Using LLM Embeddings with Similarity Search for Botnet TLS Certificate Detection

Kumar Shashwat
kshashwat@usf.edu
University of South Florida
Tampa, USA

Francis Hahn
fhahn@usf.edu
University of South Florida
Tampa, USA

Stuart Millar
stuart_millar@rapid7.com
Rapid7 LLC
Belfast, UK

Xinming Ou
xou@usf.edu
University of South Florida
Tampa, USA

Abstract

Modern botnets leverage TLS encryption to mask C&C server communications. TLS certificates used by botnets could exhibit subtle characteristics that facilitate detection. In this paper we investigate whether text features from TLS certificates can be represented by open-source and 3rd party vendor LLM text embeddings in a projected vector space, for the purpose of building a classifier to detect botnet certificates. Our method extracts informative features, generating vector representations for effective identification, creating a projected space that can be queried with test certificates via similarity search. Using a balanced dataset consisting of the publicly available SSLBL botnet certificates and TLS certificates used by popular websites, our evaluations show that C-BERT, an open-source model, emerges as the preferred choice within our proposed system rather than a vendor solution. C-BERT achieves a competitive F1 score of 0.994 on unseen test data, 97.9% accuracy on data gathered several months after an initial projected embedding space was created, and maintains performance in a simulated zero-day evaluation against four C&C groups, with an average F1 score of 0.946. Further evaluation on a random sample of 150,000 real-world certificates collected from a full internet scan between Jan 2024 to May 2024 predicts 13 potential botnet certificates, among which one was confirmed to be malicious by VirusTotal. Comparing with the scenario where no such tool exists, we randomly selected 1,300 certificates from these 150,000 certificates and ran them through VirusTotal, and *none* were confirmed to be malicious. This translates to 100 fold effort reduction in identifying botnet certificates in the wild.

CCS Concepts

• **Security and privacy** → **Intrusion detection systems**; • **Information systems** → **Language models**; **Top-k retrieval in databases**; • **Computing methodologies** → **Artificial intelligence**; **Neural networks**.

Keywords

Botnet classification, LLMs, Vector Embeddings, Similarity Search, Vector Databases, Nearest Neighbours, Clustering, kNN

ACM Reference Format:
Kumar Shashwat, Francis Hahn, Stuart Millar, and Xinming Ou. 2024. Using LLM Embeddings with Similarity Search for Botnet TLS Certificate Detection. In *Proceedings of the 2024 Workshop on Artificial Intelligence and Security (AISec '24), October 14–18, 2024, Salt Lake City, UT, USA.*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3689932.3694766

1 Introduction

A botnet is a collection of internet-connected devices infected with malware and covertly controlled by a malicious actor [15]. Compromised devices, known as bots, can be anything from traditional computers to smartphones and even Internet of Things devices. The bot communicates with a command and control (C&C) server, allowing the attacker to remotely issue commands and orchestrate large-scale attacks [7]. Botnet detection remains a critical area of cybersecurity research, as evidenced by the plethora of studies documented in various publications [15, 36]. Two main paradigms dominate detection techniques: signature-based, offering minimal false positives whilst struggling with novel threats, and anomaly-based, leveraging artificial intelligence (AI) and machine learning (ML) for broader detection however with a higher false positive rate. This trade-off highlights the importance of both approaches, with signature-based methods providing a safety net while anomaly-based techniques identifying previously unseen botnet variants, crucial in the fight against ever-evolving cyber threats.

Communication between bots and the C&C server frequently leverages Transport Layer Security (TLS) protocol for encryption [15, 36], rendering communication content indecipherable by network administrators using traditional traffic monitoring techniques. The TLS protocol requires the server to present an x.509 certificate to prove its identity, and we hypothesize botnet certificates may exhibit distinct characteristics compared to legitimate certificates. This motivates us to explore the potential of certificate analysis for botnet detection, an under-researched area in prior literature. With the sheer volume of TLS certificates, it is not feasible for security analysts to manually verify each individually. Our work

AISec '24, October 14–18, 2024, Salt Lake City, UT, USA.

ACM ISBN 979-8-4007-1228-9/24/10
https://doi.org/10.1145/3689932.3694766

aims to help solve this problem with an AI/ML based malicious certificate detection system. While previous work [32] relied on neural network training, in contrast this paper proposes a novel, lightweight method, eliminating the need for training a model from scratch by instead using pre-trained Large Language Model (LLM) embeddings. This approach further leverages a vector database for efficient similarity search during the prediction process. A test TLS certificate, the subject name and issuer name are first extracted then combined to generate a vector embedding which is queried against the vector index. The k-nearest neighbors are retrieved with a voting mechanism then being employed to classify the test certificate as malicious or benign.

While LLMs are experiencing rapid adoption across various industries due to the impressive capabilities [37], cybersecurity presents particular challenges regarding production deployment complexities and resource dependencies. This motivates another aspect of our work to investigate the efficacy of both open-source LLMs and commercial offerings from vendors like OpenAI or AWS. Furthermore, the dearth of publicly available real-world datasets has hindered AI/ML research somewhat pertaining to botnet certificate detection. In contrast, we evaluate against a real-world evaluation alongside a zero-day botnet detection scenario.

Our contributions are:

- A novel approach for botnet certificate detection using LLM vector embeddings coupled with efficient vector search techniques. Our innovative methodology presents a significant departure from traditional methods, with results demonstrating effective detection of botnet certificates.
- A rigorous evaluation assessing the potential of different open-source and vendor LLM embedding models by comparing their performance.
- Validation of the real-world applicability of our proposed approach through an evaluation on TLS certificates taken from the wild. The evaluation provides compelling evidence on our method's effectiveness in reducing human efforts for identifying botnet C&C servers under realistic operational constraints.

This paper is organized as follows: Section 2 discusses related work, Section 3 explains the methodology and Section 4 outlines the experimental setup. Detailed results are in Section 5 with Section 6 containing conclusions and future work.

2 Related Work

Torroledo et al. [32] used deep neural networks for classifying TLS certificates used by malware. The authors created a web crawler for certificate collection and feature engineering, and used this data to train a neural network model to classify whether or not a given certificate is used to facilitate malware activities. Our work differs because we leverage embeddings in pre-trained large language models combined with kNN to build a classifier. We also performed evaluation on TLS certificates in the wild, in addition to curated data sets with known ground truths.

Theofanous et. all [31] created a dataset by actively probing TLS servers and collect a large number of features from the TLS handshake meta data, including certificates. The authors investigated using machine learning on these features to classify a server into benign and botnet C&C servers. Our work focuses on TLS certificates and use LLM embeddings as latent representations to facilitate machine learning based classification. The dataset created from the authors' work could benefit our work by providing additional sources of botnet C&C TLS certificates.

Researchers have studied TLS certificates used by "Booter" websites for selling DDoS services [22], and those used by phishing websites [18]. In these studies the authors found that TLS certificates used for these malicious activities present distinct characteristics. This is consistent with our assumption on which this work is based on.

Botnet detection has been a long standing problem [38]. BotHunter [17] leveraged the evidence trails left by botnets in network traffic. Another approach [30] used deep neural networks and time-based network data utilized LTSM and RNN neural networks. Our work provides another source of indicators for detecting botnet activities, that from the TLS certificates used by botnet C&C communications.

3 Methodology

Based on our analysis of both malicious and benign certificates, we observed that malicious actors tend not to invest significant effort in obfuscating their certificates, resulting in features that differ notably from those of benign certificates. As shown in Table 1, malicious certificates often contain randomly generated characters, while benign certificates exhibit carefully selected and structured features.

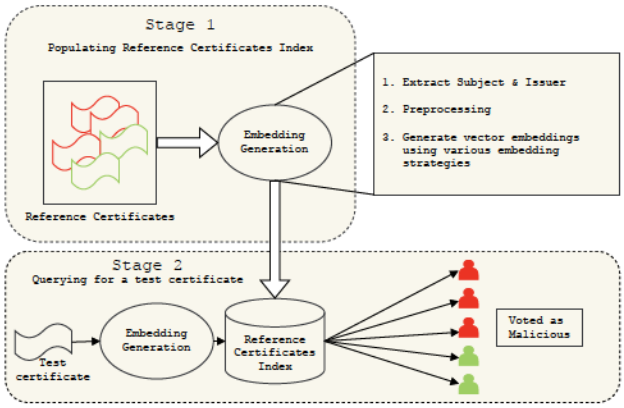

Figure 1: Approch Overview

Raw TLS certificate string features are first preprocessed to generate a set of information-rich vector embeddings, one per certificate. These vectors are projected to create a embedding space, stored in a vector database, which is queried to predict whether or not new certificates are malicious or not. Figure 1 presents the overall pipeline for using our proposed system in operation. The first step (Stage 1) involves populating a vector index of embeddings computed from known malicious and benign TLS certificates (called reference certificates in the figure). Once the index is populated, a new test certificate that is not part of the reference certificates can be queried against the embedding space already in the index to

Table 1: Example of a malicious certificate and a benign certificate

Malicious	Field	Name	Country	Organization	Organization Unit	Location	State	Email
yes	Subject	192.236.160.249	mn	xxxyz	zaaaabb	sttuvwww	noopqrrrrs	ccdde@192.236.160.249
yes	Subject	192.236.160.249	mn	xxxyz	zaaaabb	sttuvwww	noopqrrrrs	ccdde@192.236.160.249
no	Subject	www.agl.com.au	AU	AGL Energy Limited	NA	Docklands	Victoria	NA
no	Issuer	DigiCert TLS RSA SHA256 2020 CA1	US	Digicert Inc	NA	NA	NA	NA

be classified as malcious or benign. The same preprocessing steps used to generate the existing embedding space is used on the test certificate to generate the corresponding embedding, which is then queryed against the embedding space in the index to find the k-nearest neighbors. A majority voting scheme based on the labels of the k-nearest certificates then classifies the new certificate as either malicious or benign. If a majority of the k-nearest neighbors are classified as malicious, the new test certificate is classified as malicious as well, otherwise it is classified as bengin.

We take care to conduct evaluations not only to determine the optimal setup for malicious certificate detection in the general case, but in the most challenging scenarios of 1) handling brand new certificates gathered later than those used to create the embedding space, 2) a zero-day setting to simulate detection performance on emerging C&C groups from organized clusters of threat actors that use specific C&C infrastructure and 3) an evaluation against 150,000 certificates gathered in the wild. Further, not only are we interested in the discriminative power of alternative embedding representations, but this work makes considerations related to the important real-world application of our approach in a production environment, including 3rd party dependencies, open-source, information security, inference time, and cost. With this in mind multiple LLM embedding models are evaluated with different embedding strategies, and we leverage FAISS [19], an in-memory vector store, to efficiently to store, search, and retrieve the vectors that make up the embedding space.

3.1 TLS Certificate Preprocessing

Each TLS certificate contains a subject and an issuer, previously shown to be useful for botnet detection [32]. A TLS certificate's subject is who it belongs to, such as a domain, and the issuer is the trusted authority that signed it. A subject can take the form *www.agl.com.au, O=AGL Energy Limited, L=Docklands, ST=Victoria, C=AU*, and an issuer similarly constructed as *DigiCert TLS RSA SHA256 2020 CA1, O=DigiCert Inc, C=US*. While the whole subject and issuer text strings can be used in full to create vector embeddings, we wish to investigate whether or not separate embedddings for individual attributes are of value. Per Table 1, our preprocessing steps parse the subject and issuer fields for a given certificate to yield the following:

- Name
- Country
- Oraganization
- Oraganization unit

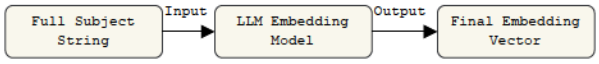

Figure 2: Embedding Strategy 1

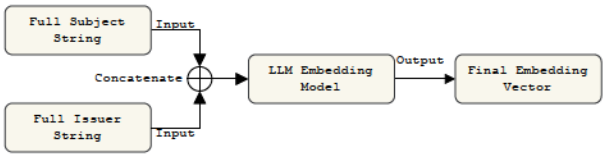

Figure 3: Embedding Strategy 2

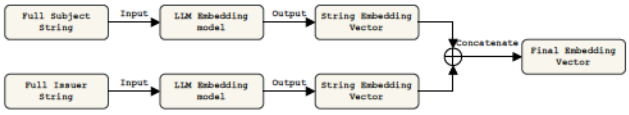

Figure 4: Embedding Strategy 3

- Location
- State
- Email

Since SSLBL offers only the 7 aforementioned features for malicious TLS certificates, our study was limited to analyzing these specific fields. Any missing fields within a certificate were imputed with the string NA to ensure consistent data representation during subsequent embedding steps. Once the full text subject and issuer, plus the individual attributes, are extracted as strings, the next step is to embed these strings to create numerical vector representations.

3.2 TLS Certificate Embedding Strategies

Learning semantic relationships through vector embeddings has transformed Natural Language Processing (NLP). Typically words or documents are mapped into high-dimensional vector spaces where data points located near each other represent semantically related concepts. This is achieved by learning a metric that captures these relationships, and the embeddings enable powerful mathematical operations on the data, such as similarity search, facilitating various NLP tasks.

Word2Vec [23], GloVe [26], and FastText [10] were seminal vector embedding methods that have been instrumental for years.

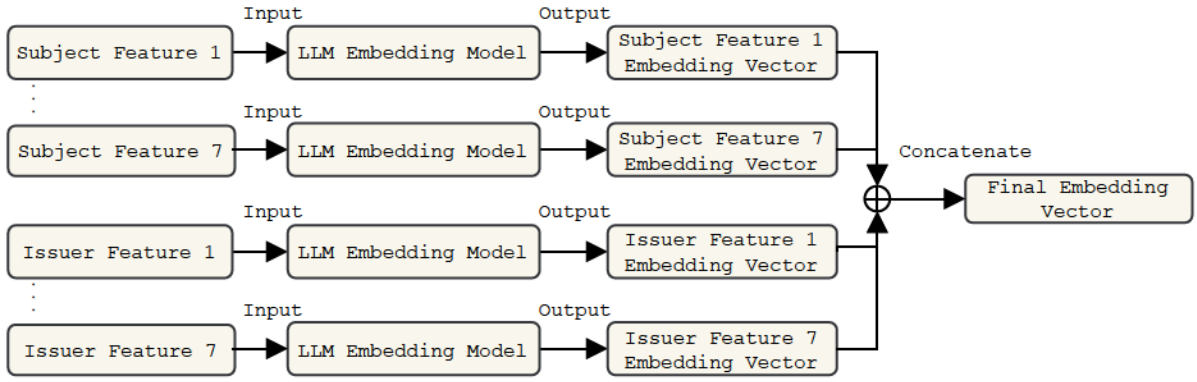

Figure 5: Embedding Strategy 4

However, recent advancements in generative pre-trained LLMs have led to the emergence of new embedding techniques. These pre-trained models can accept entire strings as input and return a fixed-length vector representation, capturing contextual information beyond individual words. As shown in Figures 2, 3, 4 and 5, after first preprocessing each certificate, the text features are then used to generate one or more feature embeddings, which are then concatenated into a final embedding vector. These final embeddings represent the certificates in a high-dimensional space, where similar certificates are in closer proximity and dissimilar certificates further apart. This final embedding creation process involves different embedding strategies to assess their impact on botnet detection performance. We explain them in turn as follows, and use n to denote the number of individual embeddings generated from a single certificate in the pre-processing stage.

- *Embedding Strategy 1*: Subject string only - here only the Subject string is used to generate the embedding so $n = 1$. The time taken to generate the embedding can be expressed as

  $$t_{cert} = t_{embed} \times n = t_{embed}$$

  where t_{embed} is the individual embedding generation time. The length of this final certificate embedding on disk is $u = l \times n = l$, with l being the length of an embedding output from a given model per Table 2.
- *Embedding Strategy 2*: Subject string and Issuer string concatenated - we first concatenate the Subject and Issuer strings to form a unified input string before embedding generation. We have $n = 1$ and the time taken to generate the embedding remains $t_{cert} = t_{embed}$. The resultant embedding vector remains a length of $u = l$.
- *Embedding Strategy 3*: Subject string and Issuer string embedded separately - embeddings are generated separately using the Subject and Issuer strings. The resulting two embeddings are then concatenated to create a unified single embedding. Here $n = 2$ and the resulting embedding vector has a dimensionality of $u = 2l$. The time taken for generating this composite embedding is $t_{cert} = 2t_{embed}$.
- *Embedding Strategy 4*: Individual features embedded separately - following the feature extraction procedures outlined in Section 3.1, we generate embeddings for each extracted feature. These embeddings are then concatenated to create a unified single embedding. The total number of features being $n = 14$, the resulting embedding vector has a length of $u = 14l$. As the embedding generation process is repeated for each feature, the total time taken to generate the final embedding vector is $t_{cert} = 14t_{embed}$, assuming no parallel processing.

In the rest of the paper, for brevity we refer to the Embedding Strategy as E, with the integers 1 to 4 representing the four strategies respectively. When running each of these embedding strategies, the choice of LLM affects the size and content of each embedding vector, discussed in the next subsection.

3.3 Generative LLM Embedding Model Selection

LLM embedding models convert discrete words or tokens, including characters or phrases, into high-dimensional numerical vectors of floating point numbers. These vectors, called embeddings, capture the semantic meaning and relationships between words in a continuous space. For our study, we selected a diverse set of embedding models as shown in Table 2, with a well-established baseline, BERT [13]. This allows for a comprehensive evaluation of the impact of different embedding techniques on our task. Different embedding models are trained on different text corpora with varying length l of their generated output embedding vectors. More data for training the model may improve performance and longer vectors are more discriminative due their higher dimensionality [16]. A summary of the embedding models used in this work can be found in Table 2.

- BERT [13]: BERT serves as our baseline model due to its widespread adoption and open-source nature. This choice facilitates the reproducibility of our research and allows comparison with other embedding models.

Table 2: Summary of vector embedding models used

Model	Year	l
BERT [13]	2018	768
C-BERT [11]	2020	768
Titan [29]	2023	1536
Titan 2 [29]	2024	1024
Cohere [33]	2023	1024
OpenAI [14]	2024	3072
VoyageAI [3]	2024	1024

- C-BERT [11]: We specifically incorporate C-BERT due to its character-level processing. This approach is particularly advantageous for capturing semantic relationships within single words, which may be effective for our task as subjects and issuers are often short and potentially ambiguous entities.
- OpenAI `text-embedding-3-large` [14]: OpenAI's `text-embedding-3-large` model is used extensively within the research community.
- AWS `titan-embed-text:v1:0` [29]: Amazon's Titan embedding models are widely used within the AWS ecosystem. Titan v1.0 serves as the default embedding model.
- AWS `titan-embed-text:v2:0` [29]: Amazon's Titan 2, released in May 2024, with the intent of giving improved performance compared to its predecessor.
- Cohere `embed-english-v3.0` [33]: Cohere's `embed-english-v3.0` model is another regular choice within the research community. Its inclusion allows for a broader comparison across various embedding models.
- Voyage AI `Voyage-large-2-instruct` [3]: At the time of conducting this work, `voyage-large-2-instruct` held the top position on the MTEB leaderboard [24]. This performance record motivated its inclusion within our model selection.

Generated vectors are stored in an in-memory FAISS [19] vector database. Vector databases are designed for highly efficient retrieval and comparison of vectors. Using a vector database to store and query certificates based on their embedded representations is advantageous to using other methods such as iterating over a lengthy CSV files. To create the FAISS index, the generated certificate embeddings are fed into the FAISS library, with FAISS data structures facilitating efficient nearest neighbor search within the embedding space.

3.4 Classifying an Unseen TLS Certificate using kNN and FAISS

Our approach leverages vector similarity search for classifying a previously unseen TLS certificate as malicious or benign. We considered three similarity metrics: cosine similarity, Euclidean distance, and dot product. Let $A = (A_1, \ldots, A_n)$ and $B = (B_1, \ldots, B_n)$ be two vectors between which the metrics are being calculated.

$$cosine\_similarity = \frac{A \cdot B}{|A||B|} \tag{1}$$

Cosine similarity excels at capturing directional alignment, making it suitable for tasks where relative orientation matters more than absolute magnitude. However, it can be insensitive to magnitude differences.

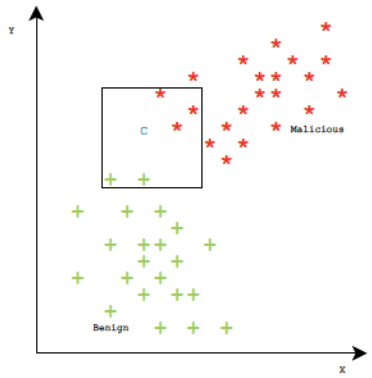

Figure 6: Certificate classified as malicious due to majority malicious neighbors.

$$euclidean\_distance = \sqrt{\sum_{i=1}^{N}(A_i - B_i)^2} \tag{2}$$

Euclidean distance provides a direct measure of distance in the vector space, but can be swayed by varying vector lengths.

$$dot\_product = A \cdot B \tag{3}$$

Dot product, while computationally efficient, inherits limitations from both cosine similarity and Euclidean distance - sensitive to both direction and magnitude without normalization.

Later in our experiments we conduct an ablation study between these three metrics in order to choose the one that appears most performant for our task.

Our choice of FAISS [19] for vector indexing and retrieval stems from its suite of optimized indexing algorithms. FAISS implements a variety of indexing techniques that significantly accelerates search times compared to brute-force approaches. This efficiency is attractive for our work dealing with datasets at several orders of magnitude in size. FAISS also demonstrates excellent scalability, allowing for efficient operations on collections that grow in size over time. This characteristic ensures our solution remains performant as our dataset expands, and likely would be of benefit in a production setting. Given a test certificate, its embedding is generated using the same embedding model employed for the reference certificates. This test certificate embedding is then used to query the pre-built FAISS index.

FAISS retrieves the k closest neighbors, that is, the k certificate embeddings most similar to the test certificate embedding from the index. The value of k represents a parameter that controls how many nearest neighbors are selected and returned. A larger k value encompasses a broader neighborhood for comparison, potentially improving robustness, but may also increase computational cost. Finally, a classification decision is made based on the majority vote from the retrieved nearest neighbors.

Figure 6 illustrates a test certificate C being queried against known malicious and benign certificates projected in a 2D embedding space, with the bounding box representing the voting process. In practice though, per Table 2, each embedding is several order of magnitudes larger than 2 (or 3 if one opts to use 3D rendering), so visualisation is not possible. But the underlying vector similarity principles are the same. If more than half, i.e., $k/2$, of the k nearest neighbors belong to the malicious class, the test certificate is classified as malicious. Otherwise it is classified as benign. Formally:

$$\text{Malicious}(C) \Leftrightarrow |\mathcal{N}_k^C \cap \mathcal{M}| > \frac{|\mathcal{N}_k^C|}{2} \tag{4}$$

where Malicious(C) represents a function that outputs `True` if the test certificate C is deemed malicious and `False` otherwise. $\mathcal{N}_k^C$ represents the set of the k-nearest neighbors of certificate C in the vector space, with $\mathcal{M}$ representing all the malicious certificates in the dataset and the intersection of both is the number of malicious certificates returned.

4 Experimental Setup

We conduct our experiments in multiple progressive stages, with the early experiments being ablation studies. To begin with each of the embedding strategies are assessed to ascertain the most performant, followed by selecting the ideal distance metric. With these two aspects of the configuration fixed, we then vary the embedding model itself, per the previous selected list in Section 3.3. In this way we can see the performance of the open source compared to closed source embedding models using the same selected optimal embedding strategy and distance metric. After this we chose the best performing open source model, and the best performing closed source model to evaluate against each other on held-out test data. The impact of varying k in the voting system is investigated plus importantly in the final stages we conduct three important experiments. The first ascertains the performance of the system with new certificate data gathered after that used to conduct the earlier ablation studies. The second subjects the system to a challenging zero-day scenario of detecting TLS certs from emerging C&C groups [4] by removing them from the dataset when creating the initial embedding space and then only using those same removed C&C certs for testing. Thirdly we evaluate our approach on TLS certificates crawled from the internet to examine its utility in real-world operations.

4.1 Datasets

A collection of malicious botnet certificates was obtained from the SSL Blacklist (SSLBL) [2], a publicly available benchmark frequently used in previous published works [21, 31, 32]. SSLBL is a project designed to identify and blacklist TLS certificates associated with botnet command and control (C&C) servers, thereby enabling research into the detection of malicious connections. We curate two datasets from SSLBL at different points in time to give more confidence in our evaluations. The first dataset contains 2,516 certificates gathered between 2014-05-04 and 2024-01-11, and the second dataset contains a further 149 certificates gathered between 2024-01-11 and 2024-06-03. Both datasets also include the C&C group attributed to each certificate. We balance the first dataset by adding 3,000 benign certificates randomly selected from the Alexa Top 1 Million list [6], a publicly available ranking of the most visited websites. We assume these websites are not malicious. Similarly we balance the second dataset with 150 randomly selected certificates from Alexa Top 1 Million list. There is no overlap in the certifcates between Dataset One and Dataset Two.

Dataset One is used in our ablation studies to select the optimal configuration for embedding strategy, distance metric, and embedding model. We then perform an evaluation in Section 5.5 holding this optimal configuration constant and testing it using Dataset Two as held-out and unseen test data, where the botnet certificates were collected later than those in Dataset One. The botnet certificates' statistics in the two datasets are summarised in Table 3, and 4. To be concise only the 10 largest attributed C&C groups are listed.

Table 3: Top 10 C&C group certs in Dataset One (malicious certs collected between 2014-05-04 and 2024-01-11 from SSLBL)

Attribution	# Count
AsyncRAT	495
Dridex	358
Gozi	174
Quakbot	145
Malware	142
BitRAT	141
TorrentLocker	135
KINS	120
Gootkit	116
QuasarRAT	98

Table 4: Top 10 C&C group certs in Dataset Two (malicious certs collected between 2024-01-11 and 2024-06-03 from SSLBL)

Attribution	# Count
AsyncRAT	58
QuasarRAT	42
PureLogStealer	24
OrcusRAT	7
DCRat	4
Latrodectus	4
VenomRAT	3
AgentTesla	2
Rhadamanthys	2
RedLineStealer	1
Malware	1
njrat	1

4.2 Data Partitioning and Cross-Validation

The first dataset gathered between 2014-05-04 and 2024-01-11 is divided into training, validation, and testing sets. Note in broader AI/ML model sense, the training step is synonymous with us creating the embedding space against which to evaluate other certificates, rather than, say, training a neural network model itself; for brevity we still refer to the portion of data used to create the embedding space as the training data. Firstly a stratified 10% holdout was created for the testing set, ensuring the test data distribution reflects

the overall dataset. The test dataset remains untouched for subsequent performance comparisons of various models as detailed in Section 5.4. The remaining 90% of the data is used in the ablation studies where, to mitigate overfitting and give more confidence in generalization potential of our approach, stratified 5-fold cross-validation[12] is performed. Each fold comprises 80% of the dataset for training and 10% for validation, and the reported results in the ablation experiments in Section 5.1, 5.2 and 5.3 are the average across the five validation splits. To further evaluate the system's ability to predict future malicious certificates, Dataset 2 with botnet certificates gathered between Jan 12th 2024 and June 3rd 2024 is used entirely as test data.

Pendlebury et al. [25] pointed out the importance of avoiding temporal bias in constructing train/test data splits for machine learning based security research. For evaluation results to reflect the machine learning model's true utility in practice, testing data shall be from later times than training data. Our Dataset 2 is for that purpose; the evaluation results on it (Section 5.5) are intended to examine our approach's effectiveness when the train/test partition follows realistic temporal orders.

4.3 Evaluation Metrics

We frame TLS certificate detection as a supervised binary classification problem. Each certificate is labelled as either malicious, which is the positive class, or benign, which is the negative class. A correct identification of a malicious certificate results in a True Positive (TP), while a correct identification of a benign certificate resulted in a True Negative (TN). A False Positive (FP) occurs when a benign certificate is mistakenly predicted as malicious, and a False Negative occurs when a malicious certificate is missed and incorrectly predicted as benign.

Hence the focus should be on minimizing the false positive rate (FPR) while concurrently monitoring the miss rate (MR), also known as false negative rate. Achieving a complete absence of FPs is impractical in real-world applications [5]. To evaluate performance, the metrics accuracy, precision, recall, F1-score, FPR, and MR are formally defined as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$precision = \frac{TP}{TP + FP} \tag{6}$$

$$recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{8}$$

$$false\ positive\ rate = \frac{FP}{FP + TN} \tag{9}$$

$$miss\ rate = \frac{FN}{TP + FN} = 1 - recall \tag{10}$$

Since miss rate can be derived from recall we do not separately report it in the experiment results.

5 Results

5.1 Embedding Strategy Ablation Study

Our first experiment aims to identify the most effective embedding strategy from the four options explained previously in Section 3. We hold the embedding model constant as `OpenAI`, $k = 5$ and the distance metric as `cosine`. Then we vary each embedding strategy E from 1 to 4, creating the vector embedding space from the training split of the first dataset, and evaluating with the validation split. This allows us to monitor any performance changes that occur as a direct result of changing the embedding strategy. Results in Table 5 show employing only the subject as an encoded feature, where $E = 1$, yields significantly lower performance compared to other strategies. Consequently, this approach can be excluded from further consideration. Considering the F1 score, $E = 3$ emerges as the clear winner, achieving an accuracy of 0.994, precision of 0.989, recall of 0.998, and F1 score of 0.994. It could be said this is expected, as embedding the subject and issuer strings separately preserves the signal in both. For $E = 4$, where every attribute of the subject and issuer are first embedded and then concatenated, performance is very good, however when compared to $E = 3$, $E = 4$ uses all possible $n = 14$ features, and hence creates fourteen feature vectors, seven from the subject and seven from the issuer that are then concatenated. Per previous in Section 3.2, this requires 7x the generation time and 7x disk space of $14t_{embed}$ and $14l$ respectively, compared to the concatenation of only two feature vectors with $2t_{embed}$ and $2l$ for $E = 3$. This further confirms $E = 3$ as our preferred strategy, where the full subject and issuer strings are embedded separately and then concatenated. Thus $E = 3$ is fixed for the remainder of the experiments.

Table 5: Performance across the four embedding strategies

E	Acc	Prec	Recall	F1	FP %
1	0.852	0.817	0.869	0.842	16.267
2	0.965	0.986	0.936	0.960	1.133
3	**0.994**	**0.989**	**0.998**	**0.994**	**0.933**
4	0.989	0.984	0.992	0.988	1.333

5.2 Distance Metric Ablation Study

Building upon the previous experiment where $E = 3$ emerged as the best choice, we next focus on identifying the optimal similarity metric for the classification task. Per previous Section 3, `cosine`, `dot product` and `Euclidean` are under scrutiny. We hold the embedding model constant as `OpenAI`, $k = 5$ and $E = 3$ whilst varying the distance metric. We also report the inference time t, measured in milliseconds, to predict a single certificate Results in Table 6 show the `cosine` metric achieves the highest F1 score of 0.994 with a marginally longer inference time than `dot product` and `Euclidean`. While a trade-off exists between inference speed and detection performance, the significant improvement in F1 score justifies fixing the use of `cosine` in the remaining experiments.

5.3 Embedding Model Ablation Study

In the previous two experiments the embedding model was held constant as `OpenAI`. Having selected which E and distance metric

Table 6: Performance across the three distance metrics

Distance Metric	t (ms)	Acc	Prec	Recall	F1	FP %
cosine	**0.177**	**0.994**	**0.989**	**0.998**	**0.994**	**0.933**
dot product	0.124	0.960	0.952	0.961	0.956	4.067
Euclidean	0.114	0.977	0.988	0.961	0.974	1.000

Table 7: Performance across the seven embedding models

Model	t (ms)	Acc	Prec	Recall	F1	FP %	OSS
OpenAI	**0.177**	**0.994**	**0.989**	**0.998**	**0.994**	**0.933**	**no**
Titan	0.078	0.990	0.986	0.993	0.989	1.200	no
Cohere	0.037	0.987	0.985	0.987	0. 986	1.267	no
Titan 2	0.046	0.987	0.984	0.987	0.986	1.333	no
Voyage	0.068	0.960	0.985	0.926	0.954	1.200	no
BERT	0.029	0.958	0.968	0.939	0.953	2.600	yes
C-BERT	**0.064**	**0.983**	**0.992**	**0.970**	**0.981**	**0.667**	**yes**

to use, here we investigate how performance varies across our chosen set of embedding models which are a mix of open and closed source. We hypothesize that for some tasks open source models may generate embedding vectors which could be as discriminative as those from 3rd party closed-source offerings, without many of the dependencies, costs, and security considerations that vendor-based solutions introduce. Hence we are curious to see if this is the case in our work in detecting botnet certificates. Holding $E = 3$, the distance metric as `cosine`, and $k = 5$, we vary the choice of embedding model across `OpenAI[14]`, `Titan[29]`, `Titan2[29]`, `Cohere[33]`, `C-BERT[11]`, `BERT[13]` and `VoyageAI[3]`.

Results in Table 7 (OSS column indicates open source or not) show C-BERT emerges as a strong open-source contender, achieving a competitive F1 score of 0.981 that is comparable to OpenAI which had an F1 score of 0.994. However notably, C-BERT's inference time t in predicting a certificate is 0.064ms, which is almost 64% faster than OpenAI's 0.177 ms. It can also be said that, unlike OpenAI, Titan, Titan 2, Cohere, and Voyage, C-BERT is open-source, reducing external dependencies, cost, and potential security risks associated with third-party API egress. Having identified this suitable open-source alternative, C-BERT, with comparable performance to OpenAI, we proceed to the evaluation phase. The next step involves a head-to-head comparison of C-BERT and OpenAI using the held-out test data.

5.4 Held-out Unseen Test Data Evaluation

As a reminder, per Section 4.2, at the outset 10% percent of the first dataset was reserved for this unseen test data evaluation. Taking both the OpenAI and C-BERT models, with $E = 3$, distance metric `cosine` and $k = 5$, we test both models with this held out 10%. Results in Table 8 demonstrate C-BERT exhibits strong competitive performance, slightly surpassing OpenAI with an F1 score of 0.994 compared to 0.990. C-BERT additionally achieves a FP rate of 0.397% which is less than half that of OpenAI's and a miss rate of 0.667% that is lower than OpenAI's 1%. Further, it can be seen C-BERT achieves this performance level with a significantly faster inference time t of 0.021 ms compared to OpenAI's 0.088 ms. Two primary factors can contribute to the observed running time difference. Firstly, C-BERT operates entirely on local hardware, while the OpenAI embedding model relies on an API for access, introducing potential network latency. Secondly, the final vector length of the C-BERT embedding output, 1,536, is smaller than that of the OpenAI model, 6,144, giving C-BERT a more favourable t. It is also likely that C-BERT's design for character-level operations contributes to its competitiveness in our use case that involves relatively short strings.

Table 8: Performance on held-out test data

Model	t (ms)	Acc	Prec	Recall	F1	FP %
OpenAI	0.088	0.991	0.988	0.992	0.990	1.000
C-BERT	**0.021**	**0.995**	**0.992**	**0.996**	**0.994**	**0.667**

5.5 Simulated Future TLS Certificate Detection Evaluation

The previous experiments have demonstrated the classification capability of our approach with strong performance in Section 5.4 using the held-out 10% test split of the first dataset. Recall this first dataset included malicious certificates identified on SSLBL [2] up to and including January 11th, 2024. A sterner test is to evaluate model performance on what we describe as unseen data that is collected later in time, accounted for in our methodology by curating a second dataset of certificates newly posted on SSLBL between January 12th, 2024 and June 3rd, 2024, comprising 149 certificates. In theory, if we had deployed our system in January 2024 with a vector embedding space using data up until that point in time, the objective now is to measure the simulated production performance between January 2024 and June 2024 on these 149 malicious certificates. With $E = 3$, $k = 5$ and distance metric `cosine` held constant, both C-BERT and OpenAI were again evaluated with this new data. The C-BERT model achieved an F1 score of 0.979, compared to OpenAI's 0.960. This further shows the potential advantage of the open-source C-BERT model in a production setting, compared to OpenAI vendor solution.

Table 9: Performance on future TLS certificates

Model	t (ms)	Acc	Prec	Recall	F1	FP %
OpenAI	0.032	0.959	0.966	0.953	0.960	3.401
C-BERT	**0.023**	**0.979**	**0.973**	**0.986**	**0.979**	**2.649**

5.6 Simulated Zero-day C&C Group Evaluation

With C&C groups emerging over time, a botnet certificate detection system should ideally still be able to identify malicious certs belonging to emerging, brand new C&C groups, akin to zero-day detection in a malware scenario. The SSLBL public benchmark dataset used in this work is curated in such a manner that each certificate comes with the responsible attributed C&C group. In this way, it becomes possible to remove a given C&C group from the setup of the vector embedding space, and then test using that same removed C&C group. To evaluate the performance of C-BERT model embeddings in this manner, we hold $E = 3$, $k = 5$ with the `cosine` distance metric per previous, while conducting a targeted

Table 10: Simulated zero-day C&C group evaluation

C&C Family	# Certs	t (ms)	Acc	Prec	Recall	F1	FP* %
Vawtrak	13	0.034	0.900	0.867	0.929	0.897	12.500
Corebot	10	0.042	1.000	1.000	1.000	1.000	0.000
VenomRAT	9	0.021	0.950	0.900	1.000	0.947	9.091
VMZeuS	9	0.033	0.900	0.900	0.900	0.900	10.000
Average	10.25	0.032	0.937	0.917	0.957	0.936	7.898

* High false positive rate is attributed to the limited sample size.

Table 11: Comparison of $k = 5$ and $k = 1$

Model	k	t (ms)	Acc	Prec	Recall	F1	FP %
C-BERT	**1**	**0.022**	**0.996**	**0.996**	**0.996**	**0.996**	**0.333**
C-BERT	5	0.021	0.995	0.992	0.996	0.994	0.667
OpenAI	1	0.093	0.995	0.992	0.996	0.994	0.667
OpenAI	5	0.088	0.991	0.988	0.992	0.990	1.00

experiment holding out four C&C groups for testing - Vawtrak MITM [20], Corebot C&C [27], VenomRAT C&C [35], and VMZeuS C&C [9]. These C&C groups are entirely withheld from the setup of the vector embedding space, which utilizes all the other remaining certificates. This results in four new, balanced zero-day test datasets containing each of the new C&C group certificates along with the same number of benign certificates. In total the four testing sets contain 41 botnet certificates and 41 benign certificates.

Results in Table 10 show excellent detection performance with an average F1 score of 0.936 across the four withheld families, a promising result given the challenging nature of the experiment to try to correctly identify certificates from C&C groups that are not present in the original vector embedding space. This suggests our approach using C-BERT may effectively generalize to unseen C&C families in the wild as they emerge. Due to the limited sample size, the measured performance metrics are much more coarse-grained than in the other experiments. For example, for a test set with 10 benign certificates, a single false positive result will yield a 10% false positive rate. This explains why the false positive rates are much higher compared to the other data sets.

5.7 Comparison between $k = 5$ and $k = 1$

The voting mechanism in our experiments uses $k = 5$, with the certificate prediction of malicious or benign depending on the majority label of the five nearest neighbours. For this experiment, we determine any change in performance for a more difficult setup where the similarity search and retrieval can only operate with $k = 1$, in that the prediction will match the label of the closest certificate in the existing embedding space compared to the projected location of a new certificate. To do so, we evaluate the performance of C-BERT and OpenAI with $k = 1$, again forcing the similarity search to select the single closest embedding in the high dimensional embedding space. Results in Table 11 show C-BERT achieves competitive performance with an F1 score of 0.996, even when $k = 1$, compared to OpenAI's 0.994.

5.8 In the Wild - Real World Data Evaluation

Curated datasets used for evaluating machine learning approaches for cyber security are inevitably biased. This is due to the fact that ground truths are hard to obtain for data related to security tasks. To create datasets with ground truths, one often has to compromise on data distribution being representative of real world. Manually examining a TLS certificate and the IP address where it was retrieved to determine its maliciousness is doable for a few cases, but is infeasible for creating a dataset large enough for machine learning research. Thus using sites' popularity as a proxy for being benign, and using published blacklisted certificates as a proxy for being malicious, is a reasonable compromise in creating a dataset. Our decision to use a mix of the SSLBL public benchmark certs and those from the Alexa Top 1 Million list is based on this consideration. However, one shall bear in mind that this mix does not represent the distribution of malicious and benign certificates in the wild. This is a dilemma researchers applying AI/ML to cybersecurity always have to deal with.

Instead of accepting this limitation as unavoidable, we adopt a different evaluation strategy to examine our approach's utility in a real world environment. We observe the key challenge in evaluating an AI/ML-based system on real-world security data is that this real-world data does not come with ground truth, and it is too expensive to curate labels manually. In binary classification such as ours, without ground truths for the positive and negative class, our desired metrics of accuracy, precision, recall, F1, false positive rate and miss rate cannot be calculated. The question is - are these the only metrics that can be used to examine an AI/ML system's utility in real world operations? Our answer is no.

We examine the utility of a security tool through the lens of effort saving. We ask the question - given the ultimate goal of a security task, how much effort is reduced by using the tool, compared with not using it? In this evaluation, we use the following as the goal of our security task:

Find at least one malicious TLS certificate in the wild.

To collect real world SSL/TLS certificate on the internet, we leveraged the extensive dataset provided by Rapid7's Project SONAR [28]. Project SONAR conducts comprehensive internet-wide scans, meticulously collecting data on SSL/TLS certificates from a vast array of websites and services. For our research, we selected a random sample of 150,000 unique SSL/TLS certificates from the trove of scans conducted between January 2, 2024, and May 26, 2024.

We then applied our classifier on these 150,000 certificates, which reported 13 of them as botnet certificates. While it is infeasible to check the ground truth of all the 150,000 certificates, it is totally feasible to check the ground truth of 13. Using VirusTotal (VT) [34], a public service for checking the maliciousness of files, URLs, or IPs, we checked the 13 IP addresses where the 13 botnet certificates were obtained. One of them was confirmed as malicious. This means that by using our classifier, *we did indeed find one malicious TLS certificate from 150,000, by examining only 13.*

The precision of our tool on the 150,000 certificates is $1/13 = 7.7\%$, significanly lower than the upper 90 percentage points from the earlier experiments. There are a number of reasons for this. The most important one is that the curated dataset is balanced, with equal number of benign and malicious certificates. In the real

world botnet certificates likely constitute a minuscule portion of the TLS certificates in the wild. Due to the base-rate fallacy [8], even if a detector has extremely high detection accuracy, when the prevelance of the target event is low, the precision will be substantially reduced. The other reason is that real-world data does not have the composition bias explained above. The bias could result in artifacts that make the classification task easier. This further demonstrates the importance of evalauting an ML-based security tool on real-world data, in addition to curated datasets.

Now, if VT can provide the ground truth, why not run all the 150,000 IP addresses through it and thus label all the TLS certificates in this real-world dataset? In fact, the ground truths from VT are not free. Each is the result of running a sample through a number of AV products, based on the vendors' threat intelligence that ultimately involves substantial human labor. For this reason, VirusTotal is not a free service. One can only query up to 500 samples each day without cost; beyond that an expensive subscription is required. Scanning all 150,000 would be cost prohibitive, either in monetary term or time. Scanning a sample through VT can thus be further seen as a proxy for human toll needed to determine if a TLS certificate is malicious.

Now we examine the cost for achieving the above stated goal *without* using our tool. The only approach will then be to randomly pick some certificates and manually determine if they are malicious. This will be a labor intensive process. We would like to see given a specific budget of human efforts, whether this approach can identify a single malicious certificate in the wild, like we did using our classifier. We decided to give this process *100 times* human efforts as in the previous case. Using VirusTotal as a proxy for such human efforts, we allowed querying VT 1,300 times (vs. 13 times when our classifier was used). We thus randomly selected 1,300 certificates from the 150,000, and ran them through VT. Out of the 1,300, *none were confirmed as malicious by VT.*

The contrast is clear: using our classifier, we only needed to query VT 13 times to identify one confirmed malicious certificate. Not using our classifier, after querying VT 1,300 times no confirmed malicious certificate had been found. Thus, *for the objective of identifying at least one malicious TLS certificate in the wild, our classifier provided at least a 100x human effort reduction for these 150,000 TLS certificates crawled from the internet.*

This analysis illustrates that standard metrics such as precision must be interpreted in the context of an operational environment to explain the benefit of a security tool, in this case the human effort reduction. A low precision score does not necessarily mean that a tool is not useful.

5.9 Discussion: the Build vs. Buy Decision

The possible selection of LLMs to solve a cybersecurity use case often includes the "build versus buy" decision, where the costs and benefits are weighed up for developing an LLM-based approach in-house, perhaps leveraging open-source foundation models, or choosing a third party vendor-owned solution, for example accessed via serverless API. Significant factors in the "buy" decision concern governance and information security risk, engineering dependencies plus naturally costs. Data privacy, egress beyond trusted boundaries, coding against 3rd party APIs vulnerable to change, and potential performance impacts from even minor 3rd party model updates all need to be considered when choosing what path to take. For botnet certificate detection, and perhaps similar use cases, this study reinforces the potential of open-source solutions like `C-BERT`, where its competitive performance across multiple experiments, coupled with inherent security and cost advantages make it a compelling alternative to commercial offerings. `C-BERT` and other appropriately licensed open-source models are free to use in production environments and can be self-hosted with internally documented and managed APIs, requiring no egress beyond a trusted boundary. Further, there should be no unexpected changes to the actual model binary due to the checks and balances associated with engineering hygiene in the enterprise. By eliminating reliance on third-party APIs and self-hosting open-source solutions for generating vector embeddings, the attack surface reduces to help mitigate the risk of LLM vulnerabilities such as model poisoning [1]. Our findings suggest that for botnet certificate detection, and possibly other tasks, if in the research phase open-source models exhibit competitive performance then serious consideration should be given to a "build" approach leveraging those models, such as `C-BERT` and others.

6 Conclusion and Future Work

This paper investigates the application of LLM text embeddings and their potential use with vector databases for botnet certificate detection. We perform a comparative analysis of various open-source and 3rd party embedding models to identify the most suitable solution for classifying malicious certificates. These evaluations show that `C-BERT`, a publicly available character-based embedding model, emerges as the preferred choice within our proposed system. `C-BERT` achieves a competitive F1 score of 0.994 on balanced unseen test data, demonstrating strong performance that outperforms `OpenAI`. `C-BERT` provides F1 score of 0.979 on balanced dataset comprising of botnet certificates gathered over several months after data used to create the initial projected embedding space, and maintains performance in a simulated zero-day evaluation of four C&C groups, with an average F1 score of 0.936. Furthermore, `C-BERT` has a rapid inference time, making it well-suited for real-time applications. Evaluation of our detection system on 150,000 TLS certificates crawled from the wild shows significant human effort reduction in identifying malicious certificates. The open-source nature of `C-BERT` eliminates reliance on external vendors and potential security vulnerabilities associated with proprietary solutions.

For future work, we aim to extend our research by incorporating a broader variety of malicious certificate databases. Additionally, we plan to explore the potential benefits of incorporating features beyond subject and issuer information into the embedding strategy. Lastly, we propose to further extend this research by generating custom embedding models through contrastive learning and fine-tuning techniques.

Acknowledgment

This work was partially supported by the National Science Foundation under award no. 2235102, and Office of Naval Research under award no. N00014-23-1-2538.

References

[1] OWASP Top 10. 2024. ML10:2023 Model Poisoning. https://owasp.org/www-project-machine-learning-security-top-10/docs/ML10_2023-Model_Poisoning. Accessed: 2024-07-04.

[2] Abuse.ch. 2024. SSLBL | Malicious SSL Certificates. https://sslbl.abuse.ch/ssl-certificates/. Accessed: 2024-06-09.

[3] Voyage AI. 2024. voyage-large-2-instruct: Instruction-tuned and rank 1 on MTEB. https://blog.voyageai.com/2024/05/05/voyage-large-2-instruct-instruction-tuned-and-rank-1-on-mteb. Accessed: 2024-07-02.

[4] Turki Al lelah, George Theodorakopoulos, Philipp Reinecke, Amir Javed, and Eirini Anthi. 2023. Abuse of cloud-based and public legitimate services as command-and-control (C&C) infrastructure: a systematic literature review. *Journal of Cybersecurity and Privacy* 3, 3 (2023), 558–590.

[5] Bushra A. Alahmadi, Louise Axon, and Ivan Martinovic. 2022. 99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 2783–2800. https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi

[6] Amazon. 2024. Alexa Top 1 million. http://s3-us-west-1.amazonaws.com/umbrella-static/top-1m.csv.zip. Accessed: 2024-07-04.

[7] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. 2017. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*. 1093–1110.

[8] Stefan Axelsson. 1999. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM conference on Computer and Communications Security (CCS'99)*. 1–7.

[9] Paul Black, Iqbal Gondal, and Robert Layton. 2018. A survey of similarities in banking malware behaviours. *Computers & Security* 77 (2018), 756–772.

[10] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics* 5 (2017), 135–146.

[11] Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Junichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for Word-Level Open-Vocabulary Representations From Characters. arXiv:2010.10392 [cs.CL] https://arxiv.org/abs/2010.10392

[12] Michael W Browne. 2000. Cross-validation methods. *Journal of mathematical psychology* 44, 1 (2000), 108–132.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[14] OpenAI et al. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[15] Maryam Feily, Alireza Shahrestani, and Sureswaran Ramadass. 2009. A Survey of Botnet and Botnet Detection. In *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. 268–273. https://doi.org/10.1109/SECURWARE.2009.48

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[17] Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. 2007. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *USENIX Security Symposium*, Vol. 7. 1–16.

[18] Kaspar Hageman, Egon Kidmose, René Rydhof Hansen, and Jens Myrup Pedersen. 2021. Can a TLS certificate be phishy?. In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT)*.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[20] A Ker, T Pevny, M Kopp, and J Kroustek. 2016. Malicons: detecting payload in favicons. *Electronic Imaging: Media Watermarking, Security, and Forensics 2016* 2016 (2016).

[21] Panagiotis Kintis, Najmeh Miramirkhani, Charles Lever, Yizheng Chen, Rosa Romero-Gómez, Nikolaos Pitropakis, Nick Nikiforakis, and Manos Antonakakis. 2017. Hiding in Plain Sight: A Longitudinal Study of Combosquatting Abuse. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 569–586. https://doi.org/10.1145/3133956.3134002

[22] Benjamin Kuhnert, Jessica Steinberger, Harald Baier, Anna Sperotto, and Aiko Pras. 2017. Booters and Certificates: An Overview of TLS in the DDoS-as-a-Service Landscape. In *2nd International Conference on Advances in Computation, Communications and Services, ACCSE 2017*. IARIA/Thinkmind, 37.

[23] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[24] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316* (2022).

[25] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. TESSERACT: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 729–746. https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[27] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla. 2016. A comprehensive measurement study of domain generating malware. In *25th USENIX Security Symposium (USENIX Security 16)*. 263–278.

[28] Rapid7. 2024. Project Sonar. https://www.rapid7.com/research/project-sonar/. Accessed: 2024-08-01.

[29] Amazon Web Services. 2024. Amazon Titan Text Embeddings models. https://docs.aws.amazon.com/bedrock/latest/userguide/titan-embedding-models.html. Accessed: 2024-07-02.

[30] Wan-Chen Shi and Hung-Min Sun. 2020. DeepBot: a time-based botnet detection with deep learning. *Soft Computing* 24, 21 (2020), 16605–16616. https://doi.org/10.1007/s00500-020-04963-z

[31] Andreas Theofanous, Eva Papadogiannaki, Alexander Shevtsov, and Sotiris Ioannidis. 2024. Fingerprinting the Shadows: Unmasking Malicious Servers with Machine Learning-Powered TLS Analysis. In *Proceedings of the ACM on Web Conference 2024 (WWW'24)*. Singapore.

[32] Ivan Torroledo, Luis David Camacho, and Alejandro Correa Bahnsen. 2018. Hunting Malicious TLS Certificates with Deep Neural Networks. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (AISec'18)*. Toronto, Canada, 64–73.

[33] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models. arXiv:2404.18796 [cs.CL] https://arxiv.org/abs/2404.18796

[34] VirusTotal. 2023. Retrieved January, 2024 from https://www.virustotal.com/gui/home/search

[35] Gaute Wangen. 2015. The role of malware in reported cyber espionage: a review of the impact and mechanism. *Information* 6, 2 (2015), 183–211.

[36] Ying Xing, Hui Shu, Hao Zhao, Dannong Li, and Li Guo. 2021. Survey on Botnet Detection Techniques: Classification, Methods, and Evaluation. *Mathematical Problems in Engineering* 2021, 1 (2021), 6640499. https://doi.org/10.1155/2021/6640499 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1155/2021/6640499

[37] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *ACM Trans. Knowl. Discov. Data* 18, 6, Article 160 (apr 2024), 32 pages. https://doi.org/10.1145/3649506

[38] Hossein Rouhani Zeidanloo, Mohammad Jorjor Zadeh Shooshtari, Payam Vahdani Amoli, M. Safari, and Mazdak Zamani. 2010. A taxonomy of Botnet detection techniques. In *2010 3rd International Conference on Computer Science and Information Technology*, Vol. 2. 158–162. https://doi.org/10.1109/ICCSIT.2010.5563555