas, recursos de hardware e superfícies de ataque, o que aumenta a complexidade de defesa.

Cita exemplos de ataques recentes para ilustrar a gravidade do cenário:

- Em 2022, o grupo russo Killnet realizou ataques de negação de serviço contra serviços governamentais de diversos estados americanos.
- No mesmo ano, hackers ligados ao governo chinês roubaram cerca de US$ 20 milhões em benefícios de alívio da COVID-19.

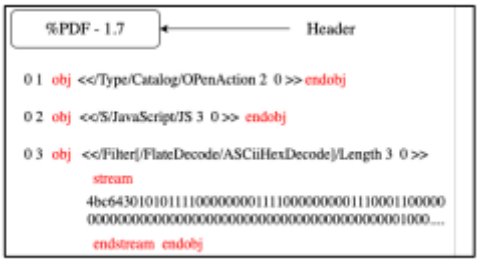
Destaca o papel do Deep Learning (DL) como tecnologia promissora para a detecção de malware. Diferentemente de técnicas clássicas, os modelos de DL têm maior capacidade de

generalização para amostras nunca vistas, o que é crucial para lidar com variantes inéditas e ataques de dia zero.

O objetivo central de qualquer hacker é conseguir instalar malware no sistema da vítima. Devido às barreiras impostas por firewalls e outras defesas perimetrais, ataques diretos são cada vez mais difíceis. Por isso, os atacantes recorrem a técnicas de engenharia social e exploração de vulnerabilidades para enganar o usuário e induzir o sistema a executar código malicioso.

Um vetor típico é o envio de documentos ou executáveis maliciosos por e-mail, phishing ou links em sites comprometidos. Quando o usuário abre o arquivo, scripts embutidos ou exploradores de vulnerabilidade (exploit kits) são executados, baixando ou extraindo o malware principal — muitas vezes backdoors, trojans ou ransomwares.

Exemplo: Ataques com PDFs Maliciosos




uso de documentos PDF como vetor de ataque. PDFs têm uma estrutura interna composta por:

- Objetos – blocos de dados que podem conter fontes, imagens, scripts ou metadados.
- Palavras-chave (keywords) – controlam o comportamento do documento (ex.: /OpenAction, /JavaScript, /EmbeddedFile).
- Fluxos de dados (streams) – que podem ser codificados, comprimidos ou até criptografados.

Hackers exploram essas características para esconder código malicioso. Um exemplo comum é o uso de /OpenAction para executar automaticamente uma ação quando o PDF é aberto, como disparar um script JavaScript embutido que baixa e executa outro arquivo.

Além disso, os atacantes aplicam técnicas de ofuscação para dificultar a análise, como:

- Codificação mista (hexadecimal, octal, ASCII combinados) para esconder strings e URLs.
- Compressão e filtros como Zlib ou ASCII85 para mascarar o conteúdo real.
- JavaScript ofuscado, com variáveis renomeadas e código desnecessário, para atrasar ferramentas de análise estática.

características que definem o funcionamento do malware, isto é, o que o código malicioso faz, como se comporta e quais são os objetivos por trás de sua execução. Entender essa natureza é essencial para o desenvolvimento de técnicas de defesa mais eficazes, pois permite aos profissionais de cibersegurança antecipar, detectar e mitigar ameaças.

- Obfuscação (Obfuscation)
  é esconder a verdadeira intenção do código, tornando-o difícil de analisar. Técnicas de ofuscação incluem criptografia de trechos de código, compressão e inserção de instruções redundantes. Isso dificulta a detecção por antivírus baseados em assinaturas, já que o código pode assumir formas diferentes a cada execução.
- Entrega de Payload (Payload Delivery)
  Todo malware possui um payload, isto é, a carga maliciosa que realiza a ação planejada:
  – roubo de informações sensíveis (credenciais, dados financeiros),
  – execução de ataques DDoS,
  – instalação de backdoors,
  – ou criptografia de arquivos para extorsão (ransomware).
- Comando e Controle (C&C)
  Muitas famílias de malware mantêm comunicação ativa com servidores de comando e controle. Isso permite ao atacante atualizar o malware, receber dados roubados ou controlar remotamente o sistema infectado, transformando-o em parte de botnets para ataques em larga escala.
- Autorreplicação (Self-Replication)
  Certos malwares são capazes de se propagar sozinhos, infectando outros arquivos, dispositivos ou máquinas na mesma rede, o que amplia rapidamente o alcance do ataque.
- Exploração de Vulnerabilidades (Exploitation)
  O malware frequentemente aproveita falhas em sistemas operacionais, softwares ou até erros humanos (como phishing) para obter acesso não autorizado e executar ações maliciosas.
- Polimorfismo e Metamorfismo
  Malware polimórfico ou metamórfico consegue alterar sua própria estrutura a cada nova execução ou infecção, mantendo a funcionalidade. Isso torna a detecção ainda mais difícil, pois cada instância parece diferente para sistemas baseados em assinatura.

- Ransomware – Um Caso Especial
  O ransomware combina criptografia forte (como RSA) com a infecção do sistema para sequestrar dados da vítima e exigir pagamento — geralmente em criptomoedas como Bitcoin — para a liberação das informações.
  O survey cita exemplos como WannaCry, que causou impacto global em 2017, e casos mais recentes, como o ataque do grupo Conti à Costa Rica, que levou o país a declarar estado de emergência em 2022, ou o roubo de dados de 270 mil pacientes de hospitais da Louisiana no mesmo ano.
  O ransomware modifica até a interface do sistema (como o wallpaper) para exibir a mensagem de resgate, exigindo valores que podem chegar a milhões de dólares.

classifica as técnicas de detecção de malware em três categorias principais: estática, dinâmica e híbrida. Cada abordagem tem vantagens e limitações, e os pesquisadores frequentemente combinam métodos para melhorar a eficácia.

Detecção Estática

A análise estática examina o código ou o arquivo executável sem executá-lo.

- Assinaturas de código e hash de funções de importação (como a técnica de *fuzzy-import hashing* de Naik et al.).
- Permissões de aplicativos (como no Android), onde modelos de ML identificam padrões suspeitos de permissões excessivas (Mohamad et al.).

Vantagem: é rápida e não requer execução do malware.
Limitação: é menos eficaz contra malware ofuscado ou polimórfico, que muda sua aparência para enganar a análise estática.

Detecção Dinâmica

A análise dinâmica consiste em executar o malware em um ambiente controlado (sandbox) para monitorar seu comportamento em tempo real — por exemplo, chamadas de sistema, acessos a rede, criação de processos e alterações em arquivos.

Vantagem: detecta comportamentos maliciosos que não são visíveis no código estático.
Limitação: é mais lenta, consome recursos e pode ser burlada por malware que detecta se está sendo executado em sandbox.

Detecção Híbrida

Método mais promissor — combina análise estática e dinâmica, aproveitando o melhor dos dois mundos.
Chaulagain et al. propuseram um sistema que coleta artefatos de ambas as análises para treinar modelos de deep learning, obtendo maior precisão e menor taxa de falsos positivos.

---

A base para qualquer método de detecção é a coleta de dados de eventos do sistema.
sistemas operacionais e softwares de infraestrutura geram uma grande quantidade de logs — locais e remotos, incluindo:

- Syslog (padrão IETF para equipamentos de rede)
- Logs de autenticação (login/logout, tentativas falhas)
- Logs de eventos de segurança e rede
- Windows Event Logs (com timestamp, usuário, processo)

A análise desses logs permite:

- Identificar brechas de segurança e comportamento suspeito.
- Fazer correlação de eventos para reconstruir ataques.
- Treinar modelos de ML/DL com dados rotulados para classificação entre comportamento benigno e malicioso.

qualidade dos dados é crítica: logs incompletos ou inconsistentes dificultam a detecção. Por isso, um dos objetivos de pesquisa é melhorar a forma como os dados são coletados, armazenados e processados para que alimentem modelos de aprendizado de máquina de forma eficiente e representativa

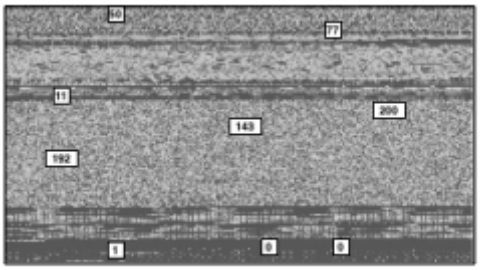
Uma das contribuições mais inovadoras do survey é mostrar como pesquisadores têm explorado representações visuais de malware para alavancar o poder das redes neurais convolucionais (CNNs) e outros modelos de visão computacional.

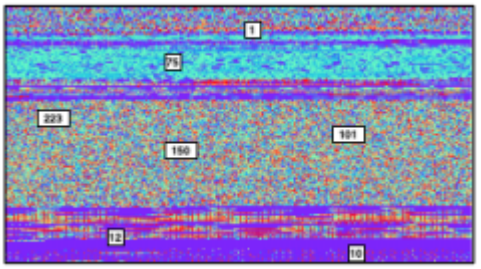
De Arquivo Binário para Imagem

Malware é, essencialmente, um conjunto de bytes, sequências de números que representam instruções de máquina. A ideia por trás da abordagem de “malware como imagem” é converter esses bytes em uma matriz bidimensional de pixels, criando uma imagem que pode ser analisada por modelos de aprendizado profundo.

Essa conversão pode ser feita de várias formas:

- Representação em tons de cinza (grayscale): cada byte é mapeado para um valor de intensidade entre 0 e 255, onde 0 é preto e 255 é branco. Assim, o arquivo se torna uma "imagem" em que padrões binários correspondem a padrões visuais.





- 
- Representação em RGB: divide-se a sequência de bytes em três canais (R, G, B), gerando imagens coloridas que podem capturar mais variação e contexto — potencialmente facilitando que redes profundas diferenciem famílias de malware.

Métodos

- Markov Images (Yuan et al.): convertem os bytes em uma matriz de probabilidade de transição (cadeia de Markov). Cada pixel representa a probabilidade de um byte ser seguido por outro. O resultado é uma imagem densa de padrões de transição.
- Bigram + DCT (Mohammed et al.): usam a frequência de pares de bytes (bigramas) como intensidade de pixels, seguida por transformada discreta do cosseno (DCT) para “desesparsificar” os dados, produzindo imagens mais compactas e informativas.
- Window Entropy Map – WEM (Euh et al.): cada byte tem sua entropia calculada, criando um mapa que representa o grau de aleatoriedade ou incerteza no código. Áreas de alta entropia geralmente correspondem a seções compactadas ou criptografadas, comuns em malware ofuscado.
- SimHash Mapping (Ni et al.): converte o malware em hashes SimHash (técnica de detecção de similaridade) e os mapeia para pixels de uma imagem em escala de cinza. Essa técnica é útil para identificar variantes de malware da mesma família.

Para pré-processar os arquivos e gerar essas representações, são usadas ferramentas como:

- IDA Pro e x64dbg – para engenharia reversa e depuração.
- HxD e PE-bear – para visualização e edição de arquivos binários.
- Yara – para identificar padrões de malware.

- Fiddler e XOR analysis – para análise de tráfego e decodificação de dados.

Importância para Deep Learning

A principal vantagem dessa abordagem é transformar o problema de detecção de malware em um problema de classificação de imagens, permitindo o uso direto de:

- CNNs como VGG16, ResNet, Inception, Xception.
- Transfer learning com modelos pré-treinados em ImageNet
- Métodos de data augmentation (rotação, espelhamento, normalização) para melhorar a robustez.

Essa técnica tem mostrado altas taxas de acurácia, chegando a superar 98% em alguns datasets como Malimg e BIG2015, e é considerada um dos caminhos mais promissores para superar limitações dos métodos tradicionais de análise estática/dinâmica.

Depois de gerar representações visuais de malware, a etapa seguinte é classificar essas imagens usando modelos de aprendizado profundo. Essa abordagem é considerada uma das mais promissoras, pois elimina a necessidade de extração manual de features e permite que a própria rede aprenda os padrões que diferenciam malwares entre si e de arquivos benignos.

Transformar malware em imagens cria uma forma padronizada de representação que pode ser usada em arquiteturas de redes neurais convolucionais (CNNs), conhecidas por sua capacidade de detectar padrões em dados visuais. Assim, em vez de lidar com bytes ou código de máquina diretamente, transformamos o problema em um desafio de visão computacional.

Principais Estudos e Abordagens

- Nataraj et al. (2011): pioneiros na visualização de bytes de malware em escala de cinza para diferenciar famílias de malware.
- Yuan et al.: propuseram uso de Markov images, que representam a probabilidade de transição de bytes, classificadas com CNNs sem necessidade de redimensionamento.
- Narayanan & Davuluru: combinaram RNN + CNN em um ensemble para classificar imagens geradas de arquivos de assembly.

- Zhu et al.: introduziram Siamese Neural Networks com meta-learning para detectar malware ofuscado, mostrando robustez a variantes.
- Chauhan et al.: exploraram diferentes espaços de cores (RGB, HSV, BGR) e usaram SVMs, obtendo acurácia de 96%.
- Darem et al.: propuseram método semi-supervisionado que alcançou 99,12% de acurácia em malware ofuscado.
- Asam et al.: apresentaram DFS-MC e DBFS-MC, com acurácia de 98,61% no dataset MalImg.
- Xiao et al.: criaram Colored Label Boxes (CoLab), técnica que mapeia seções de arquivos PE para regiões coloridas, classificando com VGG16 e SVM com acurácia acima de 96%.

Os resultados: os modelos revisados frequentemente ultrapassam 98% de acurácia em datasets públicos como MalImg e BIG-2015, demonstrando a alta eficácia do approach baseado em imagens. Além disso, esses métodos ajudam a:

- Identificar correlações entre diferentes famílias de malware.
- Detectar malware ofuscado e variantes de famílias já conhecidas.
- Reduzir dependência de análise manual, acelerando SOCs (Security Operations Centers).

Apesar de CNNs aprenderem features automaticamente, muitos pesquisadores exploram técnicas de redução de dimensionalidade para tornar o treinamento mais eficiente e melhorar a generalização.

- Eficiência Computacional: menos features → menos parâmetros → treinamento e inferência mais rápidos.
- Melhor Generalização: remove ruído e redundância, reduzindo risco de overfitting.
- Foco nas Features Mais Discriminantes: melhora a interpretabilidade e a explicabilidade dos modelos.

Métodos

1. Feature Selection – selecionar subconjunto das features originais:
    - Wrappers: usam modelos preditivos para avaliar combinações de features.
    - Filters: baseados em estatísticas, como correlação ou informação mútua.
    - Embedded: seleção feita durante o treinamento do modelo (ex.: regularização L1/LASSO).

2. Feature Extraction – transformar o espaço original em um espaço de menor dimensão:
    - PCA (Principal Component Analysis): projeta dados em eixos que explicam maior variância.
    - LDA (Linear Discriminant Analysis): maximiza separação entre classes.
    - DWT, ICA, GIST, SIFT, SURF, LBPs, KAZE: técnicas avançadas para extrair características visuais discriminantes.

Resultados citados

- Azad et al.: propuseram DEEPSEL, método de seleção profunda de features que atingiu F-measure de 82,5% em 39 famílias de malware.
- Tobiyama et al.: usaram RNNs para extrair features de chamadas de sistema, seguidas de CNNs para classificação.

O survey também traz uma tabela comparativa mostrando que modelos como Inception V3, DenseNet e Xception alcançam de 98% a 99% de acurácia em datasets MalImg e BIG-2015, reforçando o potencial de combinar feature reduction com deep learning.

**Table 2**
Comparative performance summary of Transfer Learning models for malware image classification.

| Reference | Features | Model | Files | Accuracy | Dataset |
|---|---|---|---|---|---|
| Çayır et al. [20] | gray-scale images | CapsNet | PE | 98.63% | Malimg |
| Çayır et al. [20] | gray-scale images | RCNF | PE | 98.72% | Malimg |
| Go et al. [21] | gray-scale images | ResNeXt | PE | 98.32% | Malimg |
| Bensaoud et al. [22] | gray-scale images | Inception V3 | PE | 99.24% | Malimg |
| El-Shafai et al. [23] | gray-scale images | VGG16 | PE | 99.97% | Malimg |
| Hemalatha et al. [24] | gray-scale images | DenseNet | PE | 98.23% | Malimg |
| Hemalatha et al. [24] | gray-scale images | DenseNet | PE | 98.46% | BIG 2015 |
| Lo et al. [25] | gray-scale images | Xception | PE | 99.03% | Malimg |
| Lo et al. [25] | gray-scale images | Xception | PE | 99.17% | BIG 2015 |

Uma das maiores barreiras na aplicação de deep learning para detecção de malware é a necessidade de grandes quantidades de dados rotulados e de poder computacional para treinar modelos do zero. Transfer Learning surge como uma solução estratégica: ele reutiliza modelos previamente treinados em grandes bases de imagens (como o ImageNet), aproveitando o aprendizado já consolidado para acelerar o treinamento em domínios especializados como o de malware.

- Modelos como VGG16, ResNet, Inception e EfficientNet foram treinados em milhões de imagens genéricas, aprendendo filtros de baixo nível (bordas, texturas, formas) que são úteis em diversas tarefas visuais.

- Ao invés de treinar uma rede do zero, aproveitamos essas camadas convolucionais e adaptamos apenas as camadas finais para classificar imagens de malware.
- Essa estratégia permite alcançar alta acurácia com menos dados rotulados e menor custo computacional.

1. Feature Extraction (Extração de Features):
    - Mantém-se congelada a base convolucional do modelo pré-treinado (ex.: VGG16 treinado no ImageNet).
    - Adiciona-se um novo classificador no topo (camadas densas), treinado especificamente para classificar malware.
    - É um método eficiente em recursos, pois evita re-treinar milhões de parâmetros.

2. Fine-Tuning (Ajuste Fino):
    - Descongela-se parcialmente as camadas convolucionais superiores, permitindo que sejam ajustadas ao novo domínio.
    - Essa abordagem melhora a especialização do modelo para padrões específicos de malware, aumentando a acurácia.
      Exige mais poder computacional, mas entrega resultados superiores.

Resultados:

O survey compara diferentes arquiteturas aplicadas a datasets como Malimg e Microsoft Malware Dataset:

- Go et al. (ResNeXt50 pré-treinado): 98,86% de acurácia no Malimg.
- Çayır et al. (CapsNet + Ensemble): 96,6% de F-Score no Malimg e 98,2% no BIG2015.
- Bensaoud et al.: testaram seis CNNs, sendo Inception-V3 a que apresentou melhor desempenho, estabelecendo o estado da arte.
- Khan et al.: usaram ResNet e GoogleNet para detecção a partir de bytecode de APK, com resultados robustos.
- EfficientNet (B0–B7): mostrou que aumentar a profundidade melhora a performance, mas eleva significativamente o custo de treinamento e uso de memória GPU.

Transfer learning funciona bem mesmo em domínios distantes, como malware, pois as primeiras camadas da rede aprendem representações genéricas.
Modelos mais profundos (Inception-V4, EfficientNet-B4) apresentam os melhores resultados, mas podem ser inviáveis em ambientes com restrições de hardware.

Há espaço para explorar modelos mais leves e eficientes, mantendo o bom desempenho, algo crucial para aplicações em endpoints ou IoT.

Embora boa parte da pesquisa em detecção de malware seja focada em análise binária e imagens, uma grande quantidade de informações sobre malware está contida em dados textuais ou semi-estruturados:

- Logs de sistema e rede
- Chamadas de API e sequências de opcodes
- Strings extraídas de binários
- Documentos (Word, PDF) com macros
- E-mails e tráfego de rede relacionados à distribuição de malware

O uso de NLP tem como objetivo explorar essas fontes textuais para construir modelos que consigam detectar comportamentos maliciosos sem depender exclusivamente de análise estática ou dinâmica de binários.

O primeiro passo é transformar o conteúdo textual em algo que um modelo de ML/DL possa processar. O survey apresenta várias estratégias de text encoding:

- Bag of Words (BoW) e TF-IDF – contagem de frequência de termos, útil para detectar padrões repetitivos.
- n-grams – capturam padrões de sequência (ex.: pares ou trios de chamadas de API).
- One-hot encoding ou representações ASCII – úteis para representar tokens em nível de caractere.
- Word Embeddings modernos (Word2Vec, Sent2Vec) – criam vetores semânticos para cada token, permitindo capturar relações mais sutis entre instruções de código ou chamadas de API.

Um ponto importante é que embeddings comuns (como Word2Vec treinado em texto natural) não são ideais para malware, pois não capturam semântica específica de chamadas de API ou bytecodes. Por isso, há uma tendência de treinar embeddings especializados para segurança cibernética.

arquiteturas de deep learning que obtiveram excelentes resultados, incluindo:

- CNN + LSTM híbridos – extraem padrões locais de sequência e aprendem dependências de longo prazo.

- Sequence-to-Sequence Models – originalmente criados para tradução automática, convertem sequências de chamadas ou opcodes em representações compactas e podem classificar comportamento como benigno ou malicioso.
- Modelos com Mecanismo de Atenção – superam limitações de memória dos LSTMs puros, permitindo ao modelo "focar" nas partes relevantes da sequência.
    - *Or-Meir et al.* mostraram aumento de acurácia adicionando atenção a LSTMs.
    - *Mimura e Ohminami* criaram o SLAM (Sliding Local Attention Mechanism) para sequências de execução de API.
    - *Ma et al.* propuseram o ACNN, que combina código assembly e código binário em um modelo multimodal, com CNN + atenção, resultando em desempenho superior.

Esses modelos permitem detectar padrões de comportamento malicioso mesmo em casos de malware ofuscado ou em variantes novas (zero-day), pois aprendem relações de dependência na sequência de execução, e não apenas assinaturas estáticas.

- Acurácia próxima de 99,9% em algumas arquiteturas recentes (CNN+LSTM+attention).
- Melhor capacidade de generalização para malwares novos e variantes. Possibilidade de integrar logs de múltiplas fontes (sistemas, rede, arquivos), criando modelos multimodais.

- Alto custo computacional, principalmente em modelos com atenção em sequências longas.
- Falta de padronização de datasets, o que dificulta comparar os métodos de forma justa.
- Necessidade de explicabilidade (XAI), para que analistas de SOC possam confiar e interpretar as decisões dos modelos.

discute a relação entre criptografia, ransomware e deep learning, destacando como a análise de padrões criptográficos pode auxiliar na detecção precoce de ataques.

- Confidencialidade (proteção de dados)
- Integridade (uso de hashes como SHA, MD5)
- Autenticidade e Prova de Origem
- Não-repúdio (assinaturas digitais)

Ataques comuns incluem man-in-the-middle, brute force e ataques de texto escolhido. A identificação de algoritmos de cifragem é um problema de classificação o que abre espaço para ML/DL.

- Redes neurais podem classificar tipos de cifras a partir de features como n-grams de texto cifrado, Index of Coincidence (IoC) e frequências estatísticas.
- Pesquisas mostraram sucesso em identificar algoritmos como AES, DES e RSA automaticamente.
- Modelos como MLP, CNN e LSTM foram usados para ataques diferenciais automatizados (ex.: no cipher GIMLI).

Detecção de Ransomware

- Famílias como WannaCry, Dharma, GandCrab usam criptografia assimétrica para sequestrar arquivos e exigir pagamento (normalmente em Bitcoin).
- Modelos DL como CNNs têm sido usados para detectar uso anômalo de criptografia antes do impacto final, prevenindo a execução completa do ransomware.
- Abordagens de aprendizado não supervisionado ajudam a identificar ransomware desconhecido (zero-day).
- Ferramentas também detectam uso inseguro de APIs criptográficas, reduzindo vulnerabilidades de software (AUC-ROC de até 99,2%).

Um dos grandes desafios de usar deep learning em cibersegurança é a falta de interpretabilidade — o “caixa-preta” não é suficiente para ambientes de SOC e auditorias.

O que é XAI
Explainable AI busca tornar modelos transparentes, mostrando quais features levaram à decisão de classificar algo como malicioso. Isso:

- Aumenta a confiança operacional
- Ajuda a reduzir falsos positivos
- Permite entender por que um ataque passou despercebido

Principais Técnicas

- White-box approaches (acessam gradientes internos):
  - Backpropagation, Guided Backprop, Grad-CAM, Guided Grad-CAM
- Black-box approaches (funcionam apenas com input/output):

- LIME, Occlusion Sensitivity, Feature Ablation

McLaughlin et al.: Usaram LRP e DeepLIFT para mostrar quais opcodes mais influenciam a decisão do modelo.
Hooker et al.: Removeram features consideradas importantes pelo XAI e observaram queda de acurácia (validação de explicabilidade).
Lin et al.: Compararam 7 métodos de XAI e propuseram avaliação automatizada da correção das explicações.

- Robustez: XAI pode ser alvo de ataques adversariais (explicações enganosas).
- Escalabilidade: aplicar XAI em SOCs em tempo real pode ser custoso.
- Padronização: ainda não há consenso sobre métricas para avaliar qualidade da explicação.

Um dos maiores desafios atuais para a aplicação de deep learning em cibersegurança é a vulnerabilidade a ataques adversariais. Esses ataques consistem em inputs maliciosamente modificados, quase imperceptíveis ao olho humano, capazes de enganar modelos de aprendizado profundo e forçá-los a fazer classificações erradas.

- Pequenas perturbações são adicionadas aos dados de entrada (malware, imagem, log).
- Essas alterações são cuidadosamente otimizadas para enganar o modelo sem alterar a funcionalidade do malware.
- Podem ser usadas para:
  - Evasion attacks: o malware passa despercebido pelo classificador.
  - Poisoning attacks: dados maliciosos são inseridos no processo de treinamento, corrompendo o modelo.

Métodos Recentes

- Malfox (Zhong et al.): usa GANs condicionais para gerar exemplos adversariais que mantêm a funcionalidade do malware e conseguem escapar de detectores black-box.
- SAGE (Zhao et al.): combina métodos baseados em gradiente e livres de gradiente para gerar ataques mais eficientes.
  Hu e Tan: usaram GANs para criar exemplos adversariais para Android, mantendo alta similaridade com o malware original.
- Xu et al. – Ofei Framework: gera amostras adversariais contra apps Android em plataformas DLAAS.

- Qiao et al.: propuseram usar Random Forest + LIME para identificar e detectar amostras adversariais de ELF malware.

Impacto e Desafios

- Adversarial attacks podem reduzir drasticamente a precisão de modelos, tornando-os inseguros em ambientes reais.
- Criar mecanismos de defesa é caro e computacionalmente intensivo.
- É necessário buscar modelos mais robustos, adversarial training e integração de XAI para explicar como o modelo reage a perturbações.

balanço sobre o papel do deep learning na detecção de malware, destacando avanços e desafios:

- Avanços:
  - Aplicação bem-sucedida de CNNs para classificar imagens de malware com alta acurácia (>98%).
  - Uso crescente de transfer learning e multi-task learning, reduzindo a necessidade de datasets gigantes.
  - Progresso na classificação de criptografia, detecção de ransomware e uso de NLP + atenção para analisar logs e sequências de API calls.

- Desafios Persistentes:
  - Necessidade de datasets mais realistas e balanceados.
  - Ajuste fino de hiperparâmetros e mitigação de overfitting.
    Alto custo computacional de treinamento e inferência em ambientes IoT ou SOCs.
  - Vulnerabilidade a adversarial attacks.

- Direções Futuras:
  - Desenvolvimento de modelos mais robustos, escaláveis e eficientes.
  - Integração de XAI para aumentar a confiança operacional.
  - Exploração de aprendizado federado para preservar privacidade em cenários distribuídos.
  - Combinação de abordagens multimodais (imagens, logs, tráfego de rede) para detecção mais abrangente.

Deep learning é uma ferramenta poderosa para a defesa cibernética, mas precisa evoluir para ser explicável, resistente a ataques e eficiente para uso em escala real.

**Possíveis Direções Futuras em Phishing Detection**

Delimitei o foco em phishing detection e usei ArXiv, Google Scholar, ResearchGate e ACM/IEEE para buscar artigos recentes (2024 e 2025). Usei termos como *'phishing multimodal', 'LLM phishing detection', 'adversarial robustness'* e filtrei por papers que apresentassem propostas conceituais. Li os abstracts e conclusões para selecionar apenas os que discutiam novas abordagens. Ao final, agrupei os achados em temas principais, como multimodalidade, LLMs, defesas contra evasão, uso de grafos e explicabilidade, que são hoje os tópicos mais promissores.

**1 - Multimodalidade -** Combinar análise de URLs, HTML, screenshots e elementos visuais para detectar ataques que copiam a aparência de sites legítimos, indo além da análise de texto. - Beyond the Mask: Benchmarking Multimodal Large Language Models to Expose Phishing Websites (SETEMBRO 2025)

**2 - LLMs e Agentes Inteligentes -** Usar modelos grandes para entender o contexto, tom de linguagem e elementos persuasivos para detectar ataques sofisticados que enganam modelos tradicionais. - Adaptive Linguistic Prompting (ALP) Enhances Phishing Webpage Detection in Multimodal Large Language Models (JULHO 2025)

**3 - Defesas contra Ataques Adversariais -** Treinar modelos com exemplos adversariais (adversarial training), para garantir que o detector não seja facilmente enganado por pequenas modificações na URL ou na página. - From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks | Digital Threats: Research and Practice (JUNHO 2025)

**4 - Conhecimento de Marcas e Grafos -** Detectar uso indevido de logos, cores e identidades visuais, para identificar phishing que imita sites conhecidos. - KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection (2024)

**5 - Expansão para Novos Vetores -** Analisar phishing em redes sociais, SMS e voz, por que o phishing está migrando para outras plataformas. - Multimodal framework for phishing attack detection and mitigation through behavior analysis using EM-BERT and SPCA-BASED EAI-SC-LSTM (JULHO 2025)

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 25 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta quarta Semana, o objetivo foi iniciar o estudo de 4 surveys relacionados a direções futuras que levantei para *phishing detection*. Realizei a leitura e fichamento de 2, são eles:

- **Beyond the Mask: Benchmarking Multimodal Large Language Models to Expose Phishing Websites**

Ataques de phishing estão cada vez mais sofisticados. A ideia do estudo foi testar se os LLMs multimodais, que analisam URL, código HTML e a aparência visual (screenshot) de um site poderiam ser mais eficientes. Analisaram 4 cenários:
* Apenas URL > excelente desempenho, ou seja, contêm sinais muito fortes e discriminativos para a detecção de phishing
* Apenas HTML > Trade-off entre Precisão e Recall: identificaram bem os sites maliciosos, mas gerou muitos falsos positivos.
* Apenas screenshot do site > desempenho ruim, com muitas falhas.
* Todos os três juntos > desempenho moderado, maior quantidade de informação nem sempre é a melhor situação.

- **Adaptive Linguistic Prompting (ALP) Enhances Phishing Webpage Detection in Multimodal Large Language Models**

Introduz e avalia uma nova técnica de prompt chamada Adaptive Linguistic Prompting (ALP), projetada para aprimorar a detecção de páginas de phishing usando as capacidades multimodais de LLMs como o GPT-4o e o Gemini 1.5 Pro.

* Análise da Página (MWA): Ela olha para o conteúdo e a aparência do site (código HTML e imagem) para ver se a marca parece legítima e se a linguagem é suspeita.

* Análise da URL (USA): Uma segunda análise foca apenas no endereço do site, procurando por padrões maliciosos. Depois disso, combinam os resultados das análises.

Importância do prompt > uma simples instrução adicionada ao prompt da análise de URL: *"Se você não tiver certeza e achar que um link é suspeito de atividade de phishing, rotule-o como phishing"*. Apenas essa regra fez a precisão do GPT-4o na análise de URLs saltar de 81% para 91%

Documento da Semana 04

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima Semana, pretendo definir como aplicarei IA em phishing detection.
Aprofundar o estudo nesses dois surveys:

- From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks | Digital Threats: Research and Practice - vulnerabilidade de modelos de detecção de phishing em ataques adversariais.
- KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection - identificar o logo de uma marca em uma página e verificar se o domínio dessa página corresponde ao domínio legítimo da marca.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Documento da Semana 4 - citado no Termo de Aceite de Entrega de 25 de Setembro]

Na quarta Semana, o objetivo foi iniciar o estudo das direções futuras que levantei para *phishing detection*. Dos 4 surveys selecionados, foram feitos a leitura e o fichamento de 2, sendo eles:

**Beyond the Mask: Benchmarking Multimodal Large Language Models to Expose Phishing Websites**

Apresenta o primeiro benchmark sistemático para avaliar o desempenho de Modelos de Linguagem Grandes (LLMs) multimodais de código aberto na detecção de sites de phishing. Os autores argumentam que, com o aumento da sofisticação dos ataques de phishing, as defesas tradicionais que analisam apenas URLs ou conteúdo HTML tornaram-se insuficientes. LLMs multimodais, capazes de processar simultaneamente texto, código e imagens, oferecem uma solução promissora, mas seu potencial nesta área ainda não foi explorado de forma sistemática. O objetivo do trabalho é preencher essa lacuna, fornecendo diretrizes empíricas para a seleção de modelos e modalidades de entrada em soluções de segurança baseadas em LLM.

Para avaliar os modelos, desenvolveram uma metodologia:

- Dataset: Foi criado um novo dataset de alta confiança contendo 102 instâncias (71 de phishing e 31 legítimas). Cada instância no dataset inclui três modalidades de dados: a
  URL, o código-fonte HTML bruto (truncado nos primeiros 5.000 caracteres) e uma screenshot (captura de tela) do site.
- Configurações de Detecção (Modalidades): O desempenho dos modelos foi testado em quatro cenários distintos para isolar a contribuição de cada tipo de dado:

  Entrada Combinada: O modelo recebe URL, HTML e screenshot juntos.

  Apenas URL: O modelo analisa apenas a string da URL.

  Apenas HTML: O modelo recebe apenas o código-fonte da página.

  Apenas Screenshot: A detecção é baseada unicamente na imagem do site.

- Modelos de Linguagem Selecionados: Foram escolhidos diversos LLMs multimodais de código aberto, variando em tamanho (de 2 a 11 bilhões de parâmetros) e arquitetura, como as famílias Qwen-VL, LLaVA e Phi-3.5-vision.
- Estratégias de Prompt: Foram investigadas três estratégias de prompt para entender como a formulação da instrução afeta o desempenho:

    Zero-shot Persona: Uma instrução mínima, pedindo ao modelo para atuar como um especialista em cibersegurança e classificar o site.

    Checklist-augmented Persona: Além da persona, o prompt inclui uma lista de verificação com pontos a serem considerados (ex: marca principal, elementos suspeitos, consistência).

    Multimodal Chain-of-Thought (MCoT): Fornece um processo de análise passo a passo (Análise Visual, Análise de Conteúdo, etc.) para guiar o raciocínio do modelo.

A análise revelou que o tipo de modalidade de entrada é o principal fator determinante do desempenho, com cada cenário apresentando desafios e resultados distintos.

- Detecção por URL:
    - Desempenho bom: Quase todos os modelos atingiram um desempenho próximo da perfeição, com F1-scores e recall muito altos (acima de 0.95). Isso indica que as URLs contêm sinais muito fortes e discriminativos para a detecção de phishing, que os LLMs conseguem explorar com alta confiabilidade. Modelos menores, como o Qwen2.5-VL-3B-Instruct, alcançaram pontuaçõe melhores que as dos modelos maiores.
- Detecção por Screenshot:
    - modalidade mais desafiadora. A maioria dos modelos apresentou um recall extremamente baixo, significando que eles falharam em identificar uma grande quantidade de sites de phishing. Os resultados apontam para uma deficiência fundamental na capacidade de compreensão visual detalhada dos atuais LLMs de código aberto para tarefas de segurança.
- Detecção por HTML:
    - Trade-off entre Precisão e Recall: Os modelos geralmente alcançaram um recall alto (identificaram bem os sites maliciosos), mas ao custo de uma precisão mais baixa (gerando mais falsos positivos). O conteúdo HTML parece acionar alertas de forma confiável, mas com menos exatidão.
- Detecção Combinada (URL + HTML + Screenshot):
    - Potencial e Complexidade: Embora teoricamente ofereça a maior quantidade de informação, os resultados foram mistos. Modelos grandes como o Qwen2.5-VL-7B-Instruct tiveram um desempenho excelente. No entanto,

outros, como o meta-llama-Llama-3.2-11B, falharam catastroficamente na configuração zero-shot, mostrando grandes desafios na fusão e alinhamento das diferentes modalidades de dados

| Model | Zero-shot Persona | | | | Checklist-augmented Persona | | | | Multimodal Chain-of-Thought | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | Acc | P | R | F1 | Acc | P | R | F1 | Acc |
| Qwen2-VL-2B-Instruct | **0.8780** | **0.5070** | **0.6429** | **0.6078** | 0.7719 | 0.6197 | 0.6875 | 0.6078 | 0.9032 | 0.3944 | 0.5490 | 0.5490 |
| Qwen2.5-VL-3B-Instruct | 0.9667 | 0.8169 | 0.8855 | 0.8529 | **0.9452** | **0.9718** | **0.9583** | **0.9412** | 0.9306 | 0.9437 | 0.9371 | 0.9118 |
| Qwen2-VL-7B-Instruct | 1.0000 | 0.8451 | 0.9160 | 0.8922 | 0.9701 | 0.9155 | 0.9420 | 0.9216 | **0.9706** | **0.9296** | **0.9496** | **0.9314** |
| Qwen2.5-VL-7B-Instruct | 1.0000 | 0.5211 | 0.6852 | 0.6667 | 1.0000 | 0.9014 | 0.9481 | 0.9314 | **0.9855** | **0.9577** | **0.9714** | **0.9608** |
| llava-1.5-7b-hf | 0.7209 | 0.8732 | 0.7898 | 0.6765 | **0.6961** | **1.0000** | **0.8208** | **0.6961** | 0.6931 | 0.9859 | 0.8140 | 0.6863 |
| llama3-llava-next-8b-hf | **0.9452** | **0.9718** | **0.9583** | **0.9412** | 0.6931 | 1.0000 | 0.8187 | 0.6863 | 0.6596 | 0.4493 | 0.5345 | 0.4510 |
| meta-llama-Llama-3.2-11B-Vision-Instruct-FP16 | 1.0000 | 0.0952 | 0.1739 | 0.3627 | **0.8904** | **0.9420** | **0.9155** | **0.8660** | 0.8148 | 0.6377 | 0.7154 | 0.6275 |
| UI-TARS-1.5-7B | **1.0000** | **1.0000** | **1.0000** | **1.0000** | 0.7717 | 1.0000 | 0.8712 | 0.7941 | 0.8765 | 1.0000 | 0.9342 | 0.9020 |
| Phi-3.5-vision-instruct | **0.9412** | **0.7164** | **0.8136** | **0.7451** | 0.9400 | 0.6912 | 0.7966 | 0.7353 | 0.8421 | 0.7059 | 0.7680 | 0.6667 |



embora os LLMs multimodais sejam uma fronteira promissora para a defesa contra phishing, sua aplicação prática exige uma abordagem matizada e "consciente da modalidade".

URL é a modalidade mais confiável

A dependência exclusiva de evidências visuais (screenshots) é um risco significativo com os modelos de código aberto atuais.

Prompting é essencial: A escolha do prompt pode modular o comportamento do modelo, otimizando-o para maior precisão (menos falsos positivos) ou maior recall (menos ameaças perdidas). O prompt com checklist foi o mais equilibrado para cenários complexos.

Especialização acima da escala: O desempenho não está garantido apenas pelo tamanho do modelo. Arquiteturas especializadas ou um foco de treinamento mais direcionado podem ser mais valiosos do que o número bruto de parâmetros

**Adaptive Linguistic Prompting (ALP) Enhances Phishing Webpage Detection in Multimodal Large Language Models**

Introduz e avalia uma nova técnica de prompt chamada Adaptive Linguistic Prompting (ALP), projetada para aprimorar a detecção de páginas de phishing usando as capacidades multimodais de LLMs de ponta como o GPT-4o e o Gemini 1.5 Pro. O ALP é um método de raciocínio semântico estruturado que utiliza a abordagem *few-shot* (fornecendo alguns exemplos) para guiar os LLMs. O objetivo é ensinar o modelo a analisar padrões linguísticos, identificar gatilhos de urgência e reconhecer a linguagem manipuladora que são comuns em conteúdos de phishing. A principal contribuição do trabalho é demonstrar que a engenharia de prompt, por si só, pode melhorar significativamente o desempenho da detecção, independentemente de alterações no modelo de IA.

- Modelos e Dataset: Os experimentos foram conduzidos com os modelos GPT-4o e Gemini 1.5 Pro, conhecidos por sua capacidade de correlacionar dados visuais e textuais. Foi utilizado um dataset já existente e curado (de Lee et al., 2024), que foi filtrado para otimizar a eficiência computacional, resultando em 289 instâncias de phishing e 311 instâncias de sites benignos.
- Análise Dupla com ALP: A detecção foi dividida em dois processos distintos, ambos utilizando prompts ALP de 8 exemplos (*8-shot*):

    Multimodal Webpage Analysis (MWA): Esta análise utiliza as capacidades multimodais completas, recebendo o código HTML e a screenshot da página. O prompt instrui o modelo a estruturar sua análise em reconhecimento de marca, análise de linguagem (dicção e sintaxe) e avaliação da screenshot.

    URL Structure Analysis (USA): Esta análise foca exclusivamente na URL extraída do site. O prompt guia o modelo a focar em características como nome de domínio, protocolo (HTTP vs. HTTPS) e caminho da URL para identificar indicadores de phishing.

- Combinação dos Resultados: Foi implementada uma regra "consciente do risco" para unificar os resultados de MWA e USA. Se ambos concordam, o rótulo é aceito. Se discordam, a página é classificada como phishing se a análise de URL (USA) a identificar como tal, ou se a análise multimodal (MWA) tiver uma pontuação de confiança superior a 8.5.

Os resultados confirmaram a eficácia do ALP e revelaram as forças distintas de cada modelo.

- Eficácia do ALP: A abordagem combinada, aprimorada pelo ALP, alcançou um F1-score de 0.93 com o GPT-4o, superando significativamente os métodos de linha de base e outras abordagens na literatura.
- Forças Complementares dos Modelos:
    - O Gemini 1.5 Pro se destacou na análise MWA, sendo mais eficaz em identificar anomalias na representação da marca e na consistência entre elementos visuais e textuais.
    - O GPT-4o foi superior na análise USA, se destacando na identificação de estruturas de URL suspeitas e inconsistências em protocolos de segurança.

| Model | Approach | Precision | Recall | F1 |
|---|---|---|---|---|
| GPT-4o | Baseline | 0.91 | 0.91 | 0.91 |
| | MWA | 0.80 | 0.89 | 0.84 |
| | USA | 0.91 | 0.91 | 0.91 |
| | Combined Analysis | 0.91 | 0.94 | 0.93 |
| Gemini 1.5 Pro | Baseline | 0.76 | 0.85 | 0.81 |
| | MWA | 0.94 | 0.87 | 0.90 |
| | USA | 0.88 | 0.85 | 0.87 |
| | Combined Analysis | 0.91 | 0.92 | 0.91 |

Table 1: Performance comparison across baseline, MWA, USA, & Combined Analysis

- Impacto da heurística "Suspicious-First":  uma simples instrução adicionada ao prompt da análise de URL (USA): *"Se você não tiver certeza e achar que um link é suspeito de atividade de phishing, rotule-o como phishing"*. Apenas essa regra fez a precisão do GPT-4o na análise de URLs saltar de 81% para 91%, demonstrando que a engenharia de prompt pode ser tão impactante quanto a arquitetura do modelo.

O estudo reconhece algumas limitações, como o uso de um dataset que pode não cobrir todas as táticas de phishing mais recentes, a dependência de LLMs proprietários e caros, e a falta de testes em conteúdos que não sejam em inglês.

**Multimodal framework for phishing attack detection and mitigation through behavior analysis using EM-BERT and SPCA-BASED EAI-SC-LSTM**

Propõe um framework multimodal e abrangente para detectar e mitigar ataques de phishing. A principal inovação do trabalho é sua capacidade de analisar múltiplas fontes de ataque simultaneamente, incluindo SMS, E-mail e links de URL, uma área que, segundo os autores, não foi suficientemente explorada em trabalhos anteriores. O sistema foi projetado para ir além da análise de conteúdo simples, incorporando uma análise de comportamento do usuário e extraindo uma vasta gama de características textuais, estruturais e de Javascript para alcançar uma detecção de alta precisão.

O framework utiliza uma sequência de algoritmos de machine learning e deep learning personalizados e otimizados para processar os dados em várias etapas:

- Entrada de Dados Multivetor: O sistema começa coletando dados de três fontes distintas: um dataset de SMS (SSC), um de E-mail (PEC) e um de páginas web/URLs (WPD).
- Extração e Análise de Características: A partir dos dados de entrada, o sistema realiza uma análise profunda e multifacetada:

    Análise Textual: O conteúdo de SMS e e-mails é extraído, e as URLs passam por uma análise textual exaustiva que inclui:

        Análise de Comportamento: Avalia características como nome de domínio, endereço IP e outros para prever o comportamento do usuário.

        Análise de Conteúdo, Javascript e URL: Extrai dezenas de características relacionadas a tags HTML, scripts, idade do domínio, tráfego da página, etc.

    Análise Estrutural: Para as URLs, o sistema também extrai arquivos CSS e imagens para analisar a estrutura visual e de layout da página.

- Processamento com Algoritmos Otimizados:

    EM-BERT (Word Embedding): Os dados textuais (de SMS, E-mail e da análise textual da URL) são convertidos em vetores numéricos por meio do EM-BERT, uma versão aprimorada do BERT que utiliza uma inicialização de pesos otimizada para uma convergência mais rápida e melhor desempenho.

    SPCA (Extração de Características): As características estruturais (CSS, imagens) são processadas pelo SPCA (Spherical Principal Component Analysis), um método que reduz a dimensionalidade dos dados (seleciona o

que é mais importante) preservando informações cruciais e relações geométricas, superando as limitações do PCA tradicional.

CSOA (Seleção de Características): Para otimizar o processo, o CSOA (Cauchy distribution-based Seagull Optimization Algorithm) é usado para selecionar o subconjunto ideal de características da URL e da análise estrutural, garantindo que apenas os dados mais relevantes sejam enviados ao classificador.

EAI-SC-LSTM (Classificação Final): O classificador final é o EAI-SC-LSTM, um modelo LSTM modificado.

SC (SERF COLU): Utiliza uma função de ativação especial para resolver o problema de "desvanecimento de gradiente" do LSTM, tornando o aprendizado mais eficiente.

EAI (Explainable AI): Incorpora a "IA Explicável", que torna as decisões do modelo mais transparentes e compreensíveis para os usuários humanos.

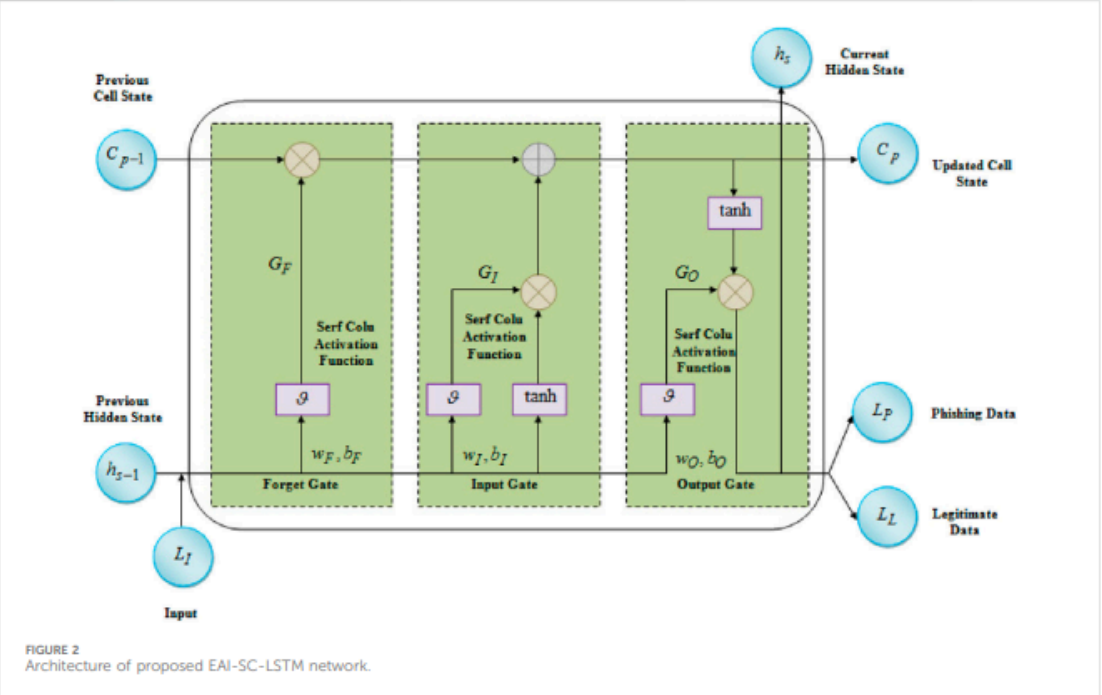
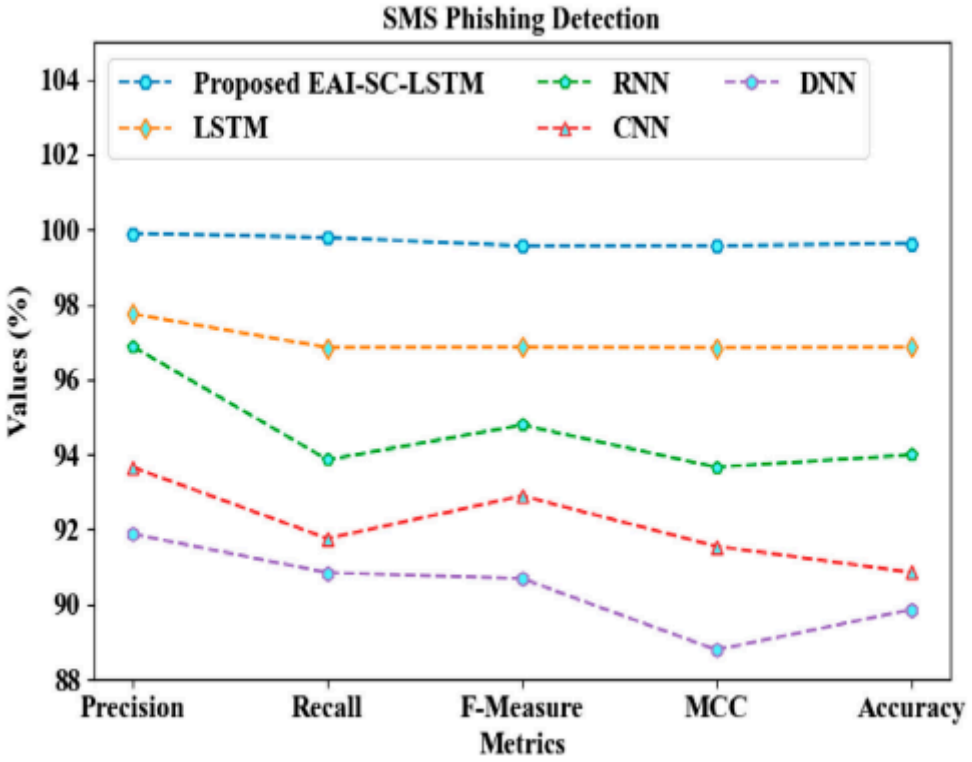


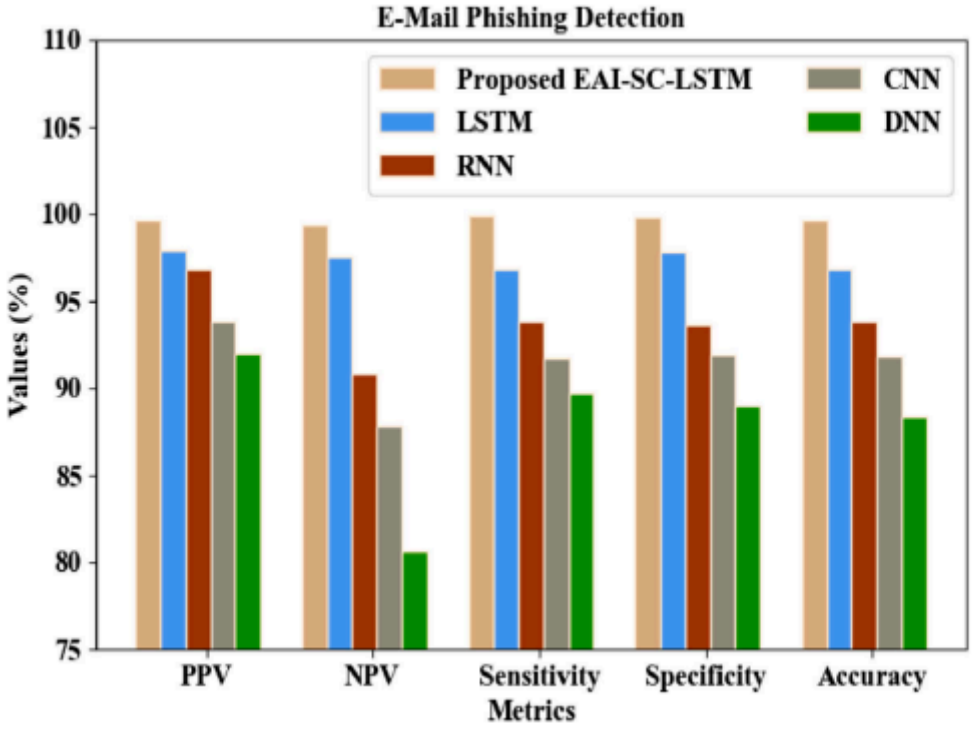

FIGURE 2
Architecture of proposed EAI-SC-LSTM network.

- Mitigação e Alerta ao Usuário: As URLs e dados identificados como phishing são adicionados a uma blacklist. Quando uma nova URL é submetida, ela é primeiro

verificada nesta lista; se houver uma correspondência, um alerta é enviado imediatamente ao usuário.

Obtiveram bons resultados:






- O modelo EAI-SC-LSTM alcançou uma acurácia superior a 99,5% nos três datasets: 99,627% para SMS (SSC), 99,645% para E-mail (PEC) e 99,541% para Páginas Web (WPD).
- O sistema superou consistentemente os modelos de deep learning tradicionais (LSTM, RNN, CNN, DNN) em todas as métricas avaliadas, incluindo acurácia, precisão, recall e tempo de treinamento.
- Graças ao uso de algoritmos otimizados para seleção de características (CSOA) e à arquitetura aprimorada do classificador, o sistema demonstrou ser mais rápido, com um tempo de treinamento significativamente menor em comparação com os outros modelos.

# APÊNDICE 3

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 1 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa quinta Semana, o objetivo foi iniciar o estudo de um artigo científico e iniciar também a sua reprodução, visando compreender a fundo as metodologias, as fontes de dados utilizadas e as técnicas de avaliação. Iniciei os primeiros passos para a reprodução prática de alguns desses conceitos, o que é fundamental para solidificar o entendimento

- From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks

Apresenta o PhishOracle, uma ferramenta que gera automaticamente páginas de phishing convincentes ao inserir características maliciosas – como pop-ups de login falsos e alterações visuais em logos – em páginas web legítimas.

- Biblioteca BeautifulSoup para extrair informações como HTML
- Transforma o HTML em uma estrutura de dados em forma de árvore, na qual cada tag (<a>, <form>, etc) vira um objeto que pode ser facilmente localizado, inspecionado e modificado.
- Aplica técnicas ( Técnicas utilizadas na geração de páginas de phishing ), por exemplo Pop-up Login): Insere um novo código HTML/JavaScript na página que, ao clicar em um botão, abre um modal de login falso. O atributo action do formulário dentro desse modal é alterado para apontar para um script controlado pelo atacante (action="local.php").

Quando testados contra essas páginas, os modelos de detecção tradicionais (baseados em Machine Learning e Deep Learning, como Stack Model, VisualPhishNet e Phishpedia) sofreram quedas drásticas em sua performance, com a detecção caindo de 16% a 40%.

- Iniciei também a reprodução do artigo(https://github.com/enzodias1/phish-oracle-residencia)

, realizei o download das páginas e geração da versão “phishing_webpage.html” e Injeção de várias features visuais/comportamentais (desabilitar clique direito, esconder status bar, mudar action de forms, look-alike domain, etc.).

* Páginas geradas por phishing

Documento da Semana 05

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima Semana, pretendo continuar na reprodução desse artigo e, se concluir. iniciar a reprodução do segundo (KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection)

**Observação: [caso precise fazer alguma observação, de qualquer “natureza”]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Técnicas utilizadas na geração de páginas de phishing - citado no Termo de Aceite de Entrega de 1 de Outubro]

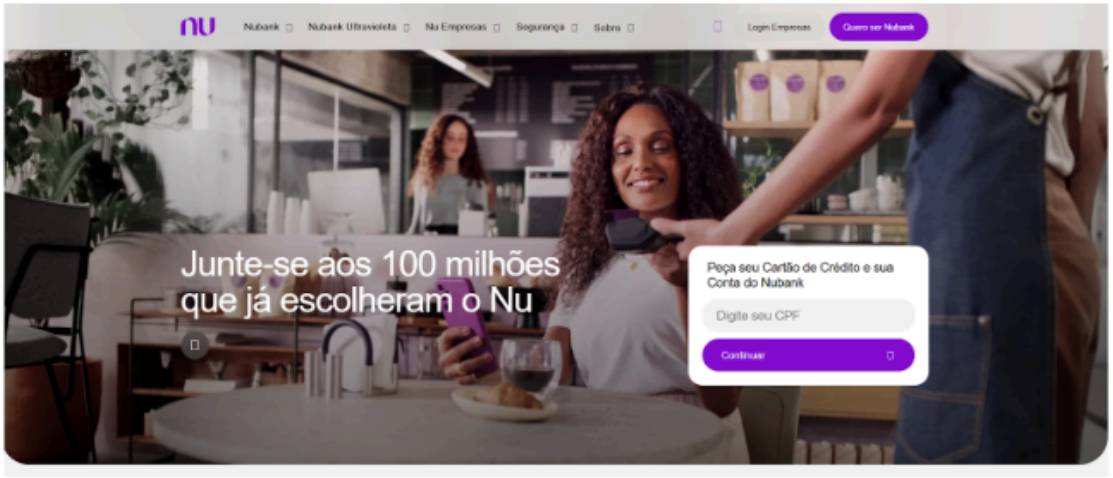
Table 2. List of Phishing Features Based on the Content and Visual Attributes in a Web page

| Type | Naming | Phishing Features | Explanation |
|---|---|---|---|
| Content | *C1* | Hypertext reference | Updating `href` with: `href="#"`, `"#content"`, `"#skip"`, or `"Javascript:void(0)"` |
| | *C2* | Disable key functions | Restrict users to view source code by disabling *f11* and *Ctrl + U* |
| | *C3* | `href` lookalike characters | Alphabets in `href` value are replaced by lookalike characters E.g.: replacing a with a lookalike character `ā, á, ă` |
| | *C4* | Hide links appearing on status bar | This feature does not allow users to view the `href` link even on hovering on the link |
| | *C5* | Disable anchor tags | Does not allow a user to navigate to other pages by clicking on the anchor tags |
| | *C6* | Replace blank space with a character | Blank spaces present in the container elements of HTML like `h1, p, span` tags are replaced with characters with `style=color: transparent;` |
| | *C7* | Save credentials | The credentials entered in the HTML form input tags are stored in a local file |
| | *C8* | Disable other login buttons | Web pages containing login buttons like Google, GitHub, LinkedIn, etc., are disabled and the credentials entered in the visible login page are stored locally. |
| | *C9* | Pop-up Login | Clicking the login or sign-up button triggers a login page to appear. The action field in the `<form>` tag is then altered, to store the credentials locally |
| | *C10* | Pop-up Login by clicking on a tags | Clicking the anchor tags, opens up a login page and the credentials are stored locally |
| | *C11* | IFrame tag with login page | `<iframe>` tags are added with a login page, the credentials entered are stored locally |
| | *C12* | Add dummy tags | Dummy `<img>`, `<link>`, `<script>`, `<a>`, and `<div>` tags are added to increase the DOM structure of web page |
| Visual | *V1* | Body Opacity | CSS `opacity` adjusts web page transparency from 0 to 1 |
| | *V2* | Text Styling | The `font-family` changes the style of web page text |
| | *V3* | Opacity on Logo | CSS `opacity` property makes the logo image transparent `<img>` tags containing images with `.png` and `.svg` extensions are considered and a `opacity` of 0.1 to 0.35 is added to make the logo transparent |
| | *V4* | Adding Watermark on logo | A watermark is added either on the bottom right corner or diagonally on logo image |
| | *V5* | Image Transformations | Techniques like adding (1) clockwise or anti-clockwise rotation, (2) Gaussian blur, (3) gray-mesh, and (4) noise, transforms the logo image |

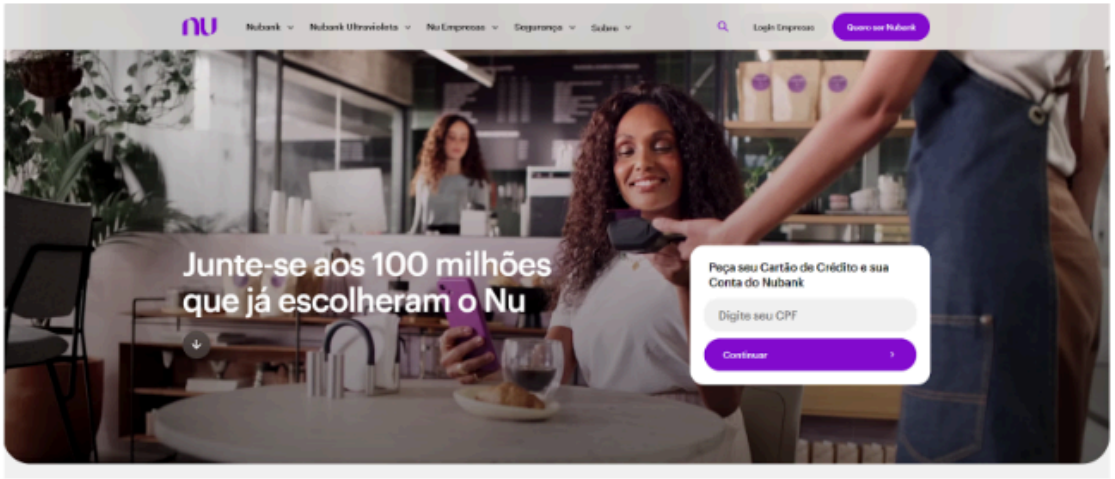
[ Páginas geradas por phishing - citado no Termo de Aceite de Entrega de 1 de Outubro]

**Nubank**

**página gerada**




**página verdadeira**




[Documento da Semana 5 - citado no Termo de Aceite de Entrega de 1 de Outubro]

Na quinta Semana, o objetivo foi iniciar o estudo de 2 surveys relacionados a direções futuras que levantei para *phishing detection*. Cada um possui um artefato interessante, por isso, pretendo reproduzir ambos, começando pelo From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks

O artigo aborda um dos problemas mais persistentes na segurança cibernética: os ataques de phishing. Apesar dos avanços na detecção automática usando Machine Learning (ML) e Deep Learning (DL), os autores identificam uma lacuna crítica: a falta de avaliação robusta da resiliência desses modelos contra ataques adversariais.

O problema central é que as ferramentas existentes para gerar páginas de phishing (como BlackEye, ZPhisher) são estáticas. Elas possuem um conjunto limitado e pré-desenhado de páginas para um número pequeno de marcas, o que:

Não reflete a diversidade do phishing real.

Impede a avaliação da capacidade dos modelos de generalizar e detectar variações novas de ataques.

Muitos desses URLs já estão bloqueados, tornando-os inúteis para testes de evasão.

- A Ferramenta PhishOracle

Para superar essas limitações, os autores desenvolvem e apresentam o PhishOracle, uma ferramenta automatizada de geração de páginas de phishing adversariais. Suas características fundamentais são:

Geração Dinâmica: Diferente das ferramentas estáticas, o PhishOracle "clone e modifica". Ele parte de uma página legítima real (de qualquer marca) e incorpora nela características de phishing.

Características Diversificadas: A ferramenta incorpora aleatoriamente um conjunto de 17 características de phishing (listadas na Tabela 2 do artigo), categorizadas em:

12 Baseadas em Conteúdo (HTML/URL): Modificam a funcionalidade da página. Exemplos: alterar ações de formulários para salvar credenciais localmente (C7), desativar botões de login de terceiros (C8), adicionar pop-ups de login (C9), desativar a navegação por links (C5), ofuscar caracteres em URLs (C3).

5 Baseadas em Elementos Visuais: Alteram a aparência para enganar tanto usuários quanto modelos baseados em visão computacional. Exemplos: ajustar a opacidade do logo (V3), adicionar marca d'água (V4), aplicar rotação, desfoque gaussiano ou ruído ao logo (V5).

Flexibilidade e Realismo: Por ser dinâmico, pode gerar múltiplas versões de uma única página legítima, criando um conjunto de testes muito mais diverso e realista. Ele também

gera URLs "lookalike" (parecidos) usando homóglifos e prefixos/sufixos (ex: facebook-login.co).

Os autores conduziram uma avaliação abrangente para responder à pergunta: "Quão robustos são os state-of-the-art detectores de phishing contra ataques adversariais?"

Modelos Selecionados para Teste:

Stack Model (ML): Um modelo clássico baseado em características de URL e HTML.

VisualPhishNet (DL): Um modelo que compara a similaridade visual de screenshots completas da página.

Phishpedia (DL): Um modelo mais avançado que identifica especificamente logos nas screenshots e os compara com uma lista de marcas legítimas.

Detector baseado em LLM (MLLM): Usa o modelo Gemini para analisar o contexto da página (via HTML ou screenshot) para identificar a marca e verificar se ela corresponde ao domínio.

Conjuntos de Dados:

CleanSet: Dados "limpos" de phishing e páginas legítimas, usados para (re)treinar os modelos.

EvasionSet: Dados adversariais gerados pelo PhishOracle, usados para testar a robustez. Diferentes EvasionSets foram criados para se alinhar aos requisitos de cada modelo.

Métricas de Avaliação: As métricas padrão de classificação foram usadas: Precisão (quantos dos que foram flagados como phishing eram realmente phishing) e Recall (quantos dos phishing reais foram corretamente detectados). A queda no Recall é a principal medida de perda de robustez.

A Tabela 6 do artigo resume os resultados, mostrando uma queda dramática no desempenho dos modelos tradicionais:

Stack Model (ML): O Recall caiu ~28% (de ~98% para ~71%). Isso mostra que alterações simples no conteúdo HTML e URL são suficientes para enganar modelos baseados em características.

VisualPhishNet (DL): O Recall caiu ~16% (de ~88% para ~73%). Técnicas de transformação de logo confundem seu algoritmo de similaridade visual.

Phishpedia (DL): Teve a queda mais severa, ~41% no Recall (de ~82% para ~40%). Por depender criticamente do reconhecimento de logos, as transformações visuais (opacidade, marca d'água) o tornam altamente ineficaz.

Imagine que, em condições normais, o modelo Phishpedia consegue identificar e bloquear 82 de cada 100 páginas de phishing. Após sofrer um ataque adversarial com as técnicas do PhishOracle (ex: logo com opacidade, marca d'água), a eficácia dele cai para apenas 40 de cada 100. Isso significa que 42 páginas que seriam bloqueadas agora passam direto, um aumento enorme no risco.

LLM (Gemini): Demonstrou a maior robustez. A versão que analisa screenshots (LLM-PD^S) teve uma queda de apenas ~10% na pontuação F1, mantendo alta precisão. O LLM mostrou capacidade de entender o contexto da página além do logo, mas ainda foi enganado em alguns casos (ex: associando um domínio signup-yahoo.com à marca Yahoo).

As páginas geradas pelo PhishOracle foram hospedadas e submetidas ao VirusTotal (+90 fornecedores de segurança) por 9 dias.

Resultado: A detecção foi lenta e inconsistente. Páginas de marcas menos conhecidas permaneceram praticamente indetectáveis. Mesmo para marcas populares (Microsoft, Yahoo), a maioria esmagadora dos fornecedores (~80%) não as detectou nas primeiras 24-48 horas, evidenciando uma janela de oportunidade crítica para os atacantes.

Um estudo com 52 participantes (estudantes e profissionais de TI) mostrou que as páginas adversariais são eficazes também contra humanos.

Resultado: Em média, ~48% das páginas de phishing geradas pelo PhishOracle foram classificadas erroneamente como legítimas quando os usuários julgaram apenas pela screenshot (sem ver a URL). Isso valida que as técnicas visuais empregadas são suficientemente sutis para enganar não apenas máquinas, mas também o alvo final do phishing: as pessoas.

Vulnerabilidade Sistêmica: Os modelos de detecção de phishing atuais (ML e DL) são fundamentalmente frágeis e não são robustos contra ataques adversariais simples e de baixo custo.

O Potencial dos LLMs: Modelos Multimodais de Grande Linguagem (MLLMs) emergem como uma abordagem muito mais promissora e resiliente, graça à sua capacidade de compreensão contextual e semântica. Eles representam um avanço significativo, mas não são uma solução perfeita, pois ainda podem ser evadidos.

Ferramenta para Melhoria Proativa: O PhishOracle não é uma ferramenta para atacantes, mas sim um instrumento de "teste de estresse" para a comunidade de segurança. Ele permite identificar pontos cegos nos detectores antes que sejam explorados maliciousmente.

Ameaça Iminente: A lentidão das soluções de segurança comerciais e a suscetibilidade dos usuários reforçam que o phishing adversarial é uma ameaça prática e imediata, não apenas um conceito acadêmico.

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 9 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta semana, foi dada continuidade ao estudo do artigo cientifico "From ML to LLM". Foram realizadas correções na estrutura de pastas do projeto para padronizar os caminhos e implementado um tratamento para imagens RGBA, que agora são convertidas para RGB ao salvar como JPEG ou mantidas como PNG para evitar erros.

Foram conduzidas rodadas de testes com as URLs para avaliar o desempenho e o tempo de geração, utilizando **CDFs (Cumulative Distribution Function)**, ou Funções de Distribuição Acumulada. Essa função mede a probabilidade de uma variável (neste caso, o tempo de geração) ser menor ou igual a um certo valor, mostrando, por exemplo, qual porcentagem das páginas foi gerada em menos de 2 segundos.

1. **Primeira Rodada:** Utilizando um split aleatório, os resultados foram baixos, como esperado, devido à pequena quantidade de dados (15 para cada classe) e features ainda em desenvolvimento. O CDF de tempo de geração mostrou que, embora a maioria das execuções seja rápida (mais da metade em menos de 2 segundos), uma minoria pode levar até 26 segundos.
2. **Segunda Rodada (GradientBoosting):** Neste teste, foi usado o **Gradient Boosting**, um método de aprendizado de máquina que constrói um modelo de previsão forte a partir da combinação sequencial de vários modelos mais fracos (geralmente árvores de decisão). As métricas foram perfeitas (AUC de 1.0) , o que indicou que o modelo estava "memorizando" características dos sites, já que domínios iguais estavam presentes nos conjuntos de treino e teste, causando vazamento de dados.
3. **Terceira Rodada (GradientBoosting + GroupShuffleSplit):** Para corrigir o vazamento, foi utilizada a estratégia **GroupShuffleSplit**, que separa os dados garantindo que todas as observações de um mesmo grupo (no caso, o mesmo domínio de site) não apareçam ao mesmo tempo nos conjuntos de treino e teste. Com isso, os resultados se mostraram mais realistas e robustos, com um AUC de 0.9. O tempo de geração também se mostrou mais consistente, com um máximo de 11 segundos.

Documento da Semana 06

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima semana pretendo focar em aumentar a escala e a variedade do dataset. Para isso, foi identificada a lista Tranco (https://tranco-list.eu/), que ranqueia 1 milhão de sites por popularidade. O plano é utilizar os 1.000 primeiros sites dessa lista para gerar um dataset diverso, relevante e baseado na popularidade real dos sites na internet.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

Não pude estar presente na apresentação dessa Semana pois estou em Natal-RN representando, com muito orgulho, a UFG nos Jogos Universitários Brasileiros (JUBS)!

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Documento da Semana 6 - citado no Termo de Aceite de Entrega de 9 de Outubro]

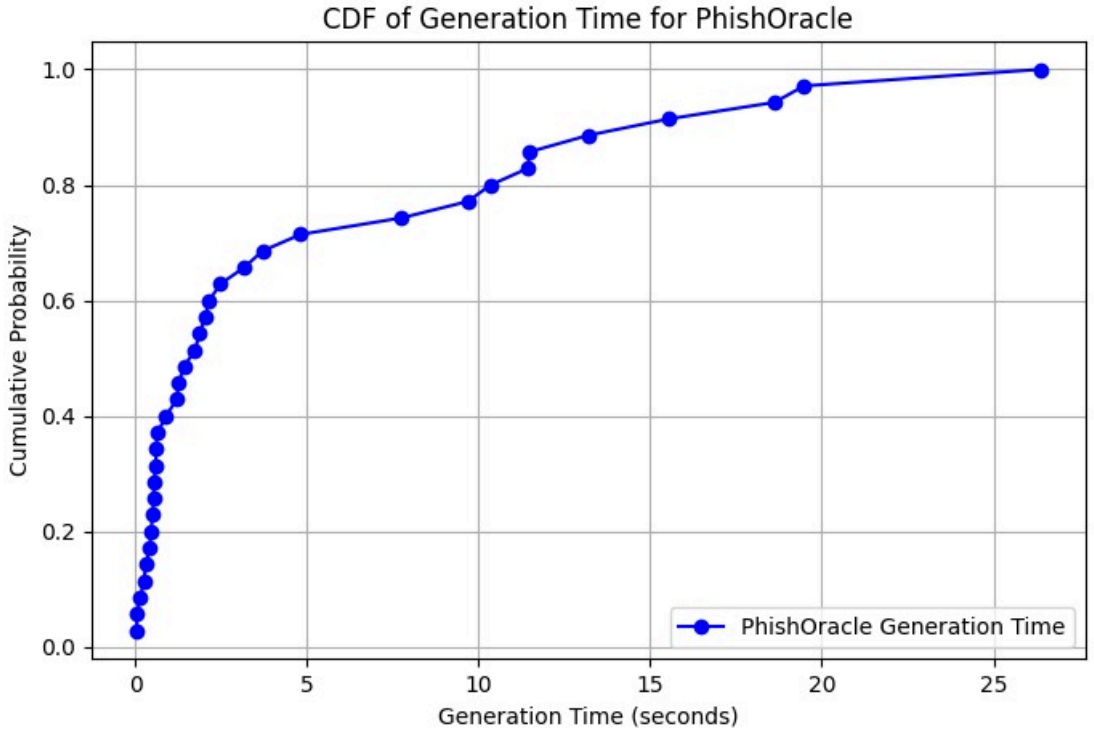
Nesta sexta Semana, o objetivo foi continuar o estudo e a reprodução do survey From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks

Revisei a árvore do projeto e vi que haviam caminhos incorretos, com isso os padronizei. Foi feito também o tratamento de imagens RGBA, quando for necessário salvar JPEG, é convertido para RGB ou salvo como PNG para não dar mais o erro de “cannot write mode RGBA as JPEG”

Testei as URLs feitas na última semana, gerando CDFs (Cumulative Distribution Function), ou Funções de Distribuição Acumulada, para o tempo de geração e também as métricas

O CDF mede o tempo por página (download/localização de recursos + injeção de features + render/screenshot, dependendo do script)

Nesse primeiro gráfico a maioria das gerações do PhishOracle é muito rápida (mais da metade em menos de 2 segundos), mas existe uma minoria significativa de casos que podem demorar bastante, chegando a até 26 segundos. Serve como *sanity check*: confirma que a maioria roda rápido, mas é otimista (variância baixa por N pequeno).




```
Distribuição de classes: {0: 15, 1: 15}

=== Relatório de classificação ===
              precision    recall  f1-score   support

           0      0.333     0.200     0.250         5
           1      0.333     0.500     0.400         4

    accuracy                          0.333         9
   macro avg      0.333     0.350     0.325         9
weighted avg      0.333     0.333     0.317         9

AUC: 0.45
Matriz de confusão:
 [[1 4]
 [2 2]]
```

Nessa primeira rodada, já era o esperado esses resultados “baixos”, visto que pouquíssimos dados (15/15), *split* aleatório, e features ainda “rasas”.

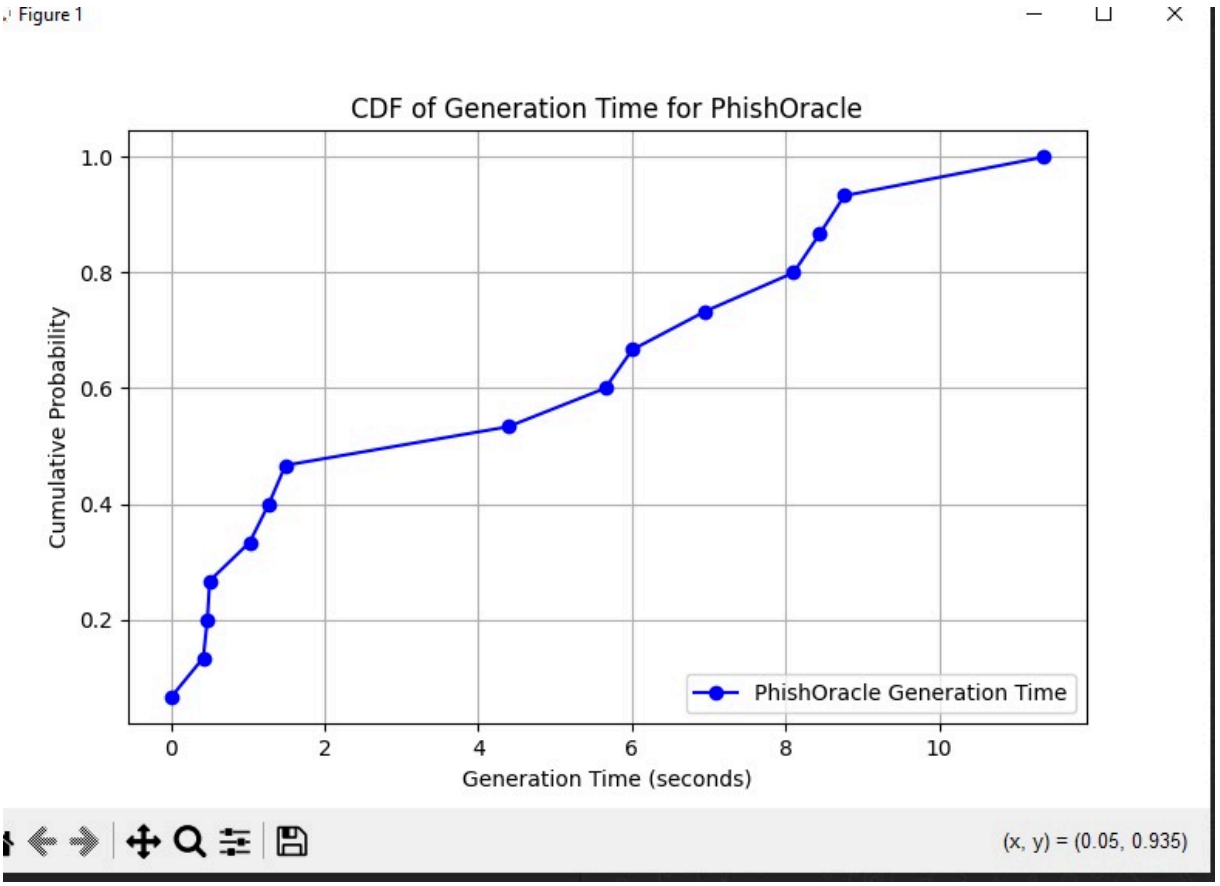
Após isso, fiz uma nova rodada de testes, porém com GradientBoosting (sem controle de grupos)

```
baseline.py
Distribuição de classes: {0: 15, 1: 15}
=== Relatório de classificação ===
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         5
           1      1.000     1.000     1.000         4

    accuracy                         1.000         9
   macro avg      1.000     1.000     1.000         9
weighted avg      1.000     1.000     1.000         9

AUC: 1.0
Matriz de confusão:
 [[5 0]
 [0 4]]
```

As métricas ficaram “perfeitas”, o que deu para perceber que páginas do mesmo domínio em treino e teste deixam pistas fáceis (mesmo layout, mesmas contagens de elementos etc.). O modelo “memoriza” peculiaridades do site.

Depois, GradientBoosting + GroupShuffleSplit por domínio (gerar divisões de treino/teste que respeitem essa estrutura, garantindo que as observações de um mesmo grupo não apareçam simultaneamente nos conjuntos de treino e de teste.)

Mostra um sistema com desempenho mais consistente e rápido no geral. O tempo máximo de geração é de apenas 11 segundos, e não há a mesma "cauda longa" de tarefas extremamente lentas.




```
Distribuição de classes: {0: 15, 1: 15}
Domínios únicos: 15
Tamanho treino: 20 (10 domínios) | teste: 10 (5 domínios)

=== Relatório de classificação ===
              precision    recall  f1-score   support

           0      1.000     0.800     0.889         5
           1      0.833     1.000     0.909         5

    accuracy                          0.900        10
   macro avg      0.917     0.900     0.899        10
weighted avg      0.917     0.900     0.899        10

AUC: 0.9
Matriz de confusão:
 [[4 1]
 [0 5]]
```

**t***reino e teste separados por domínio* (emulando “marcas novas” no teste). sem perfeição suspeita. Dá para inferir que respeitar grupos remove o vazamento.

Em relação ao artigo científico, ele treina com milhares com um número muito superior de páginas, porém já consegui definir um pipeline (geração → screenshot → features → treino com split por domínio). Agora é necessário maior escala e variedade. Com isso, encontrei o https://tranco-list.eu/ , que apresenta 1 milhão de sites ranqueados com base na sua popularidade. Com isso, pretendo pegar os 1000 primeiros para gerar um dataset diverso e relevante, baseado na popularidade real dos sites na internet

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião (“Gate”) de aprovação: 15 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta Semana, foi dada continuidade ao estudo do artigo cientifico "From ML to LLM". O objetivo foi ampliar o conjunto de páginas legítimas para treinar/testar o baseline do PhishOracle.

Alvo do artigo: avaliar robustez de detectores de phishing quando a página é “levemente” adulterada (evasões visuais + de conteúdo).

Minha ideia: montar um pipeline de dados ⇒ gerar um conjunto Clean (legítimo) e um conjunto Evasion (mesmas páginas, porém com truques que confundem modelos).
Depois treinar um classificador baseline no estilo do paper (um ensemble de árvores tipo Gradient Boosting) e medir como as métricas caem ao passar de Clean para Evasion.

Baixei a lista Tranco Top 1000, corrigindo problemas de parsing (linhas com/sem cabeçalho, linhas contendo já https:// ), os domínios foram baixados.

Problemas de tempo e alguns domínios falharam: Alguns hosts são lentos, ficam em CDNs geograficamente distantes. Alguns bloqueios, pois muitos assets (analytics/ads/fonts) retornam 403/404/timeout em requisições automatizadas. (por ex a microsoft.com) possuem proteção anti-bot. Mas mesmo quando assets falham, o index.html principal costuma ser salvo. O pipeline usa esse HTML como base.

Depois, para cada domínio baixado, foi criado uma versão “phish” (Evasion) aplicando features: desabilitar clique direito/atalhos, esconder status bar, alterar action para salvar credenciais, degradações visuais (watermark, blur, ruído JPEG etc.).

Documento da Semana 07

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana, pretendo seguir implementando a ideia:

Construir os conjuntos CleanSet1 = páginas legítimas (os nossos index.html) e EvasionSet1 = páginas phish geradas (os phishing_webpage.html).
- > gera screenshots (PNG) para Clean e Evasion; extrair contagens de links, formulários, scripts, palavras-chave (“login”, “senha”, “click”…), etc. e visual estatísticas simples do screenshot (contagem de componentes, elementos suspeitos, etc.).

Por fim, treino e avaliação. Estou usando GradientBoosting (um ensemble de pequenas árvores) porque ele aprende padrões não-lineares e combina vários classificadores fracos para formar um forte. O paper usa a família GBM/XGBoost por performance/estabilidade; comecei com essa versão por ser leve e reprodutível e posso migrar para XGBoost/LightGBM quando o dataset crescer.

**Observação: [caso precise fazer alguma observação, de qualquer “natureza”]**

## ACEITE DA ENTREGA:

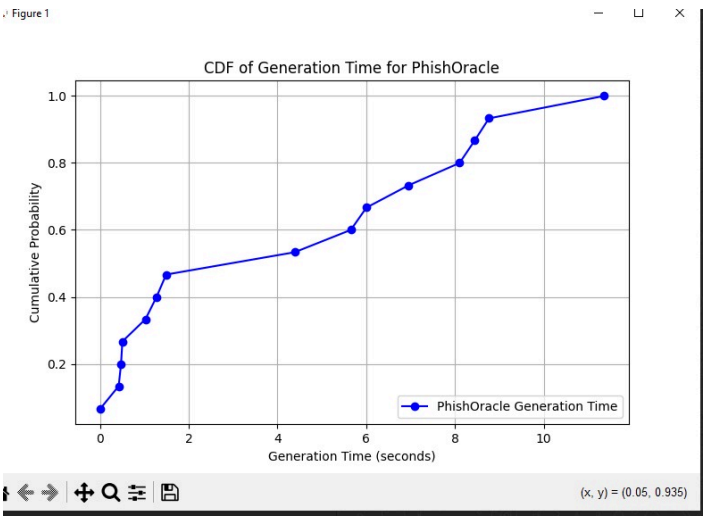
**CEDRIC LUIZ DE CARVALHO:** Go!

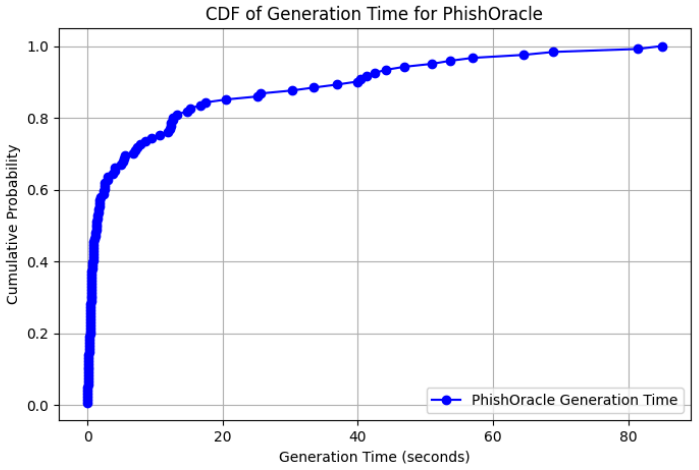
[Documento da Semana 7 - citado no Termo de Aceite de Entrega de 15 de Outubro]

Na sétima Semana, o objetivo foi continuar o estudo e a reprodução do survey From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks

O CDF mede o tempo por página (download/localização de recursos + injeção de features + render/screenshot, dependendo do script)

Repare que na primeira imagem, que é da Semana anterior, como haviam poucas páginas, a cauda está menor. Agora, com mais páginas adicionadas  a CDF “estica” para a direita (páginas pesadas/outliers) e fica mais estável.




O que foi feito:

Ingestão da Tranco Top 1000

Corrigido parsing do CSV (linhas com/sem cabeçalho; linhas cruas vs. com https://).

Pipeline cria remaining_urls.txt no formato normalizado https://<domínio>.
Downloader (simple_downloader.py) revisado:

Normalização robusta de URL.

Fallback de nome de arquivo com hash curto para evitar erros de path/charset.
Salva index.html, reescreve recursos para local_resources/ e registra source_url.txt.

Coleta
Rodada longa de download (≈10h): ~650/1000 domínios baixados.
Falhas esperadas e registradas em log:
403/404/timeout/SSL/anti-bot/CDN (ex.: Microsoft/WhatsApp/analytics/fonts).
Em muitos casos os assets falham, mas o index.html principal é salvo (suficiente para o nosso pipeline).

Geração de Evasion (opção B)
Script: add_visual_features_main.py (com medição contínua de tempo).
Features aplicadas (amostras):
esconder status bar, desabilitar clique direito/atalhos, desativar <a>, modificar action para capturar credenciais, watermark/ruído/blur/variações de logo.

Preparação para Clean/Evasion
Verificação: contagem de domínios que já possuem phishing_webpage.html.
(Com a coleta parcial e a geração em curso > o número cresce conforme o script avança.)

datasets/CleanSet1/html (dos index.html)
datasets/EvasionSet1/html (dos phishing_webpage.html)

# APÊNDICE 4

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 22 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nesta oitava Semana, foi dada continuidade ao estudo do artigo científico de referência, "From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks". O objetivo foi avaliar o desempenho e a robustez de um modelo de detecção de phishing, o **Stack Model,** contra os conjuntos de dados adversários gerados pela ferramenta PhishOracle.

O Stack Model é uma abordagem de Machine Learning que utiliza características de URL e HTML para classificar uma página como phishing ou legítima. O artigo demonstrou que este modelo não é robusto. Ao ser testado contra o EvasionSet¹ (1000 páginas de phishing adversárias), seu recall (capacidade de detecção) caiu de 98,32% para 70,86%.

Diferente do artigo, meu modelo se mostrou inesperadamente robusto, tendo 96,4% de recall.

- Isso é bom ou ruim?

O segundo modelo de detecção de phishing que o artigo testa é o **Phishpedia**, um sistema de Deep Learning que detecta phishing visualmente, identificando o logo da marca na página e comparando-o com o domínio do site. O EvasionSet² (170 páginas de phishing adversárias) foi criado especificamente com transformações de logo (opacidade, ruído, etc.). O recall do modelo despencou de 81,63% para 40%.

Gerei o EvasionSet², porém ainda não implementei o Phishpedia, então, por curiosidade quis testar e validar o que acontece com um modelo de conteúdo quando confrontado com ataques visuais. Capacidade de detecção de 6.5%, ou seja, das 170 páginas, 159 foram incorretamente classificadas como "legítimas".

Documento da Semana 08

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Para a próxima Semana, pretendo

- Investigar melhor por que minha geração do EvasionSet¹ ficou tão “robusto”.
- Implementar e treinar o modelo Phishpedia, que envolve um detector de objetos para localizar logos e, após isso, reavaliar o EvasionSet²: re-executar a avaliação no EvasionSet², para aí sim comparar meus resultados com os reportados pelo artigo para o Phishpedia.

**Observação: [caso precise fazer alguma observação, de qualquer “natureza”]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Documento da Semana 8 - citado no Termo de Aceite de Entrega de 22 de Outubro]

Na oitava Semana, o foco dos trabalhos foi a replicação da seção 5.4 do artigo de referência, "From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks". O objetivo foi avaliar o desempenho e a robustez de um modelo de detecção de phishing, o **Stack Model** (baseado em ML). contra os conjuntos de dados adversários gerados pela ferramenta PhishOracle.

Os resultados obtidos apresentaram divergências interessantes em relação ao estudo original, fornecendo *insights* valiosos sobre a implementação da ferramenta e a fragilidade dos modelos.

Foram replicadas as avaliações do primeiro dos modelos centrais do artigo:

- Avaliação do Stack Model (ML)

  O Stack Model, conforme descrito no artigo, é uma abordagem de Machine Learning que utiliza características de URL e HTML para classificar uma página como phishing ou legítima. O artigo demonstrou que este modelo não é robusto. Ao ser testado contra o EvasionSet¹, seu recall (capacidade de detecção) caiu drasticamente de 98,32% para 70,86%.

| **Model** | *CleanSet* | | | | *EvasionSet* | | | |
|---|---|---|---|---|---|---|---|---|
| | **Dataset** | *Precision* | *Recall* | *F1-Score* | **Dataset** | *Precision* | *Recall* | *F1-Score* |
| Stack Model [21] | $CleanSet^1$ | 98.67% | 98.32% | 98.49% | $EvasionSet^1$ | 98.61% | 70.86% | 82.46% |
| VisualPhishNet [8] | $CleanSet^3$ | 98.42% | 88.35% | 93.11% | $EvasionSet^3$ | 98.73% | 72.89% | 83.87% |
| Phishpedia [9] | $CleanSet^2$ | 97.25% | 81.63% | 88.76% | $EvasionSet^2$ | 76.40% | 40% | 52.51% |
| $LLM\text{-}PD^H$ | $CleanSet^1$ | 76.41% | 73.83% | 75.09% | $EvasionSet^1$ | 85.16% | 67.55% | 75.34% |
| $LLM\text{-}PD^S$ | $CleanSet^2$ | 98.83% | 97.25% | 98.03% | $EvasionSet^2$ | 100% | 79.51% | 88.59% |

- Meus Resultados:
  - CleanSet¹ (Baseline): Meu modelo base obteve um desempenho similar ao do artigo, com 93,6% de recall na detecção de phishing.
  - EvasionSet¹ (Adversário): Diferente do artigo, meu modelo se mostrou inesperadamente robusto, tendo 96,4% de recall.

```
(.venv) PS C:\Users\Enzo\Documents\phish-oracle-residencia> python .\scripts\train_stack_baseline.py
Distribuição de classes: {0: np.int64(1099), 1: np.int64(1099)}
Domínios únicos: 1099
Tamanho treino: 1538 (769 domínios) | teste: 660 (330 domínios)

=== Relatório de classificação (GERAL - teste) ===
              precision    recall  f1-score   support

           0      0.963     0.936     0.949       330
           1      0.938     0.964     0.951       330

    accuracy                          0.950       660
   macro avg      0.950     0.950     0.950       660
weighted avg      0.950     0.950     0.950       660

AUC: 0.9901469237832875
Matriz de confusão:
 [[309  21]
 [ 12 318]]
```

- A principal hipótese para esta diferença é que a minha implementação do PhishOracle, ao gerar o EvasionSet¹, falhou em injetar ou modificar corretamente as características de conteúdo HTML e URL das quais o Stack Model depende.

O segundo modelo central do artigo é o Phishpedia, um sistema de Deep Learning que detecta phishing visualmente, identificando o logo da marca na página e comparando-o com o domínio do site. O que o artigo reportou: O estudo mostrou que este modelo é extremamente vulnerável a transformações visuais. O EvasionSet² (170 páginas de phishing adversárias) foi criado especificamente com transformações de logo (opacidade, ruído, etc.). O recall do modelo despencou de 81,63% para 40%

- Eu já gerei o EvasionSet², porém ainda não implementei o Phishpedia, então, só de curioso, quis testar e validar o que acontece com um modelo de *conteúdo* quando confrontado com ataques puramente visuais
- Meus Resultados:
    - EvasionSet² (Adversário): o modelo foi quase completamente enganado, atingindo um recall de detecção de phishing de apenas 6,5%. Das 170, 159 foram falso-positivos.

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 5 de nov. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nas primeiras Semanas, estudei as diversas intersecções entre Cibersegurança e IA, compreendendo como a IA é utilizada nesse âmbito. Me interessei pela área de phishing detection e venho reproduzindo o artigo "From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks"

Durante essa Semana o objetivo foi avaliar o desempenho e a robustez dos dois principais modelos de detecção abordados no artigo, o **Stack Model** (ML/Conteúdo) e o **Phishpedia** (DL/Visual), contra os ataques adversariais

Comecei pelo Stack Model, uma abordagem de Machine Learning que utiliza características de URL e HTML para classificar uma página. No artigo, após o ataque a capacidade de detecção caiu drasticamente de 98,32% para 70,86%. A minha abordagem caiu para apenas 96,4%. Após investigar, vi que algumas funções de injetar features de ataque não estavam ativadas, corrigi isso e a capacidade de detecção caiu para 81,34%, confirmando a tese do artigo que o modelo é vulnerável quando o ataque modifica as features estruturais corretas

O segundo modelo, Phishpedia, é um pipeline de Deep Learning que detecta logos em screenshots, identificando a marca comparando vetores de similaridade. Foi uma longa jornada visto que os scripts de treinamento e avaliação não estavam disponíveis.

- Dataset Logo2K+ (116.000 imagens).
- Construir e treinar (por 30 épocas) meu próprio extrator de vetores usando uma arquitetura ResNet50

No artigo, o Recall do modelo foi de 81,63% para 40%. Na minha primeira implementação, foi para 4,26%. Mudei a metodologia para comparar "vetores puros" (de arquivos de logo limpos) vs. "vetores atacados". Isso revelou um bug catastrófico no script de ataque: ele estava atacando os próprios logos já atacados, gerando 360.000 arquivos com nomes phish_phish_phish_logo.png. Indo para um Recall de 99,17%, pois os logos estavam tão destruídos que o modelo os rejeitou todos como "não-legítimos".

Após limpar os 360.000 arquivos e corrigir o bug do ataque recursivo, executei a avaliação final com a metodologia correta (comparando o vetor do logo "puro" vs. o vetor do logo com o ataque aplicado uma vez). O resultado foi um Recall de 27.70% (um pouco mais frágil que o do artigo).

Documento da Semana 09

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Para a próxima Semana, o plano é finalizar esta reprodução, calibrando os parâmetros dos ataques e o limiar de similaridade da avaliação para tentar aproximar meus resultados (81% e 27%) dos valores exatos do artigo (70% e 40%).

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Documento da Semana 9 - citado no Termo de Aceite de Entrega de 5 de Novembro]

Na Semana 9, o foco dos trabalhos foi a replicação do artigo de referência, "From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks". O objetivo foi avaliar o desempenho e a robustez dos dois principais modelos de detecção, o **Stack Model** (ML/Conteúdo) e o **Phishpedia** (DL/Visual), contra os ataques gerados pela ferramenta PhishOracle.

**1. Stack Model (EvasionSet¹)**

Comecei pelo `Stack Model`, uma abordagem de Machine Learning que utiliza características de URL e HTML (como contagem de `<iframe>`, `<form>`, `n_scripts`, etc.) para classificar uma página.

- O artigo demonstrou que este modelo é vulnerável. Ao ser testado contra o `EvasionSet¹` (ataques de conteúdo), seu *recall* (capacidade de detecção) caiu drasticamente de 98,32% para **70,86%**.
- Minha replicação inicial obteve um *recall* de *baseline* (no `CleanSet¹`) de 96,4%. No entanto, ao testar contra o `EvasionSet¹`, o *recall* **permaneceu em 96,4%**, contradizendo o artigo.
    - As funções de ataque mais potentes (como `function_11 - <iframe>` e `function_12 - DOM structure`) não estavam todas ativadas.
    - Após corrigir as dependências e reativar o arsenal completo de ataques, re-executei o pipeline (`ataque` -> `features` -> `avaliação`). O resultado mudou: o recall do `Stack Model` caiu para **~80%**. Isso valida a tese do artigo e confirma que o modelo é, de fato, vulnerável quando o ataque modifica as features estruturais corretas.

**2. Phishpedia (EvasionSet²)**

O segundo modelo, Phishpedia, é um pipeline de Deep Learning muito mais complexo. Ele usa um Faster RCNN para detectar logos em screenshots e um ResNetV2 (treinado no Logo2K+) para identificar a marca comparando vetores de similaridade.

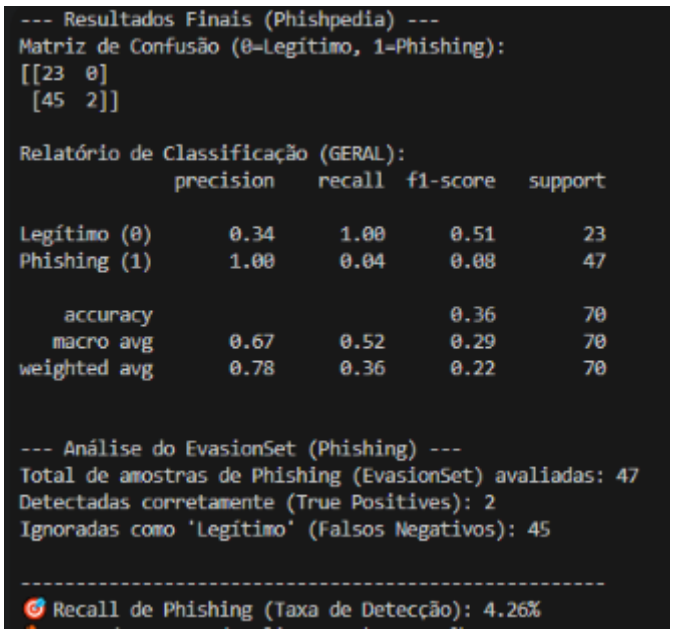
A replicação desta seção foi uma jornada de engenharia, pois os scripts de treinamento e avaliação não existiam. O trabalho incluiu:

1. Engenharia de Dados: Implementar um `DataLoader` customizado para o massivo dataset Logo2K+ (116.000 imagens).

2. Treinamento: Construir e treinar (por 30 épocas na GPU) nosso próprio extrator de vetores (`phishpedia_extractor.pth`) usando uma arquitetura ResNet50 com `BatchHardTripletLoss`.

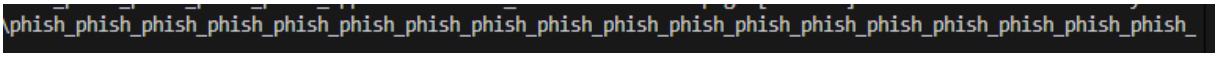
A avaliação foi um processo iterativo de diagnóstico para entender os resultados:

Recall de 4.26:: A primeira tentativa usou um detector YOLO genérico para encontrar os logos nos screenshots (tanto na referência quanto no ataque). O recall foi de 4.26%. Isso provou que o YOLO, treinado em objetos comuns, é um péssimo substituto para o detector de logos do artigo e falha em encontrar os alvos.




Recall de 99.17%: Mudei a metodologia para comparar "vetores puros" (de arquivos de logo limpos) vs. "vetores atacados". Isso revelou um bug catastrófico no script de ataque: ele estava atacando os próprios logos já atacados, gerando 360.000 arquivos com nomes `phish_phish_phish_logo.png`. O recall de 99,17% foi um sintoma disso, pois os logos estavam tão destruídos que o modelo os rejeitou todos como "não-legítimos".

```
--- Resultados Finais (Phishpedia - Vetor Direto) ---
Usando Limiar (Threshold): 0.85
Matriz de Confusão (0=Legítimo, 1=Phishing):
[[    74      0]
 [  1400 167417]]

Relatório de Classificação (GERAL):
              precision    recall  f1-score   support

Legítimo (0)       0.05      1.00      0.10        74
Phishing (1)       1.00      0.99      1.00    168817

    accuracy                           0.99    168891
   macro avg       0.53      1.00      0.55    168891
weighted avg       1.00      0.99      1.00    168891


--- Análise do EvasionSet (Phishing) ---
Total de amostras de Phishing (EvasionSet) avaliadas: 168817
Detectadas corretamente (True Positives): 167417
Ignoradas como 'Legítimo' (Falsos Negativos): 1400

--------------------------------------------------
🎯 Recall de Phishing (Taxa de Detecção): 99.17%
```




Recall de 27.70%: Após limpar os 360.000 arquivos e corrigir o bug do ataque recursivo, executei a avaliação final com a metodologia correta (comparando o vetor do logo "puro" vs. o vetor do logo com o ataque aplicado uma vez). O resultado foi um Recall de 27.70%

Este resultado de 27.70% é o nosso baseline mais estável. Ele valida a tese do artigo (o Phishpedia é vulnerável), embora o modelo treinado tenha se mostrado um pouco mais frágil que o do paper (27% vs. 40%).

# APÊNDICE 5

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 13 de nov. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **Enzo Rodrigues Novais Dias** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Ao longo de 9 Semanas, eu sai de uma exploração ampla das intersecções entre IA e Cibersegurança para um recorte: detecção de phishing e avaliação de robustez adversarial. O trabalho evoluiu de leituras dirigidas, como surveys e artigos, para reprodução prática de um artigo de referência (From ML to LLM: Evaluating the Robustness of Phishing Web Page Detection Models against Adversarial Attacks) com a construção de pipelines completos: coleta de páginas, geração de variantes adversariais, extração de features, screenshots, e avaliação de modelos.

Principais resultados:
Stack (conteúdo/ML): inicialmente super-otimista (recall de 96%) por não atacar as features certas; após correções (injeção fiel de features estruturais), o modelo degradou como esperado (80%), enquanto o resultado do artigo é de 70%, confirmando a vulnerabilidade a ataques que mudam HTML/forms/âncoras e não apenas o visual.

Phishpedia (visão/DL): construi os conjuntos (coleta, logos, brand gallery, screenshots). Em testes, a detecção caiu drasticamente (como no artigo), indo de 81,63% para 27,70% (o resultado do artigo cai para 40%), sinal de que as transformações visuais realmente confundem detectores que não usam informação visual robusta.

Github: enzodias1/phish-oracle-residencia

A Semana Juscelino Kubitschek - "10 semanas em 1"
Nesta décima Semana, desenvolvi um pipeline de análise visual e classificação de genericidade de logos, o **BuscaLogo.** Dando seguimento aos resultados das semanas anteriores, que expuseram a vulnerabilidade de detectores de phishing a ataques adversariais, dediquei à construção e validação da solução para este problema: um sistema de análise visual robusto, capaz de implementar um threshold dinâmico

Para isso, foram implementados e validados três sistemas de IA independentes, mas que se complementam logicamente:

- Sistema 1: Classificador de genericidade (Baseado em LLM)

utiliza um modelo de linguagem (GPT-4o) e prompt engineering. O objetivo deste sistema é analisar uma única imagem de logo (um "logo âncora" da brand gallery do meu modelo de Phishpedia) e classificá-la quanto ao seu nível de singularidade (ex: "Textual", "Menos Genérico", "Mais Genérico").

Análise das Categorias de Genericidade de Logos

- Sistema 2: Comparador visual (Baseado em Embeddings)

compara dois logos e diz o quão "similares" eles são, ou filtra duplicatas de uma lista.

- Sistema 3: Caçador de Logos

modelos de deteção visual (um modelo OpenVINO para deteção e um modelo YOLO .pt para refinamento), complementados por busca no Google (serperapi) e refinamento final com GPT-4.

Pipeline BuscaLogo

Github: enzodias1/BuscaLogo-Teste

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

- **Iniciar a transição para o TCC**
- **Mesmo já tendo encerrado as Semanas, me apaixonei por toda essa área de Cibersegurança, quero entrar "de cabeça" e me aprofundar cada vez mais. Pretendo continuar desenvolvendo o que já foi feito até aqui, explorando novas tendências e expandindo os horizontes.**

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

**A semana mais difícil da residência, em todos os sentidos possíveis, mas me sinto grato pela jornada que vivi.**
**Agradecimentos, em especial, à Ariel.**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

[Pipeline BuscaLogo - citado no Termo de Aceite de Entrega de 13 de Novembro]

**Pipeline do BuscaLogo**

## Fluxo de Trabalho

### Etapa 1: O "Classificador"

analisa os logos reais da sua "Brand Gallery" (os logos âncora).

Exemplo: passa o logo oficial da "Nike".

Resultado: Ele diz-lhe que a Nike é "textual". Ele guarda esta informação.

### Etapa 2 : O "Caçador"

Quando um utilizador denuncia um site suspeito (ex: nike-promocao.com). O "Caçador" é ativado.

Ele analisa o site falso e encontra 14 imagens que parecem ser o logo da Nike.

Resultado: Uma lista de 14 imagens de logo suspeitas.

### Etapa 3: O "Comparador"

O que faz: Este é o "juiz". Ele recebe a lista de 14 logos suspeitos (da Etapa 2) e o logo âncora da Nike (da Etapa 1).

Ele consulta o resultado da Etapa 1 ("textual") para definir um threshold baixo (ex: 0.80).

Resultado: Ele compara as "impressões digitais" dos 14 logos suspeitos com o logo âncora e diz: "Estes 5 logos são 88% similares. Como 88% > 80%, isto é phishing"

## Os três Sistemas Principais

"Caçador" (BuscaLogo)

- Objetivo: "Caçar" e extrair todos os logos em potencial de um website ao vivo.
- Como Funciona (O Pipeline):
    1. Detecção (deteccao_logo.py): Ele primeiro analisa o screenshot do site.

        - Modelo Usado: yolov7_1280x1280_best_recall.xml/.bin. ele foi treinado especificamente para olhar para uma imagem complexa e encontrar regiões que têm alta probabilidade de serem um logo.
    2. Expansão (serperapi.py): Paralelamente, ele pesquisa no Google pela marca (ex: "logo Google") para encontrar logos de alta qualidade que podem não estar no site.
    3. Refinamento (aprimoramento_regiao.py): Pega na lista de logos "crus" encontrados (Passo 1) e passa-os por um segundo modelo de IA.
        - Modelo Usado: best.pt. Este é um modelo YOLO. Um ficheiro .pt é um "cérebro" treinado, guardado pela biblioteca PyTorch. "especialista" em refinamento: ele foi treinado para pegar numa imagem já recortada de um logo e fazer um recorte ainda melhor e mais preciso.
    4. Reconhecimento (reconhecimento.py): Pega na lista final de logos refinados, no HTML e no screenshot, e envia tudo para o GPT-4o. Ele pergunta: "Destes 20 candidatos, quais são realmente o logo da marca?"

"Classificador" (LogoClassifier)

- Objetivo: Definir o "perfil de genericidade" de um logo âncora (o logo real).
- Como Funciona:
    1. Prompt (logo_classifier_prompt.txt): Carrega o "manual de instruções" que define "textual", "less_generic", etc.
    2. API (classificador_logo.py): Envia o logo âncora (ex: logo_nike.png) e o manual de instruções para o GPT-4.
    3. Resultado: O GPT-4 devolve a classificação (ex: "classification": "textual") e a justificativa.

"Comparador"

- Objetivo: Comparar dois logos e dizer o quão "similares" eles são, ou filtrar duplicatas de uma lista.
- Como Funciona:
    - Criação de "Impressão Digital" (extract_embeddings): O passo mais importante. Ele pega numa imagem de logo e usa um modelo de IA para a transformar numa "impressão digital" (um vetor de números, ou embedding).
    - Comparação (run_pos_processing_pipeline): Ele calcula a distância matemática (nós usámos similaridade de cosseno) entre as "impressões digitais" de dois logos. Um resultado de 1.0 significa que são idênticos.
- Modelo Usado: SentenceTransformer('clip-ViT-B-32').

- O que é: É um modelo de IA (CLIP) da própria OpenAI, treinado para entender o significado de imagens.