Lucas Albano Olive Cruz

Correlação Dinâmica de Alertas e Predição Multitarefa de Ciberataques via Transformers

Monografia apresentada ao curso de Sistemas de Informação da Universidade Federal de Minas Gerais, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Orientador: Prof.ª Dra. Michele Nogueira Lima
Coorientador: Prof.ª Dra. Ligia Francielle Borges

Belo Horizonte, Minas Gerais
2026

Ao meu pai, que não está mais aqui, mas cuja presença ainda guia cada passo que dou. Sem ele, nunca teria tido a chance de seguir meus sonhos, e esta conquista também é dele.

À minha mãe, pelo amor incondicional, por ter acreditado em mim e me apoiado em todos os meus passos.

Às professoras Michele e Ligia, por todo o apoio e pela generosidade com que compartilharam seu conhecimento ao longo desses anos, e ao professor Anderson e aos demais que contribuíram em diferentes momentos desta jornada.

Aos amigos que a graduação me deu, pela companhia, pelas risadas nos momentos certos e por tornarem essa jornada mais leve.

Aos meus familiares, pelo carinho e pelo apoio que sempre estiveram presentes, mesmo de longe.

Aos professores que cruzaram meu caminho ao longo da graduação e deixaram algo de si em cada aula.

"Aprender uma lição sem dor não tem significado, isso porque as pessoas não conseguem obter nada sem sacrificar alguma coisa. Mas quando elas superam as dificuldades e conseguem o que querem, as pessoas conquistam um coração forte que não perde para nada. É! Um coração forte como aço."

(Edward Elric - Fullmetal Alchemist: Brotherhood)

Resumo

A crescente complexidade das ameaças cibernéticas e o volume massivo de eventos gerados por sistemas de monitoramento impõem desafios aos Centros de Operações de Segurança (SOCs) diante da frequente fadiga de alertas e incapacidade de antecipar os ataques de múltiplas etapas. Este trabalho propõe uma abordagem de correlação dinâmica de alertas e predição multitarefa de ataques cibernéticos baseada em Aprendizado Profundo, por meio de arquiteturas *Transformer*. A metodologia trata o fluxo de alertas como uma narrativa sequencial, aplicando técnicas de Processamento de Linguagem Natural para vetorizar atributos heterogêneos e capturar dependências contextuais por meio de mecanismos de autoatenção. Dois codificadores são treinados por Aprendizado Contrastivo com perda tripleto, um em nível de alerta organizado pelo estágio de ataque e outro em nível de campanha responsável por separar fluxos de eventos de operações ofensivas simultâneas, com a correlação realizada pelo agrupamento HDBSCAN sobre os espaços métricos aprendidos. A partir das sequências correlacionadas, um preditor multitarefa estima conjuntamente o próximo estágio do ataque, os endereços IP de origem e destino e o intervalo temporal até o próximo evento, por meio de cabeças de saída especializadas. Para superar a escassez de dados rotulados e balanceados, um *pipeline* de geração de dados sintéticos foi desenvolvido, com cinco modos de endereçamento IP que cobrem desde endereços fixos por campanha até rotação cíclica e distribuição aleatória entre prefixos distintos. A validação experimental, realizada com o conjunto sintético e com o conjunto de dados AIT-ADS em protocolo de avaliação cruzada entre atacantes, demonstrou que o codificador de campanha atinge V-Measure de 0,9968 na separação de campanhas simultâneas com apenas 0,03% de ruído residual, e que o preditor multitarefa, treinado sobre o conjunto sintético balanceado, alcança Macro F1 de 0,8937 e acurácia Top-3 de 98,45%, com acurácia de 91,92% na predição do IP de origem dentro da janela de contexto e erro médio absoluto de 4,26 segundos na predição do intervalo temporal.

Palavras-chave: Correlação de Alertas. Predição de Ataques. Ataques de Múltiplas Etapas. Aprendizado Profundo. Transformers. Cibersegurança.

Abstract

The growing complexity of cyber threats and the massive volume of events generated by monitoring systems pose significant challenges to Security Operations Centers (SOCs), frequently leading to alert fatigue and inability to anticipate multi-stage attacks. This work proposes an approach for dynamic alert correlation and multitask cyberattack prediction based on Deep Learning, by means of Transformer architectures. The methodology treats the alert stream as a sequential narrative, applying Natural Language Processing techniques to vectorize heterogeneous attributes and capture contextual dependencies through self-attention mechanisms. Two encoders are trained by Contrastive Learning with triplet loss, one at the alert level organized by attack stage and another at the campaign level responsible for separating event streams from simultaneous offensive operations, with correlation performed by HDBSCAN clustering over the learned metric spaces. From the correlated sequences, a multitask predictor jointly estimates the next attack stage, the source and destination IP addresses, and the time interval until the next event, by means of specialized output heads. To overcome the scarcity of labeled and balanced data, a synthetic data generation pipeline was developed, with five IP addressing modes covering fixed per-campaign addresses, cyclic rotation, and random distribution across distinct prefixes. Experimental validation, conducted on the synthetic dataset and on the AIT-ADS dataset in a cross-attacker evaluation protocol, demonstrated that the campaign encoder achieves a V-Measure of 0.9968 in separating simultaneous campaigns with only 0.03% residual noise, and that the multitask predictor, trained on the balanced synthetic dataset, achieves Macro F1 of 0.8937 and Top-3 accuracy of 98.45%, with 91.92% accuracy in source IP prediction within the context window and a mean absolute error of 4.26 seconds in temporal interval prediction.

Keywords: Alert Correlation. Alert Prediction. Multi-Step Attacks. Deep Learning. Transformers. Cybersecurity.

Sumário

1 INTRODUÇÃO 8
1.1 Problema 9
1.2 Objetivo Geral 10
1.3 Objetivos específicos 11
1.4 Organização do texto 12

2 REFERENCIAL TEÓRICO 13
2.1 Sistemas de Monitoramento e Alertas 13
2.2 Ataques Multiestágio 14
2.3 Correlação de Alertas e Predição de Ataques 16
2.4 Arquiteturas Baseadas em Atenção 17
2.5 Aprendizado Contrastivo e Redes Siamesas 19
2.6 Agrupamento Baseado em Densidade 21
2.7 Redes de Ponteiros 22
2.8 Resumo 24

3 TRABALHOS RELACIONADOS 25
3.1 Correlação de Alertas e Reconstrução de Cenários 25
3.2 Predição de Ataques 28
3.3 Aprendizado Multitarefa e Análise Semântica 30
3.4 Resumo 32

4 METODOLOGIA 33
4.1 Processamento de Atributos e Representação Vetorial 34
4.2 Codificador de Campanha 35
4.3 Codificador de Alerta 36
4.4 Preditor Multitarefa 38
4.5 Caso Ilustrativo 39
4.6 Resumo 41

5 EXPERIMENTOS 42
5.1 Ambiente Experimental e Configuração 42
5.2 Conjuntos de Dados 43
5.2.1 AIT Alert Dataset 43
5.2.2 Dados Sintéticos 45
5.3 Métricas de Avaliação 48

5.4 Implementação e Hiperparâmetros 50
5.5 Avaliação dos Codificadores 54
5.5.1 Codificador de Alerta 54
5.5.2 Codificador de Campanha 56
5.6 Avaliação do Preditor Multitarefa 57
5.6.1 Dados Sintéticos 57
5.6.2 AIT-ADS 61
5.7 Estudos de Ablação 63
5.8 Discussão 65
5.9 Resumo 67

6 CONCLUSÃO 68

REFERÊNCIAS 70

1 Introdução

A crescente digitalização redefiniu os paradigmas de operação dos serviços essenciais e de infraestruturas críticas e provocou transformações profundas na sociedade. Os setores vitais, como financeiro, saúde e eletricidade, transicionaram de modelos operacionais analógicos ou isolados para ecossistemas hiperconectados e dependentes de sistemas computacionais distribuídos. A digitalização vai além da automação de tarefas e permeia aspectos do cotidiano por meio da integração de dispositivos móveis, Internet das Coisas (IoT) e computação em nuvem, o que torna a continuidade e estabilidade desses serviços pilares do mundo tecnológico moderno [1].

Em paralelo aos benefícios da digitalização, a interconectividade expandiu drasticamente a superfície de ataque disponível para ameaças cibernéticas [2]. A introdução de novos pontos de interação e a dissolução dos perímetros de rede tradicionais criaram múltiplos vetores de entrada, que podem ser explorados por agentes maliciosos. A complexidade da gestão de milhares de dispositivos interligados, muitas vezes com configurações de segurança heterogêneas, oferece um terreno fértil para a exploração de vulnerabilidades, o que permite que atacantes conduzam campanhas distribuídas e furtivas, com movimentação lateral dentro de redes corporativas e residenciais ao longo de múltiplas etapas com maior facilidade e discrição.

Nesse contexto, o volume e a velocidade têm superado a capacidade das abordagens de segurança tradicionais, que são, em sua maioria, reativas [3]. As ferramentas como os *Firewalls*, os Sistemas de Detecção e Prevenção de Intrusão (IDS/IPS) e a Detecção e Resposta de Endpoint (EDR) são essenciais, mas geralmente atuam na identificação de ataques em curso ou após a ocorrência de um dano inicial. A consequência direta dessa limitação reativa é a perda da oportunidade de gerar uma economia drástica em custos de reparo e mitigação [4], um benefício que permanece em grande parte inalcançável devido à dupla dificuldade de identificar, em meio ao volume de alertas, os sinais que pertencem a uma mesma campanha de ataque e de antecipar não apenas se um próximo passo ocorrerá, mas onde e quando ele se manifestará com base nesses sinais.

A tarefa de gerenciar esses sistemas de defesa e de buscar ativamente a predição dos ataques compete primariamente aos Centros de Operações de Segurança (SOCs). Estes centros funcionam como o núcleo de cibersegurança de uma organização, com integração de processos, pessoas e tecnologias para monitorar, analisar e responder a ameaças em tempo real. Contudo, o principal obstáculo operacional que impede os SOCs de alcançarem essa proatividade é a sobrecarga de informações, comumente referida como fadiga de alertas (em inglês, *alert fatigue*) [5]. Os sistemas de monitoramento geram um volume massivo de alertas de fontes heterogêneas, como *logs* de rede, eventos de *endpoints* e registros de aplicações, o que satura a capacidade cognitiva dos analistas humanos.

Para identificar ameaças reais em meio a este alto volume de informações, essas ferramentas utilizam métodos de correlação [6]. Estes métodos são responsáveis por filtrar falsos positivos, reduzir redundâncias, definir níveis de prioridade e enriquecer os alertas com contexto, como relações causais e temporais. O desafio central é que os métodos de correlação atuais são, em grande parte, baseados em regras estáticas e assinaturas predefinidas. Essas abordagens são frágeis, exigem configuração manual exaustiva por especialistas e falham em detectar ataques sofisticados, como ameaças persistentes avançadas (APTs) ou ataques de baixa cadência (*low-and-slow*). Como resultado, os alertas que são, na verdade, etapas de uma mesma campanha de ataque permanecem isolados, indistinguíveis do ruído e dos falsos positivos, o que não apenas impede sua correlação, mas também elimina a base de evidências sobre a qual qualquer estimativa dos próximos passos do atacante poderia ser construída. Isso força os analistas humanos a tentar conectar manualmente os pontos, uma tarefa que se torna impraticável na escala de ataques atuais, o que mantém as operações de segurança presas ao ciclo reativo e impede tanto a reconstrução de incidentes quanto a antecipação de ameaças.

1.1 Problema

A dependência de métodos de correlação estáticos e a consequente fadiga de alertas mantêm os SOCs em um ciclo reativo [3]. O impacto negativo dessa postura manifesta-se por meio do custo temporal ou tempo de permanência (em inglês, *dwell time*). Este conceito refere-se à janela de oportunidade concedida ao atacante entre a invasão inicial e a sua neutralização. Um relatório da IBM de 2024 focado no setor financeiro, por exemplo, revelou que as organizações levaram, em média, 168 dias para identificar uma violação e outros 51 dias para contê-la [7]. Esse período total de 219 dias concede aos atacantes um tempo superior a seis meses para realizar reconhecimento detalhado, mover-se lateralmente e exfiltrar dados sensíveis, muitas vezes em operação *low-and-slow* para evitar a detecção de assinaturas conhecidas.

Financeiramente, existe uma ligação direta entre o custo temporal e o prejuízo monetário. O relatório *Cost of a Data Breach Report* aponta um custo médio global de USD 4,88 milhões por incidente em 2025 [8]. O mesmo relatório indica que essa média global teve uma ligeira redução impulsionada, em parte, pela identificação e contenção mais rápidas com a ajuda de Inteligências Artificiais (IA) e da automação [8]. Ainda assim, no Brasil o custo médio de uma violação de dados atingiu o recorde de R$ 7,19 milhões em 2025, um aumento impulsionado pela complexidade dos ambientes de nuvem e pela escassez de habilidades de segurança [9]. Em mercados altamente visados e regulados, como o norte-americano, o custo escalou para um recorde de USD 10,22 milhões, o que evidencia que a ineficiência na detecção rápida independe da localização geográfica.

Esses dados confirmam, portanto, que a detecção tardia gera custos financeiros e

temporais proibitivos. Eles também sugerem que a investigação de novas abordagens, especificamente o uso de automação e IA, é o principal vetor para a mitigação desses custos. Fica evidente, assim, a necessidade de desenvolver métodos capazes de providenciar a antecipação de ataques e quebrar o ciclo reativo.

Diante desse cenário, o problema de pesquisa que esta monografia aborda é duplo. Em primeiro lugar, os métodos atuais de correlação de alertas, predominantemente baseados em regras estáticas, falham em reconhecer que alertas aparentemente isolados e de baixa severidade pertencem, na verdade, a etapas sucessivas de uma mesma campanha de ataque avançado, o que os mantém indistinguíveis do ruído gerado por sensores heterogêneos. Em segundo lugar, mesmo quando esses alertas são corretamente correlacionados, as abordagens de segurança permanecem fundamentalmente reativas, pois respondem a incidentes após sua ocorrência em vez de antecipar as ações futuras do adversário com base nos padrões já observados. A consequência combinada dessas duas lacunas é que ameaças sofisticadas, como APTs ou ataques *low-and-slow*, evitam frequentemente a detecção até que um dano significativo já tenha ocorrido [10], e mesmo quando identificadas, a postura defensiva permanece incapaz de antecipar a evolução do ataque [4].

1.2 Objetivo Geral

Para superar a limitação de métodos estáticos e reativos, este trabalho propõe uma nova abordagem para a correlação dinâmica de alertas e a predição de ataques cibernéticos. A hipótese central é que o fluxo de alertas, definido como a sequência temporal de eventos gerados por diferentes sensores (IDS/IPS, *Firewalls*, EDRs), pode ser tratado de forma análoga a como os Grandes Modelos de Linguagem (LLMs) processam o texto. Assim como palavras em uma frase, os alertas de segurança ganham significado a partir do contexto sequencial em que aparecem. Por exemplo, um alerta de Scan de Portas, seguido por uma tentativa falha de login em SSH e, dias depois, por um upload de Web Shell vindos do mesmo bloco de IP, contam uma história coesa que alertas individuais não conseguem expressar, e essa mesma história, uma vez reconhecida, fornece a base para estimar qual será o próximo passo nessa narrativa.

Para capturar esse contexto, este trabalho adota modelos de Aprendizado Profundo (em inglês, *Deep Learning*). A escolha justifica-se pela capacidade desses algoritmos de aprender representações complexas e não lineares diretamente dos dados brutos, o que elimina a necessidade de engenharia de características manual. Especificamente, a arquitetura selecionada baseia-se em modelos *Transformer*. A escolha por esta arquitetura, em detrimento de redes recorrentes tradicionais, deve-se ao mecanismo de autoatenção (em inglês, *self-attention*) [11]. Enquanto redes recorrentes tendem a perder o contexto em sequências muito longas, os *Transformers* conseguem mensurar a influência direta de eventos passados sobre os atuais independentemente da distância temporal entre eles. Essa

característica é essencial para o domínio da cibersegurança, pois permite correlacionar sinais fracos de um ataque que podem estar separados por longos intervalos de tempo, e fornece a base contextual sobre a qual estimativas futuras podem ser construídas.

Diante disso, este trabalho apresenta o projeto, implementação e validação de uma abordagem de *Deep Learning* baseada em modelos *Transformer* para a correlação dinâmica de alertas heterogêneos e a predição multitarefa de ataques cibernéticos. A correlação visa à separação de campanhas de ataque simultâneas em sequências coesas de múltiplas etapas, enquanto a predição multitarefa busca estimar conjuntamente o próximo estágio do ataque, os endereços envolvidos e o intervalo temporal até sua ocorrência, a partir das sequências correlacionadas.

1.3 Objetivos específicos

Para alcançar o objetivo geral, este trabalho define os seguintes objetivos específicos:

1. Uma revisão sistemática das abordagens de correlação de alertas e de predição de ataques, com identificação das limitações dos métodos estáticos e sequenciais existentes e das oportunidades para aplicação de *Deep Learning*.

2. Um método de representação vetorial de alertas de fontes heterogêneas, capaz de traduzir características categóricas, numéricas e textuais em um formato numérico denso adequado ao processamento pelo modelo.

3. Uma arquitetura baseada em *Transformer* com mecanismos de autoatenção adaptados ao processamento de sequências de *embeddings* de alertas, com produção de representações enriquecidas pelo contexto temporal e sequencial.

4. Uma análise comparativa de duas estratégias de aprendizado contrastivo com granularidades distintas de supervisão, sendo a primeira em nível de alerta individual com pares positivos definidos pelo estágio do ataque, e a segunda em nível de sequência de alertas com pares positivos definidos pela campanha de ataque.

5. Um *pipeline* de geração de dados sintéticos capaz de produzir sequências de alertas rotuladas por estágio e por campanha, adequadas ao treinamento e à avaliação em condições de classes balanceadas.

6. Um modelo preditor multitarefa baseado em *Transformer*, capaz de estimar conjuntamente o próximo estágio do ataque, os endereços envolvidos e o intervalo temporal até a ocorrência do próximo evento, a partir das sequências de alertas correlacionadas.

7. Uma avaliação experimental da abordagem proposta quanto à sua capacidade de separar campanhas de ataque simultâneas e de predizer os atributos do próximo

evento, com posicionamento dos resultados em relação às capacidades e limitações dos métodos discutidos na revisão da literatura.

1.4 Organização do texto

Esta monografia está estruturada como segue. O Capítulo 2 detalha o referencial teórico necessário. O Capítulo 3 apresenta uma revisão dos trabalhos relacionados em correlação de alertas, predição de ataques e aprendizado multitarefa. O Capítulo 4 descreve a abordagem proposta e a metodologia de pesquisa. O Capítulo 5 apresenta os experimentos e a análise dos resultados obtidos. Finalmente, o Capítulo 6 conclui o trabalho e apresenta as contribuições e as direções para trabalhos futuros.

2 Referencial Teórico

Este capítulo apresenta os conceitos fundamentais que sustentam a abordagem proposta. Inicialmente, a Seção 2.1 discute o ecossistema de monitoramento de cibersegurança e a natureza heterogênea dos alertas, e a Seção 2.2 apresenta os modelos de ataques multiestágio que evidenciam a necessidade de correlação além do nível de alerta individual. Em seguida, a Seção 2.3 aborda os processos teóricos de correlação de alertas e de predição de ataques. Posteriormente, a Seção 2.4 detalha as arquiteturas baseadas em atenção, que fornecem a base para a modelagem contextual das sequências de alertas. Complementarmente, a Seção 2.5 apresenta o Aprendizado Contrastivo e as Redes Siamesas, que fundamentam a construção do espaço métrico de representações. Por fim, a Seção 2.6 discute os algoritmos de agrupamento baseados em densidade, e a Seção 2.7 apresenta o mecanismo de Redes de Ponteiros, empregado na predição de endereços IP em sequências de alertas.

2.1 Sistemas de Monitoramento e Alertas

Os sistemas de monitoramento de cibersegurança são a base das estratégias de defesa cibernética atuais. Esses sistemas coletam e analisam dados em tempo real de diversas fontes (e.g., tráfego de rede, *logs* de sistema e atividades de *endpoints*) para identificar indícios de ataques. Dentre esses sistemas de monitoramento, os Sistemas de Detecção de Intrusão (IDS) são dos mais empregados. Os IDSs baseados em rede (NIDS) examinam pacotes ou fluxos de rede em busca de assinaturas de ataques conhecidos ou de anomalias estatísticas, enquanto os IDSs baseados em *host* (HIDS) monitoram atividades em nível de sistema operacional, como chamadas de sistema e modificações de arquivos, para detectar adulterações ou intrusões [12]. As soluções de Detecção e Resposta de *Endpoint* (EDR) oferecem uma visão detalhada sobre o comportamento dos dispositivos, o que permite a identificação de ameaças sofisticadas como *malware* sem arquivo (do inglês, *fileless*) e movimentação lateral (i.e., técnica de expansão de acesso a outros dispositivos da infraestrutura) [13].

Para gerenciar essa diversidade de sistemas de monitoramento, as plataformas de Gerenciamento e Correlação de Eventos de Segurança (SIEM) agregam *logs* e alertas de múltiplas fontes, o que permite correlações básicas, geralmente por meio de mecanismos baseados em regras [14]. No entanto, esses recursos de correlação integrados são frequentemente estáticos, predefinidos e limitados em sua capacidade de descobrir padrões de ataque complexos ou em evolução, principalmente aqueles que abrangem múltiplos estágios ou fontes [3]. Independentemente de sua origem (NIDS, HIDS, EDR ou SIEM), o produto desses sistemas é, por definição, um alerta (i.e., uma estrutura de dados que descreve um evento de segurança observado). Formalmente, um alerta é uma tupla de atributos

que descreve um evento de segurança observado [6]. Seja $\mathcal{A}$ o espaço de todos os alertas possíveis, $a \in \mathcal{A}$ representa um alerta individual, tal como:

$$a = (t, src, dst, proto, p, sig, \dots) \tag{2.1}$$

onde, nessa definição, t representa o carimbo de tempo (*timestamp*), *src* e *dst* os endereços IP de origem e destino, *proto* o protocolo de rede, p a porta de comunicação e *sig* a assinatura ou classificação da ameaça. É necessário ressaltar que os atributos desta tupla não são fixos. A composição e a granularidade dos metadados podem e irão variar significativamente, dependendo da natureza do evento e do sistema de monitoramento que o produziu [15].

Para ilustrar essa definição na prática, a Figura 1 apresenta um registro extraído do AIT Alert Dataset [16], conjunto de dados utilizado neste trabalho. O alerta, gerado pelo Wazuh, uma plataforma de código aberto para coleta e correlação de eventos de segurança, exemplifica a estrutura e a complexidade dos metadados disponíveis. Observando a Figura 1, nota-se que o mapeamento com a Equação 2.1 ocorre em meio a um amplo conjunto de informações auxiliares, sendo que o atributo `@timestamp` corresponde a t, `data.srcip` a *src* e `rule.description` a *sig*. A presença de campos adicionais, como `rule.pci_dss` e `full_log`, demonstra o desafio de processar esses dados brutos. Apesar de conterem metadados essenciais, os formatos de alertas variam significativamente entre ferramentas e fornecedores, desde formatos estruturados como o Formato de Troca de Mensagens de Detecção de Intrusão (do inglês, *Intrusion Detection Message Exchange Format*, IDMEF) [15] e esquemas baseados em JSON, até entradas de *log* não estruturadas. Essa heterogeneidade, somada ao alto volume de falsos positivos e à falta de contexto, torna a triagem manual impraticável, motivando o desenvolvimento de métodos automatizados de correlação e predição.

2.2 Ataques Multiestágio

Os ataques mais elaborados raramente se manifestam como um evento único e isolado. Em vez disso, um adversário precisa executar uma sucessão de ações intermediárias até atingir seu objetivo final, *i.e.*, conjunto de passos conhecido como ataque multiestágio (do inglês, *multi-step attack* ou *multi-stage attack*), também denominado cenário de ataque [17]. Essa característica se destaca em Ameaças Persistentes Avançadas (do inglês, *Advanced Persistent Threats*, APT), nas quais atacantes com muitos recursos conduzem campanhas prolongadas e furtivas contra alvos específicos [18]. A natureza multiestágio desses ataques é justamente o que dificulta sua detecção, pois cada ação isolada pode parecer benigna ou de baixa severidade, e somente a correlação de múltiplos eventos ao longo do tempo revela a estratégia subjacente e a real intenção do adversário [17].

```
{
  "agent": {
    "ip": "192.168.131.215",
    "name": "wazuh-client",
    "id": "21"
  },
  "manager": {
    "name": "wazuh.manager"
  },
  "data": {
    "protocol": "GET",
    "srcip": "10.237.1.238",
    "id": "404",
    "url": "/javascript/xmlrpc"
  },
  "rule": {
    "firedtimes": 12688,
    "mail": false,
    "level": 5,
    "pci_dss": ["6.5", "11.4"],
    "tsc": ["CC6.6", "CC7.1", "CC8.1", "CC6.1", "CC6.8", "CC7.2", "CC7.3"],
    "description": "Web server 400 error code.",
    "groups": ["web", "accesslog", "attack"],
    "id": "31101",
    "nist_800_53": ["SA.11", "SI.4"],
    "gdpr": ["IV_35.7.d"]
  },
  "decoder": {
    "name": "web-accesslog"
  },
  "full_log": "10.237.1.238 - - [08/Feb/2022:07:30:16 +0000] \"GET /javascript/xmlrpc
  HTTP/1.1\" 404 363 \"-\" \"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)\"",
  "input": {
    "type": "log"
  },
  "@timestamp": "2022-02-08T07:30:16.000000Z",
  "location": "/var/log/apache2/intranet-access.log",
  "id": "1687025674.40247962"
}
```


Figura 1 – Exemplo de alerta do *dataset* AIT-ADS.

Para estruturar e antecipar essas etapas, a literatura propõe modelos que descrevem o ciclo de vida de uma intrusão. A mais difundida é a *Cyber Kill Chain*, proposta por Hutchins, Cloppert e Amin [18], que decompõe uma intrusão em sete fases sequenciais: reconhecimento (*reconnaissance*), armamento (*weaponization*), entrega (*delivery*), exploração (*exploitation*), instalação (*installation*), comando e controle (*command and control*, C2) e ações sobre os objetivos (*actions on objectives*). A premissa do modelo é que o atacante deve necessariamente progredir por essas fases para obter sucesso, consequentemente, a detecção e a interrupção em qualquer estágio anterior à conclusão são suficientes para frustrar o ataque, o que incentiva o deslocamento da defesa de uma postura reativa para uma postura proativa.

Em contraste com a *Cyber Kill Chain*, que oferece uma visão linear e de alto nível, o *framework* MITRE ATT&CK [19] fornece uma taxonomia mais granular e fundamentada empiricamente no comportamento adversário, organizada em táticas e técnicas. As táticas representam os objetivos de curto prazo do atacante (i.e., o porquê de uma ação, como

persistência, escalonamento de privilégios ou exfiltração), enquanto as técnicas descrevem os meios concretos para alcançá-los (i.e., o como, a exemplo do roubo de credenciais do sistema ou *download* de arquivo malicioso). Por ser construído a partir de observações de ataques reais, o ATT&CK tornou-se uma referência amplamente adotada para rotular e categorizar atividades maliciosas, e os identificadores de tática (`mitre.tactic`) figuram entre os metadados associados aos alertas analisados neste trabalho.

A consequência prática desses modelos é o reenquadramento do problema de detecção. Um alerta individual constitui apenas o sintoma de uma fase do ataque. O cenário completo surge quando os alertas são ordenados temporalmente e associados às etapas da cadeia de ataque. Essa perspectiva sustenta o tratamento do fluxo de alertas como uma sequência ordenada de eventos, na qual a posição de cada alerta e suas relações com os demais carregam informações essenciais para a reconstrução da campanha, fundamentação que motiva tanto a correlação de alertas, discutida a seguir, quanto a modelagem sequencial adotada posteriormente neste trabalho.

2.3 Correlação de Alertas e Predição de Ataques

Em termos gerais, a correlação se refere ao processo de identificar e estabelecer relacionamentos entre elementos distintos de tal forma que sua interpretação coletiva revele uma estrutura ou padrão coerente [20]. No contexto da cibersegurança, a correlação de alertas é o processo interpretativo por meio do qual múltiplos alertas de baixo nível são analisados conjuntamente. Em vez de tratar alertas como ocorrências independentes, a correlação os interpreta como componentes inter-relacionados de um cenário mais amplo, o que permite que analistas ou sistemas automatizados descubram as estratégias de ataque subjacentes [20].

O processo de correlação de alertas abrange um conjunto de operações projetadas para refinar e contextualizar a informação. A Normalização e Redução de alertas tem como objetivo a remoção de falsos positivos e a fusão de alertas redundantes que se referem ao mesmo evento [6]. O Enriquecimento corresponde à adição de informações contextuais, como o número de alertas repetidos recentes ou a categoria da vulnerabilidade identificada, e a Inferência de Relacionamentos realiza a identificação de ligações causais ou temporais entre eventos distintos [20, 14]. Por fim, a Priorização corresponde à atribuição de pontuações de severidade baseadas na relevância do alvo ou na confiança da detecção [6]. Essas operações transformam fluxos de alertas complexos e ruidosos em representações estruturadas de atividade de ameaça, indispensáveis para as etapas posteriores [3].

Conforme a taxonomia apresentada por Mirheidari, Arshad e Jalili [14], os métodos de correlação classificam-se em três categorias. Os métodos baseados em similaridade correlacionam alertas pela comparação de atributos, como endereços IP de origem e destino, portas e intervalos de tempo, identificando eventos contextualmente relacionados

por sua proximidade no espaço de características. Os métodos baseados em conhecimento empregam modelos predefinidos, como regras de correlação, grafos de ataque e bibliotecas de cenários conhecidos, para estabelecer relacionamentos entre alertas com base em estruturas de ataque documentadas, tendo como principal limitação a incapacidade de detectar variações ou ataques inéditos não contemplados na base de conhecimento. Os métodos baseados em estatística identificam correlações pela análise de propriedades como frequência, coocorrência e distribuições temporais, sem exigir conhecimento prévio dos cenários de ataque [14], o que os torna mais adaptáveis a novas ameaças ao custo de maior complexidade de interpretação.

Uma vez que os alertas são correlacionados e o cenário de ataque reconstruído, torna-se possível ir além da detecção e antecipar os próximos movimentos do adversário. Neste trabalho, adota-se a definição de predição de ataques proposta por Husák et al. [21], segundo a qual o objetivo é antecipar os próximos passos de um adversário com base em sequências observáveis de ações, inferindo não apenas o tipo de evento malicioso esperado, mas também os ativos envolvidos e o momento em que ele deve ocorrer. Essa capacidade move o paradigma de segurança de uma postura reativa, que responde a ataques em andamento, para uma postura proativa, na qual estratégias de defesa podem ser acionadas antes que o dano seja causado.

2.4 Arquiteturas Baseadas em Atenção

A modelagem de dependências sequenciais e contextuais é um dos principais desafios em aprendizado profundo. Embora Redes Neurais Recorrentes (RNNs) e *Long Short-Term Memory* (LSTMs) tenham sido historicamente utilizadas para processar sequências, elas apresentam limitações quanto ao paralelismo e à captura de dependências de longo alcance, devido à natureza sequencial de sua computação [22]. Como alternativa, a arquitetura *Transformer*, introduzida por Vaswani et al. [11], prescinde de recorrência e convolução, baseando-se inteiramente em mecanismos de atenção para computar representações globais de suas entradas e saídas.

O núcleo dessa arquitetura é o mecanismo de Autoatenção (em inglês, *Self-Attention*). Conceitualmente, esse mecanismo mapeia uma consulta e um conjunto de pares chave-valor para uma saída. Diferente de abordagens anteriores, o *Transformer* permite que cada elemento de uma sequência (e.g., uma palavra ou um evento) pondere sua relação com todos os outros elementos da mesma sequência simultaneamente, independentemente da distância posicional entre eles. Matematicamente, as entradas são projetadas em três vetores distintos: Consulta (Q - *Query*), Chave (K - *Key*) e Valor (V - *Value*). Formalmente, a operação de atenção, denominada *Scaled Dot-Product Attention*, recebe como entrada as matrizes Q e K de dimensão d_k, e a matriz de Valores V, que contém o conteúdo informacional ou as características latentes associadas a cada chave. O cálculo é realizado

através da função *softmax* aplicada ao produto escalar de Q e K, escalonado por $\sqrt{d_k}$, conforme a Equação 2.2:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

onde o produto QK^T calcula a similaridade entre as consultas e as chaves. O fator de escala $\frac{1}{\sqrt{d_k}}$ é introduzido para evitar que o produto escalar cresça excessivamente em magnitude, o que leva *softmax* a regiões com gradientes extremamente pequenos, dificultando o treinamento via *backpropagation* [11].

Entretanto, aplicar um único mecanismo de atenção limita a capacidade do modelo de focar em diferentes posições e subespaços de representação simultaneamente. Para mitigar isso, o *Transformer* utiliza a Autoatenção com Múltiplos Cabeçotes (em inglês, *Multi-Head Attention*). Este mecanismo projeta linearmente as consultas, chaves e valores h vezes com diferentes projeções lineares aprendidas para dimensões d_k, d_k e d_v, respectivamente.

Matematicamente, dado um número de cabeçotes h, a saída é a concatenação dos resultados de cada cabeçote projetada novamente, como descrito nas Equações 2.3 e 2.4:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.3}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

onde as projeções são matrizes de parâmetros $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ e $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. Essa abordagem permite que o modelo atenda, de forma conjunta, a informações de diferentes subespaços de representação em diferentes posições.

Essa capacidade de ponderação dinâmica é o que permite ao modelo mapear o verdadeiro significado de um evento de segurança. Por exemplo, um alerta de *Port Scan*, que isoladamente pode ser ruído, recebe um peso de atenção significativamente maior quando contextualmente associado a um alerta subsequente de *Web Shell Upload* originado do mesmo IP. O resultado do processamento pelo *Transformer Encoder* é uma sequência de *embeddings* contextuais, onde cada vetor não representa mais apenas o alerta isolado, mas o alerta enriquecido pelo contexto dos eventos que o cercam.

É importante notar que, diferentemente das RNNs, a arquitetura *Transformer* não processa os dados sequencialmente e, portanto, é invariante à ordem dos elementos. Para corrigir essa invariância e permitir que o modelo distinga a posição relativa dos elementos em uma sequência, injeta-se uma Codificação Posicional (em inglês, *Positional Encoding*) aos vetores de entrada. Isso garante que o modelo consiga distinguir a posição relativa dos eventos dentro da janela de alertas.

Para implementar essa injeção, utilizam-se frequências de funções senoidais e cossenoidais. Para uma posição *pos* e a dimensão i do vetor de *embedding*, o *Positional Encoding* (PE) é definido pelas Equações 2.5 e 2.6:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.5}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.6}$$

Dessa forma, cada dimensão da codificação posicional corresponde a uma sinusoide, o que permite ao modelo aprender a atender posições relativas facilmente, uma vez que para qualquer deslocamento fixo k, PE_{pos+k} pode ser representado como uma função linear de PE_{pos} [11].

2.5 Aprendizado Contrastivo e Redes Siamesas

A discriminação entre diferentes classes de ataque em um espaço vetorial não depende somente da arquitetura da rede, mas também da função objetivo empregada no treinamento. Contudo, em ambientes reais a obtenção de rótulos confiáveis é um grande problema. A triagem é cara e demorada, e a maior parte dos alertas chega a um sistema de produção sem qualquer indicação de *ground truth*, frequentemente em regime contínuo (em inglês, *online*) [23]. Esse cenário dificulta abordagens de classificação supervisionada que pressupõem um conjunto fechado e previamente conhecido de classes, sobretudo porque novos tipos de ataque surgem continuamente.

Diante dessa limitação, o Aprendizado de Representação (do inglês, *Representation Learning*) oferece uma alternativa mais adequada. Em vez de aprender fronteiras de decisão entre categorias fixas, o objetivo passa a ser aprender um mapeamento dos dados para um espaço vetorial no qual a distância geométrica reflete a similaridade semântica entre os eventos [24]. Uma vez aprendido esse espaço, a descoberta de cenários pode ser delegada a métodos não supervisionados, como o agrupamento discutido na próxima seção, sem exigir a enumeração prévia das classes de ataque. A propriedade de aprender uma estrutura útil sem depender de rótulos densos ou de um conjunto pré-definido de classes é o que torna o Aprendizado Contrastivo particularmente atrativo no contexto de cibersegurança.

A construção desse espaço de representação se apoia na arquitetura de Redes Siamesas (em inglês, *Siamese Networks*), introduzidas por Bromley et al. [25] no contexto de verificação de assinaturas. Uma rede siamesa é composta por duas ou mais sub-redes idênticas que compartilham os mesmos pesos e que processam, em paralelo, entradas distintas. Como os ramos são idênticos, entradas semelhantes são mapeadas para regiões próximas do espaço de saída, e a comparação entre as representações resultantes é feita por uma métrica de distância. O treinamento não ajusta a rede para simplesmente emitir um rótulo, mas para organizar o espaço de *embeddings* de modo que a distância entre as saídas seja pequena para pares similares e grande para pares dissimilares. Essa formulação foi posteriormente generalizada para o aprendizado de métricas profundas (em inglês, *deep metric learning*), consolidando o uso de funções de perda baseadas em

distância [26]. No presente trabalho, os ramos da rede siamesa correspondem ao mesmo *encoder* Transformer descrito na Seção 2.4, aplicado de forma compartilhada a cada elemento de um tripleto de alertas.

A função objetivo que guia esse processo é o Aprendizado Contrastivo (do inglês, *Contrastive Learning*), cujo princípio é aproximar representações de eventos correlacionados enquanto se maximiza a distância entre eventos não relacionados [26]. Um caso particular amplamente adotado é a Perda Tripleto (em inglês, *Triplet Loss*), popularizada em tarefas de reconhecimento de padrões [27]. Nessa formulação, o modelo é otimizado com base em tripletos compostos por uma Âncora (A), um exemplo Positivo (P) similar à âncora e um exemplo Negativo (N) dissimilar. A função de perda satisfaz a seguinte inequação:

$$\mathcal{L}(A, P, N) = \max(0, |f(A) - f(P)|_2^2 - |f(A) - f(N)|_2^2 + \alpha) \tag{2.7}$$

onde $f(\cdot)$ é o mapeamento aprendido para um espaço Euclidiano d-dimensional, $\| \cdot \|_2$ denota a norma Euclidiana L_2, e α é uma margem que impõe uma separação mínima entre o par positivo e o negativo.

A eficácia do treinamento contrastivo depende essencialmente de como os tripletos são construídos. Tripletos formados por exemplos triviais, nos quais a distância entre a âncora e o negativo já supera a distância entre a âncora e o positivo pela margem α, produzem perda nula e não contribuem para o aprendizado, pois a restrição da Equação 2.7 já está satisfeita sem atualização dos pesos. Por isso, a literatura propõe empregar a Mineração de Negativos Difíceis (em inglês, *Hard Negative Mining*, HNM), que seleciona, para cada âncora, os exemplos negativos mais próximos no espaço atual, ou seja, aqueles que a rede neural ainda confunde com o positivo [27]. Como o espaço de *embeddings* evolui a cada época, o processo é iterativo. A cada ciclo, a rede neural é treinada e os negativos difíceis são re-minerados sobre as novas representações, o que refina progressivamente as fronteiras do espaço métrico. Na formulação supervisionada, os pares positivos e negativos são definidos a partir de rótulos estruturais dos dados (e.g., alertas pertencentes ao mesmo estágio da cadeia de ataque ou à mesma campanha são tratados como positivos). Essa estratégia reduz a dependência de rotulação manual, pois a supervisão surge a partir da estrutura relacional dos alertas, mas ainda pressupõe a existência desses rótulos estruturais.

Em um cenário de produção estritamente não supervisionado, no qual mesmo esses rótulos estruturais são indisponíveis, a mineração de negativos difíceis exige uma fonte alternativa de supervisão. A resposta na literatura é o ciclo de pseudo-rótulos via agrupamento, exemplificado pelo *DeepCluster* [28]. A abordagem consiste em agrupar os *embeddings* correntes com um algoritmo de agrupamento, atribuir a cada alerta o identificador de seu grupo como um rótulo temporário (i.e., pseudo-rótulo) e, então, minerar os tripletos e os negativos difíceis a partir desses pseudo-rótulos. Após o re-treinamento, os *embeddings* são reagrupados e os pseudo-rótulos atualizados, repetindo-se o ciclo. Dessa forma, o

agrupamento passa a fornecer o sinal de supervisão necessário para refinar a própria representação, viabilizando a aplicação do método em regime contínuo e sem rótulos.

2.6 Agrupamento Baseado em Densidade

A correlação de alertas em cenários de ataque pode ser formalizada como um problema de agrupamento (do inglês, *clustering*), cujo objetivo é particionar o conjunto de dados de modo que elementos semântica ou temporalmente similares sejam alocados no mesmo grupo. No entanto, a natureza dos dados de ataques impõe desafios aos algoritmos clássicos de particionamento, como o K-Means [29]. Tais métodos exigem a definição *a priori* do número de grupos (k) e tendem a assumir que os *clusters* possuem formato esférico e variância similar. Essas premissas são violadas em intrusões reais, onde o número de ataques simultâneos é desconhecido e os padrões de ataque podem assumir formatos arbitrários no espaço de características. Adicionalmente, algoritmos rígidos de particionamento forçam a inclusão de todos os pontos em algum grupo, o que é inadequado para dados de segurança que contêm grande volume de ruído (i.e., falsos positivos). Para superar essas limitações, a literatura aponta a eficácia dos algoritmos baseados em densidade, que definem *clusters* como regiões de alta concentração de pontos separadas por regiões de baixa densidade [30, 31].

Um dos principais algoritmos desta categoria é o DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [30]. O algoritmo classifica os pontos com base na densidade local. Formalmente, para um ponto p e um raio ε, a vizinhança $N_\varepsilon(p)$ é dada por:

$$N_\varepsilon(p) = \{q \in D \mid \text{dist}(p, q) \leq \varepsilon\} \tag{2.8}$$

Um ponto p é classificado como núcleo (em inglês, *core point*) se sua vizinhança contém no mínimo $MinPts$ pontos. Essa estratégia permite identificar grupos com formatos arbitrários e a exclusão automática de *outliers*, o que é necessário para distinguir alertas de campanhas de ataque reais de falsos positivos.

Apesar de sua eficácia, o DBSCAN assume uma densidade similar para todos os grupos, o que pode falhar em cenários heterogêneos onde diferentes ataques apresentam densidades distintas. Uma evolução para este problema é o algoritmo hierárquico HDBSCAN (*Hierarchical Density-Based Spatial Clustering of Applications with Noise*) [31], que elimina a dependência de um parâmetro ε global. O HDBSCAN introduz o conceito de Distância de Alcançabilidade Mútua (em inglês, *Mutual Reachability Distance*) para transformar o espaço métrico, o que o torna mais separável. Formalmente:

$$d_{mreach_k}(a, b) = \max\{\text{core}_k(a), \text{core}_k(b), d(a, b)\} \tag{2.9}$$

onde $\text{core}_k(x)$ é a distância do ponto x ao seu k-ésimo vizinho. Com a conversão do problema de agrupamento em um problema de otimização de estabilidade ao longo da hierarquia de

densidade, o HDBSCAN apresenta uma abordagem mais adaptativa para a descoberta de estruturas latentes em dados complexos e ruidosos, sem exigir a parametrização rígida de limiares de distância.

Cabe ressaltar, contudo, que o HDBSCAN é um algoritmo de processamento em lote (do inglês, *batch*). A construção da hierarquia de densidade e a subsequente extração dos *clusters* por estabilidade exigem acesso simultâneo a todo o conjunto de pontos, não havendo atualização incremental à medida que novos dados chegam. Essa característica é compatível com o regime *offline* adotado neste trabalho, no qual a descoberta de cenários é realizada de forma analítica sobre janelas de alertas previamente coletadas. Nesse contexto, o custo computacional é amortizado e o acesso à totalidade dos dados está garantido, de modo que as vantagens do algoritmo (i.e., a dispensa de um número de grupos pré-definido, a tolerância a densidades variáveis e o tratamento automático de ruído) podem ser plenamente exploradas, o que justifica sua escolha para a etapa de descoberta.

Em um cenário de produção, entretanto, os alertas chegam continuamente como um fluxo, e reexecutar o HDBSCAN sobre a base completa a cada nova observação torna-se proibitivo. Para esse caso, a literatura oferece algoritmos de agrupamento por fluxo de natureza incremental, que mantêm resumos compactos dos dados (i.e., micro-*clusters*) e os atualizam em uma única passagem, com memória limitada. Um exemplo é o DenStream [32], que preserva as propriedades do agrupamento baseado em densidade, adaptando-as ao processamento *online*. Esses métodos trocam parte da qualidade de agrupamento obtida em modo *batch* pela capacidade de operação contínua, e se associam naturalmente com o ciclo de pseudo-rótulos discutido na seção anterior, no qual o agrupamento atua como fonte de supervisão em ambientes sem rótulos.

2.7 Redes de Ponteiros

Na tarefa de predição do próximo endereço IP em uma sequência de alertas, os endereços de atacante e vítima tendem a se repetir ao longo dos estágios de um incidente. Essa propriedade sugere selecionar um endereço já observado na janela de contexto em vez de gerá-lo a partir de um espaço de saída de tamanho 2^{32}. Para formalizar essa intuição, este trabalho recorre às Redes de Ponteiros (em inglês, *Pointer Networks*), introduzidas por Vinyals, Fortunato e Jaitly [33] como uma extensão dos modelos de sequência para sequência (*seq2seq*) que incorporam atenção [34]. O problema motivador era a solução de tarefas combinatórias como o Problema do Caixeiro Viajante (em inglês, *Travelling Salesman Problem*, TSP) e o fecho convexo (em inglês, *convex hull*), nas quais a saída corresponde a uma subsequência ordenada dos próprios elementos de entrada, não a um vocabulário fixo. Nesses problemas, uma cabeça gerativa convencional, cujo tamanho de vocabulário é definido estaticamente no momento do treinamento, não pode generalizar

para instâncias com número de cidades ou pontos diferente do visto no treino.

A principal diferença em relação aos mecanismos de atenção convencionais, descritos na Seção 2.4, está no uso que se faz da distribuição de atenção. Nos modelos convencionais, os pesos de atenção são utilizados para computar uma média ponderada dos estados ocultos do codificador, resultando em um vetor de contexto que alimenta o decodificador. O índice de posição com maior peso não é diretamente observável na saída. Em redes de ponteiros, a própria distribuição de atenção sobre as posições de entrada $\{1, \ldots, K\}$ é a saída do modelo, o que permite ao modelo selecionar valores diretamente do contexto observado sem depender de um vocabulário de saída fixo. Essa propriedade torna o modelo generalizável para sequências de tamanho variável. O *token* selecionado é aquele para o qual o peso de atenção é máximo. Formalmente, dados os estados do codificador $\mathbf{e}_1, \ldots, \mathbf{e}_K$ e um estado de consulta (*query*) $\mathbf{q}$, os logits de atenção são:

$$u_i = \mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{e}_i + \mathbf{W}_2 \mathbf{q}), \tag{2.10}$$

e a distribuição de ponteiro é $p(i \mid \mathbf{q}) = \text{softmax}(\mathbf{u})_i$. Durante a inferência, o índice selecionado é $i^* = \arg\max_i p(i \mid \mathbf{q})$, e o elemento de entrada correspondente é copiado literalmente para a saída.

Trabalhos posteriores [35] aprofundaram esse mecanismo no contexto de sumarização automática de textos, propondo as redes ponteiro-geradoras (do inglês, *pointer-generator networks*). Nessas redes, um escalar de portão $p_{\text{gen}} \in [0, 1]$ controla o modo de produção da saída. Quando $p_{\text{gen}} = 1$, o modelo gera um *token* inteiramente a partir do vocabulário do decodificador; quando $p_{\text{gen}} = 0$, copia um *token* da entrada pela distribuição de atenção. Valores intermediários interpolam linearmente entre as duas distribuições. Adicionalmente, foi introduzida uma variante de treinamento supervisionado em que, quando o token-alvo está presente na entrada, a perda de entropia cruzada sobre a distribuição de atenção é calculada diretamente em relação à posição correta na entrada, em vez de ser calculada sobre a distribuição de vocabulário. Essa estratégia de perda de ponteiro supervisionada (do inglês, *supervised pointer loss*) reforça que a atenção se concentre na posição que contém o valor-alvo, melhorando o ajuste do mecanismo de cópia independentemente do valor de p_{gen}.

O mecanismo de ponteiro é naturalmente adequado ao problema de predição de endereços IP em sequências de alertas gerados por IDS. O mesmo *host* atacante persiste por múltiplos estágios da cadeia de ataque, e o alvo permanece fixo ou migra de forma previsível durante o movimento lateral. Formalmente, dado um conjunto de K alertas anteriores com endereços IP representados como vetores de octetos normalizados $\mathbf{c}_1, \ldots, \mathbf{c}_K \in [0, 1]^4$, e um vetor de consulta $\mathbf{q}$ derivado da representação agregada da sequência, a distribuição de ponteiro sobre as posições do contexto é obtida por atenção de produto escalar escalonado [11]:

$$\mathbf{s} = \frac{\mathbf{q}\,\mathbf{C}^{\top}}{\sqrt{d_a}}, \qquad \boldsymbol{\alpha} = \mathrm{softmax}(\mathbf{s}) \in \mathbb{R}^{K}, \tag{2.11}$$

em que $\mathbf{C} \in \mathbb{R}^{K \times d_a}$ é a matriz de chaves obtida projetando os octetos IP do contexto, e d_a é a dimensão de atenção. O endereço predito na inferência é a cópia direta dos octetos da posição $i^* = \arg\max \boldsymbol{\alpha}$. Durante o treinamento, a perda de ponteiro supervisionada [35] direciona a atenção para as posições cujo IP coincide com o alvo, sem impor que apenas uma posição seja válida.

2.8 Resumo

Neste capítulo, foram estabelecidos os fundamentos teóricos necessários para a compreensão da abordagem proposta. Inicialmente, discutiu-se o ecossistema de monitoramento de segurança e a natureza heterogênea dos dados gerados por ferramentas como IDS e EDR, destacando as limitações dos métodos de correlação tradicionais baseados em regras estáticas. Em seguida, apresentaram-se os modelos de ataques multiestágio (i.e., a *Cyber Kill Chain* e o MITRE ATT&CK), que estruturam a progressão do adversário em fases sequenciais e evidenciam por que a detecção de ameaças exige a correlação de múltiplos eventos ao longo do tempo. Para superar esses desafios, fundamentou-se a aplicação de técnicas de Aprendizado Profundo. Detalharam-se os conceitos de representação vetorial, essenciais para transformar dados simbólicos e temporais em informação computável, e a arquitetura *Transformer*, que por meio de mecanismos de atenção permite a modelagem de dependências sequenciais e contextuais entre eventos de segurança. Além da arquitetura, exploraram-se as Redes Siamesas e o Aprendizado Contrastivo com *Triplet Loss*, que fundamentam a construção de espaços métricos semânticos em cenários com escassez de rótulos, incluindo a extensão ao regime não supervisionado por meio do ciclo de pseudo-rótulos via agrupamento. Apresentou-se ainda a fundamentação dos algoritmos de agrupamento baseados em densidade (i.e., DBSCAN e HDBSCAN), sua adequação ao regime *offline* e as alternativas incrementais, como o DenStream, para operação contínua em regime *online*. Por fim, introduziu-se o mecanismo de Redes de Ponteiros, que fundamenta a predição de endereços IP em sequências de alertas ao copiar valores já observados no contexto em vez de gerá-los a partir de um vocabulário fixo. Esse referencial teórico compõe a base sobre a qual a abordagem foi desenvolvida. O próximo capítulo apresenta os trabalhos relacionados, contextualizando a abordagem proposta em relação ao estado da arte.

3 Trabalhos Relacionados

Este capítulo apresenta a revisão dos trabalhos relacionados à correlação dinâmica de alertas de segurança, à predição de ataques e à aplicação de aprendizado profundo multitarefa em cibersegurança. A literatura sobre correlação de alertas e predição de ataques é extensa, mas a abordagem dinâmica do problema, na qual múltiplas campanhas de ataque ocorrem simultaneamente e o conjunto de táticas observáveis evolui continuamente, permanece pouco explorada na maioria dos trabalhos, voltados a cenários estáticos com um único contexto de ataque. Essa lacuna tem motivado evoluções recentes na área, impulsionadas pela necessidade de superar as limitações dos métodos estáticos diante da crescente sofisticação das ameaças atuais. Para fornecer uma visão estruturada do estado da arte, o capítulo organiza-se em três eixos conforme detalhado a seguir.

A Seção 3.1 discute as abordagens clássicas de correlação e as técnicas modernas de reconstrução de cenários de ataque, incluindo métodos baseados em grafos de proveniência e na atribuição de campanhas simultâneas. A Seção 3.2 examina os métodos voltados à predição de ataques, com foco em arquiteturas sequenciais para estimativa de estágios da *kill-chain* e antecipação dos próximos passos do adversário. A Seção 3.3 abrange as abordagens de aprendizado multitarefa e de análise semântica de eventos de segurança, examinando como técnicas de Processamento de Linguagem Natural (NLP) e de representação vetorial têm sido aplicadas à detecção e à classificação de ameaças. Por fim, a Seção 3.4 sintetiza as lacunas identificadas que motivam a abordagem proposta neste trabalho.

3.1 Correlação de Alertas e Reconstrução de Cenários

A correlação de alertas evoluiu para um componente indispensável na segurança de redes, atuando como uma camada de pós-processamento para Sistemas de Detecção de Intrusão (IDS) [14, 6]. A principal motivação está em uma limitação inerente aos IDS, a geração de um volume massivo de alertas de baixa qualidade, dos quais a literatura aponta que cerca de 99% podem ser falsos positivos [6]. Nesse contexto, a correlação é definida como o processo interpretativo pelo qual múltiplos alertas são analisados conjuntamente, de modo que um novo significado emerja do conjunto, como a identificação de uma campanha de varredura de portas seguida por uma tentativa de autenticação bem-sucedida no mesmo host, que isoladamente seriam dois eventos de baixa severidade, mas que em conjunto indicam um possível comprometimento em andamento [20]. Conforme a taxonomia proposta em [14], os métodos de correlação classificam-se em três categorias principais, discutidas a seguir.

Os métodos baseados em similaridade agrupam alertas pela coincidência de atributos como endereços IP, portas e intervalos de tempo, com o objetivo de reduzir a redundância

sem exigir a definição prévia dos tipos de ataque. Embora eficazes na fusão e deduplicação de alertas, apresentam limitações na detecção de sequências complexas, pois dependem de regras simples de agrupamento que não capturam relações causais entre eventos de estágios distintos. A coincidência de atributos é uma condição necessária, mas insuficiente para correlacionar eventos pertencentes a diferentes fases de uma mesma campanha [14].

Os métodos baseados em conhecimento utilizam modelos predefinidos, como regras de correlação, grafos de ataque e bibliotecas de cenários documentados, para estabelecer relacionamentos entre alertas com base em estruturas de ataque conhecidas. Embora ofereçam alta precisão na detecção de ameaças conhecidas, são limitados pela necessidade de construir e manter manualmente bases de conhecimento especializadas, tornando-os ineficazes contra variantes ou ataques inéditos. A dependência de codificação explícita cria um ciclo de manutenção insustentável que não acompanha a velocidade de evolução das táticas adversariais [14].

Os métodos baseados em estatística identificam correlações ao analisar propriedades como frequência, coocorrência e distribuições temporais entre eventos, sem exigir conhecimento prévio explícito sobre cenários de ataque [36]. Esses algoritmos operam sobre dados históricos para armazenar relações causais entre incidentes, o que lhes confere certa adaptabilidade a novas ameaças em comparação às abordagens baseadas em conhecimento. No entanto, sua aplicabilidade é restrita a domínios em que os atributos podem ser rigorosamente modelados, e o desempenho se degrada significativamente quando os sensores fornecem informações incompletas ou quando os padrões de ataque são raros no conjunto de treinamento [14].

Como evolução às técnicas estáticas, surgiram propostas focadas na construção automática de cenários de ataques multiestágio. Liu et al. [37] propuseram um modelo híbrido que combina Redes Neurais e Redes Bayesianas, no qual a rede neural filtra falsos positivos dos registros brutos enquanto um Grafo de Ataque Bayesiano modela as dependências causais entre alertas para reconstruir cenários de ataque multi-estágio. Embora o método tenha demonstrado alta precisão na reconstrução de cenários no *dataset* DARPA 2000, ainda depende de regras de associação causal predefinidas para gerar as sequências iniciais, o que mantém uma dependência parcial de conhecimento especializado [37].

Nadeem et al. [38] propõem o SAGE, uma abordagem alternativa que extrai grafos de ataque diretamente dos alertas de intrusão sem exigir conhecimento prévio de especialistas. O sistema explora a dependência temporal e probabilística entre alertas por meio de um Autômato Finito Determinístico Probabilístico Baseado em Sufixos (em inglês, *Suffix-based Probabilistic Deterministic Finite Automaton*, S-PDFA), um modelo que representa sequências de eventos como estados de um autômato, atribuindo probabilidades às transições entre eles e dando maior destaque a alertas raros de alta severidade, extraindo subgrafos orientados por pares vítima-objetivo. Em avaliação sobre *datasets* de competições de segurança, o SAGE comprimiu mais de 330 mil alertas em 93 grafos interpretáveis,

evidenciando que a estrutura sequencial dos alertas contém informação suficiente para a reconstrução de cenários sem supervisão especializada [38].

Kannan et al. [39] propõem o MAAT, sistema que aborda o desafio de identificar o atacante original quando múltiplos fluxos de rede se misturam ao longo da cadeia de comprometimento em ataques multiestágio. O sistema emprega Redes Definidas por Software (SDN) para embaralhar periodicamente as atribuições de clientes a réplicas de servidores, mantendo o estado de risco de cada fluxo e isolando progressivamente os fluxos maliciosos. Embora eficaz em sua proposta, a abordagem depende de infraestrutura SDN especializada e opera por meio da movimentação de alvo, em vez de aprender representações dos padrões comportamentais do atacante [39].

Milajerdi et al. [40] propõem o HOLMES, sistema que realiza a correlação de ataques persistentes avançados (APTs) por meio da análise de grafos de proveniência, estruturas que representam processos, arquivos e conexões de rede como nós e as relações causais entre eles como arestas, possibilitando rastrear fluxos de informação suspeitos entre essas entidades. A abordagem mapeia os fluxos detectados para as táticas do MITRE ATT&CK, a base de conhecimento de táticas e técnicas adversariais apresentada na Seção 2.2, construindo automaticamente cenários de ataque sem depender de assinaturas predefinidas. Em avaliação sobre *datasets* do DARPA, o HOLMES demonstrou capacidade de detecção em tempo real com precisão e baixa taxa de falsos alarmes [40], embora abordagens baseadas em grafos de proveniência em escala enfrentem, de forma geral, o desafio da explosão de dependências, no qual o número de relações causais cresce exponencialmente com o tamanho do grafo, dificultando a extração de cenários interpretáveis [41].

Recentemente, Liu et al. [42] introduziram o Trace2Vec, sistema que modela eventos do sistema como um grafo heterogêneo dinâmico para a detecção de ataques multiestágio, com ênfase na explicabilidade. Para mitigar a escassez de dados reais de ataque, o sistema emprega uma função de aumento de dados que gera amostras sintéticas de ataques raros a partir dos grafos de eventos existentes. Adicionalmente, o sistema emprega busca em árvore de Monte Carlo para identificar quais eventos contribuíram mais significativamente para a classificação de anomalia, mitigando o problema da caixa-preta comum em modelos de aprendizado profundo. Assim como o HOLMES, o Trace2Vec opera sobre um único contexto de ataque por grafo, sem mecanismos para separar campanhas simultâneas de múltiplos atacantes que se entrelaçam nos mesmos eventos [42].

As revisões mais recentes da literatura [6, 3] reforçam que, apesar dos avanços, a maioria dos sistemas de correlação ainda opera de forma retrospectiva sobre alertas já observados, sem capacidade de antecipar o que vem a seguir. Os métodos baseados em grafos de proveniência, como HOLMES e Trace2Vec, ampliam o poder de reconstituição de cenários, mas assumem implicitamente que os eventos analisados pertencem a um único contexto de ataque. A separação dinâmica de campanhas simultâneas, na qual múltiplos atacantes operam em paralelo e seus alertas se entrelaçam no mesmo fluxo, permanece

um problema em aberto cuja solução requer modelagem de representações capazes de distinguir incidentes distintos sem rótulos de atribuição disponíveis.

3.2 Predição de Ataques

A predição de ataques representa uma mudança de paradigma da defesa reativa para a proativa, com o objetivo de inferir as próximas ações do adversário antes de sua execução, com base em sequências de alertas observadas. Husák et al. [21] organizam as abordagens preditivas em cibersegurança em quatro tarefas principais. A projeção de ataque e o reconhecimento de intenção estimam, respectivamente, o próximo estágio imediato do adversário e seu objetivo final, com base no histórico de eventos observados, como inferir que uma sequência de varreduras de porta seguida por tentativas de autenticação indica uma próxima tentativa de exploração de serviço (i.e., projeção de ataque), ou que esse mesmo padrão, combinado com o perfil do alvo, revela a intenção de exfiltração de dados sensíveis (i.e., reconhecimento de intenção). A predição de intrusão antecipa a ocorrência de ataques sem exigir a observação de eventos anteriores, como identificar que um host apresenta indicadores de comprometimento compatíveis com infecção iminente antes de qualquer alerta ser gerado, enquanto a previsão da situação de segurança projeta o estado global de ameaça da rede sem se ater a um atacante específico, como estimar a probabilidade de a rede sofrer um incidente crítico na próxima janela de tempo com base no volume agregado de alertas [21].

Shen et al. [43] propõem o TIRESIAS, sistema pioneiro ao reformular a predição de ataques como a estimativa da exata próxima ação do adversário, em vez de apenas detectar a ocorrência de um ataque. O sistema emprega Redes Neurais Recorrentes com Memória de Curto-Longo Prazo (em inglês, *Long Short-Term Memory*, LSTM) para aprender padrões históricos de sequências de alertas e antecipar o próximo passo específico, com desempenho superior ao de métodos probabilísticos como Cadeias de Markov. Os autores demonstraram que a formulação sequencial captura dependências temporais que técnicas estáticas não conseguem modelar, posicionando as LSTMs como ponto de partida para trabalhos subsequentes, embora a predição se limite à categoria do próximo evento, sem estimar quando ele ocorrerá ou quais atributos de rede estarão envolvidos, restrição que se mantém na maior parte dos trabalhos subsequentes na área [43].

Van Ede et al. [44] propõem o DeepCASE, sistema que realiza análise contextual semi-supervisionada de eventos de segurança. Em vez de depender inteiramente de rótulos, o sistema agrupa sequências de eventos por contexto comportamental por meio de uma rede neural que aprende representações capazes de distinguir comportamentos maliciosos de ruído sem exigir anotação completa do conjunto de treinamento. O DeepCASE demonstrou que a incorporação de contexto, isto é, a janela de eventos que precedem o alerta sob análise, melhora substancialmente a precisão da classificação em comparação ao TIRESIAS,

com redução da carga de rotulação exigida pelo analista [44].

Ansari et al. [45] propõem o uso de Unidades Recorrentes com Portão (em inglês, *Gated Recurrent Units*, GRUs) para predição de alertas de intrusão, motivados pela menor complexidade de parâmetros das GRUs em relação às LSTMs para desempenho preditivo equivalente ao do TIRESIAS. Uma contribuição relevante desse trabalho é a predição conjunta de múltiplos atributos do próximo alerta, que abrange categoria, volume, tempo aproximado e os prefixos /24 dos endereços IP de origem e destino, com foco no perfil comportamental de fontes maliciosas. Os autores demonstraram que as GRUs atingem precisão equivalente às LSTMs com tempo de treinamento reduzido, o que as torna adequadas para ambientes com restrições de recursos computacionais, embora a predição de endereços se limite ao prefixo /24 e o método assuma um fluxo de eventos de um único contexto de ataque, sem tratar o entrelaçamento de campanhas simultâneas [45].

A analogia entre fluxos de eventos de segurança e linguagem natural motivou a proposta de Fredj [46], que trata cenários de ataque como sentenças e alertas individuais como palavras. Com LSTMs treinadas sobre sequências de identificadores de assinatura de alertas, o autor obteve acurácia de 98% na predição do próximo passo de ataque, e demonstra que a representação sequencial de identificadores de alerta captura a intenção do atacante de forma eficaz. A formulação como problema de modelagem de linguagem simplifica a arquitetura ao dispensar atributos de rede complexos, embora limite a predição ao tipo de ação sem estimar atributos como endereços IP ou intervalos temporais [46].

Phan e Bauschert [47] propõem o StageFinder, que combina Redes Neurais de Grafos (GNNs) e LSTMs sobre dados de proveniência, isto é, registros que capturam as relações entre processos, arquivos e conexões de rede em um sistema, combinando fontes provenientes de *hosts* e da rede para a estimativa de estágios da *kill-chain*. As GNNs codificam as dependências estruturais entre processos, arquivos e conexões, enquanto as LSTMs capturam a dinâmica temporal para estimar probabilidades de estágio alinhadas ao MITRE ATT&CK, e o sistema atinge Macro F1 de 0,96 em avaliação sobre os *datasets* DARPA OpTC e DARPA TC. É importante distinguir que o StageFinder estima o estágio corrente do ataque em observação e não o próximo estágio a ocorrer, o que o posiciona como sistema de classificação contextual em vez de predição antecipada [47].

Zhang et al. [48] propõem o DeepOP, um *framework* híbrido para predição de sequências ATT&CK que utiliza um mecanismo de autoatenção de janela causal embutido em uma arquitetura Transformer, capaz de capturar relações causais locais e dependências globais entre técnicas de ataque. Para superar a escassez de dados rotulados em nível de técnica, os autores constroem um conjunto de dados a partir de relatórios de inteligência de ameaças cibernéticas (em inglês, *Cyber Threat Intelligence*, CTI), que extrai eventos causalmente conectados e os mapeia para táticas e técnicas do MITRE ATT&CK por meio de raciocínio ontológico. A combinação de atenção causal com conjunto de dados estruturado por ontologia permite predições de ataques multiestágio com precisão em nível de técnica

individual, embora o modelo seja treinado sobre sequências de um único atacante por campanha, sem considerar o entrelaçamento de alertas provenientes de múltiplos atacantes simultâneos nem estimar atributos adicionais como endereços IP e intervalos temporais [48].

Os trabalhos desta seção demonstram progresso significativo na predição do próximo estágio ou ação do adversário, mas apresentam lacunas em relação à abordagem proposta neste trabalho. Com exceção de Ansari et al. [45], que inclui estimativas de tempo e volume de fluxo, nenhum dos sistemas prediz simultaneamente o próximo estágio da *kill-chain*, o endereço IP de origem e de destino e o intervalo temporal até o próximo evento em uma única arquitetura. Adicionalmente, todos assumem um fluxo de alertas proveniente de um único contexto de ataque, sem abordar o entrelaçamento de campanhas simultâneas que dificulta a organização das janelas de contexto sobre as quais a predição é realizada. A abordagem proposta neste trabalho aborda essas duas lacunas por meio de um preditor multitarefa que estima conjuntamente estágio, endereços IP e intervalo temporal, organizado sobre janelas de contexto delimitadas por um codificador de campanha dedicado à separação de incidentes simultâneos.

3.3 Aprendizado Multitarefa e Análise Semântica

Paralelamente aos avanços em predição de ataques, uma linha de pesquisa tem explorado o aprendizado multitarefa e a análise semântica de eventos de segurança, aproximando-se da combinação de objetivos heterogêneos adotada neste trabalho. O aprendizado multitarefa consiste em treinar um único modelo para resolver simultaneamente múltiplos objetivos relacionados, com a expectativa de que o aprendizado compartilhado entre tarefas produza representações mais robustas do que o treinamento isolado de cada objetivo [49]. Já a análise semântica explora técnicas de NLP para extrair significado contextual dos campos textuais dos alertas, superando representações baseadas unicamente em identificadores categóricos.

Lan et al. [50] propõem o MEMBER, modelo multitarefa para detecção de intrusão em redes, combinando uma Rede Neural Convolucional (em inglês, *Convolutional Neural Network*, CNN) com mecanismos de atenção que destacam as partes mais relevantes da entrada processada pela rede a duas tarefas auxiliares, sendo um *autoencoder* com módulo de memória e uma rede prototípica baseada em distância. As tarefas auxiliares têm como objetivo melhorar a capacidade de generalização do modelo e mitigar a degradação de desempenho em cenários de classes desbalanceadas, problema recorrente em conjuntos de dados de intrusão. Os experimentos demonstraram que a combinação de tarefas produz representações mais robustas do que o treinamento de tarefa única, embora o objetivo permaneça centrado na classificação do tráfego corrente, sem componente preditivo sobre eventos futuros [50].

Wang [51] propõe uma arquitetura de fusão multimodal para detecção precoce de APTs, processando registros de eventos do Windows, fluxos de rede e alertas de *endpoints*

por meio de codificadores neurais paralelos. Um mecanismo de atenção cruzada alinha temporalmente as características provenientes das diferentes fontes, enquanto uma cabeça multitarefa identifica eventos maliciosos e classifica simultaneamente a fase do ataque segundo o MITRE ATT&CK. Os autores reportam redução da latência média de detecção para 8,7 minutos após o comprometimento inicial, com efetividade na identificação de extração de credenciais e movimento lateral, embora a tarefa de classificação de fase do ataque opere sobre o estado corrente, sem estimar a fase seguinte [51].

A aplicação de modelos de linguagem pré-treinados a registros de segurança tem explorado a riqueza semântica dos campos textuais dos alertas. Afnan et al. [52] propõem o LogShield, que utiliza o RoBERTa [53] para processar rastros extraídos de grafos de proveniência e classificar sequências de eventos do sistema como benignas ou maliciosas, identificando a presença de APTs sem engenharia manual de características. Niknami et al. [54] propõem o CrossAlert, sistema que classifica sequências de alertas como pertencentes ou não a um ataque de múltiplos estágios, combinando BERT [55] para extrair *embeddings* semânticos das mensagens de alerta com uma Rede Prototípica para lidar com o problema de *domain shift*, no qual modelos treinados em um ambiente de rede degradam significativamente ao serem aplicados a outro. Os *embeddings* semânticos são concatenados com características básicas e baseadas em sequência, além de um *score* de anomalia, e alimentados a uma rede classificadora que aprende, por meio da Rede Prototípica, um espaço métrico no qual a classificação ocorre pela distância a representações de protótipo de cada classe. Em avaliação cruzada entre os *datasets* DARPA 2000 e ISCX 2012, o CrossAlert manteve desempenho superior aos métodos de referência mesmo com apenas cinco amostras do domínio de destino, evidenciando a relevância da generalização entre domínios também para a avaliação cruzada de cenários deste trabalho [54].

Abordagens mais recentes têm explorado LLMs para correlação de alertas sem treinamento específico. Du et al. [56] propõem o MAD-LLM, que emprega engenharia de *prompts* para instruir LLMs a correlacionar alertas diretamente a partir de suas descrições textuais, dispensando o treinamento de modelos supervisionados especializados. Embora promissora pela flexibilidade, essa abordagem depende da qualidade e do custo computacional dos LLMs disponíveis, e sua aplicação em fluxos de alertas de alto volume permanece limitada pelas restrições de janela de contexto e latência de inferência. Wu et al. [57] propõem o Alert2vec, que explora o aprendizado de representações de subgrafos para alertas de segurança, capturando relações estruturais entre eventos em uma representação vetorial compacta utilizada em tarefas de classificação *downstream*.

A escassez de dados rotulados, balanceados e representativos é um obstáculo recorrente em todos os trabalhos discutidos neste capítulo. Shadabfar et al. [58] abordam esse problema propondo o DSRL-APT-2023, que utiliza uma Rede Adversária Generativa Condicional para Dados Tabulares (em inglês, *Conditional Tabular Generative Adversarial Network*, CTGAN) para gerar uma versão balanceada do *dataset* DAPT2020, demonstrando que

algoritmos de aprendizado de máquina supervisionados atingem desempenho competitivo quando treinados sobre os dados sintéticos gerados. Essa abordagem gera dados por meio de modelagem estatística da distribuição de atributos de um conjunto existente, sem a atribuição de rótulos de estágio ou de campanha aos exemplos sintéticos produzidos.

Os trabalhos desta seção evidenciam que o aprendizado multitarefa melhora a robustez das representações aprendidas e que a análise semântica enriquece a caracterização dos eventos de segurança além dos atributos categóricos. No entanto, em nenhum dos trabalhos as tarefas combinadas incluem a predição simultânea do próximo estágio da *kill-chain*, do endereço IP de origem e de destino e do intervalo temporal até o próximo evento, e a geração de dados sintéticos balanceados permanece restrita à modelagem estatística de *datasets* existentes, sem a atribuição de rótulos de campanha que permitam avaliar a separação de incidentes simultâneos.

3.4 Resumo

Este capítulo revisou os trabalhos relacionados à correlação de alertas, à predição de ataques e ao aprendizado multitarefa em cibersegurança, organizados em três eixos. Os sistemas de correlação evoluíram de regras estáticas para grafos de proveniência capazes de reconstruir cenários de ataque, mas operam de forma retrospectiva e assumem implicitamente um único contexto de ataque por vez, o que limita sua aplicabilidade em cenários com múltiplos atacantes simultâneos. Os sistemas de predição avançaram de modelos probabilísticos para arquiteturas recorrentes capazes de estimar o próximo passo do adversário, mas restringem o escopo da predição ao próximo estágio ou ação, sem estimar conjuntamente atributos como endereços IP e intervalos temporais. Os sistemas multitarefa e semânticos demonstraram que objetivos auxiliares e representações contextuais melhoram a robustez e a generalização dos modelos, mas seu uso permanece centrado na detecção e classificação do estado corrente, sem componente preditivo sobre eventos futuros.

A análise da literatura revela três lacunas que motivam a abordagem proposta neste trabalho. Primeiramente, a ausência de mecanismos de separação dinâmica de campanhas simultâneas por meio de aprendizado de representação, capazes de distinguir incidentes entrelaçados sem depender de rótulos de atribuição. Em segundo lugar, a ausência de uma arquitetura que combine, em uma única predição, o próximo estágio da *kill-chain*, os endereços IP de origem e destino e o intervalo temporal até o próximo evento. Por fim, a dependência de conjuntos de dados rotulados e balanceados, cuja escassez em ambientes de produção limita o treinamento e a avaliação de modelos preditivos. A abordagem proposta neste trabalho endereça essas lacunas por meio de dois codificadores especializados, um preditor multitarefa e um *pipeline* sintético de geração de dados, conforme detalhado no capítulo seguinte.

4 Metodologia

Este capítulo apresenta a abordagem proposta para a correlação dinâmica e a predição multitarefa de ataques cibernéticos. A abordagem foi desenvolvida para superar as limitações dos sistemas tradicionais baseados em regras, que frequentemente resultam em fadiga de alertas e na incapacidade de predizer ataques de múltiplos estágios [3]. O fluxo utilizado fundamenta-se em aprendizado profundo com *Transformers*, tratando os alertas como componentes de uma narrativa sequencial, em vez de simples eventos isolados. A partir dessa representação, a abordagem abrange dois objetivos complementares. O primeiro é a correlação dos alertas e a construção de cenários de ataque coesos, com a condensação de grandes volumes de eventos em incidentes acionáveis. O segundo é a predição multitarefa, que antecipa os atributos do próximo evento da sequência, a saber, o estágio do ataque, os endereços IP envolvidos e o intervalo temporal até sua ocorrência.

A Figura 2 apresenta a visão geral da abordagem. O processo inicia-se com a etapa de vetorização, que transforma os atributos heterogêneos de cada alerta em representações numéricas. A partir dessas representações, dois codificadores baseados na arquitetura *Transformer* e treinados por aprendizado contrastivo operam em paralelo com objetivos distintos. O codificador de campanha processa o fluxo de alertas e sua saída é submetida ao agrupamento por campanha, que delimita as fronteiras entre operações ofensivas simultâneas, enquanto o codificador de alerta produz a representação vetorial em nível de evento, utilizada como base para a predição. Por fim, o preditor multitarefa recebe as sequências delimitadas pelo agrupamento por campanha, representadas pela saída do codificador de alerta, para estimar o próximo estágio do ataque, os endereços IP envolvidos e o intervalo temporal até sua ocorrência. As seções subsequentes detalham cada uma dessas etapas.

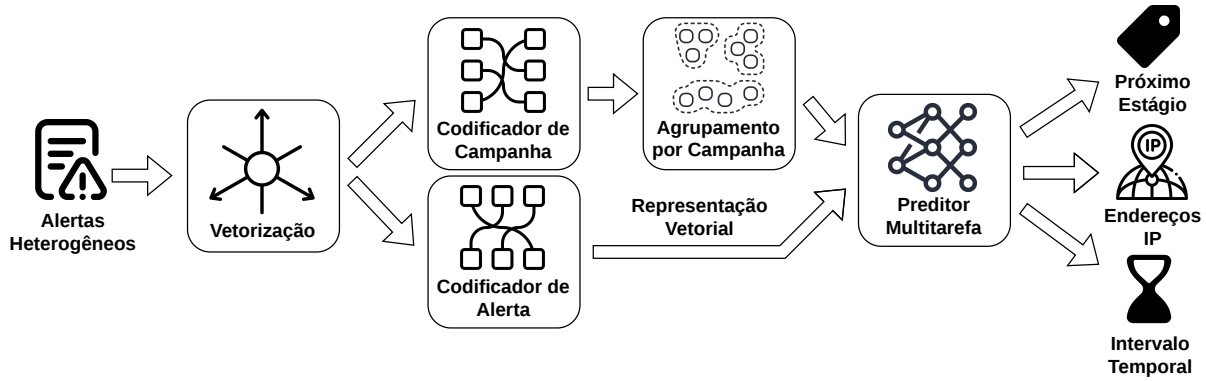

Figura 2 – Visão Geral da Abordagem

4.1 Processamento de Atributos e Representação Vetorial

A primeira etapa da abordagem consiste em transformar os alertas heterogêneos em representações numéricas que possam ser processadas pelos codificadores. Um alerta bruto é definido como uma tupla contendo atributos variados, tais como carimbo de tempo, endereços IP de origem e destino, portas, protocolo, categorias e severidade do evento, além de descrições textuais. Cada tipo de atributo é tratado de forma específica, de modo a preservar sua semântica própria, antes de ser disponibilizado aos codificadores.

Os atributos categóricos (e.g., portas, protocolo e categorias de ataque) são mapeados por uma camada de *embedding*, que associa cada categoria a um vetor denso aprendido. Para o tratamento de endereços IP, a proposta emprega uma estratégia de decomposição hierárquica em octetos, em vez de mapear cada endereço completo como uma categoria isolada, porque a alta cardinalidade do espaço de endereçamento IPv4 inviabilizaria tanto o aprendizado de um *embedding* denso por endereço quanto a generalização para endereços não vistos em produção. Cada octeto é então vetorizado individualmente, o que permite ao modelo capturar relações espaciais de sub-rede e escalar eficientemente sem a necessidade de representar todo o espaço de endereçamento IPv4. Para garantir robustez diante de dados não vistos em produção, categorias ausentes do treinamento são mapeadas para um *token* especial de desconhecido (`<UNK>`).

As descrições textuais, em especial as assinaturas de IDS (e.g., *"Web server 400 error code"*), são processadas pelo modelo de linguagem pré-treinado BERT (*Bidirectional Encoder Representations from Transformers*) [55]. A escolha pelo BERT justifica-se pela sua arquitetura de atenção bidirecional, capaz de capturar nuances semânticas e o contexto das palavras em descrições técnicas curtas. Esta etapa converte a descrição textual em um vetor semântico, o que permite que a abordagem compreenda a similaridade entre assinaturas sintaticamente distintas, mas que referenciam contextos de ataque análogos.

A dimensão temporal recebe um tratamento voltado tanto à cadência quanto à periodicidade dos eventos. Para a cadência, calcula-se o intervalo de tempo (Δt) em relação ao evento imediatamente anterior, métrica que permite distinguir ataques de força bruta (i.e., intervalos curtos) de técnicas de evasão temporal (e.g., ataques *low-and-slow*). Diferenças de cadência entre as fases de um ataque ajudam a separar comportamentos automatizados de ações manuais ou deliberadamente furtivas.

Para a periodicidade, trata-se a representação do horário do evento. Em uma escala linear (e.g., de 0 a 23 horas), a transição de 23:59 para 00:00 introduz uma descontinuidade artificial, sugerindo uma distância que não corresponde à proximidade temporal real. Para corrigir isso e capturar padrões de rotina, o horário é projetado em um círculo unitário por meio de uma codificação trigonométrica. O instante do evento é convertido para segundos decorridos no dia (t_{seg}), normalizado pelo total de segundos diários ($T_{dia} = 86400$) e mapeado em dois componentes ortogonais, definidos pela Equação 4.1.

$$t_{sin} = \sin\left(\frac{2\pi \cdot t_{seg}}{T_{dia}}\right), \quad t_{cos} = \cos\left(\frac{2\pi \cdot t_{seg}}{T_{dia}}\right) \tag{4.1}$$

Assim, eventos próximos à meia-noite preservam sua adjacência no espaço de características, de modo a possibilitar a identificação de padrões cíclicos e de rotina.

Por fim, os atributos numéricos, como estatísticas de fluxo de rede, são normalizados antes de compor a representação. Em particular, contadores de cauda longa recebem transformação logarítmica para conter sua dispersão. Isso impede que atributos de escalas muito distintas dominem o vetor resultante, preservando a contribuição equilibrada de cada característica.

O resultado dessa etapa é um conjunto de representações vetoriais para cada alerta (i.e., octetos de IP, *embeddings* categóricos, vetor textual do BERT, atributos numéricos e temporais). É a partir desse conjunto que os dois codificadores constroem suas representações, cada um consumindo o subconjunto de atributos adequado ao seu objetivo, conforme detalhado nas seções seguintes. Essa separação entre a extração de atributos e a codificação propriamente dita permite que ambos os codificadores partam de uma base comum, sem duplicar o pré-processamento.

4.2 Codificador de Campanha

O codificador de campanha é o componente responsável pela correlação dinâmica dos alertas, que produz para cada alerta uma representação que sintetiza o comportamento da campanha ativa em torno dele. Para isso, o codificador opera sobre uma janela de W alertas consecutivos centrada no alerta a ser representado, integrando o contexto local da sequência em uma única representação por janela. Nos alertas próximos às extremidades de uma campanha, a janela é deslocada para permanecer inteiramente dentro dos limites da campanha (i.e., estratégia denominada *boundary clamping*), de modo a garantir que todas as W posições sempre contenham alertas reais e evitar a discrepância entre distribuições de treino e inferência que surgiria com *zero-padding* (i.e., preenchimento das posições faltantes com vetores nulos). Ao deslizar essa janela sobre o fluxo completo de alertas, cada alerta recebe o *embedding* da janela centrada nele, o que permite que a correlação seja realizada sem depender de regras pré-definidas.

Cada alerta da janela é descrito pela concatenação de três grupos de atributos, sendo estes uma projeção linear do vetor textual do BERT, os octetos de IP de origem e destino, e o intervalo temporal Δt em relação ao alerta anterior. Antes do processamento pelo *Transformer*, uma codificação posicional é somada a cada vetor por alerta, o que permite ao modelo distinguir a posição de cada alerta na janela e preservar a ordem dos eventos. Esses vetores são processados por um *Transformer* com pré-normalização (*Pre-Norm*), composto por blocos de atenção multi-cabeça que permitem que cada posição da janela integre informações de todas as demais. A representação final da janela é obtida pela

média dos *tokens* de saída (*mean pooling*) e normalizada pela norma L_2 (i.e., divisão do vetor por sua magnitude, preservando apenas a direção), de modo a produzir um vetor que sintetiza o comportamento coletivo da sequência.

O espaço latente resultante é uma hiperesfera unitária, no qual o treinamento posiciona janelas de uma mesma campanha em regiões compactas e bem separadas das demais. Essa separabilidade é sustentada pelo fato de que janelas de uma mesma campanha compartilham os mesmos endereços IP, a mesma progressão de estágios e padrões temporais semelhantes, ao passo que janelas de campanhas distintas diferem nesses atributos. As janelas de transição entre campanhas, cujo conteúdo é misto, tendem a ocupar regiões de baixa densidade no espaço latente.

O treinamento utiliza a perda tripleto apresentada na Equação 2.7, com tripletos definidos em nível de campanha. A âncora e o exemplo positivo são janelas distintas extraídas de uma mesma campanha, enquanto o exemplo negativo pertence a uma campanha diferente. Para viabilizar o treinamento sem exigir o armazenamento de todos os tripletos possíveis, eles são gerados dinamicamente a cada lote, reduzindo substancialmente o consumo de memória. Concluído o treinamento, o agrupamento por densidade HDBSCAN é aplicado sobre os *embeddings* de todas as janelas, e cada *cluster* denso resultante corresponde a uma campanha de ataque delimitada, sendo o rótulo de cada alerta determinado pelo *cluster* da janela centrada nele.

4.3 Codificador de Alerta

O codificador de alerta é o componente responsável por produzir, para cada alerta, uma representação que organiza os eventos segundo o estágio do ataque ao qual pertencem, servindo de base para a predição realizada pelo preditor multitarefa. Para isso, o codificador de alerta opera sobre uma janela de L alertas consecutivos e produz, para cada alerta central dessa janela, uma representação contextualizada que captura tanto suas características individuais quanto as dependências com os demais alertas da janela. Ao contrário do codificador de campanha, que agrega os *tokens* por média para representar a sequência como um todo, o codificador de alerta extrai a representação do *token* central após o processamento da atenção, para preservar a identidade do alerta-alvo enquanto a enriquece com o contexto da janela. Esse *embedding* por alerta é gerado de forma que alertas de estágios semelhantes ocupem regiões próximas no espaço latente, independentemente da campanha de que fazem parte.

A entrada do codificador de alerta é formada pela fusão de todos os grupos de atributos processados na etapa anterior, incluindo o vetor textual do BERT, os *embeddings* de octetos de IP, os *embeddings* de portas e demais campos categóricos, e os atributos numéricos e temporais, concatenados em um único vetor por alerta. Esse vetor é projetado por camadas densas aplicadas de forma independente a cada posição da janela, realizando

uma fusão não-linear das características e reduzindo a representação a uma dimensão d fixa. O resultado é o *embedding* base de cada alerta, uma representação que sintetiza todas as suas propriedades antes do processamento contextual.

Antes do processamento pelo *Transformer*, uma codificação posicional é somada a cada *embedding* base, o que permite ao modelo distinguir a posição de cada alerta na janela e preservar a ordem dos eventos. Nos primeiros alertas, quando a janela não pode se estender à esquerda por exceder os limites do conjunto, ela fica fixada no início, de modo que o alerta ocupa as primeiras posições em vez da posição central. Para os últimos alertas, as posições à direita são preenchidas com a replicação do último alerta válido, o que garante uma janela de tamanho L em todos os casos. Os vetores resultantes são processados por um *Transformer* com pré-normalização (*Pre-Norm*), composto por blocos de atenção multi-cabeça que permitem a cada alerta integrar informações de todos os outros alertas da janela. Na pré-normalização, a normalização de camada é aplicada antes de cada bloco de atenção e *feed-forward*, em vez de após a adição residual como no *Transformer* original [11]. Essa escolha estabiliza os gradientes ao longo das camadas, prevenindo divergências numéricas no treinamento. Ao final do processamento, apenas a representação do token central é extraída e normalizada pela norma L_2, de modo a produzir o *embedding* final de dimensão d para o alerta a ser representado.

Como o objetivo do codificador de alerta é organizar os eventos segundo o estágio da cadeia de ataque, e não separar incidentes, o treinamento emprega a perda tripleto apresentada na Equação 2.7 com tripletos definidos em nível de estágio, em vez de em nível de campanha. Essa escolha favorece pares positivos semanticamente coesos, pois alertas do mesmo estágio compartilham padrões de comportamento mais homogêneos entre campanhas distintas do que alertas de uma mesma campanha em estágios diferentes. A âncora e o exemplo positivo são, portanto, alertas pertencentes ao mesmo estágio da cadeia de ataque, enquanto o exemplo negativo faz parte de um estágio diferente. Para garantir tripletos informativos e evitar que o modelo treine com exemplos triviais, utiliza-se a mineração de negativos difíceis, que seleciona, para cada âncora, os exemplos negativos mais próximos no espaço latente atual. O processo é iterativo, de modo que ao final de cada ciclo de treinamento os *embeddings* são atualizados e o processo de mineração é repetido para selecionar novos negativos difíceis, melhorando progressivamente as fronteiras do espaço latente.

Concluído o treinamento, os *embeddings* produzidos pelo codificador de alerta são organizados em sequências com base nos rótulos de campanha gerados pelo codificador de campanha. Os alertas classificados como ruído pelo agrupamento HDBSCAN, identificados pelo rótulo -1, são previamente excluídos, de modo que apenas eventos pertencentes a campanhas identificadas compõem as sequências. Cada sequência resultante reúne os *embeddings* dos alertas de uma mesma campanha em ordem temporal, sendo então fornecida ao preditor multitarefa, conforme descrito na seção seguinte.

4.4 Preditor Multitarefa

O preditor multitarefa é o componente responsável pelo segundo objetivo da abordagem, a antecipação de eventos em uma campanha de ataque em andamento. A partir de uma sequência de alertas de uma mesma campanha, representados pelos *embeddings* do codificador de alerta, o preditor estima simultaneamente o estágio do próximo evento, seus endereços IP de origem e destino e o intervalo temporal Δt até sua ocorrência. A formulação multitarefa permite que as predições sobre estágio e intervalo temporal compartilhem uma representação contextual comum, explorando as dependências entre elas em vez de tratá-las de forma isolada.

A arquitetura do preditor possui duas entradas que alimentam conjuntos distintos de cabeças de saída. A sequência de *embeddings* de alerta é processada por um ramo baseado em *Transformer* com pré-normalização, que modela as dependências contextuais ao longo da sequência e fornece a representação utilizada pelas cabeças de estágio e de intervalo temporal. Paralelamente, a sequência de octetos de IP dos alertas do contexto é fornecida diretamente às cabeças de ponteiro, atuando como o reservatório a partir do qual os endereços de origem e destino são copiados. Dessa forma, cada tarefa recebe a representação mais adequada ao seu objetivo, sem a necessidade de uma fusão das duas entradas.

A cabeça de estágio é um classificador cujo alvo é uma distribuição suavizada sobre múltiplos eventos futuros da sequência. Isso leva o modelo a aprender a topologia de progressão da cadeia de ataque, em vez de memorizar as raras transições de etapa presentes nas sequências. A predição do intervalo temporal Δt, por sua vez, é tratada como uma tarefa de regressão.

A predição de endereços IP baseia-se no mecanismo de redes de ponteiros, fundamentado na Seção 2.7. Em vez de gerar um endereço a partir de um vocabulário fixo, cada cabeça de ponteiro seleciona um endereço já presente na janela de contexto. Para identificar qual posição é mais provável, a cabeça utiliza uma Unidade Recorrente com Portão (*Gated Recurrent Unit*, GRU) para processar a sequência de octetos de IP do contexto e produzir uma estimativa do próximo endereço, $\hat{a}$. A GRU é uma arquitetura recorrente que mantém um estado oculto ao longo da sequência e utiliza mecanismos de porta para controlar quais informações passadas são retidas ou descartadas, sendo adequada para capturar padrões temporais como persistência de IP ou ciclos de rotação entre endereços. A distribuição de atenção sobre as posições da janela é então computada pela distância L_2 negativa entre $\hat{a}$ e cada endereço do contexto, escalada por uma temperatura aprendível T:

$$\alpha_i \propto \exp!\left(-T \cdot |\hat{a} - a_i|^2\right)$$

Assim, posições cujo IP está geometricamente mais próximo da estimativa recebem maior peso de atenção por construção, sem depender de projeções lineares. Na inferência, a seleção torna-se determinística, dada pela posição de maior peso de atenção.

O treinamento do preditor é conduzido de forma conjunta, otimizando uma função de perda combinada que soma as perdas individuais de cada cabeça de saída. Cada termo recebe um peso que reflete sua importância relativa e a escala de sua perda, equilibrando a contribuição das tarefas de classificação, cópia e regressão durante a otimização. Dessa forma, o modelo aprende uma representação contextual compartilhada que serve de base para todas as tarefas de predição.

4.5 Caso Ilustrativo

Para ilustrar o funcionamento da abordagem proposta, considera-se um cenário teórico com nove alertas gerados por dois atacantes simultâneos, cujos eventos chegam de forma intercalada no fluxo. A Tabela 1 apresenta a sequência completa, incluindo os endereços IP de origem e destino, as assinaturas detectadas e as táticas MITRE correspondentes. O alerta e_9, originado de um endereço sem relação com as campanhas e separado dos demais por um longo intervalo temporal, representa um evento ruidoso.

Tabela 1 – Fluxo bruto de alertas do cenário ilustrativo, em ordem cronológica.

ID	Δt	**Origem**	**Destino**	**Assinatura**	**Tática**
e_1	T+0s	185.23.4.100	10.0.0.80	ET SCAN Nmap Scripting Engine	Descoberta
e_2	T+4s	185.23.4.100	10.0.0.80	ET SCAN HTTP Directory Brute Force	Reconhecimento
e_3	T+9s	91.45.22.18	10.0.0.95	ET SCAN SYN Port Sweep	Reconhecimento
e_4	T+13s	185.23.4.100	10.0.0.80	ET WEB_SERVER SQL Injection Attempt	Acesso Inicial
e_5	T+15s	185.23.4.100	10.0.0.80	ET WEB_SERVER PHP Remote File Upload	Persistência
e_6	T+27s	91.45.22.18	10.0.0.95	ET WEB_SERVER Directory Traversal	Acesso Inicial
e_7	T+31s	10.0.0.80	185.23.4.100	ET MALWARE Generic HTTP Beacon	Cmd. e Controle
e_8	T+48s	10.0.0.80	10.0.0.81	ET POLICY SMB Scanner Detected	Mov. Lateral
e_9	T+620s	203.0.113.5	10.0.0.80	ET BRUTE SSH Login Attempt	Força Bruta

A primeira etapa transforma cada alerta em uma representação vetorial combinando características textuais e estruturais, conforme descrito na Seção 4.1. O codificador de campanha processa janelas centradas em cada evento e aplica o agrupamento HDBSCAN sobre os *embeddings* resultantes, identificando automaticamente as fronteiras entre as campanhas. O resultado da separação é apresentado na Tabela 2.

Tabela 2 – Separação do fluxo bruto em campanhas e ruído pelo codificador de campanha.

Grupo	**Eventos**	**IPs característicos**
Campanha A	$e_1, e_2, e_4, e_5, e_7, e_8$	185.23.4.100 $\leftrightarrow$ 10.0.0.80
Campanha B	e_3, e_6	91.45.22.18 $\leftrightarrow$ 10.0.0.95
Ruído ($\ell = -1$)	e_9	203.0.113.5

Os eventos e_1, e_2, e_4, e_5, e_7 e e_8 são agrupados como Campanha A, com base na recorrência dos endereços 185.23.4.100 e 10.0.0.80 e na progressão de táticas. Os eventos e_3 e e_6 formam a Campanha B, pois compartilham o par 91.45.22.18 e 10.0.0.95 com intervalos temporais coerentes entre si. O evento e_9, cujos endereços e intervalo temporal diferem substancialmente dos demais, recebe o rótulo $\ell = -1$ do HDBSCAN e é excluído das sequências fornecidas ao preditor.

O codificador de alerta processa cada evento em sua janela de contexto, extraindo a representação do *token* central enriquecida pela atenção sobre os alertas vizinhos. Para e_7, o vetor de entrada inclui 10.0.0.80 como IP de origem, o mesmo endereço que aparecia como destino em e_4 e e_5, e o mecanismo de atenção, ao processar a janela, correlaciona e_7 com esses alertas anteriores, o que produz um *embedding* contextualizado do estágio de comunicação pós-comprometimento. Os *embeddings* da Campanha A ocupam regiões do espaço latente correspondentes aos seus respectivos estágios, com os alertas de reconhecimento próximos entre si, os de exploração e persistência em uma região adjacente e os alertas de C2 e movimento lateral em regiões progressivamente distintas.

O preditor multitarefa recebe os cinco *embeddings* mais recentes da Campanha A, correspondentes a e_2, e_4, e_5, e_7 e e_8, produzidos pelo codificador de alerta. A Tabela 3 apresenta a janela de entrada do preditor para estimativa do próximo evento da Campanha A. Os endereços de origem e destino presentes nas cinco posições constituem o reservatório de cópia para as cabeças de ponteiro.

Tabela 3 – Janela de entrada do preditor para predição do próximo evento.

Posição	**Alerta**	**Estágio**	**IP de Origem**	**IP de Destino**
1	e_2	Reconhecimento	185.23.4.100	10.0.0.80
2	e_4	Acesso Inicial	185.23.4.100	10.0.0.80
3	e_5	Persistência	185.23.4.100	10.0.0.80
4	e_7	Cmd. e Controle	10.0.0.80	185.23.4.100
5	e_8	Mov. Lateral	10.0.0.80	10.0.0.81

Com base na progressão observada, a cabeça de estágio atribui probabilidade de 0,72 ao estágio de Movimento Lateral, o que reflete a tendência de persistência de estágio ao longo de sequências de ataque. A distribuição suavizada atribui probabilidade de 0,15 a Descoberta e 0,08 a Exfiltração como etapas seguintes plausíveis, que figuram entre as três predições de maior pontuação e totalizam 0,95 da probabilidade total. A Tabela 4 consolida a saída do modelo para o próximo evento da Campanha A.

Tabela 4 – Saída do preditor para o próximo evento da Campanha A.

Cabeça	**Predição**
Estágio	Mov. Lateral ($p = 0{,}72$), Descoberta ($p = 0{,}15$), Exfiltração ($p = 0{,}08$)
IP de origem	10.0.0.80
IP de destino	10.0.0.81
Δt	≈ 18 s

As cabeças de ponteiro selecionam os endereços 10.0.0.80 e 10.0.0.81 como origem e destino mais prováveis, indicando a persistência do servidor comprometido como agente ativo e a continuação do movimento lateral sobre o *host* atingido. O intervalo temporal estimado é coerente com a cadência observada nos alertas de movimento lateral da campanha. Esse cenário ilustra como a abordagem processa um fluxo heterogêneo de alertas intercalados de forma dinâmica, sem depender de regras estáticas predefinidas para cada cenário de ataque, separando campanhas simultâneas a partir dos padrões observados nos próprios dados, enriquecendo as representações com contexto sequencial e antecipando a próxima ação de um adversário em andamento.

4.6 Resumo

Este capítulo apresentou a abordagem desenvolvida para a correlação dinâmica e a predição multitarefa de ataques cibernéticos. O pipeline proposto parte do processamento multimodal dos atributos de cada alerta, combinando características textuais, de endereçamento, categóricas e temporais, para alimentar dois codificadores que operam em paralelo com objetivos distintos. O codificador de campanha identifica as fronteiras entre operações ofensivas simultâneas a partir dos padrões de endereçamento e progressão tática presentes nos próprios dados, sem depender de regras ou assinaturas pré-definidas para cada cenário de ataque, enquanto o codificador de alerta produz *embeddings* contextualizados em nível de evento que alimentam as etapas subsequentes.

O preditor multitarefa recebe as sequências definidas pelo codificador de campanha e estima simultaneamente o próximo estágio, os endereços IP de origem e destino e o intervalo temporal, por meio de cabeças de saída especializadas que incluem um mecanismo de ponteiro para a predição de endereços. O treinamento dos codificadores fundamenta-se no aprendizado contrastivo com perda tripleto e mineração de negativos difíceis. O agrupamento HDBSCAN delimita as campanhas a partir dos espaços métricos aprendidos pelo codificador de campanha. Um caso ilustrativo demonstrou o funcionamento da abordagem sobre um fluxo heterogêneo de alertas de dois atacantes simultâneos, o que evidencia a separação automática e dinâmica de campanhas em andamento, o enriquecimento contextual das representações e a estimativa dos atributos do próximo evento. O próximo capítulo dedica-se à validação experimental, com a apresentação dos dados, configuração e análise dos resultados.

5 Experimentos

Este capítulo apresenta a avaliação experimental da abordagem proposta para correlação dinâmica e predição multitarefa de ataques cibernéticos. Os experimentos foram conduzidos sobre dois conjuntos de dados, sendo o primeiro composto por dados do AIT Alert Dataset (AIT-ADS) [16] e o segundo por alertas sintéticos desenvolvidos para a validação da proposta. As seções seguintes descrevem o ambiente experimental, os conjuntos de dados, as métricas de avaliação, a configuração dos hiperparâmetros, os resultados por componente, os estudos de ablação e uma discussão geral sobre os resultados.

5.1 Ambiente Experimental e Configuração

Os experimentos foram executados em um notebook ASUS Zephyrus G15, equipado com processador AMD Ryzen 9 6900HS (8 núcleos, 16 *threads*), GPU NVIDIA GeForce RTX 3060 Laptop e 40 GB de memória RAM DDR5. O treinamento das redes neurais e a inferência dos *embeddings* BERT foram acelerados pela GPU, com acesso ao ambiente CUDA por meio do subsistema Windows para Linux (WSL2). Para garantir a reprodutibilidade dos resultados, sementes aleatórias fixas foram configuradas em todos os componentes do *pipeline*.

A implementação foi realizada em Python 3.11.13, utilizando um duplo *backend* com o TensorFlow[1] 2.20.0 responsável pelo treinamento dos modelos e pela execução do *pipeline*, e o PyTorch[2] 2.9.0 via HuggingFace Transformers[3] 4.57.1 dedicado à inferência do BERT, por ser o *backend* de referência para modelos pré-treinados dessa biblioteca. Para as etapas de agrupamento não supervisionado e avaliação das métricas de *clustering*, foram utilizados o HDBSCAN[4] 0.8.40 e o Scikit-learn[5] 1.7.2. A manipulação de dados estruturados e as operações vetoriais foram realizadas com Pandas[6] 2.3.3 e NumPy[7] 2.3.3, e a visualização dos resultados com Matplotlib[8] 3.10.6 e Seaborn[9] 0.13.2, sendo o ambiente de desenvolvimento gerenciado pelo JupyterLab[10] 4.4.9.

[1] `https://www.tensorflow.org`
[2] `https://pytorch.org`
[3] `https://huggingface.co/docs/transformers`
[4] `https://hdbscan.readthedocs.io`
[5] `https://scikit-learn.org`
[6] `https://pandas.pydata.org`
[7] `https://numpy.org`
[8] `https://matplotlib.org`
[9] `https://seaborn.pydata.org`
[10] `https://jupyterlab.readthedocs.io`

5.2 Conjuntos de Dados

Esta seção descreve os dois conjuntos de dados utilizados na avaliação experimental. O primeiro, o *AIT Alert Dataset* (AIT-ADS), é um conjunto baseado em simulação de rede empresarial desenvolvido pelo *Austrian Institute of Technology*, utilizado para a avaliação dos codificadores e do preditor em condições próximas a um ambiente de produção. O segundo é um conjunto sintético gerado especificamente para este trabalho, projetado para viabilizar a validação do preditor multitarefa em condições controladas e balanceadas, permitindo separar o impacto do desbalanceamento de classes da capacidade arquitetural da abordagem.

5.2.1 AIT Alert Dataset

Landauer, Skopik e Wurzenberger [16] propuseram o AIT Alert Data Set (AIT-ADS), um conjunto de alertas gerado a partir do AIT Log Data Set versão 2 (AIT-LDSv2), um conjunto de *logs* de rede e de *host* coletados em ambiente virtual que simula uma rede empresarial completa, desenvolvido pelo *Austrian Institute of Technology* para suprir a escassez de dados públicos adequados à análise de ataques de múltiplos estágios. O ambiente de coleta consiste em uma infraestrutura virtual que simula uma rede empresarial completa, composta por uma zona de intranet com servidor de arquivos e servidor de intranet, uma zona desmilitarizada com servidor VPN, servidor de e-mail e armazenamento em nuvem, e uma zona de Internet com servidor DNS e servidores de e-mail externos, todos interconectados por um *firewall*. O tráfego legítimo é gerado por máquinas de estados que simulam colaboradores interagindo com os serviços disponíveis, garantindo um volume realista de atividade de fundo.

A narrativa de ameaças é composta por dois casos de ataque que ocorrem em paralelo às atividades normais. O primeiro consiste em uma intrusão de múltiplos estágios, com varreduras de serviços e *hosts* por meio do Nmap, enumeração de diretórios com o Dirb e varredura da plataforma WordPress com o WPScan, evoluindo para a exploração de um *plugin* WordPress vulnerável para *upload* de *webshell*, quebra de senhas, instalação de *reverse shell* e escalada de privilégios. O segundo foca na exfiltração de dados sensíveis do servidor de arquivos por tunelamento DNS furtivo com a ferramenta DNSteal, cujo padrão temporal, com a exfiltração ativa desde o início da captura e cessando em um momento específico, impõe desafios adicionais à detecção por anomalia.

Para estabelecer o *ground truth* (i.e., o conjunto de rótulos de referência considerados corretos para fins de avaliação) necessário para a avaliação, os alertas são rotulados por estágio do ataque por meio de correspondência temporal, conforme proposto pelos autores do *dataset* [16]. Como o comportamento do atacante é orquestrado por uma máquina de estados determinística, cada etapa da cadeia de ataque possui marcas de tempo precisas de início e fim, e todos os alertas dentro dessa janela recebem o rótulo do

estágio correspondente. É importante destacar que essa estratégia opera em nível de estágio e não em nível de campanha, uma vez que o conjunto não disponibiliza identificadores de incidente que permitam associar alertas de diferentes estágios a um mesmo atacante.

O conjunto disponibiliza oito cenários de execução distintos (*fox, harrison, russell-mitchell, santos, shaw, wardbeck, wheeler* e *wilson*), todos operando sobre a mesma topologia de rede e incluindo os mesmos casos de ataque, mas com variações de intensidade de varredura, perfis de usuário e infraestrutura implantada. Essa diversidade viabiliza uma estratégia de avaliação cruzada entre cenários, obrigando o modelo a aprender padrões comportamentais abstratos em vez de memorizar parâmetros fixos de uma execução específica. Neste trabalho, utilizou-se o cenário *wardbeck* para treinamento, *santos* para validação e *shaw* para teste.

A Tabela 5 apresenta a distribuição de alertas por classe nos três cenários selecionados, demonstrando o desbalanceamento do conjunto. A classe *dirb* domina os três cenários com contagens praticamente idênticas (4.459 a 4.475 alertas), indicando que a intensidade das varreduras de diretório é consistente entre atacantes, enquanto classes críticas como *webshell*, *privilege_escalation* e *reverse_shell* somam menos de 120 amostras no conjunto completo. O símbolo − indica ausência total da classe naquele cenário, como ocorre com *network_scans* em *wardbeck* e *reverse_shell* em *santos*.

Tabela 5 – Distribuição de alertas por classe nos cenários selecionados do AIT-ADS (treino = *wardbeck*, validação = *santos*, teste = *shaw*).

Classe	**Wardbeck**	**Santos**	**Shaw**	**Total**
Estágios de ataque				
dirb	4.459	4.459	4.475	13.393
wpscan	747	3.268	739	4.754
cracking	734	726	1.322	2.782
dnsteal	8	386	58	452
service_scans	53	90	13	156
network_scans	−	82	48	130
webshell	4	4	42	50
privilege_escalation	10	10	26	46
reverse_shell	6	−	16	22
Subtotal ataque	**6.021**	**9.025**	**6.739**	**21.785**
Ruído de fundo				
noise_dovecot	26.337	20.169	11.886	58.392
noise_clamav	1.080	558	380	2.018
Subtotal ruído	**27.417**	**20.727**	**12.266**	**60.410**
Total	**33.438**	**29.752**	**19.005**	**82.195**

Os alertas foram coletados a partir dos arquivos JSON exportados pelo agente Wazuh disponibilizados no AIT-ADS, que consolidam registros de múltiplas fontes de *log*, incluindo capturas de pacotes de rede e registros de níveis variados, como *logs* de auditoria do sistema operacional, *logs* de acesso Apache, DNS e aplicações diversas. Os alertas gerados pelo AMiner, que requer uma fase de treinamento por anomalia específica para cada cenário,

não foram incluídos neste trabalho por ser incompatível com a estratégia de avaliação cruzada entre cenários adotada. No entanto, o conjunto apresenta desbalanceamento severo de classes, conforme evidenciado na Tabela 5, com o ruído de fundo representando entre 65% e 82% dos alertas por cenário e as classes de ataque mais raras totalizando menos de 120 amostras no conjunto completo, resultando em uma razão de desbalanceamento superior a 46.000:1.

O pré-processamento dos alertas, implementado como parte do *pipeline* proposto, ocorre em duas etapas. Na primeira, os oito arquivos JSON exportados pelo agente Wazuh são achatados, campos com alta taxa de ausência são removidos e os campos de IP e porta são consolidados em representações canônicas, com o resultado armazenado em um CSV estruturado por fonte de coleta. Na segunda etapa, os CSVs são concatenados e os campos textuais de descrição da regra, assinatura e categoria do alerta são concatenados em uma única cadeia e codificados pelo modelo BERT [55], que produz um vetor de representação semântica de 768 dimensões por alerta. As características estruturadas utilizadas são os endereços IP de origem e destino decompostos por octeto, as portas de origem e destino, o protocolo de rede, a severidade e o nível do alerta como campos categóricos, as estatísticas de fluxo de pacotes e bytes nas duas direções em escala logarítmica para reduzir a assimetria da distribuição, e os atributos temporais derivados Δt, *time_sin* e *time_cos*.

5.2.2 Dados Sintéticos

O severo desbalanceamento de classes do AIT-ADS limita a avaliação da capacidade arquitetural do preditor, pois classes críticas como *reverse_shell*, *webshell* e *privilege_escalation* fornecem sinal supervisionado insuficiente para o aprendizado de transições raras de *kill-chain*. Embora o desbalanceamento também ocorra em ambientes reais, a ausência de amostras suficientes para certas transições impede distinguir se eventual baixo desempenho decorre de uma limitação da arquitetura ou da escassez de dados de treinamento. Para estabelecer essa distinção, desenvolveram-se dados sintéticos com formato idêntico ao AIT-ADS. Essa equivalência de formato permite que os dados sintéticos percorram as mesmas etapas de transformação, incluindo a extração de *embeddings* BERT, o treinamento do codificador de alerta e, por fim, o treinamento do preditor multitarefa.

O conjunto sintético compreende 400 campanhas de ataque e 213.610 alertas ao total, divididos em 280 campanhas para treinamento, 60 para validação e 60 para teste, com partição realizada em nível de campanha para evitar vazamento de informação entre as partições. A taxonomia de estágios adota 18 classes de *kill-chain*, sendo 10 diretamente mapeadas a partir das classes do AIT-ADS e 8 estágios adicionais inspirados na MITRE ATT&CK Enterprise Matrix [19], cobrindo táticas ausentes nos cenários de captura do conjunto original, conforme detalhado na Tabela 6. As campanhas são geradas a partir de 11 *templates* distintos de *kill-chain*, cada um representando uma estratégia de ataque com fundamentação em incidentes documentados pelo Cyber Kill Chain [18], pela

MITRE ATT&CK e por relatórios de inteligência de ameaças, com sequências estocásticas que introduzem variações estruturais realistas entre campanhas do mesmo template. As 400 campanhas distribuem-se de forma quase uniforme entre os *templates* por meio de atribuição cíclica, conforme a coluna N da Tabela 7, que indica o número de campanhas geradas nativamente a partir de cada *template* antes da sobreposição de modos de IP descrita a seguir.

Tabela 6 – Taxonomia dos 18 estágios de *kill-chain* do conjunto sintético.

Estágio sintético	**Correspondência AIT-ADS**	**Tática MITRE ATT&CK**
Estágios mapeados do AIT-ADS		
`recon_passive`	`network_scans`	Reconnaissance
`recon_active`	`service_scans`	Reconnaissance
`web_discovery`	`dirb`	Reconnaissance
`web_enumeration`	`wpscan`	Reconnaissance
`brute_force`	`cracking`	Credential Access
`exfil_dns`	`dnsteal`	Exfiltration
`persistence_webshell`	`webshell`	Persistence
`priv_escalation`	`privilege_escalation`	Privilege Escalation
`c2_comms`	`reverse_shell`	Command and Control
`noise_background`	`noise_dovecot / noise_clamav`	—
Estágios adicionais		
`initial_access`	—	Initial Access
`lateral_movement`	—	Lateral Movement
`defense_evasion`	—	Defense Evasion
`collection`	—	Collection
`cred_dumping`	—	Credential Access
`data_staging`	—	Collection
`persistence_scheduled`	—	Persistence
`impact`	—	Impact

A geração de endereços IP adota cinco modos de atribuição ajustados por *template*, refletindo diferentes estratégias adversariais. O modo *campaign* representa ataques com IP de origem fixo ao longo de toda a *kill-chain* e o modo *targeted* fixa tanto o IP de origem quanto o de destino, modelando ataques direcionados a uma vítima específica. O modo *distributed* simula *botnets* com *pool* de 5 a 20 IPs provenientes de prefixos /8 distintos e o modo *subnet_distributed* modela *botnets* de IoT onde os nós compartilham um prefixo /24. O modo *rotating*, introduzido especificamente para testar a sensibilidade posicional do mecanismo de atenção do preditor, emprega um *pool* de 2 a 3 IPs de um mesmo prefixo /24 que se alternam em ciclos fixos de 3 a 5 alertas por posição, criando janelas de contexto não triviais nas quais o endereço de origem correto frequentemente não ocupa a posição mais recente da janela. Para introduzir esses modos sem alterar a diversidadee textual das campanhas, adotou-se uma estrutura de sobreposição por faixa de campanha, na qual as campanhas de índice 0 a 199 mantêm o modo de IP nativo de seu *template*, as campanhas de 200 a 299 têm o modo forçado para *subnet_distributed* e as campanhas de 300 a 399 têm o modo forçado para *rotating*, preservando as tuplas

Tabela 7 – Templates de kill-chain utilizados na geração do conjunto sintético.

Template	**Modo IP nativo**	**N**	**Estratégia de ataque**
`web_app_attack`	campaign	37	Intrusão via aplicação web pública com webshell e exfiltração
`network_intrusion`	campaign	37	Varredura de rede com exploração de serviço e movimento lateral
`ransomware`	targeted	37	Exploração inicial com escalada de privilégios e impacto por criptografia
`apt_espionage`	targeted	37	APT com espionagem prolongada, coleta e exfiltração furtiva
`credential_attack`	distributed	36	Força bruta distribuída de credenciais com múltiplos IPs de origem
`dns_exfiltration`	targeted	36	Exfiltração furtiva de dados por tunelamento DNS
`insider_threat`	campaign	36	Ameaça interna com acesso legítimo utilizado para exfiltração
`supply_chain`	targeted	36	Comprometimento via cadeia de suprimento com persistência e impacto
`brute_webshell`	campaign	36	Força bruta de formulários web seguida de implantação de webshell
`recon_exploit_lateral`	campaign	36	Kill chain completo de reconhecimento até impacto com movimento lateral
`iot_botnet`	subnet_distributed	36	Botnet de IoT com nós em mesmo prefixo /24, força bruta e impacto opcional

textuais originais de cada *template* e alterando apenas a semântica de endereçamento. A distribuição final por modo é de aproximadamente 91 campanhas *campaign*, 73 *targeted*, 18 *distributed*, 118 *subnet_distributed* e 100 *rotating*. Para evitar vazamento de informação entre partições, os *pools* de IPs de destino são disjuntos entre treinamento, validação e teste, com 45, 15 e 15 endereços exclusivos, respectivamente. Essa configuração permite avaliar se o preditor aprende padrões estruturais de endereçamento em vez de memorizar IPs observados durante o treinamento.

Em contraste com o AIT-ADS, os dados sintéticos mantêm uma razão máxima de desbalanceamento de 10,8:1 entre as classes mais e menos frequentes, reduzindo em mais de três ordens de grandeza a disparidade presente no conjunto. Adicionalmente, cada alerta recebe seu rótulo de estágio diretamente no momento da geração, eliminando as fontes de imprecisão da rotulagem temporal por janelas do AIT-ADS, como sobreposição de estágios e contaminação por tráfego legítimo. Diferentemente do AIT-ADS, o conjunto também disponibiliza identificadores de campanha para cada alerta, viabilizando a avaliação separada da qualidade de agrupamento por estágio e por incidente. A combinação de classes balanceadas, rótulos precisos e em dois níveis de granularidade permite uma avaliação controlada a fim de isolar o efeito do desbalanceamento ao comparar os resultados entre os dois conjuntos.

5.3 Métricas de Avaliação

Diferentemente da classificação supervisionada, a avaliação da correlação de alertas baseada em agrupamento requer métricas que analisem a qualidade estrutural dos grupos formados, verificando se o modelo foi capaz de aproximar corretamente eventos com relacionamentos semânticos reais. Para este fim, foram selecionadas seis métricas, divididas entre avaliações externas, dependentes dos rótulos, e internas, baseadas na geometria dos dados. As métricas de agrupamento são aplicadas tanto ao codificador de alerta, avaliado em relação aos rótulos de estágio, quanto ao codificador de campanha, avaliado em relação aos identificadores de campanha.

Para mensurar a pureza e a integridade dos grupos, utilizaram-se a Homogeneidade, a Completude e a V-Measure. A Homogeneidade avalia se cada *cluster* contém apenas membros de uma única classe, e a Completude verifica se todos os membros de uma determinada classe foram atribuídos ao mesmo *cluster*, sendo definidas como:

$$h = 1 - \frac{H(C \mid K)}{H(C)}, \qquad c = 1 - \frac{H(K \mid C)}{H(K)}, \tag{5.1}$$

em que $H(C \mid K)$ é a entropia condicional das classes verdadeiras dado o agrupamento obtido, $H(C)$ é a entropia das classes verdadeiras, $H(K \mid C)$ é a entropia condicional dos *clusters* dado o rótulo verdadeiro, e $H(K)$ é a entropia dos *clusters*. A V-Measure consiste na média harmônica entre as duas e fornece uma pontuação equilibrada sobre a qualidade da partição, dada por

$$V = \frac{2 \cdot h \cdot c}{h + c}. \tag{5.2}$$

Para mitigar o viés de agrupamentos aleatórios, empregaram-se ainda o Índice de Rand Ajustado (ARI) e a Informação Mútua Ajustada (AMI), ambos ajustados para o acaso e particularmente robustos em conjuntos desbalanceados. O ARI compara as atribuições par a par entre o agrupamento obtido e os rótulos verdadeiros, corrigindo pelo acordo esperado ao acaso, sendo dado por

$$\mathrm{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] \Big/ \binom{n}{2}}, \tag{5.3}$$

em que n_{ij} é o número de elementos pertencentes simultaneamente à classe verdadeira i e ao *cluster* j, a_i e b_j são as somas marginais da tabela de contingência, e n é o número total de elementos. A AMI, por sua vez, ajusta a informação mútua entre os agrupamentos pelo seu valor esperado sob permutações aleatórias, sendo dada por

$$\mathrm{AMI} = \frac{\mathrm{MI}(C, K) - \mathbb{E}[\mathrm{MI}(C, K)]}{\frac{1}{2}\left[H(C) + H(K)\right] - \mathbb{E}[\mathrm{MI}(C, K)]}, \tag{5.4}$$

em que $\mathrm{MI}(C, K)$ é a informação mútua entre as classes verdadeiras e os *clusters*, e $\mathbb{E}[\mathrm{MI}(C, K)]$ é o seu valor esperado sob o modelo de permutação aleatória.

Para validação interna, o Coeficiente de Silhueta avalia a coesão intra-*cluster* e a separação inter-*cluster*, sendo definido para cada ponto i como

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \tag{5.5}$$

em que $a(i)$ é a distância média entre o ponto i e os demais pontos do mesmo *cluster*, e $b(i)$ é a menor distância média entre o ponto i e os pontos de qualquer outro *cluster*. O coeficiente varia de -1 a 1, onde valores mais altos indicam grupos bem definidos e geometricamente separados.

Adicionalmente, a redução de alertas quantifica a compressão obtida ao substituir os alertas individuais pelos *clusters* encontrados, sendo definida como

$$\text{Redução} = \left(1 - \frac{N_{\text{clusters}}}{N_{\text{alertas}}}\right) \times 100\%, \tag{5.6}$$

em que N_{alertas} é o número total de alertas e N_{clusters} é o número de *clusters* obtidos.

A avaliação do preditor multitarefa requer métricas distintas para cada cabeça de saída. Para a classificação do próximo estágio, utiliza-se o Macro F1, que pondera igualmente todas as classes independentemente de sua frequência, definido como

$$\text{Macro F1} = \frac{1}{N} \sum_{c=1}^{N} \frac{2 \cdot P_c \cdot R_c}{P_c + R_c}, \tag{5.7}$$

em que P_c e R_c são, respectivamente, a precisão e a revocação para a classe c, e N é o número total de classes. Complementarmente, utiliza-se a acurácia Top-1, que mede a fração dos alertas em que o estágio de maior probabilidade predito corresponde ao rótulo alvo, e a acurácia Top-3, que mede a fração em que o rótulo alvo figura entre os três estágios de maior probabilidade predita. A acurácia Top-3 é particularmente relevante porque campanhas de ataque podem evoluir para múltiplos estágios a partir de um mesmo ponto da *kill-chain*, e o modelo é treinado com distribuições de rótulo suavizadas que refletem essa possibilidade.

Para a predição de endereços IP, utilizam-se três medidas complementares. A acurácia /32 verifica a correspondência exata do endereço, enquanto a acurácia /24 verifica a correspondência do prefixo de sub-rede. Por fim, a cobertura representa a fração dos alertas para os quais o IP de origem a ser predito já aparece na janela de contexto de K eventos, sendo definida como

$$\text{Cobertura} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left[\mathrm{IP}_i \in \{\mathrm{IP}_{i-K}, \ldots, \mathrm{IP}_{i-1}\}\right], \tag{5.8}$$

em que $\mathbb{1}[\cdot]$ é a função indicadora, igual a 1 quando o endereço IP de origem do evento i está presente na janela de contexto dos K eventos anteriores, e 0 caso contrário. É essa

métrica que delimita o subconjunto em que a predição exata pelo mecanismo de ponteiro é possível, dado que ele só pode selecionar endereços previamente observados na janela.

Para o intervalo temporal, utiliza-se o erro absoluto médio (MAE) em segundos, sendo definido como

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \,, \tag{5.9}$$

em que y_i é o intervalo temporal observado até o próximo evento e $\hat{y}_i$ é o valor predito pelo modelo. Complementarmente, um *baseline* de cópia ingênua (*naive-copy*) prediz os próximos endereços IP de origem e de destino, bem como o próximo intervalo temporal, reproduzindo o valor mais recente da janela de contexto, sem qualquer capacidade preditiva aprendida, servindo como referência para quantificar o ganho efetivo do modelo sobre a simples persistência do último evento observado.

5.4 Implementação e Hiperparâmetros

A implementação dos componentes foi realizada por meio da API funcional do Keras integrada ao TensorFlow, com os codificadores e o preditor treinados de forma independente e sequencial, de modo que cada componente se especialize em seu respectivo objetivo antes de os *embeddings* serem consumidos pelo estágio seguinte. Os blocos Transformer dos codificadores adotam o esquema de normalização Pré-Norma, no qual a *LayerNorm* é aplicada à entrada de cada sub-camada antes das operações de atenção e de alimentação direta, invertendo a ordem do Transformer original [11], que normaliza após a soma residual. Essa estabilidade numérica tem impacto direto na qualidade das representações aprendidas, pois instabilidades no fluxo de gradientes durante o treinamento contrastivo tendem a corromper o espaço latente e a impedir a convergência da *Triplet Loss*, anulando a capacidade do codificador de organizar os *embeddings* por estágio. O treinamento foi conduzido em precisão simples (*float32*), na qual cada valor de ponto flutuante é representado com 32 bits, em vez da precisão mista, que alterna entre representações de 16 e 32 bits para ganho de eficiência computacional mas com intervalo dinâmico reduzido que aumenta o risco de *overflow* ou *underflow* em operações de distância como a *Triplet Loss*. O preditor multitarefa adota o esquema Pós-Norma em seus blocos Transformer, onde a normalização é aplicada após a soma residual, configuração mais comum em tarefas supervisionadas de classificação e regressão simultâneas.

O pré-processamento textual utiliza o modelo `bert-base-uncased` disponibilizado pela biblioteca HuggingFace Transformers[11], com os pesos mantidos fixos durante todo o treinamento para aproveitar as representações semânticas pré-treinadas sem o custo computacional do ajuste fino. Cada alerta possui quatro atributos textuais que descrevem o

[11] `https://huggingface.co/google-bert/bert-base-uncased`

evento de segurança detectado, sendo a assinatura (`signature`) o identificador da regra de detecção disparada, a categoria (`category`) a classificação do tipo de ameaça, a descrição (`description`) o detalhamento do comportamento observado e a tática MITRE (`tactic`) o mapeamento do evento para uma fase da kill-chain. Esses atributos são concatenados como um texto estruturado no formato `SIG: {signature} | CAT: {category} | DESC: {description} | TAC: {tactic}` e submetidos ao BERT, que produz para cada alerta um vetor de 768 dimensões extraído do *token* `[CLS]`, um marcador especial inserido no início da sequência cujo estado oculto final agrega a representação semântica global do texto.

O codificador de alerta é implementado como uma rede siamesa cujas ramificações compartilham pesos e processam âncora, positivo e negativo com a mesma sequência de transformações. Cada ramificação recebe a fusão multimodal de 904 dimensões e aplica, a cada alerta da janela de contexto, duas camadas densas de projeção ($Dense(512) \rightarrow Dense(256)$) seguidas de codificação posicional aprendível. A sequência resultante passa por um bloco Transformer composto por uma sub-camada de atenção multi-cabeça e uma sub-camada de alimentação direta, cada uma precedida por *LayerNorm* e seguida por conexão residual. O *embedding* de saída é extraído do *token* central da janela e normalizado por L_2, produzindo um vetor de 256 dimensões na esfera unitária como representação contextualizada do alerta, e o treinamento utiliza *Triplet Loss* com amostragem de tripletos por rótulo de estágio a cada lote, conforme os hiperparâmetros da Tabela 8.

Tabela 8 – Hiperparâmetros do Codificador de Alerta.

Hiperparâmetro	**Valor**
Arquitetura	
Janela de contexto	16 alertas
Dimensão do modelo (d_{model})	256
Cabeçotes de atenção	4
Dimensão *feed-forward* (d_{ff})	128
Dropout	0,1
Embedding de saída	256-dim (L_2-normalizado)
Treinamento	
Batch size	1024
Taxa de aprendizado	5×10^{-6}
Otimizador	Adam ($clipnorm = 1{,}0$)
Épocas	12
Função de custo	*Triplet Loss*
Margem (α)	1,0

O codificador de campanha adota uma representação de entrada mais compacta em contraste com a fusão multimodal de 904 dimensões do codificador de alerta. Para cada alerta da janela de W eventos consecutivos da mesma campanha, a entrada é formada pela concatenação da projeção do vetor BERT de 768 dimensões para 64 dimensões por meio de uma camada densa sem viés, dos oito octetos de IP de origem e destino normalizados para o intervalo $[0, 1]$ e do intervalo temporal Δt em escala logarítmica, totalizando 73 dimensões.

Essa representação compacta concentra os atributos que identificam a consistência de um incidente ao longo do tempo, conforme descrito na Seção 4.2. A sequência de 73 dimensões é projetada por uma camada densa para $d_{model} = 128$, somada à codificação posicional aprendível e processada por dois blocos Transformer empilhados, cada um composto por uma sub-camada de atenção multi-cabeça e uma sub-camada de alimentação direta, cada uma precedida por *LayerNorm* e seguida por conexão residual. Ao contrário do codificador de alerta, que extrai o *token* central para contextualizar um alerta específico, o codificador de campanha aplica *pooling* médio sobre todos os W *tokens* da sequência para agregar a representação da campanha como um todo, normalizando o resultado por L_2 para produzir um vetor de 128 dimensões. O treinamento utiliza *Triplet Loss* com tripletos formados por janelas gerados dinamicamente a cada lote, em que âncora e positivo são duas janelas de W alertas consecutivos extraídas da mesma campanha e o negativo é uma janela de campanha distinta. Isso contrasta com o codificador de alerta, cujos tripletos são formados por alertas individuais agrupados por estágio em vez de por campanha, e para os quais se emprega mineração de negativos difíceis em vez de amostragem dinâmica. Os hiperparâmetros do codificador de campanha são apresentados na Tabela 9.

Tabela 9 – Hiperparâmetros do Codificador de Campanha.

Hiperparâmetro	**Valor**
Arquitetura	
Dimensão de entrada	73
Comprimento de sequência	20 alertas
Dimensão do modelo (d_{model})	128
Cabeçotes de atenção	4
Dimensão *feed-forward* (d_{ff})	256
Camadas Transformer	2
Embedding de saída	128-dim (L_2-normalizado)
Treinamento	
Batch size	512
Taxa de aprendizado	5×10^{-4}
Otimizador	Adam (*clipnorm* = 1,0)
Épocas	15
Tripletos por época	20.000
Margem (α)	1,0

O HDBSCAN é empregado em três contextos com configurações e objetivos distintos. Sobre o AIT-ADS, o agrupamento é aplicado aos *embeddings* de 256 dimensões do codificador de alerta, com parâmetros otimizados por busca em grade na partição de validação maximizando a V-Measure, e desempenha duplo papel. Os rótulos de *cluster* produzidos delimitam as janelas de contexto do preditor, identificando incidentes inferidos sem utilizar os rótulos do *dataset*, e o agrupamento é simultaneamente avaliado em relação a esses rótulos para mensurar a qualidade dos *embeddings* aprendidos. No pipeline sintético, esses dois papéis ficam separados entre os codificadores. Sobre os *embeddings* do

codificador de campanha (128 dimensões), o agrupamento desempenha papel funcional análogo ao do AIT-ADS, com os rótulos de *cluster* delimitando janelas de contexto do preditor por incidente, sendo também avaliado em relação aos rótulos de campanha do conjunto sintético. Sobre os *embeddings* do codificador de alerta no pipeline sintético, os parâmetros do HDBSCAN são fixos e não otimizados, pois o objetivo é apenas mensurar a separabilidade dos *embeddings* por estágio, sem papel funcional no preditor. Nesse contexto, os *embeddings* são pré-normalizados com *StandardScaler* antes do agrupamento, diferentemente do contexto do AIT-ADS. A Tabela 10 detalha os parâmetros adotados em cada contexto.

Tabela 10 – Configurações do HDBSCAN por contexto de aplicação.

Parâmetro	**AIT-ADS**	**Sint. — Cod. Campanha**	**Sint. — Cod. Alerta**
min_cluster_size	5	200	10
min_samples	5	5	5
cluster_selection_epsilon	0,1	0,0	0,0
Pré-normalização	—	*StandardScaler*	*StandardScaler*
Otimização	*Grid Search*	*Grid Search*	Fixo

O preditor multitarefa opera sobre dois fluxos de entrada distintos, sendo o primeiro uma janela de K *embeddings* de 256 dimensões do codificador de alerta restritos ao mesmo incidente inferido pelo agrupamento do codificador de campanha, e o segundo os oito octetos de IP de origem e destino dos K alertas da janela de contexto. Os *embeddings* são processados por dois blocos Transformer empilhados com atenção multi-cabeça de quatro cabeçotes, alimentando uma cabeça de classificação para o próximo estágio da kill-chain e uma cabeça de regressão para o intervalo temporal Δt. Os octetos de IP alimentam duas cabeças de ponteiro independentes para predição do endereço de destino e do endereço de origem. Em cada cabeça, uma GRU processa a sequência de octetos do contexto e produz uma estimativa do próximo endereço $\hat{a}$. A distribuição de atenção sobre as posições da janela é então computada pela distância L_2 negativa entre $\hat{a}$ e cada endereço do contexto, escalada por uma temperatura aprendível T, conforme descrito na Seção 4.4. A classificação de estágio utiliza distribuições de rótulo suavizadas sobre os $K_{\text{future}} = 10$ próximos eventos da kill-chain, com coeficiente de suavização de 0,1, refletindo que uma mesma posição na sequência pode preceder múltiplos estágios válidos dependendo do *template* de ataque. Os pesos da função de custo são 4,0 para o estágio, 1,5 para cada componente de predição de IP e 0,3 para o intervalo temporal, priorizando a classificação de estágio, que requer maior discriminação semântica entre as classes. O critério de parada antecipada monitora o Macro F1 na partição de validação em vez da perda de validação, por refletir mais diretamente a qualidade da predição de estágio em conjuntos desbalanceados, com *patience* de 8 épocas. A Tabela 11 apresenta os demais hiperparâmetros adotados.

Tabela 11 – Hiperparâmetros do Preditor Multitarefa.

Hiperparâmetro	**Valor**
Arquitetura	
Janela de contexto (K)	3 alertas
Dimensão do modelo	256
Cabeçotes de atenção	4
Dimensão por cabeçote (*key_dim*)	64
Camadas Transformer	2
Estágios futuros (K_{future})	10
Suavização de rótulos	0,1
Dropout	0,1
Dimensão oculta da GRU (cabeça de IP)	32
Treinamento	
Batch size	64
Épocas	60 (com *early stopping*)
Patience (Macro F1)	8
Otimizador	Adam ($lr = 5 \times 10^{-4}$, *clipnorm* = 1,0)
Pesos da função de custo	
Estágio	4,0
IP de destino / IP de origem	1,5 / 1,5
Intervalo temporal (Δt)	0,3

5.5 Avaliação dos Codificadores

Esta seção avalia a qualidade dos *embeddings* produzidos pelos dois codificadores por meio das métricas de agrupamento definidas na Seção 5.3. O codificador de alerta é avaliado pela separabilidade dos *embeddings* em relação aos rótulos de estágio nos dois conjuntos de dados, enquanto o codificador de campanha é avaliado pela qualidade do agrupamento por incidente no conjunto sintético. Os resultados fundamentam as escolhas arquiteturais da abordagem e motivam a separação das duas tarefas em componentes independentes.

5.5.1 Codificador de Alerta

A avaliação sobre os dados sintéticos compara duas variantes de treinamento, distinguindo entre tripletos amostrados por rótulo de estágio e tripletos amostrados por rótulo de campanha, com métricas computadas separadamente em relação a ambos os rótulos. Essa comparação permite verificar se o objetivo de treinamento determina a semântica das representações aprendidas e em que medida o sinal de supervisão utilizado influencia a organização do espaço latente. A Tabela 12 apresenta os resultados obtidos em cada variante.

As duas variantes obtiveram desempenho semelhante em ambas as dimensões de avaliação, com a variante por campanha apresentando vantagem marginal tanto em V-Measure de estágio (0,5445 contra 0,5368) quanto em V-Measure de campanha (0,2383

Tabela 12 – Avaliação do Codificador de Alerta nos dados sintéticos.

Métrica	**Tripletos por estágio**	**Tripletos por campanha**
H / C / V (estágio)	0,4866 / 0,5987 / 0,5368	0,4942 / 0,6060 / 0,5445
ARI (estágio)	0,0572	0,0601
AMI (estágio)	0,4206	0,4264
H / C / V (campanha)	0,2519 / 0,2106 / 0,2294	0,2622 / 0,2185 / 0,2383
ARI (campanha)	0,0179	0,0191
AMI (campanha)	0,1807	0,1875
Silhueta	0,1391	0,1639
Clusters	288	306
Ruído	18,2%	19,7%
Redução de alertas	99,12%	99,06%

contra 0,2294). Essa proximidade entre as variantes indica que o objetivo de treinamento tem influência limitada sobre a organização global do espaço latente quando os atributos disponíveis em nível de alerta não são suficientes para discriminar incidentes simultâneos. O principal resultado, comum a ambas as variantes, é a V-Measure de campanha inferior a 0,25, evidenciando que o codificador de alerta não consegue separar incidentes simultâneos com base nos atributos de alerta, independentemente do objetivo de treinamento, o que motiva diretamente a introdução do codificador de campanha.

A avaliação sobre o AIT-ADS seguiu o protocolo de avaliação cruzada descrito na Seção 5.2, com o modelo treinado no cenário *wardbeck*, validado em *santos* e avaliado em *shaw*. Os *embeddings* foram agrupados pelo HDBSCAN com os parâmetros da Tabela 10 e os rótulos de *cluster* comparados aos rótulos de estágio do *dataset*. A Tabela 13 apresenta os resultados obtidos.

Tabela 13 – Avaliação do Codificador de Alerta sobre o AIT-ADS (teste: *shaw*).

Métrica	**Valor**
Homogeneidade	0,8749
Completude	0,7062
V-Measure	0,7815
ARI	0,9187
AMI	0,7802
Silhouette	0,0544
Clusters formados	44
Ruído residual	1,87%
Redução de alertas	99,77%

A V-Measure de 0,7815 e o ARI de 0,9187 indicam forte concordância entre os agrupamentos inferidos e os rótulos reais de estágio, demonstrando que os *embeddings* aprendidos generalizam entre cenários de rede distintos. A redução de 99,77% no volume de alertas, consolidando os eventos brutos em 44 incidentes agrupados, demonstra a relevância operacional da abordagem em ambientes com alto volume de alertas. O Coeficiente de Silhueta

de 0,0544, embora baixo em termos absolutos, situa-se dentro da faixa esperada para *embeddings* em espaços de alta dimensionalidade, onde o fenômeno de concentração de normas faz com que as distâncias entre pontos convirjam para valores similares, reduzindo o contraste entre pares intra e inter-*cluster* e tornando a métrica menos discriminativa do que em espaços de baixa dimensão [59].

Durante o treinamento do codificador de alerta sobre os dados sintéticos, foram identificadas três causas-raiz independentes de valores inválidos (*NaN*) na função de perda, cada uma corrigida individualmente. A primeira foi a presença de campos textuais vazios em tuplas de alerta não populados, que produzia *embeddings* BERT degenerados com baixa variância semântica, corrigida pelo preenchimento completo das 118 tuplas distintas do conjunto sintético. A segunda foi o cálculo de distâncias tripleto sobre *embeddings* não normalizados, gerando magnitudes instáveis propagadas durante a retropropagação, corrigida pela normalização L_2 dos *embeddings* antes do cálculo de distância. A terceira foi o uso do esquema Pós-Norma no bloco Transformer, que produzia *NaN* já no segundo lote de treinamento devido à acumulação de instabilidades numéricas, corrigida pela adoção do esquema Pré-Norma, motivando a escolha arquitetural descrita na Seção 5.4.

5.5.2 Codificador de Campanha

A avaliação do codificador de campanha verifica se uma arquitetura dedicada ao processamento sequencial de janelas supera as variantes do codificador de alerta na tarefa de separação de incidentes simultâneos, que demonstraram V-Measure de campanha inferior a 0,25 na seção anterior. O comparativo inclui três variantes avaliadas sobre o conjunto sintético de teste com os mesmos rótulos de campanha como referência. As duas primeiras são o codificador de alerta treinado com tripletos por estágio e o codificador de alerta treinado com tripletos por campanha, ambos operando sobre alertas individuais como unidade de representação. A terceira é o codificador de campanha, que processa janelas de W alertas consecutivos como unidade de representação. A Tabela 14 apresenta os resultados.

Tabela 14 – Comparativo de agrupamento por campanha entre variantes do codificador de alerta e o codificador de campanha.

Métrica	**Alerta — estágio**	**Alerta — campanha**	**Cod. de campanha**
V (campanha)	0,2294	0,2383	**0,9968**
H (campanha)	0,2519	0,2622	**1,0000**
C (campanha)	0,2106	0,2185	**0,9936**
ARI (campanha)	0,0179	0,0191	**0,9893**
AMI (campanha)	0,1807	0,1875	**0,9967**
V (estágio)	0,5368	0,5445	0,2408
Clusters	288	306	62
Ruído	18,2%	19,7%	0,03%
Silhueta	0,1391	0,1639	0,4826
Redução de alertas	99,12%	99,06%	99,81%

O codificador de campanha obteve V-Measure de campanha de 0,9968, Homogeneidade de 1,0000, Completude de 0,9936, ARI de 0,9893 e AMI de 0,9967 no *split* de teste, produzindo 62 *clusters* para as 60 campanhas de teste com apenas 0,03% de ruído residual (i.e., 11 alertas não atribuídos). A Homogeneidade perfeita indica que cada *cluster* contém alertas de uma única campanha, enquanto a Completude de 0,9936 indica que 99,36% dos alertas de cada campanha caem no mesmo *cluster*. Os 2 *clusters* excedentes resultam da divisão das campanhas 76 e 373 em dois *clusters* cada, consequência de descontinuidades temporais internas a essas campanhas que o HDBSCAN interpreta como regiões de densidade distintas. Esse resultado representa um salto de mais de 75 pontos percentuais em V-Measure de campanha em relação à melhor variante do codificador de alerta (0,2383), com o número de *clusters* convergindo para a granularidade de campanha e a redução de alertas atingindo 99,81%, confirmando que a separação de incidentes simultâneos requer o processamento sequencial de janelas de campanha. A V-Measure de estágio do codificador de campanha (0,2408) é inferior à das variantes do codificador de alerta (0,54), resultado esperado dado que o modelo foi treinado com tripletos por incidente, de modo que os *embeddings* agrupam alertas pelo incidente ao qual pertencem, independentemente do estágio. A granularidade dos *clusters* de campanha faz com que cada *cluster* atravesse múltiplas combinações de estágio, o que é consistente com a V-Measure de estágio reduzida, confirmando a especialização complementar dos dois codificadores e justificando sua operação em paralelo na abordagem proposta.

5.6 Avaliação do Preditor Multitarefa

Esta seção apresenta os resultados do preditor multitarefa sobre os dois conjuntos de dados. Sobre os dados sintéticos, avaliou-se o impacto da integração do codificador de campanha, a influência do tamanho da janela de contexto, o desempenho por tipo de transição de estágio e a cobertura de predição de IP por modo de endereçamento. Sobre o AIT-ADS, avaliou-se o desempenho em condição de avaliação cruzada entre cenários, com foco no impacto do desbalanceamento de classes sobre as métricas de classificação.

5.6.1 Dados Sintéticos

O primeiro conjunto de experimentos comparou três configurações do preditor, variando a origem da delimitação de campanha utilizada para restringir a janela de contexto. A configuração oráculo utilizou diretamente o *campaign_id* verdadeiro de cada alerta para delimitar a janela, sem depender do codificador de campanha, servindo como limite superior teórico para as demais configurações. As configurações com codificador de campanha restringiram a janela aos alertas do *cluster* previsto pelo HDBSCAN, com *min_cluster_size* (MCS) igual a 200 (configuração final) e a 15, parâmetro descrito na Seção 5.4. Um *baseline* de cópia ingênua (*naive-copy*) foi incluído como referência para as métricas de IP e de

intervalo temporal, conforme descrito na Seção 5.3. O tamanho da janela de contexto (K) foi mantido fixo em 3 alertas durante estes experimentos, valor adotado como configuração final e discutido em detalhe na análise de sensibilidade apresentada a seguir. A Tabela 15 apresenta os resultados.

Tabela 15 – Comparativo de configurações do preditor nos dados sintéticos ($K = 3$).

Configuração	**F1**	**Top-1**	**Top-3**	**Dest /32**	**Src /32 geral**	**Src /32 (janela)**	**Δt MAE**
Oráculo (*campaign_id*)	0,8940	91,14%	98,51%	99,97%	65,70%	91,89%	4,35s
Cod. campanha MCS=200	0,8937	91,09%	98,45%	99,97%	65,71%	91,92%	4,26s
Cod. campanha MCS=15	0,9000	91,98%	98,60%	99,98%	65,13%	91,50%	4,20s
Baseline de referência							
Naive-copy	—	—	—	99,97%	66,56%	93,09%	10,42s

A configuração com codificador de campanha MCS=200 produziu Macro F1 de 0,8937, praticamente idêntico ao da configuração oráculo (0,8940), com Src /32 geral equivalente entre as duas configurações (65,71% e 65,70%, respectivamente). Essa equivalência indica que os *clusters* previstos pelo codificador de campanha são estatisticamente indistinguíveis do *campaign_id* verdadeiro ao nível do preditor, consistente com a elevada qualidade de agrupamento reportada na Seção 5.5.2 (V-Measure de 0,9968). A configuração MCS=15 produziu o maior Macro F1 (0,9000), porém esse resultado permanece inflacionado pela filtragem de aproximadamente 12% dos alertas de fronteira como ruído pelo HDBSCAN, que reduziu o conjunto de treino de 147.178 para 121.505 sequências. O *baseline* de cópia ingênua atingiu Dest /32 de 99,97%, evidenciando a forte persistência do IP de destino em sequências de ataque, e Δt MAE de 10,42s, bem superior aos cerca de 4,3s do preditor, confirmando que o modelo aprendeu padrões temporais além da simples cópia do último intervalo observado na janela, que o *baseline* ingênuo emprega. A acurácia Src /32 dentro da janela do *baseline* ingênuo (93,09%) superou ligeiramente a do preditor (91,89% a 91,92%), resultado dominado pelos modos de endereçamento *campaign* e *targeted*, nos quais o endereço de origem permanece constante ao longo de toda a campanha e a cópia do último valor observado é trivialmente correta para ambas as abordagens. Esses dois modos respondem pela maior parte dos alertas dentro da janela, de modo que a métrica agregada de Src /32 esconde o desempenho real do mecanismo de ponteiro nos casos não triviais, detalhado por modo de endereçamento na Tabela 18.

O segundo experimento avaliou o impacto do tamanho da janela de contexto K sobre o preditor multitarefa, variando entre 3 e 15 alertas no conjunto sintético descrito na Seção 5.2.2. Além da cobertura de IP de origem, definida na Seção 5.3, foram avaliadas três medidas adicionais específicas desta varredura. A primeira foi a acurácia Src /32 restrita às campanhas do modo *rotating* dentro da janela. A segunda foi o peso médio de atenção máxima (MMA) atribuído pela cabeça de ponteiro de origem, que indica o quanto o modelo concentrou a atenção em uma posição específica da janela em vez de distribuí-la uniformemente. A terceira foi a temperatura T_{src} aprendida pelo mecanismo de atenção por distância L_2 do ponteiro de origem. A Tabela 16 apresenta os resultados.

Tabela 16 – Varredura da janela de contexto K no preditor multitarefa nos dados sintéticos.

Métrica	$K = 3$	$K = 5$	$K = 7$	$K = 10$	$K = 15$
F1	0,8940	0,8921	0,8865	0,8839	0,8840
Top-1	91,14%	90,96%	90,45%	90,28%	90,49%
Top-3	98,51%	98,51%	98,37%	98,49%	98,33%
Dest /32	99,97%	99,97%	99,97%	99,97%	99,97%
Src /32 (janela)	91,89%	80,22%	76,04%	75,81%	68,66%
Rotating /32 (janela)	94,83%	66,38%	64,42%	76,25%	62,21%
Src cobertura	71,50%	77,46%	83,28%	87,47%	92,65%
MMA Src	0,411	0,288	0,207	0,133	0,085
T_{src}	8,34	8,55	8,29	6,06	5,41
Δt MAE	4,35s	4,21s	4,12s	4,43s	4,70s

O Macro F1 variou pouco com o tamanho da janela, situando-se entre 0,8839 e 0,8940 em todas as configurações testadas, com pico em $K = 3$. Esse comportamento estável contrastou fortemente com o da acurácia Src /32 dentro da janela, que caiu de 91,89% em $K = 3$ para 68,66% em $K = 15$, e com a acurácia restrita às campanhas do modo *rotating* dentro da janela, que caiu de 94,83% em $K = 3$ para um mínimo de 62,21% em $K = 15$. A cobertura de IP de origem seguiu a tendência oposta, crescendo de 71,50% em $K = 3$ para 92,65% em $K = 15$, o que indica que janelas maiores aumentam a probabilidade de o IP-alvo estar presente entre os eventos disponíveis, ao custo de introduzir posições concorrentes que dificultam a seleção correta pelo ponteiro.

O MMA do ponteiro de origem decresceu monotonicamente de 0,411 em $K = 3$ para 0,085 em $K = 15$, acompanhando a queda da temperatura aprendida T_{src}, de 8,34 para 5,41 no mesmo intervalo. Em todas as configurações da Tabela 16, o MMA observado superou o valor de atenção uniforme correspondente (i.e., $1/K$), indicando que o mecanismo de atenção por distância L_2 com temperatura aprendida, descrito na Seção 4.4, concentrou ativamente a atenção em posições específicas da janela em vez de distribuí-la de forma indiscriminada. Esse comportamento é consistente com a expectativa de que janelas maiores dificultam a identificação de uma única posição relevante, reduzindo a concentração da atenção à medida que mais candidatos competem pela mesma seleção.

A varredura indicou $K = 3$ como a configuração recomendada, valor adotado como configuração final do preditor multitarefa. Em $K = 5$, a acurácia Src /32 dentro da janela caiu 11,67 pontos percentuais e a acurácia Rotating /32 caiu 28,45 pontos percentuais em relação a $K = 3$, em troca de um ganho de apenas 5,96 pontos percentuais em cobertura, uma troca desfavorável diante da diferença de F1 entre as duas configurações, de apenas 0,19 ponto percentual. A causa estrutural está associada ao ciclo das campanhas do modo *rotating*, que alternam entre 2 e 3 endereços IP a cada 3 a 5 alertas, conforme descrito na Seção 5.2.2. Em $K = 3$, a janela capturou exatamente um ciclo completo de rotação, de modo que a posição mais recente da janela correspondeu, na maioria dos casos, ao próximo endereço de origem correto, ao passo que em $K = 5$ posições adicionais

introduziram candidatos concorrentes incorretos que degradaram a seleção pelo ponteiro. O ganho aparente de Rotating /32 em $K = 10$ (76,25%) em relação a $K = 7$ (64,42%) é consistente com variação estocástica entre execuções, dado que não acompanha uma tendência monotônica nas demais métricas.

A Tabela 17 expande a avaliação de desempenho do preditor por tipo de transição de estágio, computada para a configuração final ($K = 3$). A distribuição do conjunto de teste foi dominada por transições de persistência, nas quais o próximo estágio é idêntico ao último observado na janela, representando 66,5% dos casos. As transições não vistas na janela, em que o próximo estágio não está presente entre os K alertas da janela de contexto, representaram 28,5% dos casos.

Tabela 17 – Desempenho do preditor por tipo de transição de estágio (K=3).

Tipo	**N**	**%**	**Top-1**	**Top-3**	**F1**
Persistência (alvo = último)	21.526	66,5%	90,7%	98,1%	0,894
Vista, não última	1.600	4,9%	90,8%	98,6%	0,885
Transição não vista	9.224	28,5%	91,1%	98,3%	0,891

O Macro F1 permaneceu estável entre 0,885 e 0,894 nas três categorias de transição, a acurácia Top-1 permaneceu próxima de 91% e a Top-3 acima de 98% inclusive nas transições não vistas na janela, que responderam por 28,5% dos casos na configuração final ($K = 3$), proporção maior do que a observada em janelas mais largas devido à menor probabilidade de o próximo estágio já ter aparecido dentro de uma janela curta. Esse resultado é consistente com a diversidade do conjunto sintético, que ampliou a cobertura de combinações entre campanha e estágio observadas no treinamento, de modo que uma transição ausente de uma janela específica tendeu a ter sido observada em outras campanhas do conjunto de treino. Essa avaliação indica que o desempenho do preditor nessas condições dependeu mais da cobertura de transições no conjunto de treinamento como um todo do que da presença do próximo estágio na janela imediata de um exemplo específico.

A Tabela 18 detalha a acurácia de predição do IP de origem por modo de endereçamento, com separação entre os casos dentro e fora da janela de contexto. Nos modos *campaign* e *targeted*, que juntos responderam por 14.936 dos 23.126 alertas dentro da janela, o endereço de origem permanece constante ao longo de toda a campanha, de modo que a acurácia Src /32 atingiu 100% em ambos, resultado trivial que não depende da qualidade do mecanismo de ponteiro. O modo *rotating*, introduzido especificamente para testar a sensibilidade posicional da atenção, atingiu 94,56% de acurácia Src /32 dentro da janela, evidenciando que o ponteiro de origem aprendeu a identificar a posição correta do ciclo de rotação em vez de copiar mecanicamente a posição mais recente da janela. O modo *subnet_distributed* atingiu apenas 38,45% em Src /32 dentro da janela, pois os múltiplos nós de uma mesma sub-rede IoT são indistinguíveis pelos atributos disponíveis ao ponteiro, mas alcançou 100% em Src /24 tanto dentro quanto fora da janela, confirmando que o modelo aprendeu

a estrutura de prefixo de rede compartilhada por esses nós. O modo *distributed*, cujos endereços provêm de *pools* aleatórios sem estrutura de prefixo compartilhada, apresentou o desempenho mais baixo entre os modos, com 37,10% em Src /32 e Src /24 dentro da janela e 0% nos casos fora da janela, configurando o limite informacional do mecanismo de cópia por atenção, dado que nenhum padrão de endereçamento aprendível estava disponível para esses casos.

Tabela 18 – Acurácia de predição do IP de origem por modo de endereçamento ($K = 3$).

Modo	N (janela)	Src /32 (janela)	Src /24 (janela)	N (fora)	Src /32 (fora)	Src /24 (fora)
campaign	8.301	100,00%	100,00%	0	—	—
targeted	6.635	100,00%	100,00%	0	—	—
rotating	5.660	94,56%	100,00%	1.873	0,00%	100,00%
subnet_distributed	2.247	38,45%	100,00%	6.851	0,00%	100,00%
distributed	283	37,10%	37,10%	500	0,00%	0,00%

5.6.2 AIT-ADS

A avaliação sobre o AIT-ADS seguiu uma metodologia de avaliação cruzada de atacantes, com treinamento em *wardbeck*, validação em *santos* e teste em *shaw*. O conjunto apresenta desbalanceamento severo, com *dirb* e *cracking* representando 68% e 19% do conjunto de teste, respectivamente, enquanto as demais sete classes raramente aparecem nos dados de treinamento, como ilustrado na Tabela 5. Foram avaliadas três variantes de tratamento do ruído de fundo. A variante sem ruído exclui os alertas das classes de ruído (i.e., *noise_dovecot* e *noise_clamav*) integralmente das sequências de avaliação. A variante com ruído mantém os alertas de ruído na janela de contexto, mas sem incluí-los como alvos de predição. A variante ruído como alvo inclui os alertas de ruído tanto na janela de contexto quanto como alvos de predição, expandindo o conjunto de avaliação para 7.901 amostras. Os resultados são apresentados na Tabela 19.

Tabela 19 – Avaliação do preditor multitarefa sobre o AIT-ADS.

Variante	F1	Top-1	Top-3	Dest /32	Src /32 geral	Src /32 (janela)	Src cobertura	Δt MAE	N
Sem ruído	0,2147	81,76%	95,01%	88,84%	89,18%	91,98%	97,0%	0,084s	6.569
Com ruído	0,1964	83,76%	96,91%	88,18%	87,09%	94,27%	92,4%	0,060s	6.514
Ruído como alvo	0,2537	79,07%	96,60%	88,14%	87,13%	93,13%	93,6%	0,605s	7.901

As três variantes (i.e., sem ruído, com ruído e ruído como alvo) obtiveram acurácia Top-1 entre 79% e 84% e Top-3 entre 95% e 97%, valores consideráveis para uma avaliação cruzada entre atacantes. O Macro F1 permanece baixo em todas as variantes, com máximo de 0,2537, resultado do colapso de sete classes para F1=0, conforme ilustrado na matriz de confusão da Figura 3. A matriz revela que a confusão concentra-se predominantemente na classe *Cracking*, para a qual *Net. Scans*, *Rev. Shell*, *Svc. Scans*, *Webshell* e *Priv. Escal.* são majoritariamente classificadas, enquanto a *DNS Steal* divide-se quase igualmente entre a classificação correta e a confusão com *Cracking*, e o *wpscan* segue um padrão distinto, sendo confundido com *Dirb* em 92% dos casos. Dessas, seis são genuinamente raras no treinamento,

enquanto o *wpscan*, apesar de representar cerca de 11% do conjunto de teste de ataque, colapsa por falha de transferência de *embeddings* entre atacantes, padrão consistente com sua confusão sistemática especificamente com a classe majoritária *Dirb*. A classe *Noise*, por sua vez, mantém uma separação quase perfeita, refletindo a distinção clara entre o ruído de fundo e os padrões de ataque da *kill-chain*. Com o Macro F1 ponderando igualmente todas as classes, a ausência de predição correta nessas classes domina a métrica. A predição de IP manteve Dest /32 entre 88% e 89%, Src /32 dentro da janela entre 92% e 94% e Src /32 geral entre 87% e 89%, valor superior aos 65,71% obtidos na configuração final dos dados sintéticos. Essa diferença não indica melhor generalização do preditor no AIT-ADS, mas reflete a composição dos modos de endereçamento dos dois conjuntos, já que o conjunto sintético inclui deliberadamente os modos *distributed* e *subnet_distributed*, nos quais o IP de origem é estruturalmente imprevisível fora da janela de contexto, conforme detalhado na Tabela 18, enquanto o AIT-ADS não possui uma proporção comparável de casos equivalentes.

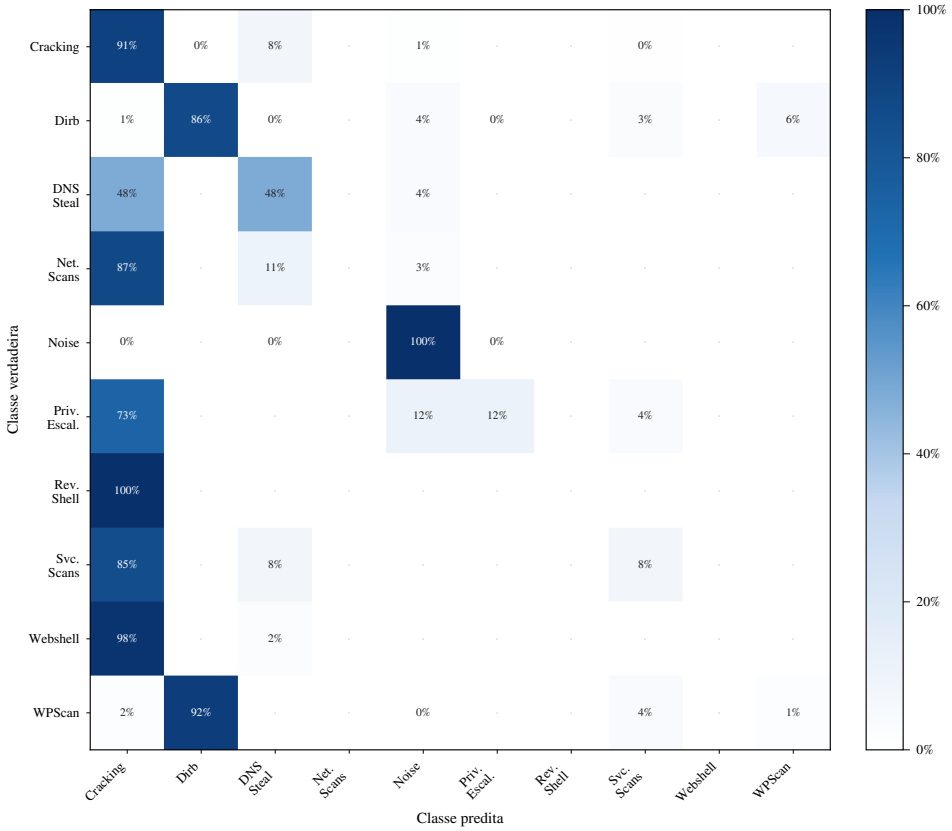

Figura 3 – Matriz de confusão da predição de estágio no AIT-ADS (*ruído como alvo*).

O Δt MAE nas variantes sem ruído como alvo (0,084s e 0,060s) indica baixo erro de predição temporal nesse conjunto, embora a comparação direta com os dados sintéticos (4,26s) seja limitada pela maior homogeneidade temporal do AIT-ADS, cujos alertas são gerados em condições com menor variância de intervalo. A variante que trata o ruído como alvo elevou o Δt MAE para 0,605s, pois os alertas de ruído não seguem padrões de *kill-chain* e o modelo não consegue predizer seus intervalos temporais com precisão.

Em conjunto com os resultados sintéticos, onde o preditor atingiu Macro F1 de 0,8937 em dados balanceados, esses resultados confirmam que o fator limitante no AIT-ADS é o desbalanceamento de classes, não a capacidade arquitetural da abordagem.

5.7 Estudos de Ablação

Esta seção avalia quatro decisões de projeto por meio de estudos de ablação, metodologia que substitui ou modifica um componente específico e mede o impacto no desempenho, mantendo as demais condições fixas. Os estudos foram conduzidos sobre uma configuração de referência dos dados sintéticos, com $K = 5$ e 200 campanhas de treinamento, que fornece rótulos precisos e classes balanceadas para isolar o efeito de cada variante de forma independente da configuração final adotada. Dois estudos avaliam escolhas de treinamento e dois avaliam escolhas arquiteturais.

O primeiro estudo investiga se a substituição da perda tripleto por entropia cruzada (*cross-entropy, CE*) no codificador de alerta melhora o desempenho do preditor, partindo da hipótese de que agrupamentos de estágio mais puros forneçam representações latentes de maior qualidade para a predição de sequências. O segundo investiga se a adição do histórico de Δt como entrada explícita do preditor reduz o erro de predição temporal. A hipótese é que os intervalos recentes dos K alertas de contexto forneçam sinal adicional além daquele capturado implicitamente pelos mecanismos de atenção sobre os *embeddings*.

O terceiro estudo avalia a *PointerGeneratorIPHead*, uma variante da cabeça de predição de IP. Na configuração final, a cabeça de ponteiro seleciona o próximo IP diretamente entre os endereços presentes na janela de contexto por atenção, sem capacidade de predizer endereços ausentes. A variante sob ablação introduz um portão aprendível que estima a probabilidade de o próximo IP estar na janela, favorecendo o ponteiro quando essa confiança é elevada e acionando um ramo gerador para produzir o endereço nos demais casos, com a hipótese de que esse mecanismo melhore a predição nos modos de endereçamento onde o IP atacante não consta na janela de contexto. O quarto estudo compara o treino em estágios sequenciais, no qual os pesos de cada componente são congelados antes de iniciar o treino do seguinte, com o treino unificado ponta a ponta, que propaga gradientes por todos os componentes simultaneamente, com a hipótese de que a otimização conjunta melhore o desempenho global.

Os resultados dos quatro estudos estão apresentados na Tabela 20. O estudo com entropia cruzada elevou a V-Measure de estágio de 0,63 para 0,91, mas reduziu o Macro F1 do preditor em 0,35 pontos percentuais. A adição do histórico de Δt reduziu o MAE de 4,20s para 4,19s, com leve queda de 0,0008 no Macro F1. O portão da *PointerGeneratorIPHead* colapsou para 100% de utilização do ponteiro durante o treinamento, produzindo F1 inferior à configuração base. O treino ponta a ponta unificado produziu Macro F1 de 0,8778, contra 0,8831 do treino em estágios.

Tabela 20 – Resultados dos estudos de ablação.

Ablação	**Resultado**
Enc. de alerta com CE	V(estágio): 0,63→0,91; F1: −0,35pp
Histórico de Δt	MAE: 4,20→4,19s; F1: −0,0008
PointerGeneratorIPHead	Portão colapsa a 100% ponteiro; F1 inferior
Treino unificado	F1: 0,8778 vs. 0,8831 (−0,53pp)

O estudo com entropia cruzada aponta uma dissociação entre qualidade de agrupamento e desempenho preditivo, pois a perda tripleto preserva a variância geométrica entre estágios que o preditor utiliza para modelar transições, enquanto a entropia cruzada produz *embeddings* mais compactos intra-classe ao custo de menor separação inter-estágio. O histórico de Δt não trouxe ganho mensurável, sugerindo que os mecanismos de atenção do Transformer já capturam a informação temporal relevante a partir dos *embeddings* de contexto. Em conjunto, esses dois resultados indicam que as escolhas de treinamento da configuração final são adequadas e não há ganho em substituí-las pelas alternativas testadas.

Durante o treinamento, o portão da *PointerGeneratorIPHead* convergiu para $p_{\text{gen}} \approx 0$ todos os casos, efetivamente zerando o ramo gerador. Esse colapso ocorre porque o desequilíbrio entre os dois regimes de dados cria um gradiente dominante a favor do ponteiro. Nos casos em que o IP alvo está na janela (aproximadamente 95% dos exemplos), a cópia por atenção já atinge 98% de acerto, gerando gradiente forte para $p_{\text{gen}} = 0$. Nos casos restantes, os IPs provêm de *pools* aleatórios sem padrão aprendível, e o gerador recebe gradiente incoerente, de modo que o portão aprende a ignorá-lo completamente. A saída final combina os dois ramos segundo $\hat{y} = p_{\text{gen}} \cdot \hat{y}_{\text{gen}} + (1 - p_{\text{gen}}) \cdot \hat{y}_{\text{ptr}}$, onde o portão $p_{\text{gen}} \in [0, 1]]$ deveria aprender a acionar o gerador nos casos em que o IP alvo não está na janela e o ponteiro nos demais, mas o desequilíbrio de gradiente descrito impede esse comportamento. O colapso sugere que o problema de predizer IPs completamente ausentes da janela não tem solução arquitetural, pois não existe padrão nos dados que permita essa inferência, e a rede aprende isso corretamente ao zerar o ramo gerador.

A hipótese de que a otimização conjunta supera o treino em estágios não se confirmou, e a causa identificada é a combinação de dois fatores. O primeiro é a compressão das representações BERT de 768 para 256 dimensões, que introduz perda de informação que o treino conjunto não recupera. O segundo é a tarefa de cópia de IP, de perda imediata baixa, que monopoliza o gradiente nas iterações iniciais e impede que a representação de estágios se consolide. Esses dois fatores persistem independentemente da janela de contexto, e o F1 praticamente idêntico para $K = 5$ e $K = 16$ no modelo unificado (0,8778 e 0,8779, respectivamente), valores testados exclusivamente neste estudo de ablação, confirma que o pré-treino contrastivo dos codificadores em estágios separados produz representações de maior qualidade do que o treino conjunto, ganho que não é recuperado pelo simples aumento da janela de contexto. Esses resultados confirmam que as escolhas arquiteturais da configuração final são mais efetivas do que as alternativas testadas.

5.8 Discussão

Os resultados de avaliação dos codificadores validam a hipótese de que a separação das tarefas de discriminação de estágio e de separação de incidentes em componentes especializados é necessária. O codificador de alerta aprendeu representações que organizam os eventos por estágio da *kill-chain*, atingindo V-Measure de 0,7815 no AIT-ADS com redução de 99,77% no volume de alertas, mas apresentou separação de campanha consistentemente inferior a 0,25 em ambos os conjuntos de dados. O codificador de campanha supriu essa lacuna, atingindo V-Measure de campanha de 0,9968 com apenas 0,03% de ruído residual, o que valida a decisão de dedicar um componente exclusivamente à tarefa de correlação de incidentes.

A avaliação do preditor multitarefa sobre os dados sintéticos demonstrou que a arquitetura é capaz de predizer o próximo estágio, os endereços IP e o intervalo temporal com elevada precisão quando os dados são balanceados, atingindo Macro F1 de 0,8937 e acurácia Top-3 de 98,45%. A comparação com os resultados sobre o AIT-ADS, onde o Macro F1 máximo foi de 0,2537 com colapso de sete classes, confirma que o fator limitante é o desbalanceamento de classes do conjunto real e não uma limitação arquitetural. Essa dissociação sugere que a abordagem tem potencial para cenários com maior cobertura de técnicas de ataque ou conjuntos enriquecidos com dados sintéticos balanceados.

A análise da predição de IP revelou uma assimetria entre os casos cobertos pela janela de contexto e os casos externos. Para os 71,50% de alertas com IP de origem na janela, a acurácia Src /32 do preditor atingiu entre 91,89% e 91,92% conforme a configuração, valor próximo ao do *baseline* ingênuo no mesmo subconjunto (93,09%), pois ambas as abordagens acertam trivialmente os modos *campaign* e *targeted*, que respondem pela maior parte dos casos dentro da janela. O modo *rotating* isola a contribuição real do mecanismo de ponteiro, atingindo 94,56% de acurácia Src /32 dentro da janela apesar do endereço de origem se alternar ciclicamente, e o modo *subnet_distributed* demonstrou que o modelo aprendeu a estrutura de prefixo de rede, atingindo 100% em Src /24 tanto dentro quanto fora da janela. A acurácia Src /32 geral de 65,70% a 65,71%, inferior ao *baseline* ingênuo (66,56%), não reflete falha do modelo, mas a agregação de subproblemas qualitativamente distintos, com desempenho próximo ao *baseline* nos modos triviais e nos casos fora da janela, e desempenho superior nos casos não triviais dentro da janela, como o modo *rotating*. Para os casos do modo *distributed*, a ausência de padrão de endereçamento aprendível constitui um limite informacional da janela deslizante, confirmado pelo colapso da *PointerGeneratorIPHead*, indicando que a solução para esses casos requer informação externa à arquitetura, como inteligência de ameaças sobre *pools* de IPs conhecidos.

A análise de transições revelou que 28,5% dos casos correspondem a transições cujo próximo estágio não aparece na janela de contexto, e mesmo nesses casos o preditor manteve Top-1 de 91,1% e F1 de 0,891, próximos aos das transições de persistência. Esse resultado sugere que o preditor generaliza a partir da cobertura de transições observada

no conjunto de treinamento como um todo, e não apenas da presença do próximo estágio na janela imediata, capacidade que tende a depender da diversidade de campanhas e combinações de estágio disponíveis no treinamento. Em ambientes reais, onde campanhas de ataque evoluem com técnicas inéditas, essa generalização pode ser mais limitada do que a observada nos dados sintéticos, o que representa uma limitação a ser considerada. Em contrapartida, a varredura do tamanho da janela mostrou que o peso médio de atenção máxima do ponteiro de origem diminui conforme K cresce, indicando que o modelo concentra a atenção em posições específicas da janela em vez de distribuí-la uniformemente, o que reflete capacidade de raciocínio posicional sobre a sequência.

Em relação aos trabalhos da literatura, a abordagem proposta não é diretamente comparável em métricas numéricas, pois cada sistema é avaliado sobre conjuntos de dados distintos e resolve subconjuntos diferentes da tarefa, o que impede a equiparação de valores de F1 ou acurácia entre estudos. A análise comparativa é, portanto, conduzida em termos das capacidades de cada sistema e das lacunas que a abordagem proposta busca suprir. Os sistemas TIRESIAS [43], DeepCASE [44], StageFinder [47] e DeepOP [48], bem como a abordagem de Fredj [46], restringem a predição à categoria do próximo evento ou ao estágio corrente, sem estimar atributos de rede como endereços IP e intervalo temporal. A abordagem de Ansari et al. [45] é a mais próxima em escopo, por incluir estimativas de tempo e prefixo /24 dos endereços IP de origem e destino, mas limita-se ao prefixo /24, sem estimar o endereço completo, e não trata o entrelaçamento de campanhas simultâneas que dificulta a organização das janelas de contexto. A abordagem proposta avança em duas dimensões complementares em relação a esses sistemas, ao estimar conjuntamente o próximo estágio, os endereços IP completos (/32) de origem e destino e o intervalo temporal em uma única arquitetura, e ao realizar a correlação dinâmica de campanhas simultâneas que organiza as janelas de contexto sobre as quais a predição é realizada, sem depender de rótulos de incidente disponíveis em tempo de inferência. Essa combinação não é abordada por nenhum dos sistemas da literatura revisada, o que posiciona a proposta como uma contribuição complementar ao estado da arte.

Em conjunto, os resultados indicam que a abordagem proposta constitui uma contribuição válida para a correlação dinâmica e a predição de ataques multiestágio, com limitações claramente identificadas e quantificadas. A redução de 99,77% no volume de alertas e a elevada precisão preditiva sobre o conjunto sintético, obtida em condições de classes balanceadas, apontam para a viabilidade operacional do sistema como ferramenta de apoio a analistas de segurança. Os estudos de ablação fundamentam as escolhas de projeto, e as limitações identificadas, especialmente o desbalanceamento de classes nos dados e a predição de IPs fora da janela, delineiam caminhos para pesquisas futuras.

5.9 Resumo

Este capítulo apresentou a avaliação experimental da abordagem proposta sobre o AIT-ADS e um conjunto sintético desenvolvido para a validação do *pipeline* completo. Os codificadores foram avaliados pela qualidade dos agrupamentos produzidos, com o codificador de alerta atingindo V-Measure de 0,7815 no AIT-ADS com redução de 99,77% no volume de alertas, e o codificador de campanha atingindo V-Measure de 0,9968 no conjunto sintético, confirmando a necessidade da arquitetura de dois codificadores especializados. O preditor multitarefa demonstrou Macro F1 de 0,8937 sobre dados sintéticos balanceados, e a comparação com os resultados sobre o AIT-ADS evidenciou que o fator limitante é o desbalanceamento de classes, não a capacidade arquitetural da abordagem.

Os estudos de ablação confirmaram as principais escolhas de projeto, mostrando que a perda tripleto organiza o espaço latente de forma mais adequada à predição de sequências do que a entropia cruzada, que o treino em estágios sequenciais supera o treino unificado ponta a ponta e que a predição de IPs fora da janela de contexto constitui um limite informacional sem solução arquitetural. A análise de transições revelou que o desempenho do preditor permanece estável entre os três tipos de transição, incluindo os casos cujo próximo estágio não aparece na janela de contexto, indicando que a generalização depende da cobertura de transições no conjunto de treinamento como um todo. O capítulo seguinte apresenta as conclusões do trabalho, discutindo as contribuições alcançadas e os caminhos para investigações futuras.

6 Conclusão

As limitações dos métodos estáticos de correlação de alertas frente à sofisticação das ameaças cibernéticas motivaram o desenvolvimento deste trabalho. A dependência de regras fixas e assinaturas predefinidas mostra-se insuficiente para lidar com o volume massivo de eventos gerados pelos sistemas de monitoramento modernos, perpetuando o problema da fadiga de alertas nos SOCs e dificultando a identificação e a predição de ataques de múltiplos estágios. Para superar esse cenário, esta monografia apresentou e validou uma abordagem fundamentada em aprendizado profundo com arquitetura *Transformer*, na qual a correlação dinâmica de alertas e a predição multitarefa de eventos futuros são tratadas como problemas complementares de modelagem de sequências.

A abordagem proposta organiza-se em torno de dois codificadores que operam em paralelo sobre representações vetoriais dos alertas. O codificador de campanha aprende, por aprendizado contrastivo com perda tripleto em nível de incidente, um espaço métrico no qual janelas de alertas de uma mesma campanha formam regiões compactas e separadas das demais, permitindo que o agrupamento HDBSCAN delimite automaticamente as fronteiras entre operações ofensivas simultâneas sem depender de regras predefinidas. O codificador de alerta aprende representações em nível de evento organizadas por estágio da cadeia de ataque, que servem de base para o preditor multitarefa, responsável por estimar simultaneamente o próximo estágio, os endereços IP de origem e destino e o intervalo temporal até a próxima ocorrência por meio de cabeças de saída especializadas, incluindo um mecanismo de ponteiro para a predição de endereços.

A avaliação experimental sobre o conjunto sintético e o AIT-ADS validou as hipóteses centrais da proposta. O codificador de campanha atingiu V-Measure de 0,9968, Homogeneidade de 1,0000 e Completude de 0,9936 no conjunto sintético, com apenas 0,03% de ruído residual, resultado que representa um salto de mais de 75 pontos percentuais em relação às variantes do codificador de alerta avaliadas como referência. O codificador de alerta atingiu V-Measure de 0,7815 no AIT-ADS, com redução de 99,77% no volume de alertas, consolidando os eventos brutos em 44 incidentes agrupados com alta correspondência aos rótulos reais de estágio. O preditor multitarefa atingiu Macro F1 de 0,8937 e acurácia Top-3 de 98,45% nos dados sintéticos balanceados, com Dest /32 de 99,97% e acurácia de predição do IP de origem dentro da janela de 91,92%. A análise por modo de endereçamento revelou que o mecanismo de ponteiro aprendeu padrões não triviais, atingindo 94,56% de acurácia Src /32 no modo *rotating* e 100% em Src /24 no modo *subnet_distributed* mesmo para alertas fora da janela de contexto.

A avaliação cruzada entre atacantes no AIT-ADS expôs as limitações do sistema em cenários com desbalanceamento severo de classes. O Macro F1 máximo de 0,2537 reflete o colapso de sete classes para F1=0, sendo seis delas genuinamente raras no treinamento

e uma, o *wpscan*, afetada por falha de transferência de *embeddings* entre atacantes. A comparação com os resultados sintéticos confirma que essa limitação é estrutural ao conjunto de dados, não à capacidade arquitetural da abordagem. Uma segunda limitação identificada é a predição de IPs de origem completamente ausentes da janela de contexto nos modos de endereçamento sem estrutura de prefixo aprendível, para os quais o mecanismo de ponteiro atinge 0% de acurácia Src /32, configurando um limite informacional sem solução arquitetural. Os estudos de ablação confirmaram que as escolhas de projeto adotadas, a perda tripleto em vez de entropia cruzada, o treino em estágios sequenciais em vez do treino unificado ponta a ponta e a cabeça de ponteiro sem ramo gerador, são superiores às alternativas testadas.

Os resultados obtidos apontam caminhos para investigações futuras. A limitação do Macro F1 no AIT-ADS sugere que o enriquecimento do conjunto de treinamento com dados sintéticos balanceados ou técnicas de aumento de dados para classes raras pode melhorar substancialmente o desempenho do preditor em cenários reais ou desbalanceados. A predição de IPs fora da janela de contexto no modo *distributed* indica que a incorporação de inteligência externa de ameaças, como listas de reputação de IPs e bases de dados de *pools* conhecidos, pode suprir a informação estruturalmente ausente da janela deslizante. A generalização entre atacantes, evidenciada pelo colapso do *wpscan*, aponta para a investigação de técnicas de adaptação de domínio ou aprendizado por transferência que preservem a separabilidade de estágios ao longo de cenários de rede distintos. Por fim, a operação atual em modo *offline* sobre conjuntos fixos de dados constitui uma limitação para ambientes de produção, e a adaptação do pipeline para processamento incremental de fluxos de alertas em tempo real representa uma evolução relevante para a viabilidade operacional da abordagem.

Em conjunto, as contribuições desta monografia avançam o estado da arte em dois eixos complementares, a correlação dinâmica de incidentes sem dependência de regras predefinidas e a predição multitarefa de atributos do próximo evento de uma campanha em andamento. A arquitetura de dois codificadores demonstrou que as tarefas de separação de incidentes e de discriminação de estágios requerem objetivos de aprendizado distintos e não podem ser resolvidas por um único componente, resultado que orienta o projeto de sistemas futuros de correlação dinâmica de alertas de segurança.

Referências

[1] ARGYROUDIS, S. A. et al. Digital technologies can enhance climate resilience of critical infrastructure. *Climate Risk Management*, v. 35, p. 100387, 2022. ISSN 2212-0963. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2212096321001169>.

[2] ALHIDAIFI, S. M.; ASGHAR, M. R.; ANSARI, I. S. A survey on cyber resilience: Key strategies, research challenges, and future directions. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 56, n. 8, abr. 2024. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3649218>.

[3] ALBANO, L. et al. Cyberattack prediction by dynamic alert correlation: Taxonomy, challenges and directions. Submitted to IEEE Network Magazine. 2025.

[4] ALBANO, L. et al. Predição de ataques ddos pela correlação de séries temporais via padrões ordinais. In: SBC. *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*. [S.l.], 2023. p. 69–82.

[5] TARIQ, S. et al. Alert fatigue in security operations centres: Research challenges and opportunities. *ACM Computing Surveys*, ACM New York, NY, v. 57, n. 9, p. 1–38, 2025.

[6] ALBASHEER, H. et al. Cyber-attack prediction based on network intrusion detection systems for alert correlation techniques: a survey. *Sensors*, MDPI, v. 22, n. 4, p. 1494, 2022.

[7] BONDERUD, D. *Cost of a data breach 2024: Financial industry*. 2024. `https://www.ibm.com/think/insights/cost-of-a-data-breach-2024-financial-industry`. Acessado em 15 de novembro de 2025.

[8] IBM Corporation. *Cost of a Data Breach Report 2025*. [S.l.], 2025. Disponível em: <https://www.ibm.com/reports/data-breach>.

[9] IBM Brasil. *Relatório da IBM: Custo médio de uma violação de dados no Brasil atinge R$ 7,19 milhões*. 2025. Acessado em: 04 dez. 2025. Disponível em: <https://brasil.newsroom.ibm.com/2025-07-30-Relatorio-da-IBM-Custo-medio-de-uma-violacao-de-dados-no-Brasil-atinge-R-7,19-milhoes>.

[10] CHU, W.-L.; LIN, C.-J.; CHANG, K.-N. Detection and classification of advanced persistent threats and attacks using the support vector machine. *Applied Sciences*, MDPI, v. 9, n. 21, p. 4579, 2019.

[11] VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

[12] OZKAN-OKAY, M. et al. A comprehensive systematic literature review on intrusion detection systems. *IEEE Access*, IEEE, v. 9, p. 157727–157760, 2021.

[13] ARFEEN, A. et al. Endpoint detection & response: A malware identification solution. In: IEEE. *2021 international conference on cyber warfare and security (ICCWS)*. [S.l.], 2021. p. 1–8.

[14] MIRHEIDARI, S. A.; ARSHAD, S.; JALILI, R. Alert correlation algorithms: A survey and taxonomy. In: SPRINGER. *Cyberspace Safety and Security: 5th International Symposium, CSS 2013, Zhangjiajie, China, November 13-15, 2013, Proceedings 5*. [S.l.], 2013. p. 183–197.

[15] DEBAR, H.; CURRY, D.; FEINSTEIN, B. *The intrusion detection message exchange format (IDMEF)*. [S.l.], 2007.

[16] LANDAUER, M.; SKOPIK, F.; WURZENBERGER, M. Introducing a new alert data set for multi-step attack analysis. In: *Proceedings of the 17th Cyber Security Experimentation and Test Workshop*. [S.l.: s.n.], 2024. p. 41–53.

[17] NAVARRO, J.; DERUYVER, A.; PARREND, P. A systematic survey on multi-step attack detection. *Computers & Security*, Elsevier, v. 76, p. 214–249, 2018.

[18] HUTCHINS, E. M.; CLOPPERT, M. J.; AMIN, R. M. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, v. 1, n. 1, p. 80, 2011.

[19] STROM, B. E. et al. *MITRE ATT&CK: Design and Philosophy*. [S.l.], 2018.

[20] POUGET, F.; DACIER, M. *Alert Correlation: Review of the State of the Art*. [S.l.], 2003.

[21] HUSÁK, M. et al. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys & Tutorials*, IEEE, v. 21, n. 1, p. 640–660, 2018.

[22] BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, IEEE, v. 5, n. 2, p. 157–166, 1994.

[23] SOMMER, R.; PAXSON, V. Outside the closed world: On using machine learning for network intrusion detection. In: IEEE. *2010 IEEE symposium on security and privacy*. [S.l.], 2010. p. 305–316.

[24] GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1.

[25] BROMLEY, J. et al. Signature verification using a "Siamese" time delay neural network. In: *Advances in Neural Information Processing Systems (NIPS)*. [S.l.: s.n.], 1993. v. 6, p. 737–744.

[26] HADSELL, R.; CHOPRA, S.; LECUN, Y. Dimensionality reduction by learning an invariant mapping. In: IEEE. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.], 2006. v. 2, p. 1735–1742.

[27] SCHROFF, F.; KALENICHENKO, D.; PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 815–823.

[28] CARON, M. et al. Deep clustering for unsupervised learning of visual features. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018. p. 132–149.

[29] MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. [S.l.], 1967. v. 1, n. 14, p. 281–297.

[30] ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*. [S.l.: s.n.], 1996. v. 96, n. 34, p. 226–231.

[31] CAMPELLO, R. J.; MOULAVI, D.; SANDER, J. Density-based clustering based on hierarchical density estimates. In: SPRINGER. *Pacific-Asia conference on knowledge discovery and data mining*. [S.l.], 2013. p. 160–172.

[32] CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*. [S.l.: s.n.], 2006. p. 328–339.

[33] VINYALS, O.; FORTUNATO, M.; JAITLY, N. Pointer networks. *Advances in neural information processing systems*, v. 28, 2015.

[34] BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. In: *3rd International Conference on Learning Representations (ICLR 2015)*. [s.n.], 2015. Disponível em: <https://arxiv.org/abs/1409.0473>.

[35] SEE, A.; LIU, P. J.; MANNING, C. D. Get to the point: Summarization with pointer-generator networks. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. [S.l.: s.n.], 2017. p. 1073–1083.

[36] SADODDIN, R.; GHORBANI, A. A. An incremental frequent structure mining framework for real-time alert correlation. *Computers & Security*, Elsevier, v. 28, n. 3-4, p. 153–173, 2009.

[37] LIU, J. et al. Multi-step attack scenarios mining based on neural network and bayesian network attack graph. In: SPRINGER. *International Conference on Artificial Intelligence and Security*. [S.l.], 2019. p. 62–74.

[38] NADEEM, A. et al. Alert-driven attack graph generation using s-pdfa. *IEEE Transactions on Dependable and Secure Computing*, v. 19, n. 2, p. 731–746, 2022.

[39] KANNAN, S. et al. Maat: Multi-stage attack attribution in enterprise systems using software defined networks. *ICST Transactions on Security and Safety*, v. 4, n. 11, 2017.

[40] MILAJERDI, S. M. et al. Holmes: real-time apt detection through correlation of suspicious information flows. In: IEEE. *2019 IEEE symposium on security and privacy (SP)*. [S.l.], 2019. p. 1137–1152.

[41] ZIPPERLE, M. et al. Provenance-based intrusion detection systems: A survey. *ACM Computing Surveys*, ACM, v. 55, n. 7, p. 1–36, 2022.

[42] LIU, W. et al. Trace2vec: Detecting complex multi-step attacks with explainable graph neural network. *Pattern Recognition*, Elsevier, v. 162, p. 111363, 2025.

[43] SHEN, Y. et al. Tiresias: Predicting security events through deep learning. In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. [S.l.: s.n.], 2018. p. 592–605.

[44] EDE, T. V. et al. Deepcase: Semi-supervised contextual analysis of security events. In: IEEE. *2022 IEEE Symposium on Security and Privacy (SP)*. [S.l.], 2022. p. 522–539.

[45] ANSARI, M. S.; BARTOŠ, V.; LEE, B. Gru-based deep learning approach for network intrusion alert prediction. *Future Generation Computer Systems*, Elsevier, v. 128, p. 235–247, 2022.

[46] FREDJ, O. B. A nlp-inspired method to predict multi-step cyberattacks. In: IEEE. *2022 15th International Conference on Security of Information and Networks (SIN)*. [S.l.], 2022. p. 1–6.

[47] PHAN, T. V.; BAUSCHERT, T. *Learning the APT Kill Chain: Temporal Reasoning over Provenance Data for Attack Stage Estimation*. 2026. `https://arxiv.org/abs/2603.07560`. Accepted to IEEE Global Communications Conference (GLOBECOM) 2026. arXiv:2603.07560.

[48] ZHANG, S.; XUE, X.; SU, X. Deepop: A hybrid framework for mitre att&ck sequence prediction via deep learning and ontology. *Electronics*, MDPI, v. 14, n. 2, p. 257, 2025.

[49] CARUANA, R. Multitask learning. *Machine learning*, Springer, v. 28, n. 1, p. 41–75, 1997.

[50] LAN, J. et al. Member: A multi-task learning model with hybrid deep features for network intrusion detection. *Comput. Secur.*, Elsevier Advanced Technology Publications, GBR, v. 123, n. C, dez. 2022. ISSN 0167-4048. Disponível em: <https://doi.org/10.1016/j.cose.2022.102919>.

[51] WANG, Y. Application of a deep learning model integrating multi-source logs for early detection of apt attacks. In: IEEE. *2025 10th International Conference on Cyber Security and Information Engineering (ICCSIE)*. [S.l.], 2025. p. 421–427.

[52] AFNAN, S. et al. *LogShield: A Transformer-based APT Detection System Leveraging Self-Attention*. 2023. `https://arxiv.org/abs/2311.05733`. Preprint. arXiv:2311.05733.

[53] LIU, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. `https://arxiv.org/abs/1907.11692`. Preprint. arXiv:1907.11692.

[54] NIKNAMI, N.; MAHZOON, V.; WU, J. Crossalert: Enhancing multi-stage attack detection through semantic embedding of alerts across targeted domains. In: IEEE. *2024 IEEE Conference on Communications and Network Security (CNS)*. [S.l.], 2024. p. 1–9.

[55] DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. [S.l.: s.n.], 2019. p. 4171–4186.

[56] DU, D. et al. Mad-llm: A novel approach for alert-based multi-stage attack detection via llm. In: IEEE. *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*. [S.l.], 2024. p. 2046–2053.

[57] WU, S. et al. Alert2vec: Eliminating alert fatigue by embedding security alerts through subgraph learning. *IEEE Transactions on Dependable and Secure Computing*, IEEE, 2025.

[58] SHADABFAR, H.; DEHGHAN, M.; SADEGHIAN, B. Dsrl-apt-2023: A new synthetic dataset for advanced persistent threats. *ISeCure*, v. 17, n. 2, p. 107–116, 2025.

[59] ROUSSEEUW, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, v. 20, n. 1, p. 53–65, 1987.