CrossAlert: Enhancing Multi-Stage Attack Detection through Semantic Embedding of Alerts Across Targeted Domains

***Abstract*—With the continuous evolution of attack methods, cyber attacks have become more distributed and complex, employing techniques such as multi-stage network attacks (MSA). Monitoring tools generate numerous alerts during these attacks, but the high dimensionality and diverse features of alert data often result in poor detection performance. Manual analysis of MSAs is time-consuming, leading to limited labeled data. Additionally, changes in attack types and new domains cause Intrusion Detection Systems (IDSs) to perform poorly, presenting a significant challenge known as domain shift. In this paper, we address these issues by proposing a multi-stage network attack detection algorithm that enhances MSA detection through the analysis of high-dimensional alerts and the integration of various alert aspects. Our algorithm incorporates multiple facets including semantic similarity, anomaly scores, and feature extraction to identify relevant entities, detect intricate relationships, and uncover hidden patterns, enhancing the detection of multi-stage network attacks. The model was tested successfully using the DARPA 2000 and ISCX 2012 datasets, with completeness and soundness measured to evaluate its effectiveness.**

***Index Terms*—Alert correlation, Bert, Intrusion Detection System (IDS), multi-stage attack(MSA), Prototypical Network(PTN).**

I. Introduction

A Network Intrusion Detection System (NIDS) is a crucial network security technology for detecting intruder attacks. Different intrusion detection methods are needed across various environments, requiring multiple IDS implementations within a security domain for optimal results [1]. However, there are some open issues with intrusion detection over networks. First, deploying diverse sensors enhances coverage but introduces challenges in interpreting reports, often generating a flood of low-level alerts with over 90% false positives, overwhelming analysts and leading to missed critical alerts. For heterogeneous sensors, a correlation engine should identify when reports from multiple sensors refer to the same incident. By organizing related alerts, correlation engines can reduce alert volume and enhance detection capabilities, providing a more comprehensive attack overview. Second, not all attacks are simple to detect for IDS. Multi-Stage Network Attacks (MSAs) involve a series of steps targeting a network, with each step appearing harmless independently but dangerous when combined [2]. For MSAs, not only do all single-step attacks need to be identified, but the relationships between each step also need to be inferred, which makes it more difficult to describe the similarities between attacks. A MSA with its stages is shown in Fig. 1. In this example,

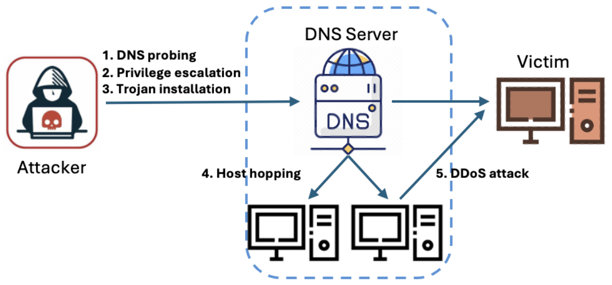


Fig. 1: Multi-stage attack.

an MSA begins with DNS probing, followed by privilege escalation, Trojan installation, lateral movement, and culminating in a disruptive attack like DDoS. Detecting MSAs is challenging because it requires identifying both individual steps and their interrelationships. The high-dimensional alert data from IDS, containing positive, false, and irrelevant alerts, complicates MSA detection [3]. Traditional machine learning models struggle with clustering this data and accounting for the non-linear, variable-timed nature of multi-step attacks.

Existing methods often struggle with false and incomplete alert correlations due to raw alerts. Those methods often fail to consider the complexity and high dimensionality of MSA alert data, leading to inefficiencies in detection. The timestamp of each step is important, similar to word order in a sentence. Additionally, the number of attack types is limited, like vocabulary size in NLP. Therefore, using BERT (Bidirectional Encoder Representations from Transformers) [4] to find semantic similarities between alerts, helping the IDS to detect multi-stage attacks more accurately. Semantic similarity techniques in NLP can identify similarities between different stages of an attack.

One major issue is domain shift, where IDS models trained on specific rules do not perform well under different configurations, highlighting the need for adaptable and robust IDS. Prototypical networks are particularly useful in this context because they can generalize well to new domains with very limited available labeled data by learning a metric space where data points from the same class are clustered together. This allows the IDS to maintain high performance even when the operational environment changes, thereby addressing the domain shift challenge effectively. Our paper proposes a new approach that considers multiple aspects of alerts, including alert messages, potential alert stages, and some extracted features within the dataset. Our algorithm leverages NLP to extract semantic embeddings from alert messages and determine the corresponding potential alert stages. These embeddings are then integrated with the anomaly scores, some extracted features to effectively detect MSAs. To address domain shift, these concatenated embeddings are subsequently passed through a prototypical network, ensuring a reliable IDS. Our main contributions are summarized as follows:

- We proposed a NIDS that leverages NLP to extract semantic embeddings from raw alert messages using BERT, enhancing detection accuracy and reducing false positives.
- We utilize integrating some extracted features, potential alert stages, anomaly scores, and obtained semantic embedding of alert messages to improve the system's ability to detect multi-stage attacks with higher accuracy.
- We addressed the domain shift problem in IDSs by implementing a few-shot learning methodology, specifically using the Prototypical Network.
- We demonstrate the high performance of our methodology on real-world datasets in both within-dataset and cross-dataset scenarios.

II. Preliminary and Related Work

A. Natural Language Processing (NLP)

NLP technology is essential for understanding and analyzing natural language. As NLP has matured, its use in cybersecurity has increased, particularly in binary code similarity comparison, where binary code is represented as vectors using techniques like word2vec [5]. NLP involves tasks such as lexical analysis or tokenization, sentence analysis, semantic analysis, and information extraction. Transformer-based models like BERT [4], GPT-2 [6], and XLNet [7] dominate the field of Large Language Models (LLMs) [8], [9] in NLP. BERT is highly effective in finding semantic similarity [10] between texts by understanding the context and meaning of words within sentences [11].

Lira *et al.* [12] proposed a model demonstrating the capability of LLMs to identify normal and anomalous traffic from 23 different types of attacks with higher accuracy and a lower false positive rate than other researched models. Their model processes and comprehends large volumes of network log data, autonomously learns and adapts to evolving network behavior, and effectively differentiates between regular activities and potential threats. In the context of cyber attack scenario reconstruction, semantic similarity techniques can identify similarities between incident reports, facilitating the detection of common attack patterns and tactics.

B. Multi-Stage Attack (MSA)

MSAs are among the most critical security threats in cyberspace, defined as a series of steps involving at least two actions taken by one or more attackers targeting a specific network [2]. MSAs consist of multiple steps through which attackers infiltrate various networks and systems. Unlike single-stage attacks, detecting MSAs is challenging because each stage often appears harmless when executed alone [13] [14]. However, the combined execution of these steps results in a highly dangerous attack, posing significant risks to industries and individuals. Analyzing MSA events involves extracting high-level security events from low-level alert logs. Each step in an MSA is related, with the previous step often leading to the next. By analyzing the early stages, it is possible to predict subsequent attack events and proactively prevent major damage [15], [16].

Wang *et al.* [17] propose a multistage network attack detection algorithm based on a Gaussian mixture hidden Markov model and transfer learning. Mao *et al.* [18] addressed the problem of numerous redundant features in the high-dimensional alert data by leveraging IDS alerts corresponding to abnormal traffic to correlate attacks, reconstruct MSA scenarios, and discover attack chains. They introduced a Graph-based Fusion Module that uses risk assessment and time information to create a weighted attack scenario, enhancing reconstruction accuracy. Hu *et al.* [19] propose an attack graph-based alert correlation approach to capture network connectivity and vulnerabilities, mapping alerts to the attack graph to derive and cluster attack sequences. Their approach detects unreported true negative alerts and merging broken scenarios. Haas *et al.* [20] introduce GAC, which clusters alerts and connects clusters based on host communication to form a complete attack. However, GAC can only identify predefined scenarios and ignores detailed IDS alert information.

III. Methodology

Network-based IDSs generate a high volume of low-quality alerts, and it arise from the inability of alert generation rules to distinctly differentiate between normal and malicious activities, leading to numerous alerts that do not represent genuine security threats. Additionally, IDSs may miss certain attacks. As mentioned earlier, IDSs generate individual alerts for each step of a MSA. Therefore, to detect MSAs, it is crucial to identify alerts related to each potential stage.

Definition 1. Attack Stage:

The attacker's attack actions are implemented step by step. A whole attack scenario includes multiple stages and each stage has its characteristics.

We categorize alerts into four attack stages: scan, exploit, get-access-privilege, and post-attack, as shown in Table I. Note that not all alerts are associated with MSAs. The purpose of stage clustering is to identify the potential stage of an alert if it were to be part of an MSA. The scan is the first step in an attack, aiming to probe and collect information through various methods to prepare for subsequent attacks. The exploit stage involves the actual attack process, which may include exploiting vulnerabilities or running malicious documents on the victim host. When the attack is successful, the attacker needs to obtain the necessary permissions to control the victim host and perform further attacks, marking the get-access-privilege stage. Finally, after successfully compromising a host, the

TABLE I: Alert Stages.

No.	Alert Stage	Alert Types
1	Reconnaissance or Scan	IP address scan, Port scan, Version scan, Vulnerability scan, Social engineering
2	Exploit	Malicious file in network traffic and host, Command injection, Vulnerability attack
3	Get Access	SSH login, RDP login, shell connect
4	Post-Penetrate or Post-Attack	Data transfer, command & control, backdoor communication

TABLE II: Categorization of Different Network Threats

Stage	Alerts	Tag
Reconnaissance or Scan	scan behavioral unusually fast terminal server traffic potential scan or infection (outbound) scan suspicious inbound to postgresql port 5432	Fast Terminal Scan PostgreSQL Inbound Scan
Exploit	imap fetch overflow attempt exploit echo command attempt	HP Power Manager Overflow Echo Command Exploit
Get Access	web_server /bin/sh in uri possible shell command execution attempt trojan possible metasploit payload common construct bind_api (from server)	Shell Command Execution Metasploit Payload
Post-Penetrate or Post-Attack	policy incoming basic auth base64 http password detected unencrypted trojan blue bot ddos blog request	Unencrypted HTTP Password Blue Bot DDoS Request

TABLE III: Alerts related to the reconnaissance or scan stage.

Stage	Alerts	Tag
Reconnaissance or Scan	info observed dns query to .biz tld scan nmap os detection probe scan potential ftp brute-force attempt response et scan suspicious inbound to mssql port 1433 et info dynamic_dns query to a suspicious no-ip domain et info dotted quad host dll request	DNS Query to .biz TLD Nmap OS Detection Probe FTP Brute-Force Attempt MSSQL Inbound Scan Dynamic DNS Query DLL Request to Quad Host

attacker conducts follow-up actions such as information theft, categorized as post-attack. The actions of the attacker at each stage are detailed in Table I. Steps are according to the kill chain model [21]. Table II shows two alerts for each stage of an attack along with their associated tags. For simplicity, we use tags instead of lengthy alert messages. Table III provides more detailed alerts related to the reconnaissance stage and their respective tags.

For an anomaly detection system, it is crucial to address both *within-dataset detection* and *domain shift detection* for comprehensive coverage and adaptability. Ignoring these can reduce performance and increase false positives or negatives, undermining system reliability.

Definition 2. Within-Dataset Detection:

Identifying anomalies within a single dataset where data distribution remains the same.

Definition 3. Domain Shift Detection:

Identifying anomalies in datasets when data distributions change over time.

Detection within a dataset focuses on identifying anomalies when the model is trained and tested on the same dataset, ensuring consistent data distribution and unchanged statistical properties. This makes it straightforward to detect outliers based on the learned model. Domain shift detection, on the other hand, addresses identifying anomalies when the model is trained on one dataset and tested on another. This involves changing data distributions over time or across domains, complicating detection, and introducing different attack types.

The overview of CrossAlert is shown in Fig. 2 which contains three main modules. The diagram illustrates a framework for detecting multi-stage network attacks by leveraging IDS alerts. Raw alerts generated by IDS sensors are first processed through feature extraction to obtain Basic and Sequence-based features. Feature extraction module extracts Basic features from raw alerts, Sequence-based features from alert sequences. These features are then fed into an anomaly detection module that processes the data, generating anomaly scores that indicate potential threats. The core of this framework utilizes BERT (Bidirectional Encoder Representations from Transformers) [4] to analyze the semantic similarity between alert messages and obtain rich and meaningful embedding from alert messages. In this model, the alert messages undergo an initial tokenization process and are then transformed into vector representations via an embedding layer. Positional embeddings are then added to these token vectors, providing the model with information about the order of tokens in the sequence. These token representations are subsequently passed through multi-head attention mechanisms to capture complex dependencies and contextual relationships between words. The obtained vectors are further processed by feed-forward layers to generate the final embeddings for each token. Due to the high dimensionality of these representations, Principal Component Analysis (PCA) is applied to compress the embeddings, making them more manageable while retaining essential information.

Finally, the compressed embeddings are concatenated with the obtained anomaly score, the Basic and Sequence-based features, and then fed into a neural network that classifies the

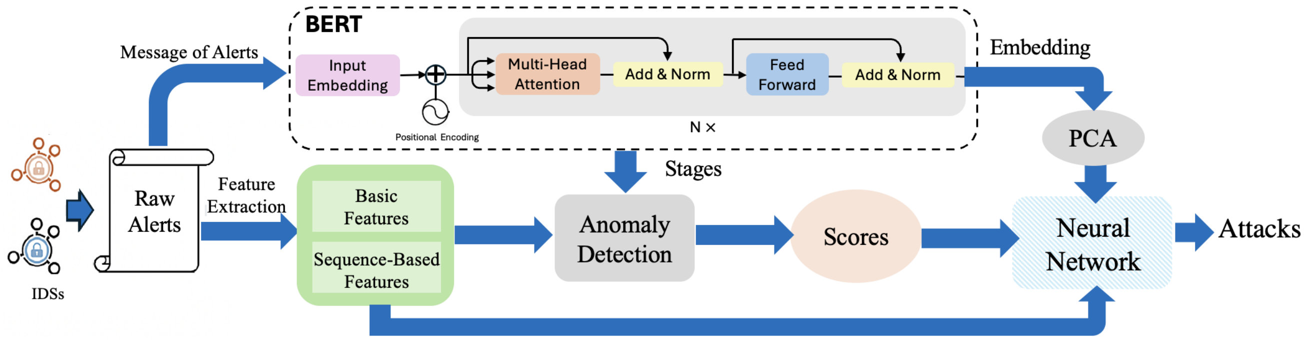


Fig. 2: CrossAlert:An Alert-based NIDS for Multi-Stag Attacks over Different Domains.

alerts to detect multi-stage attacks. This approach enhances the accuracy of intrusion detection by identifying hidden patterns and relationships within the alert data, reducing false positives, and improving the overall detection of complex attack scenarios. This framework addresses several challenges in cybersecurity, such as the high dimensionality of alert data, the prevalence of false positives, and the difficulty in correlating alerts from diverse sources. By combining feature extraction, anomaly detection, semantic analysis with BERT, dimensionality reduction with PCA, and classification with neural networks, the framework provides a robust solution for detecting and mitigating multi-stage network attacks.

A. Phase 1: Alert Preprocessing and Feature Extraction

The goal is to transform the text-style alert data generated in Phase 1 into data that can be used by machine learning algorithms. The main part of this step in the framework is feature extraction, where the raw alerts are processed to derive both *Basic features* (the fundamental attributes that are independent of the alert sequence) and *Sequence-based features* (features derived from the alert sequence and capture temporal or contextual information). These extracted features are crucial for understanding the alert data in a more structured manner. In multi-step attack detection, the inter-correlation between alerts is crucial. Therefore, CrossAlert takes advantage of both Basic and Sequence-based features to calculate anomaly scores. *Feature 1* and *2* are Source IP (SIP) and Destination IP (DIP). In these features, instead of using one-hot encoding which leads to sparse high-dimensional vectors, we used frequency encoding. These features represent participants in the alert, either an attacker or a victim. *Feature 3* is attack stage. The attack stage of an alert provides information about the progression of the attack in the MSA process. We divide the MSA process into four stages: reconnaissance, exploitation, get access privilege, and post-penetration. Stages of a MSA are shown in Table I. In the next section, we will explain how we derive the potential attack stage from the alerts.

Sequence-based features are extracted from a sequence of alerts triggered by a MSA. An alert sequence refers to a collection of alerts that share the same (SIP, DIP) within a specific time interval, arranged in chronological order. While an individual alert might seem low-risk when considered in isolation, within the context of an MSA, it could escalate to a critical level, necessitating a more thorough analysis. To improve the detection rate of multistep attacks, we extract Sequnece-based features to capture these relationships. *Feature 5* and 6 are defined as the number of alerts with the same alert message generated by the same SIP and DIP, respectively, across all alerts. *Feature 7* is also defined as the count of alerts with the same alert message generated by the same combination of both SIP and DIP. These features measure how many alerts share the same attacker or victim with the alert in sequence. A large number of such alerts could indicate either false positives from normal behavior or concentrated attacks on a single target.

B. Phase 2: Semantic Embedding with BERT

The semantic description of alerts can be seen as a sequence of statements, and if the context of two alert descriptions is similar, it can be considered that they have similar semantics. In MSAs, the attacker's actions are intentional, and the alerts from attack stages also exhibit certain characteristics. Similar attack methods result in similar alert information. Therefore, by learning alert semantic representations from a large number of alert sequences, it is possible to effectively represent alerts.

We leverage the power of Encoder Representations from Transformers (BERT) which is a powerful language model that excels in understanding the semantics of text. BERT is utilized to analyze the semantic similarity between different alert messages. This involves converting the alert messages into rich embeddings through a series of steps: tokenization, initial embedding via embedding layer, positional embedding, multi-head attention, and feed-forward layers to capture the sequential nature of the data, as shown in Fig. 2.

1) Tokenization and Vector Representation: BERT takes a sequence as an input and tokenize it using WordPiece [4] and will add special tokens such as [CLS] as the first token of every sequence and then convert each token into a dense vector representation via WordPiece embeddings [22] with a $30,000$ token vocabulary.

2) Positional Embedding: The BERT model utilizes the transformer architecture [23], which is permutation invariant, meaning the order of input tokens does not affect the model's output. To provide the model with information about the

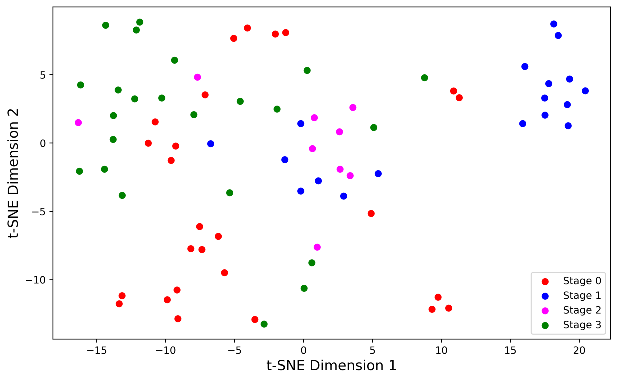


Fig. 3: t-SNE for different stages without prior knowledge.

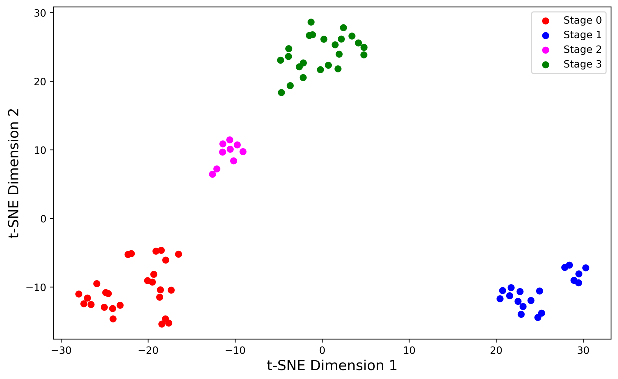


Fig. 4: t-SNE for different stages with prior knowledge.

position of each token, BERT employs positional encoding. This technique incorporates the position of tokens through sinusoidal functions, defined as follows:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{1}$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d}}\right), \tag{2}$$

where pos is the position of the token, i is the dimension index and d is the dimension of the embeddings. Obtained embeddings from the first two steps are fed into BERT's architecture, enabling it to capture rich contextual relationships within the text.

3) Multi-Head Attention: The multi-head attention mechanism allows the model to focus on different parts of the alert message simultaneously. We denote the obtained embedding of one alert message from the first two steps by $X \in \mathbb{R}^{L\times d}$, where L is the sequence length and d is the embedding dimension. BERT consists of multiple layers of multi-head attention, in which a new embedding for each token is obtained. We denote the final embedding for layer l as $E^{(l)}$. In each head of the first multi-head attention layer, initially three different representations of X are obtained, called Q (Query), K (Key), and V (Value), through linear projection using three trainable matrices W_i^Q, W_i^K, W_i^V with dimensions $\mathbb{R}^{d\times d_k}$ as follows:

$$Q = XW_i^Q, \quad K = XW_i^K, \quad V = XW_i^V$$

Then, for each head:

$$\mathrm{head}_i = \mathrm{softmax}\left((QW_i^Q)(KW_i^K)^T/\sqrt{d_k}\right)VW_i^V. \tag{3}$$

Next, the heads are concatenated and multiplied by another trainable matrix W^O:

$$\mathrm{MultiHead} = \mathrm{Concat}(\mathrm{head}_1, \mathrm{head}_2, \dots, \mathrm{head}_h)W^O.$$

In order to obtain the final embedding, residual connection [24], layer normalization (LN) [25] and Feed Forward Neural Network (FNN) have been utilized as follows:

$$E^{(1)} = \mathrm{LN}(\mathrm{FNN}(\mathrm{out}) + \mathrm{out}), \quad \mathrm{out} = \mathrm{LN}(\mathrm{MultiHead} + X),$$

where $E^{(1)}$ is the final embedding for the first multi-head attention layer. This new embedding will then serve as the input for the second multi-head attention layer, and so on.

BERT model has been pretrained on the massive corpus of text data in two tasks: Masked Language Model and Next Sentence Prediction. Then it has been fine-tuned on 11 NLP tasks. Pretraining and fine-tuning makes BERT model capable of understanding the context of words in a sentence by considering the entire sequence bidirectionally. The high-dimensional embeddings generated by BERT are then processed using PCA. PCA reduces the dimensionality of the embeddings, making them more manageable for further processing while retaining the most important information. This step is crucial for handling the complexity and volume of alert data.

In Fig. 3 which is a t-SNE [26] plot, the data points representing different stages (Stage 0, Stage 1, Stage 2, and Stage 3) are scattered across the plot without clear boundaries between them. This figure indicates that BERT, without fine-tuning on cybersecurity task, struggles to distinguish between the different stages of alert messages. The clusters for each stage are not well-defined, and there is significant overlap between the stages. This scattering suggests that BERT has difficulty understanding and categorizing the alerts accurately, leading to a mixed and less interpretable clustering.

To solve this issue we need to fine-tun BERT on a cybersecurity task to make the BERT model specialized in this field. Our data did not have any labels regarding alert stage. We manually labeled 20% of it to determine the alert stage. Then we fine-tuned BERT with a classifier head on this data and obtained embeddings for all the alerts and apply t-SNE on those to get a 2D embedding. Finally, we performed clustering on these compressed embeddings. Now, we need to determine which stage of a multistage attack each of these clusters corresponds to. To do this, we use the labeled 20% to see which samples belong to which cluster, and thus, all members of that cluster will correspond to that stage.

Fig. 4 shows t-SNE plot where BERT has been fine-tuned with labeled data, data points form distinct and well-separated clusters for each stage. Each stage has clearly defined clusters with minimal overlap. This indicates that BERT, equipped with prior knowledge, can effectively differentiate between the stages, resulting in clearer and more distinct groupings of alerts. The enhanced clustering demonstrates BERT's im-

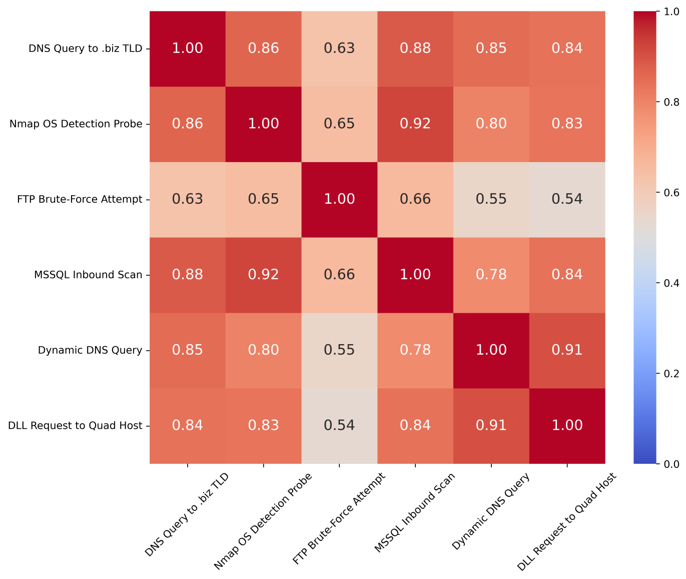


Fig. 5: Similarity heatmap for alert of Scan stage.

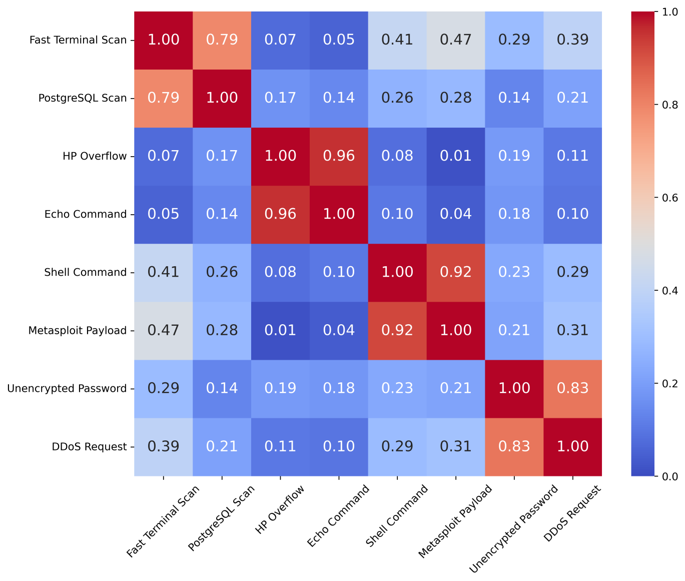


Fig. 6: Similarity heatmap for alerts.

proved understanding and categorization of the alert messages when it has access to labeled data.

Fig. 5 visualizes the similarity between alert messages of the same stages in a multistage attack. For this figure, we considered the alert messages in Reconnaissance or Scan stage. We utilized the fine-tuned BERT to obtain the embedding for these alerts and compute the cosine similarity between them. Each cell in the matrix represents the correlation between two types of alert messages, with values ranging from 0 to 1. High values, closer to 1, indicate a strong similarity, while lower values suggest a weaker relationship.

We performed the same analysis between alert messages from different and same stages of an attack, illustrated in Fig. 6. Notably, *HP Overflow* and *Echo Command* exhibit a very high correlation, suggesting these alerts frequently occur together, likely within the same stage. On the other hand, *Fast Terminal Scan* and *HP Overflow* show a very low correlation, indicating these alerts are less likely to be observed together, potentially occurring in different stages. Additionally, *Unencrypted Password* and *DDoS Request* have a significant correlation, suggesting a potential relationship between these alerts in similar stages. This heatmap effectively highlights the relationships and patterns between various alerts, aiding in the understanding of multistage attack behaviors and improving the efficiency of detection systems

C. Phase 3: Anomaly Score

The Basic and Sequence-based features are fed into an anomaly detection module to calculate the anomaly score for each alert. We incorporate various and novel features to provide the isolation forest algorithm with sufficient context. This module uses sophisticated algorithms to analyze the features and detect deviations from normal patterns, which are indicative of potential security threats. The output of this module is a set of scores that quantify the likelihood of an alert being part of an ongoing attack. In our methodology, we employed the isolation forest [27] algorithm. This algorithm is an unsupervised anomaly detection technique that employs multiple decision trees to compute anomaly scores, which are then averaged. Each decision tree recursively partitions the alert messages by randomly selecting a feature and a split value within the feature's range, aiming to isolate the anomalous alerts. In the isolation forest algorithm, each tree assigns higher anomaly scores to alerts that can be isolated with shorter path lengths or with fewer splits.

D. Phase 4: Classification

The reduced embeddings are fed into a neural network. This neural network is designed to classify the alerts, effectively detecting multi-stage attacks. By analyzing the embeddings, the neural network can identify hidden patterns and relationships that signify an ongoing multi-stage attack. This integrated approach enhances the accuracy of intrusion detection systems by reducing false positives and providing a comprehensive understanding of complex attack scenarios.

Algorithm 1 provides the pseudocode for the Cross-Alert methodology for training a classifier to detect MSAs given alerts in chronological order $\{a_1, a_2, \ldots, a_N\}$, with a few labeled alerts. Initially, we manually determine the alert stage for a small subset of alerts and fine-tune a BERT model with a classifier head (*BERTClassifier*) using these alerts. For each aler a_i, we extract Basic and Sequence-based features (F_i) and determine its alert stage (S_i) using the fine-tuned BERT model. We then train an isolation forest on the features and alert stages in an unsupervised manner. The trained isolation forest provides an anomaly score (AN_i) for each alert. For each alert a_i, we create a window of alerts $[a_{i-5}, \ldots, a_i, \ldots, a_{i+5}]$ to obtain a rich embedding (E_i) from the BERT model. Finally,

Algorithm 1 Cross-Alert Algorithm

```
1: Input $\{a_1, a_2, \ldots, a_N\}$: Alerts in chronological order,
   $\{(a_{s_1}, y_{s_1}), \ldots, (a_{s_M}, y_{s_M})\}$: Subset of labeled alerts
2: Output Trained Classifier $f_W$ with parameters $W$
3: $B \leftarrow$ Manually determine the alert stage of very few alerts
4: Fine-tune BERT model with a classifier head over $B$
5: for each alert $a_i$ in $\{a_1, a_2, \ldots, a_N\}$ do
6:     $F_i \leftarrow$ EXTRACTFEATURES$(a_i)$
7:     $S_i \leftarrow$ BERTCLASSIFIER$(a_i)$                ▷ Alert stage
8:     $E_i \leftarrow$ BERTEMBEDDING$([a_{i-5}, \ldots, a_i, \ldots, a_{i+5}])$
9: end for
10: Train Isolation Forest on $\{(F_1, S_1), \ldots, (F_N, S_N)\}$
11: for each alert $a_i$ in $\{a_1, a_2, \ldots, a_N\}$ do
12:     $AN_i \leftarrow$ ISOLATIONFOREST$(a_i)$        ▷ Anomaly score
13: end for
14: Randomly initialize network parameters $W$
15: $L \leftarrow 0$                                      ▷ Initialize the loss
16: while Accuracy is improving do
17:     for each labeled alert do
18:         $P_i \leftarrow$ CLASSIFIER$(E_i, F_i, S_i, AN_i)$
19:         $L \leftarrow L +$ CrossEntropy $(P_i, y_i)$
20:     end for=
21:     Perform Adam optimizer on $W$ to minimize $L$
22: end while
```

we concatenate the semantic embedding, alert stage, anomaly score, and extracted features, and pass them to a classifier (f_W) to detect MSAs. This classifier is trained using the labeled alerts $\{(a_{s_1}, y_{s_1}), \ldots, (a_{s_M}, y_{s_M})\}$. The sigmoid output of the classifier provides the likelihood P_i of the class labels for each alert. Using this output and the ground truth label (y_i), we calculate the CrossEntropy loss and update the model parameters with the Adam optimizer [28].

E. Detection in domain shift via Prototypical Network

In the case of domain shift in our dataset, we are dealing with a more challenging problem. To deal with that, we will use Prototypical Network (PTN) [29] in the classification phase to perform few-shot classification. PTNs are designed to learn a metric space where classification is achieved by measuring distances to the prototype representations of each class. In training our PTN, we start by generating multiple tasks from our source task. This process involves creating support sets and query sets for each task. After defining the tasks, we process the support set through the neural network, which is a crucial step as it converts the input data into embeddings. These embeddings are high-dimensional vectors that represent the features of each input. We then calculate prototypes for each label within the support set. The training process involves comparing the embeddings from the query set with these prototypes, with the goal of minimizing the distance between the query embeddings and the corresponding class prototype. Here, we choose the Feed Forward Neural Network as our embedding function for PTN.

IV. EXPERIMENTAL

We evaluate our proposed approach CrossAlert on real-world datasets. Alerts are needed to test this method, but there is currently no dataset composed entirely of alerts. Therefore, we first use IDS to generate alerts based on the traffic packet and logs in the dataset [30]. We use attack detection rules that can generate alerts as the evaluation set, including snort rule sets, as well as rules collected from Github. The datasets used in the experiments are described as follows:

- DARPA 2000: It comprises two multistage attack samples, LLDOS 1.0 and LLDOS 2.0, both generated using the same experimental network. In this dataset, there are 51 MSAs out of 750 alerts.
- UNB ISCX IDS 2012 datase. It contains both normal and malicious network traffic activity of seven days. There are four attack scenarios in the entire dataset: infiltrating the network from inside, HTTP denial of service, DDoS, and brute force SSH. This dataset comprises $12,463$ alerts, of which only 107 are MSAs.

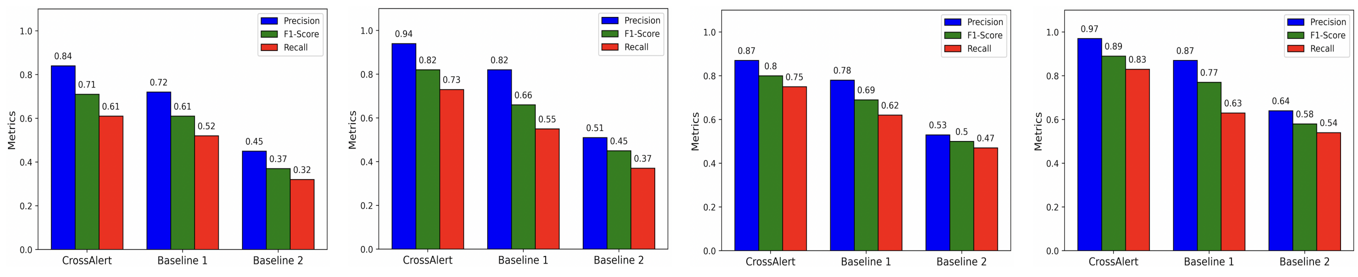
Both datasets are PCAP data and we deploy Snort 2.9 with Emerging Threats rules to generate alerts. For consistency, we assumed four stages for each multi-stage attack according to Table I. We evaluated our methodology in two scenarios: within-dataset and domain shift. For the within-dataset case, we considered two baselines. *Baseline1* is our CrossAlert methodology without incorporating the BERT embedding and potential alert stage. This baseline utilizes only the Basic and Sequnece-based features along with the anomaly score as inputs to the neural network. *Baseline2* employs a framework based on the isolation forest. Here, we determine a threshold using grid search on our training data, and using the obtained anomaly score and this threshold, we classify the alerts. In the case of domain shift, we considered the *Baseline2* along with CrossAlert algorithm in a zero-shot scenario. In this setting, we only train our methodology on the source dataset without seeing any data from target dataset. For all methodologies, we divided our data into training and testing sets. The training data comprises less than 10% of the total data, presenting a significant challenge due to its limited size. To evaluate all models, we utilized three metrics: precision, recall, and F1-score for the MSA label. We repeated each experiment multiple times and reported the average metric value.

A. Effect of Features

Table IV presents a comparative analysis of three different feature sets as input for the isolation forest (*Baseline2*)—Basic features, Sequence-based features, and a combination of Basic and Sequence-based features to perform anomaly detection across two datasets. The analysis clearly indicates that incorporating both Basic and Sequence-based features significantly enhances the performance of anomaly detection models, as evidenced by the higher Precision, Recall, and F1-scores across both datasets. Note that the metric values obtained by the isolation forest are still not acceptable, as the isolation

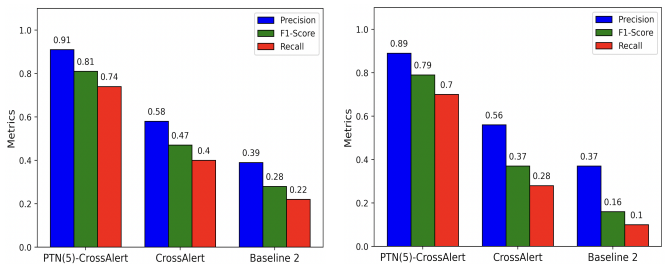
TABLE IV: Summary of features contribution to initial alert ranking in anomaly detection.

Datasets	Basic Features			Sequence-based Features			Basic and Sequence-based Features		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
DARPA 2000	0.41	0.28	0.34	0.51	0.42	0.46	0.64	0.54	0.58
ISCX 2012	0.33	0.15	0.21	0.44	0.22	0.29	0.51	0.37	0.45



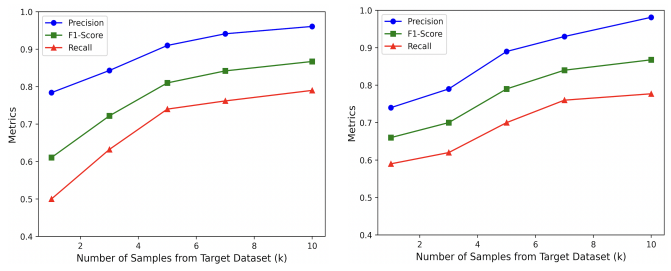
(a) Scenario 1 - ISCX dataset (b) Scenario 2 - ISCX dataset (c) Scenario 1 - DARPA dataset (d) Scenario 2 - DARPA dataset

Fig. 7: Comparing the performance of detection models on different datasets. Scenario 1: training dataset with 10 MSAs and 100 normal labels; Scenario 2: training dataset with 20 MSAs and 200 normal labels.



(a) ISCX to DARPA (b) DARPA to ISCX

Fig. 8: Evaluation of detection models' performance in domain shift problem.



(a) ISCX to DARPA (b) DARPA to ISCX

Fig. 9: Evaluation of PTN(k)-CrossAlert for different values of k for ISCX to DARPA.

forest results in many false predictions. In the following sections, we apply our methodology to address this issue.

B. Results for Within Dataset

We consider two scenarios in different training and testing sets. Scenario 1 in which the training dataset contains 10 MSAs and 100 normal labels, and scenario 2 in which the training dataset contains 20 MSAs and 200 normal labels. We then evaluated CrossAlert, *Baseline1*, and *Baseline2*, across two datasets, in terms of different measurements, as illustrated in Fig. 7. For the DARPA dataset, in the second scenario, CrossAlert achieves the highest precision of 0.97 and F1-Score of 0.83, whereas *Baseline1* and *Baseline2* lag behind significantly. Similar trends are observed with the ISCX dataset, where CrossAlert maintains superior performance with a precision of 0.94 and F1-Score of 0.73. The results demonstrate that CrossAlert provides more reliable and precise detection capabilities compared to the baseline models, indicating its effectiveness in identifying security threats across different environments. Additionally, the results demonstrate that the semantic embedding of the BERT model plays a vital role in our methodology, as evidenced by the low performance of *Baseline1*. Our methodology effectively integrates information from semantic embeddings, Basic and Sequnece-based features, and anomaly scores to make accurate decisions.

C. Results for Domain Shift

In the domain shift scenario, we evaluated our methodology with Prototypical Network, PTN(k)-CrossAlert, where the model can only see k MSA and normal samples from the target dataset in addition to the entire source dataset. We conducted evaluations with ISCX as the source dataset and DARPA as the target dataset (ISCX to DARPA), and vice versa. We also considered CrossAlert in zero-shot learning setting and isolation forest (*Baseline2*) as the baselines.

Fig. 8 displays the comparison of the model in terms of different metrics in the domain shift scenario for both cases, ISCX to DARPA and DARPA to ISCX. Results show that our methodology with the PTN, seeing only 5 samples from the target dataset, outperforms the baselines across all metrics in both domain shift scenarios. The baselines perform poorly due to the domain shift and varying attack types in these two datasets. Our methodology effectively resolves the domain shift by integrating semantic embeddings, Basic and Sequnece-based features, and anomaly scores.

Fig. 9 illustrates the performance of our methodology, PTN(k)-CrossAlert, as the value of k varies, allowing the model to see different numbers of samples from the target dataset. It can be observed that as the number of available samples from the target dataset increases, the metric values improve. With just 10 samples for ISCX to DARPA, we achieve 0.96 for precision and 0.86 for the F1-score. Compared to Fig. 7(c), where we also had 10 samples, these metrics are significantly better. This improvement is attributed to training our model on source data from ISCX, which enabled our model to learn more effectively how to detect multi-stage attacks.

V. Conclusion

In this paper, we propose CrossAlert, a semantic-based method that leverages NLP to extract semantic embeddings from raw alert messages using BERT. These embeddings are integrated with anomaly scores and various extracted features to effectively detect MSAs. By passing the concatenated embeddings through a prototypical network, we provide a reliable IDS. Our evaluations demonstrated that integrating both Basic and Sequence-based features significantly enhances anomaly detection performance. CrossAlert consistently outperforms baseline models across different datasets and scenarios. Its robust performance in domain shift scenarios, even with limited samples from the target dataset, highlights its precision and reliability. As the number of target samples increases, CrossAlert's performance continues to improve, demonstrating its adaptability and scalability in real-world applications.

References

[1] N. Niknami and J. Wu, "Enhancing load balancing by intrusion detection system chain on sdn data plane," in *Proc. of IEEE Conf. on Communications and Network Security (CNS)*, 2022, pp. 264–272.

[2] J. Navarro, A. Deruyver, and P. Parrend, "A systematic survey on multi-step attack detection," *Computers & Security*, vol. 76, pp. 214–249, 2018.

[3] M.-S. Shahryari, L. Mohammad-Khanli, M. Ramezani, L. Farzinvash, M.-R. Feizi-Derakhshi *et al.*, "Nesting circles: An interactive visualization paradigm for network intrusion detection system alerts," *Security and Communication Networks*, vol. 2023, 2023.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[7] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[8] M. Hassanin and N. Moustafa, "A comprehensive overview of large language models (llms) for cyber defences: Opportunities and directions," *arXiv preprint arXiv:2405.14487*, 2024.

[9] M. A. Ferrag, F. Alwahedi, A. Battah, B. Cherif, A. Mechri, and N. Tihanyi, "Generative ai and large language models for cyber security: All insights you need," *arXiv preprint arXiv:2405.12750*, 2024.

[10] D. Chandrasekaran and V. Mago, "Evolution of semantic similarity—a survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–37, 2021.

[11] N. Peinelt, D. Nguyen, and M. Liakata, "tbert: Topic models and bert joining forces for semantic similarity detection," in *Proc. of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 7047–7055.

[12] O. G. Lira, A. Marroquin, and M. A. To, "Harnessing the advanced capabilities of llm for adaptive intrusion detection systems," in *International Conference on Advanced Information Networking and Applications*. Springer, 2024, pp. 453–464.

[13] S. Moothedath, D. Sahabandu, J. Allen, A. Clark, L. Bushnell, W. Lee, and R. Poovendran, "A game-theoretic approach for dynamic information flow tracking to detect multistage advanced persistent threats," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5248–5263, 2020.

[14] R. Bozkır, M. Cicioğlu, A. Çalhan, and C. Toğay, "A new platform for machine-learning-based network traffic classification," *Computer Communications*, vol. 208, pp. 1–14, 2023.

[15] X. Wang, X. Gong, L. Yu, and J. Liu, "Maac: Novel alert correlation method to detect multi-step attack," in *Proc. of the 20th IEEE Intl. Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2021, pp. 726–733.

[16] W. Ma, Y. Hou, M. Jin, and P. Jian, "Anomaly based multi-stage attack detection method," *Plos one*, vol. 19, no. 3, p. e0300821, 2024.

[17] Q. Wang, W. Wang, Y. Wang, J. Ren, and B. Zhang, "Multi-stage network attack detection algorithm based on gaussian mixture hidden markov model and transfer learning," *IEEE Transactions on Automation Science and Engineering*, 2024.

[18] B. Mao, J. Liu, Y. Lai, and M. Sun, "Mif: A multi-step attack scenario reconstruction and attack chains extraction method based on multi-information fusion," *Computer Networks*, vol. 198, p. 108340, 2021.

[19] H. Hu, J. Liu, Y. Zhang, Y. Liu, X. Xu, and J. Tan, "Attack scenario reconstruction approach using attack graph and alert data mining," *Journal of Information Security and Applications*, vol. 54, p. 102522, 2020.

[20] S. Haas and M. Fischer, "Gac: graph-based alert correlation for the detection of distributed multi-step attacks," in *Proc. of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 979–988.

[21] F. Wilkens, F. Ortmann, S. Haas, M. Vallentin, and M. Fischer, "Multi-stage attack detection via kill chain state machines," in *Proc. of the 3rd Workshop on Cyber-Security Arms Race*, 2021, pp. 13–24.

[22] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. on computer vision and pattern recognition*, 2016, pp. 770–778.

[25] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[26] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. of 18th IEEE Int. Conf. on data mining*, 2008, pp. 413–422.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[30] X. Wang, X. Yang, X. Liang, X. Zhang, W. Zhang, and X. Gong, "Combating alert fatigue with alertpro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection," *Computers & Security*, vol. 137, p. 103583, 2024.