An Evaluation Benchmark for Generative AI In Security Domain

Mina Ghashami, Mikhail Kuznetsov, Vianne Gao, Ganyu Teng, Phil Wallis, Joseph Xie,
Ali Torkamani, Baris Coskun and Wei Ding
{ghashami,mikuzne,gaov,tenganyu,phwallis,xietiank,alitor,barisco,dingwe}@amazon.com
Security Analytics and AI Research (SAAR)
Amazon Web Services

ABSTRACT

As computing environments become increasingly complex and distributed, the volume and complexity of security data generated across various systems have grown exponentially. Extracting useful insights from this security data is crucial for effective security analytics, anomaly detection, and threat identification. However, there is a lack of comprehensive evaluation benchmarks for assessing the performance of large language models trained on any security log dataset, hindering progress in this domain. This paper proposes a comprehensive evaluation benchmark for security data, addressing this critical gap. The benchmark is easily adoptable to any security log dataset and comprises four diverse categories of tasks: supervised evaluations, unsupervised evaluations, anomaly detection, and semantic similarity evaluations. By providing a standardized framework for evaluation, the benchmark enables objective comparison and reproducible assessment of state-of-the-art embedding models across various computing environments and security log sources.

KEYWORDS

security domain, evaluation benchmark, embedding models, large language models, generative AI

ACM Reference Format:
Mina Ghashami, Mikhail Kuznetsov, Vianne Gao, Ganyu Teng, Phil Wallis, Joseph Xie,, Ali Torkamani, Baris Coskun and Wei Ding. 2024. An Evaluation Benchmark for Generative AI In Security Domain. In *Proceedings of KDD workshop on Evaluation and Trustworthiness of Generative AI Models (KDD).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the era of generative AI, the analysis and interpretation of security telemetry data from cloud computing environments have become increasingly crucial. As the adoption of cloud services continues to grow, the volume and complexity of security logs generated by these environments have escalated exponentially. These logs contain invaluable information about security events, user activities, and system behaviors. Large language models trained on security logs have the potential to learn meaningful representations of this data, enabling downstream applications such as anomaly detection [19, 20], threat mitigation [14, 27], and policy enforcement [17]. By leveraging the power of these models, security teams can gain a deeper understanding of the patterns and behaviors captured in the logs, empowering them to proactively identify and respond to potential security threats.

In the field of natural language processing (NLP), evaluation benchmarks have played a pivotal role in driving progress and enabling researchers and practitioners to objectively assess the performance of their models [3, 10, 11, 18]. These benchmarks include standardized datasets, tasks, and evaluation metrics, allowing for fair and consistent comparisons across different models and approaches. Notable examples include the GLUE benchmark [26] for natural language understanding, SQuAD [21] for question answering, the WMT [1] series for machine translation, and MTEB [18] the massive text embedding benchmark that is the most comprehensive benchmark of text embeddings to date.

However, in the domain of security analytics there is a lack of comprehensive, and widely accepted evaluation benchmarks. This gap hinders the ability to effectively evaluate and compare the performance of large language models trained on security telemetry data. These models aim to learn meaningful representations of fine-grained security entities such as account, usernames, APIs, and larger-grained entities such as events and user activities. Through these representations, they enable advanced analytics, anomaly detection, and threat identification.

The development of an evaluation benchmark for large language models in security data is crucial for several reasons: (1) **standardization**; a benchmark would provide a standardized dataset and set of tasks, allowing for consistent and reproducible evaluation of large language models across different research groups and organizations. (2) **objective comparison**; with a common benchmark, researchers and practitioners can objectively compare the performance of their large language models, fostering healthy competition and driving innovation in the field. (3) **reproducibility**; a well-defined benchmark ensures that evaluation results are reproducible, promoting transparency and facilitating knowledge sharing within the research community. (4) **identifying strengths and weaknesses**; benchmarks can reveal the strengths and weaknesses of different large language models, guiding future research efforts and model improvements.

Contribution. In this paper, we propose a pioneering evaluation benchmark tailored explicitly for security telemetry data, representing, to the best of our knowledge, the first endeavor of its kind in this domain. Our evaluation benchmark presents a versatile methodology tailored to accommodate any security data, encompassing a diverse array of tasks and datasets meticulously curated to deliver

KDD, August 25-26, 2024, Barcelona, Spain

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

a comprehensive and holistic assessment of state-of-the-art large language models. Specifically, it includes:

(1) **Unsupervised evaluation** (section 4); these tasks measure the expressivity and learnability of the learned representation space via embedding models.

(2) **Supervised evaluation** (section 5); these tasks measure the extent to which a model can distinguish between different classes of entities.

(3) **Semantic Similarity evaluation** (section 6); these tasks evaluate a model's ability to position similar items in close proximity within the embedding space, while ensuring dissimilar items are placed farther apart from each other. This evaluation comprises of two subtasks:

(3.1) **Event Correlation** (section 6.1); correlating and connecting related security events is crucial for identifying potential threats and understanding the broader context of security incidents. This task assesses the capability of embedding models to accurately associate and link related events based on their representations.

(3.2) **User Behavior Profiling** (section 6.2); understanding and profiling user behavior patterns is essential for detecting insider threats and identifying potential policy violations. This task evaluates the performance of embedding models in capturing and representing user behavior patterns from security data.

(4) **Downstream evaluation of Anomaly Detection** (section 7); this task evaluates if embedding models can enhance anomaly detector's ability to identify anomalous security events and user activities within the security environments. The datasets for this task include a mostly benign security log dataset, and a synthetic attack dataset.

While the proposed evaluation benchmark is designed to be general and applicable to any security data, in this paper, we leverage real-world AWS security telemetry logs to demonstrate its efficacy. The selected AWS security logs encompass a diverse range of services, configurations, and security event types. This diversity ensures that the evaluated models are exposed to a wide spectrum of scenarios, enabling them to generalize effectively to real-world deployments. We utilize the developed evaluation benchmark to measure the quality of encoder-based language models fine-tuned on AWS log data. As a baseline, we employed RoBERTa-large [15], a pre-trained language model, and performed domain adaptation by continuing its pre-training on CloudTrail, one of the AWS security telemetry logs. The comprehensive comparison results are presented in Section 8. Throughout this paper, we refer to the model that is adapted to security logs as *log-adapted*, and the model which is not adapted as *non-adapted*.

In Section 9, we delve into a crucial characteristic of security data: its dynamic nature. We emphasize that our benchmark must be dynamically re-created at a defined cadence to account for the ever-evolving landscape of security telemetry data. The dynamic nature of security data necessitates a proactive approach to ensure the continuous relevance and accuracy of our evaluation benchmark. By acknowledging and addressing this dynamic aspect, our benchmark maintains its robustness and adaptability, enabling it to consistently provide a reliable and comprehensive assessment of embedding models' performance in the face of the rapidly changing security landscape. This proactive approach ensures that the benchmark remains a valuable resource for researchers and practitioners, fostering the development of more resilient and effective solutions for securing cloud environments and safeguarding sensitive data.

2 RELATED WORK

2.1 Large Language Models

Large language models [2, 25] have emerged as powerful tools for learning dense vector representations of discrete entities, such as words, phrases, or tokens. These representations, known as embeddings, capture semantic and contextual information, enabling various natural language processing (NLP) tasks to be performed more effectively. Among the different types of embedding models, transformer-based models [22, 25] have gained significant traction due to their remarkable performance and ability to capture long-range dependencies.

While embedding models have primarily been explored and applied in the NLP domain, their potential value extends to other areas, including *security analytics*. In the context of security telemetry data, embedding models can be leveraged to learn meaningful representations of security entities, such as user identities, resource names, IP addresses, and event types. These learned embeddings can capture the intricate relationships and patterns present in security logs, enabling more effective anomaly detection, threat identification, and user behavior profiling.

2.2 Cybersecurity

The security domain has witnessed a significant surge in the adoption of machine learning techniques to tackle various challenges, including anomaly detection [19, 20], IP reputation analysis [13], and threat identification [12]. Traditional rule-based and signature-based approaches, while valuable, are often limited in their ability to keep pace with the ever-evolving landscape of cyber threats. Machine learning, with its ability to learn from data and adapt to new patterns, has emerged as a powerful tool for enhancing security analytics and fortifying defenses against sophisticated adversaries.

Anomaly detection, for example, is a critical task in the security domain, as it enables the identification of deviations from normal behavior, which may indicate potential threats or compromised systems. Previous works [9], have proven effective in learning the characteristics of normal behavior from security telemetry data and flagging anomalous patterns. IP reputation analysis [13] is another area where machine learning has made significant contributions. By analyzing various features associated with IP addresses, such as geolocation, autonomous system numbers, and historical behavior, machine learning models can classify IP addresses as malicious or benign. This capability is crucial for identifying potential sources of attacks, and implementing access controls.

2.3 Benchmarks

Evaluation benchmarks have played a pivotal role in driving progress and enabling objective comparisons in the field of NLP. Benchmarks such as (Super)GLUE [26] and Big-BENCH [24], along with evaluation frameworks [6] have been instrumental in advancing NLP research and development.

While these benchmarks have proven effective in the NLP domain, the field of cloud security lacks comprehensive and widely accepted evaluation benchmarks. As cloud computing environments generate vast amounts of security telemetry data, there is a pressing need for standardized benchmarks to assess the performance of large language models trained on this data. These models aim to learn meaningful representations of security entities, events, and behaviors, enabling advanced analytics, anomaly detection, and threat identification.

The successful adoption and impact of these benchmarks in the NLP field highlight the potential value that a dedicated evaluation benchmark could bring to the domain of cloud security. By providing a standardized platform for evaluation, researchers and practitioners could objectively assess the performance of their embedding models, identify areas for improvement, and facilitate the development of more robust and effective security solutions tailored to the unique challenges of cloud computing environments.

3 TASKS AND EVALUATION

To offer a holistic evaluation, our benchmark considers multiple facets. First, it encompasses both upstream evaluations, through unsupervised tasks (section 4) and semantic similarity assessments (section 6), as well as downstream evaluations, including supervised tasks (section 5) and anomaly detection (section 7). This comprehensive approach examines the quality of entity representations from various vantage points.

Secondly, the benchmark assesses entities not only in isolation through *non-contextual* embeddings but also in the *context* of other entities, enabling an understanding of how representations perform in practical situations. For instance, it evaluates the embedding of a service in the context of the user identity that utilized that service.

Thirdly, our benchmark considers a wide range of security entities spanning different granularities. Table 2 lists down the entities our evaluation benchmark utilizes. We focused on the most crucial entities that either have a direct application in downstream usecases or are of immediate importance from a subject matter expert's perspective.

Although the proposed methodology is intentionally designed to be generic and applicable to any security data logs, we strategically leverage AWS CloudTrail logs as the primary data source for extracting and constructing the various benchmark tasks. CloudTrail furnishes a comprehensive audit trail that encompasses activities across AWS services and resources, enabling us to carve out realistic and practical test cases. Table 1 presents a comprehensive listing of all our evaluation tasks.

4 UNSUPERVISED EVALUATION TASKS

The following tasks are unsupervised [4, 16] and do not need a labelled evaluation dataset.

4.1 Cluster Learnability (CL)

This task [16] evaluates the learnability of representations. A higher Cluster Learnability (CL) score indicates that the learned representation is more effective at separating distinct clusters in the input data space. The CL metric measures the learning speed of a K-Nearest Neighbor (KNN) classifier trained to predict labels obtained by clustering the representations using K-means. The process is as follows:

(1) Pick the hyper-parameters k and k' as number of clusters, and number of neighbors. Let x_i denote the representations of $i-$th data point.
(2) Perform k-means clustering on the dataset to obtain k clusters. Assign a cluster ID to each data point x_i as its label, denoted as $\hat{y_i}$. This cluster ID serves as the ground truth label for the corresponding data point within the context of this clustering task.
(3) Run KNN classification on each datapoint in a prequential manner to obtain a predicted label through majority voting. Specifically, for a given data point $(x_i, \hat{y_i})$, consider all data points with indices less than i, denoted as $x_j | j < i$, and apply the KNN algorithm to these data points to determine the majority class label. This majority class label will be the predicted label $\tilde{y_i}$ for the data point x_i.
(4) Then the CL metric would be $CL = \frac{1}{N} \sum_{i=1}^{N} [\hat{y_i} = \tilde{y_i}]$

The "prequential" approach, also known as the "test-then-train" method, is a way of evaluating the performance of a machine learning model, such as KNN, in an online learning setting. In this task, through the prequential manner of KNN computation, the model is evaluated and updated sequentially, one instance at a time, using a stream of data. Since the data is unlabeled, this method uses clustering in step (2) to synthetically generate labels i.e. cluster ids, and use them to compute the speed of cluster learnability in an online (or prequential) manner.

4.2 Expressiveness

This task assesses expressiveness of representations through a metric called *Intrinsic Dimension (ID)* [16]. This metric is based on the intuition that as more and more fine-grained categories emerging in the representation space, we expect the manifold complexity to go up, which can be characterized as the number of parameters needed to describe the representation manifold without losing information[16]. While inferring intrinsic dimension is a challenging problem in its own, [16] uses nearest neighbors to infer ID. The metric computation process is as following:

(1) Let $z_1, \cdots, z_N$ be N data points.
(2) For each datapoint z_i, compute the distance to its k-th nearest neighbor, i.e. $r_{ik} = Dist(z_i, NN(z_i, k))$
(3) Define $\mu_i = r_{i2}/r_{i1}$ as ratio of distances for i-th datapoint.
(4) Sort μ_i ascendingly, i.e. $\mu_1 \leq \cdots \leq \mu_N$ and estimate the cumulative distribution as $F_i^{emp} = i/N$.
(5) Fit a straight line on the dataset $\{(\log \mu_i, -\log(1-F_i^{emp}))\}_{i=1}^{N}$ in two dimensional plane.
(6) The slope of the line is the *estimated ID*.

To gain an intuition on the approach, assume that data points are sampled on a manifold with intrinsic dimension d. It can be shown [5] that under the assumption of local uniformity, μ_i follows a Pareto distribution with parameter $d + 1$ on $[1, \infty)$, i.e. $\mu_i \sim P(\mu d) := d\mu^{-(d+1)}$. The cumulated distribution associated with $P(\mu d)$ is $F(\mu) = 1 - \mu^{-d}$, and we can use this to estimate d. The way authors[16] estimate it, as described in above steps, is

Table 1: Our evaluation benchmark tasks

Evaluation Type		Task	SubTask
Upstream	Unsupervised	Cluster Learnability	
		Expressiveness	
	Semantic Similarity	User behavior profiling	Service Linked Roles
			Account ID
			Principal ID
			Account ID + Principal ID
			account ID + Username
		Event correlation	Service
			Service & API
			Whole Event
			IP
Downstream	Supervised	IP Reputation	
		Risk API	
		Suspicious ASNorg	
		Prod User Identity	
	Anomaly Detection	Overall Detection Volume	
		Attack Detection Volume	
		Benign ASN Detection	
		Malicious ASN Detection	

Table 2: AWS security entities used in our benchmark

Entity	Description
accountId	An account number uniquely identifying a user account.
username	A single sentence description that represents the user or entity associated with the recorded security event or activity.
userIdentity	A pair consisting of an accountId and a username.
service	A service or resource, such as S3, from which the security event originates.
API	The name of a specific API or operation within an service, e.g., ListS3Buckets() in the S3 service.
event	A timestamped record capturing a user's interaction with AWS, containing the request and response details.
IP	The Internet Protocol (IP) address associated with the security event.
ASNorg	The Autonomous System Number (ASN) organization associated with the IP address involved in the security event.
PrincipalId	A unique identifier that represents the entity (user, role, or service) responsible for performing an action or operation.

through linear regression on the empirical cumulate of the distribution i.e. $F_i^{emp} = i/N$.

5 SUPERVISED EVALUATION TASKS

This section encompasses classification tasks involving security entities present in table 2. The motivation for these tasks is that a log-adapted model should learn informative representations for each of these entities, and enable accurate discrimination across various classes.

5.1 Classification on IP Reputation

This task classifies an IP address to either benign or malicious. The motivation for this task is that benign and malicious IPs have very different behaviors that could be captured by the security logs. To enable the binary classification, we use output of an internal labelling tool as target labels. This tool assigns +1 if an IP is a threat and assigns 0 otherwise.

A snippet of the dataset is as follows, where IP addresses are masked for privacy, where the input and the corresponding label are separated by comma:

- 100.200.300.400, 0
- 100.500.600.700, 1

5.2 Classification on Risk API

This task classifies an API on their risk category (low, mid, high). The rationale for this task is that the risk level of an API can be captured by the sets of events it happens in. A snippet of the curated dataset for this task is as follows:

- CreateChangeSet, MediumRisk
- ExecuteChangeSet, HighRisk
- UpdateStack, HighRisk

5.3 Suspicious ASNorg or IP address

This task classifies a given IP address/ASNorg as suspicious or not based on a static list. The suspicious ASNorgs are ones we've observed attacks from more often than others. The rationale for this task is that the suspicious ASNorg/IPs will have different behavior than benign IPs, and embedding model trained on security logs should be able to catch that distinction. A snippet of the dataset is as following, where names of ASNorgs are masked:

- asnOrg:asn-org-name, 1
- asnOrg:another-asn-org-name, 0

5.4 Production vs Non-production User Identity Detection

This is a binary classification task in which we classify a userIdentity which is a pair of (accountId, username) as either internal production or non-production. We make sure the accountId is unique in the training and testing data to avoid potential memorizing. We obtain ground truth labels from a list of internal accounts.

6 SEMANTIC SIMILARITY TASKS

The *semantic similarity* tasks are designed to quantify how well domain-specific semantics are learned via an embedding space or, equivalently, a model producing these embedding vectors. These tasks assess whether the model can infer relationships within the representation space, via semantic similarity, that are meaningful to a subject matter expert. These benchmark tasks can be used to evaluate and compare various language models, including generally pre-trained base models and various domain-adapted variants. The domain-adapted variants incorporate pre-training from security log sources. Performance across these tasks will be used to determine the degree to which models have learned useful, domain-specific relationships.

A dataset for each semantic similarity task comprises a list of (anchor, positive, negatives) triplets. An anchor is a reference data point that is observed in the training data. To enhance the quality and diversity of our training data, we meticulously selected a diverse set of anchor points (see section 6.3). The corresponding positive example is also observed in the training data, and selected by a process designed to ensure they are *similar* to the anchor. The negative example is selected to be semantically *dissimilar* to the anchor, but ideally not too easy for the model to distinguish from a positive.

Semantic similarity evaluation encounters two task categories: **event correlation**, and **user behavior profiling**, which we describe below in more detail.

6.1 Event Correlation

This task measures the efficacy of an embedding model at capturing associations between different events happened for the same users. Overall principle is that an embedding model is supposed to assign a higher similarity score to the events that were triggered by the user, and a lower similarity score when one of the events hasn't been triggered by a user and therefore is an outlier for this user. To measure this similarity, we define anchor, positive, and negative as follows:

+ anchor = (accountId, username, event_i) where the (accountId, username) has interacted with event_i

+ positive = (accountId, username, event_j) where the (accountId, username) has interacted with event_j and $\text{event}_i \neq \text{event}_j$

+ negative = (accountId, username, event_k) where (accountId, username) has not interacted with event_k.

We define the events at different levels of granularity, where each granularity level corresponds to a specific semantic similarity sub-task, described below.

User: Service. This task measures the efficacy of an embedding model at capturing associations between user identities i.e. (accountId, username) pair and AWS services. Note that, this task limits users' behaviors to only the services they have interacted with in AWS. In this task, an input anchor/positive/negative instance takes the form (accountId, username, eventSource).

User: Service and API.. This task aims to assess the efficacy of a model at capturing associations between user identities and service and API pairs. In this task, an input anchor/positive/negative instance takes the form (accountId, username, service, API).

User: Whole event. In the previous two tasks, we compare the contextual representations of a service, or a (service, api) pair , in the context of the userIdentity i.e. (accountId, username). In this task, an anchor/positive/negative instance takes the form (accountId, username, event).

User: IP.. This task considers IP instead of event, and assesses whether two distinct IP which a user was connecting from are more similar to each other in the embedding space, than to an IP unrelated to this user.

6.2 User Behavior Profiling

This category of tasks evaluates the performance of embedding models in capturing and representing user behavior patterns from security telemetry data, which is essential for detecting insider threats and identifying potential policy violations. This task consists of the following subtasks:

Role Similarity Identification. The goal of this task is to assess a model's ability to identify similar roles or privileges that are used across multiple entities (e.g., accounts, organizations, or users) within a security dataset. Certain roles or privileges are often pre-defined and assigned to specific services, applications, or user types within an organization or system. Since these pre-defined

roles are installed, managed, and used exclusively for a particular purpose, they tend to have identical or very similar permissions and usage patterns across different entities. This makes them a good candidate for this task. The motivation for this task is that the log-adapted models are expected to assign higher semantic similarity scores to the pre-defined roles or privileges from the same service, application, or user type across different, unrelated entities. In contrast, the adapted models should assign lower scores to pre-defined roles or privileges associated with different services, applications, or user types.

Let r denote a pre-defined role or privilege, then each example in the data is as follows:

+ anchor = (accountId$_1$, r_i) where r_i occurs with accountId$_1$ in the training data

+ positive = (accountId$_2$, r_i) where r_i occurs with accountId$_2$ in the training data

+ negative = (accountId$_3$, $r_{k\neq i}$) where r_k does not occur with accountId$_1$ in the training data

That is, the anchor and the positive are entityIds that are associated with the same pre-defined role or privilege in the data, and the negatives are all other accountId + role/privilege pairs in the evaluation dataset, where the role or privilege is different from the positive role or privilege.

User Similarity by Service and API Usage. The goal of this evaluation task is to measure similarity of representations of users based on service and API usage, such that users with similar service and API usage will have high similarity scores, and users with very different Service and API usage will have low scores. This will allow us to measure how well user representations can capture a user's behavior with respect to the services and APIs they use.

The ability to identify users exhibiting similar behavior patterns is valuable for multiple security use cases, such as detecting malicious activity, mitigating false positives, and clustering principals based on their common activities and resource interactions.

For this task, we define anchor as a user with enough number of events in the training data, a corresponding positive as a user who has similar service and API usage to anchor, and a corresponding negative as a user who has dis-similar service and API usage to anchor. A user is defined in four distinct ways:

- **Account ID** represents the AWS account associated with the user,
- **Principal ID** is a unique identifier for the user within the account,
- **A combination of Account ID and Principal ID** provides a more granular representation of the user,
- **A combination of Account ID and Username** incorporates the user's chosen name or alias along with their account information. Note that this case essentially extends the Service Linked Roles task over all usernames presented in the data.

We capture the service and API usage by constructing an event bag for each user as a list of (service, API, volume) where volume is the logarithmic quantization of the (userIdentity, service, API) triplet's volume. We consider two user similar/positive pair, if their constructed event bags have high Jaccard similarity. Otherwise they are dis-similar.

6.3 Sampling Data for Evaluation

One of the key constraints on the evaluation dataset is its size. While the number of events in security logs can reach billions in one hour, it would be infeasible to run model inference on a dataset of that order. For this matter, we constraint the dataset size to be not more than one thousand of events (e.g. not more than 1000 anchor-positive-negative instances for a semantic similarity task). Since the evaluation dataset comprises a small fraction of the whole data, uniform sampling of anchors could lead to a biased evaluation towards the over-represented instances, and developing a proper sampling mechanism becomes essential.

To deal with the sampling problem, we employ a sampling mechanism which accounts for data diversity. The mechanism is based on selecting diverse data in a metric space. A metric is defined differently for a different task: for example, for User Behavior Profiling (section 6.2) the metric is defined as the Jaccard distance of the event bags between users.

Having a metric defined on an entity space (e.g. event-based Jaccard distance on a user space), we construct the anchor set for a task as the metric k-centers [8], that is we perform clustering in a metric space, and use the cluster centroids as the anchors. More specifically, we employ a greedy selection of cluster centers by a farthest-first traversal [7]: assume we have a set of cluster centers C, then in the next iteration we add as a new center the datapoint for which the maximum distance to the closest point in C is minimized.

In this way, we build a set of anchors C of size k (in practice, we set $k = 1000$), so that the anchors comprise the diverse set of entities spanning the overall space of entities. After anchor selection, for a given anchor in C, we select negatives as a entities from the original space which have low similarity to anchor. Similarly, we select positives as entities from the original space which have high similarity to the anchor.

7 DOWNSTREAM EVALUATION: ANOMALY DETECTION FINDING QUALITY

Identifying anomalous events within security systems has been a challenging problem due to several reasons including but not limited to the followings: **(1) high-dimensional and heterogeneous data**: security systems often deal with large volumes of high-dimensional data from various sources. This data can be heterogeneous, consisting of structured and unstructured data, making it difficult to process and analyze. **(2) imbalanced and evolving data distributions**: in security contexts, anomalous events are typically rare occurrences, leading to highly imbalanced data distributions. Additionally, the nature of anomalies can evolve over time as new threats and attack vectors emerge. **(3) lack of labeled data**: obtaining labeled data for anomalous events is often difficult and costly. and **(4) high-cardinality categorical data**: security data frequently includes high-cardinality categorical features, such as user IDs, IP addresses, and event types. Handling these high-dimensional and sparse categorical features can be computationally expensive and may require specialized techniques for effective representation and modeling.

To address these challenges, the state-of-the-art anomaly detection (AD) models consist of an encoder and a decoder. The encoder receives embeddings of selected entities such as user IDs, resource

names, network identifiers, and action types as input and passes them through further encoding layers. The decoder reconstructs a subset of the input entities from the latent representations conditioned on the rest of the input. Finally, a categorical cross-entropy loss is computed between the reconstructed entity and the original input entity.

While the AD models can use randomly initialized embeddings for their inputs and exhibit reasonable effectiveness, the primary objective of this task is to assess whether pre-trained representations can enhance the detection capabilities of AD models.

For this task, we use an autoencoder-based AD model that takes features related to user profile such as accountId and username, and features related to user activity such as service and API as input. We then compute the following two subtasks:

First, we compute the total volume of the predicted anomalous events on a new day conditioned on the assumption that the behavior on the prior 45 days are normal activities. Under this assumption, the total volume should be relatively low if the model is correctly specified and trained, since we expect most events to be normal user activities.

Second, to assess the model's sensitivity to malicious events, we generate a synthetic malicious dataset based on attack templates from *Pacu: The Open Source AWS Exploitation Framework* [23], and evaluate the model's ability to detect such potential attacks.

8 EXPERIMENTS AND RESULTS

To demonstrate the efficacy of the benchmark, we employed RoBERTa-large, a pre-trained language model, as our starting point and performed domain adaptation by continuing its pre-training on CloudTrail, one of AWS security telemetry logs. We refer to the original RoBERTa-large model as the *non-adapted model*, while the domain-adapted version, fine-tuned on CloudTrail logs, is referred to as the *log-adapted model*. We split our results into three tables: table 3 summarizes the results of unsupervised, and supervised tasks; while table 4 states the result for semantic similarity tasks, and table 5 lists down the results for anomaly detection task.

For the unsupervised task, we computed the non-contextual embedding of IP as a candidate security entity by mean pooling its output from the model. We report *CLID* metric which is a summation of Cluster Learnability(CL) and Intrinsic Dimensionality(ID). The higher the CLID value, the greater the expressivity and learnability of the learned embedding space.

For supervised evaluation tasks, we developed a classification head on top of the language model. While the classification head can be linear or non-linear we chose a two layer fully connected network with ReLU activation function. We created train set and test set consistent with the pretrain data format, and embeded the input using the language model and mean pooling option. We trained the classifier for 10 epochs, and evaluated its performance on the test set after every epoch. For each task, we record the highest Area Under the Receiver Operating Characteristic (AUROC) score.

For semantic similarity evaluation tasks, we report AUROC. For each subtask, we collected 1000 anchors, where each anchor has one corresponding positive, and 10 corresponding negatives. To compute the AUROC score, we calculated the cosine similarity between the anchor and positive example, as well as the mean cosine similarity between the anchor and the 10 negative examples. These similarities provide the probabilities that a positive example (and similarly, a negative example) belongs to the anchor class. We then compute the AUROC score by averaging these probabilities over all examples in the dataset.

The results presented in tables 3 and 4 demonstrate that the log-adapted model achieves up to 20% performance improvement in both unsupervised and supervised tasks, and more than 100% improvement in semantic similarity tasks, when compared to the non-adapted RoBERTa-large. This significant improvement highlights the model's capability to effectively capture and leverage the semantic information embedded within the CloudTrail logs. Consequently, the model exhibits remarkable proficiency in accurately classifying entities, as evidenced by its strong performance in the supervised evaluation tasks. Additionally, the log-adapted model can position anchor and positive examples closer together in the semantic space, while separating anchor and negative examples, thereby excelling in semantic similarity tasks.

For the downstream anomaly detection task, the primary objective is to assess whether pretrained representations can enhance the precision and recall of anomaly detection models. To achieve this, we generate three types of representations: 1) random representations for each entity, 2) representations from a log-adapted model, and 3) representations from a non-adapted model. For each case, we feed the respective entity representations into the AD model, resulting in three distinct AD model variants:

(1) Random-init AD (using random representations),
(2) Log-adapted AD (using representations from a log-adapted embedding model),
(3) Non-adapted AD (using representations from a non-adapted model).

As shown in table 5, we observe that log-adapted AD model has improved sensitivity for malicious events compared to the non-adapted AD model. Specifically, we observe 0.9% higher detection volume from the synthetic attack dataset, as compared to non-adapted model which has 4.5% lower detection volume. In addition, the log-adapted AD model has a 35.7% lower detection volume from a largely benign set of CloudTrail data, as compared to the non-adapted AD which has 32.2% lower detection volume.

9 DYNAMIC BENCHMARKING

A crucial characteristic of security log data is its dynamic nature. New accounts and users are continuously introduced into the cloud environment on a daily basis, while others cease to utilize the service. To accommodate these fluctuations, both the log-adapted model and benchmark datasets should be periodically recomputed at a defined cadence, ensuring their relevance and accuracy in responding to the ever-evolving landscape.

Figure 1 illustrates the evaluation results where the benchmarks (*Event Correlation: Service+API*, and *User Behavior Profiling: Account+Principal*) are recomputed daily for seven consecutive days. The figure clearly demonstrates that the log-adapted model consistently outperforms the non-adapted model. Since the non-adapted model is static, the daily variance in its metrics can be attributed to data drift. In other words, the daily fluctuation in the metric for the

Table 3: Unsupervised and Supervised Evaluation Results

	Unsupervised	Supervised		
Model	CLID	IP Reputation	API Risk	UserIdentity cls
Non-Adapted	12.81	0.68	0.64	0.61
Log-Adapted	15.46	0.82	0.70	0.71

Table 4: Semantic similarity Evaluation Results

	Event Correlation				User Behavior Profiling				
Model	Service	Service+ API	Whole Event	IP	SLR	Account	Account+ Username	Principal	Account+ Principal
Non-Adapted	0.52	0.50	0.77	0.58	0.63	0.48	0.78	0.54	0.54
Log-Adapted	0.63	0.66	0.88	0.6	1.0	0.58	0.89	0.64	0.60

Table 5: Downstream Anomaly Detection Evaluation Results

Model	Overall Detection Volume	Attack Detection Volume
Random-init AD	4594	1218
Non-Adapted AD	3117 (-32.2%)	1163 (-4.5%)
Log-Adapted AD	2974 (-35.7%)	1229 (+0.9%)

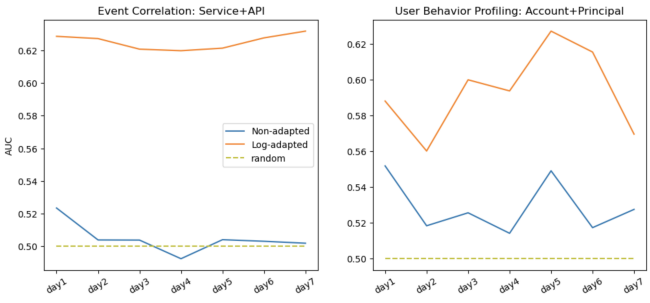

Figure 1: Dynamic evaluation with daily updated benchmarks

non-adapted model is caused by the benchmark version, specifically the introduction of a new set of users and a new set of actions performed by users on a given day compared to the previous one.

In practice, we employ the non-adapted baseline for daily benchmarking to monitor any sudden, unexpected changes in the log-Adapted model's performance. If there is a significant deviation in the metric difference between the two models, we trigger an alert and debug the training process for the log-adapted model. This approach allows us to proactively identify and address any potential issues, ensuring the model's reliability and consistent performance.

10 CONCLUSION AND FUTURE WORK

The proposed evaluation benchmark for security datasets represents a significant step towards advancing the field of security analytics. By providing a comprehensive and standardized framework for evaluating embedding models trained on security logs, this benchmark addresses a critical gap in the research community. The benchmark's diverse set of tasks, including anomaly detection, event correlation, and user behavior profiling ensures a holistic assessment of the state-of-the-art embedding models.

As part of future work, we can enhance the current benchmark by incorporating additional tasks such as information retrieval and summarization. Summarization tasks could involve generating concise summaries of security incidents or events based on the learned embeddings, which would be valuable for security analysts to quickly grasp the essence of complex situations. Information retrieval tasks, on the other hand, could involve querying the security logs and events using natural language queries, and retrieving relevant entries based on the learned embeddings. This would enable security analysts to efficiently search and retrieve specific information from large volumes of security data.

REFERENCES

[1] Ondrej Bojar, Christian Federmann, Barry Haddow, Philipp Koehn, Matt Post, and Lucia Specia. 2016. Ten years of WMT evaluation campaigns: Lessons learnt. In *Proceedings of the LREC 2016 Workshop "Translation Evaluation–From Fragmented Tools and Data Sets to an Integrated Ecosystem.* 27–34.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[3] Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C Wallace. 2019. ERASER: A benchmark to evaluate rationalized NLP models. *arXiv preprint arXiv:1911.03429* (2019).

[4] Linus Ericsson, Henry Gouk, Chen Change Loy, and Timothy M Hospedales. 2022. Self-supervised representation learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine* 39, 3 (2022), 42–62.

[5] Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2017. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports* 7, 1 (2017), 12140.

[6] L Gao, J Tow, B Abbasi, S Biderman, S Black, A DiPofi, C Foster, L Golding, J Hsu, A Le Noac'h, et al. [n. d.]. A framework for few-shot language model evaluation, 12 2023. *URL https://zenodo. org/records/10256836* 7 ([n. d.]).

[7] Teofilo F Gonzalez. 1985. Clustering to minimize the maximum intercluster distance. *Theoretical computer science* 38 (1985), 293–306.

[8] Dorit S Hochbaum and David B Shmoys. 1985. A best possible heuristic for the k-center problem. *Mathematics of operations research* 10, 2 (1985), 180–184.

[9] Wenjie Hu, Yihua Liao, and V Rao Vemuri. 2003. Robust Support Vector Machines for Anomaly Detection in Computer Security.. In *ICMLA*. 168–174.

[10] Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. *arXiv preprint arXiv:2004.12376* (2020).

[11] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, et al. 2021. Dynabench: Rethinking benchmarking in NLP. *arXiv preprint arXiv:2104.14337* (2021).

[12] Terran D Lane. 2000. *Machine learning techniques for the computer security domain of anomaly detection.* Purdue University.

[13] Jared Lee Lewis, Geanina F Tambaliuc, Husnu S Narman, and Wook-Sung Yoo. 2020. IP reputation analysis of public databases and machine learning techniques. In *2020 international conference on computing, networking and communications (ICNC)*. IEEE, 181–186.

[14] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor CM Leung. 2018. A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE access* 6 (2018), 12103–12117.

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[16] Yuchen Lu, Zhen Liu, Aristide Baratin, Romain Laroche, Aaron Courville, and Alessandro Sordoni. 2023. Using Representation Expressiveness and Learnability to Evaluate Self-Supervised Learning Methods. *Transactions on Machine Learning Research* (2023).

[17] Jesus Mena. 2011. *Machine learning forensics for law enforcement, security, and intelligence.* CRC Press.

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

[19] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. 2021. Machine learning for anomaly detection: A systematic review. *Ieee Access* 9 (2021), 78658–78700.

[20] Salima Omar, Asri Ngadi, and Hamid H Jebur. 2013. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications* 79, 2 (2013).

[21] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2383–2392.

[22] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[23] Rhino Security Labs. 2021. Pacu: The Open Source AWS Exploitation Framework. Software. https://github.com/RhinoSecurityLabs/pacu.

[24] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615* (2022).

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[26] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[27] Mingfu Xue, Chengxiang Yuan, Heyi Wu, Yushu Zhang, and Weiqiang Liu. 2020. Machine learning security: Threats, countermeasures, and evaluations. *IEEE Access* 8 (2020), 74720–74742.