USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Facade: High-Precision Insider Threat Detection Using Deep Contextual Anomaly Detection

Alex Kantchelian, *Google;* Casper Neo, *Palace Cybersecurity;* Ryan Stevens, Hyungwon Kim, Zhaohao Fu, Sadegh Momeni, Birkett Huber, Cem Topcuoglu, Senaka Buthpitiya, Elie Bursztein, Yanis Pavlidis, Martin Cochran, and Massimiliano Poletto, *Google*

https://www.usenix.org/conference/usenixsecurity26/presentation/kantchelian

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by

جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

Facade: High-Precision Insider Threat Detection Using Deep Contextual Anomaly Detection

Alex Kantchelian*, Casper Neo†, Ryan Stevens*, Hyungwon Kim*, Zhaohao Fu*, Sadegh Momeni*, Birkett Huber*, Cem Topcuoglu*, Senaka Buthpitiya*, Elie Bursztein*, Yanis Pavlidis*, Martin Cochran*, Massimiliano Poletto*

*Google, {akant,stevensr,clintk,zhaohaofu,samomi, bthuber,cemtopcuoglu,senaka,elieb,ypavlidis, martincochran}@google.com, max.poletto@gmail.com

†Palace Cybersecurity, casper@palacecyber.com

Abstract

Insiders with privileged access have the power to cause great harm to their organization. Even a single insider threat incident can be catastrophic, resulting in both financial losses and reputation damage. These threats are some of the most difficult to detect, as attack activity is interspersed in large volumes of legitimate activity. Although it is a serious threat, the literature is sparse aside from a few studies with various limitations, preventing their practical deployment in large scale organizations.

We present Facade: fast and accurate contextual anomaly detection, a high-precision, deep-learning system that has served as the last line of defense against insider threats at Google since 2018. Facade is an innovative self-supervised system that detects suspicious events by considering the context surrounding each event, including relevant facts about the user and resource involved. It is built around a new multi-modal model that is trained on corporate document access, SQL table access, and HTTP/RPC request logs. To overcome the scarcity of labeled incident data, Facade employs a novel contrastive learning strategy trained exclusively on benign activity.

Facade detects insider attackers with an extremely low false positive rate, lower than 0.01%. For single rogue events, such as the illegitimate access to a sensitive document, the false positive rate is as low as 0.0003%. To the best of our knowledge, Facade is the only proposed insider risk anomaly detection system with a false positive rate low enough for use in large corporate environments.

1 Introduction

Insider threats are a fast-growing attack vector that organizations have to contend with. The number of companies experiencing between 20 and 41 insider-led incidents increased from 53% in 2018 to 67% in 2022 [48]. In those incidents, insiders use their *authorized* access to cause harm to their organization, either *wittingly* or *unwittingly* [2]. Larger organizations face more attempts and incidents as risks increase with a more numerous and distributed workforce that is harder to vet, a more complex infrastructure that offers more attack surface, and a more expansive user base that appeals to a larger group of bad actors [48].

With a workforce of over 100,000 full-time employees (and additional temps, vendors, and contractors), one of the largest technical infrastructures in the world, and a multi-billion user-base, Google is one of the most targeted global companies, including by insiders. To protect its user data against threats ranging from nation-state infiltrated employees, malevolent employees trying to sell data on the black market, and well-intentioned employees being tricked or making mistakes; Google heavily invests in detecting and remediating insider threat attacks—including ongoing efforts to innovate and deploy ML-based anomaly detection systems.

Despite the importance of insider threats, there is a scarcity of published research on the subject. This is partly due to the sensitivity of the topic and partly due to the technical difficulty of building robust detection systems.

Existing literature in this domain is often hindered by the use of unrealistic datasets that lack a representative volume of benign activity. Consequently, many proposed systems report false-positive rates (FPR) that are operationally prohibitive for large-scale organizations. Notably, studies in the literature often report FPRs greater than 1% [34]. At our scale, such rates would produce too many spurious alerts to be usable. In large corporate environments, extremely low FPRs are crucial to ensure manageable alert precision. While this approach inherently leads to lower recall and higher false-negative rates (FNR), it is a necessary design trade-off. Without it, alert fatigue and processing bottlenecks would render the system unusable.

To help bridge the gap in the literature, this paper presents Facade, a deep-learning anomaly detection system actively providing a last line of defense against insider threats at Google since 2018. Facade is uniquely capable of detecting anomalies at the single-event level, in contrast to many

†Work done while the author was at Google.

other anomaly detection systems that operate at the volumetric level. This allows Facade to operate and detect low-volume targeted attacks such as exfiltrating high-value documents. Upon deployment, within its first weeks of operation, Facade uncovered multiple high-profile but previously unknown and unsuspected malicious events, including early-stage offensive security reconnaissance activity.

Our design learns the relevant context around access events, using features that indicate what a principal works on and who typically accesses a given resource. Intuitively, Facade learns which resources employees are likely to access and highlights deviations from the norm.

We use a Machine Learning architecture akin to two-tower recommendation systems [15, 52] to learn the relationship between principals and resources in our system. This architecture allows us to embed semantically similar principal-resource pairs (benign activity) close to each other, while the non-similar pairs (anomalies) are far from each other. Facade uses an innovative training technique that enables training even in the absence of incident data.

We evaluate Facade within Google by simulating a diverse set of real-world insider threat scenarios. This simulated attack activity is mixed with the extremely large volume of regular activity happening in the organization – tens of billions of actions per year, even after deduplication. Facade can identify attackers with an extremely low false positive rate, lower than 0.01%. For single rogue events, such as the illegitimate access to a sensitive document, the false positive rate is as low as 0.0003%.

In summary, we make the following contributions:

- We present Facade, the first high-precision contextual anomaly detection system for insider threat detection that can find realistic red team attacks with a false positive rate as low as 0.0003% for single access events.
- We propose a novel contrastive learning strategy that reduces the unsupervised anomaly detection problem to a supervised one. Our technique, *positive sampling*, allows Facade to be trained using only benign data.
- Facade is a multi-modal model that can detect anomalous access across diverse resource types, including documents, data-stores, and internal websites.
- Facade is robust against distribution shift and generalizes to unseen-at-training-time users and resources, eliminating the need for frequent model retraining.
- Facade is a multi-scale detector. In addition to its single-event detection capabilities, it can catch red-team *attackers* with a daily false positive per user budget lower than 0.01%.

2 Background

2.1 Rule-Based Detection

Rule-based systems are a mainstay for security teams. They enable the security engineers to express complex detection logic over the raw features of the application domain. For instance, the Zeek (formerly Bro) network monitoring system [46] supports stateful intrusion detection, and Yara [4] enables threat analysts to author static detection rules to hunt for known Tactics, Techniques, and Procedures (TTPs).

The explicit rules, when tuned to target specific known malicious behaviors, offer excellent precision and recall. However, adversary TTPs, company internal systems, and access controls are all fast-evolving, which makes writing up-to-date detection rules that cover the entire attack surface near-impossible [8].

This is why, in conjunction with a rule-based system, anomaly detection is needed to provide defense-in-depth and limit blind spots. The development of such a system for detecting insider attacks has specific requirements compared to traditional anomaly detection systems.

2.2 Enterprise Level Insider Threat Detection

Insider threat detection is a fundamentally out-of-distribution problem. We expect the next attack to involve different actors, targets, strategies, and overall goals from the known previous attacks. Therefore, insider threat detection *requires catching rare, novel behaviors.*

In this setting, a straightforward supervised learning approach does not succeed at recognizing novelty [12, 28]. This is because supervised learning can only detect patterns that have previously manifested in the training set – a limitation it shares with a rule-based detection systems. The domains where supervised learning is successfully employed are also considerably more predictable: detection of deceptive reviews [43], unwanted behavior on social networks [60], and adversarial advertisements [57].

Traditional anomaly detection often relies on volumetric signals. For instance, an IP hitting significantly more machines or ports than normal can be a sign of network reconnaissance activity. This is unsuitable for our application domain as it will only catch egregious attacks after the fact, such as the exfiltration of a large volume of documents. Such an approach fails to detect low-volume targeted attacks aimed at exfiltrating a single critical piece of data, such as chip design plans, financial information, or proprietary technical designs. Hence, enterprise level insider threat detection requires *detection granularity down to single resource access.*

Lastly, traditional anomaly detection systems infamously suffer from high false alarm rates and security-irrelevant alerts [59]. Since the produced alerts are ultimately reviewed by security analysts who are expensive to hire and train, the

detection system must operate within a stringent alert budget. Unfortunately, even at seemingly low false positive rates, say 0.1%, the large base rate of events in the activity logs of a large-scale company, say 10 million events per day, will result in 10 thousand alerts per day. This is an unmanageable volume for a typical size security team and will quickly result in analysts ignoring the alerts. At scale, the vast amount of legitimate activity together with the rarity of insider attacks *require extremely low false positive detection rates.*

2.3 Research Gaps

Prior work on insider threat detection suffers from data sets with unrealistic benign activity, false positive rates that are unsuitable for large enterprises.

To the best of our knowledge, no publicly available data set is suitable for developing insider threat detection systems in large-scale organizations. The CERT Insider Threat Test Dataset [1, 23] is one of the most pervasively used data sets. It is created by randomly sampling benign user activity under probabilistic assumptions. For any real large-scale enterprise, however, we expect and in practice observe, that the benign behaviors possess a diversity and unpredictability that would be extremely challenging to replicate under the random sampling schemes used for the CERT data set. Other works completely eschew the question of generating realistic benign activity and focus instead on creating richer, more representative attack instances [47, 65].

Unfortunately, a significant number of insider threat works rely on the CERT data set [34,44,49,64]. There, the highly predictable benign activity may result in under-estimated false positive rates in practical large-scale organization deployments. Even without this issue, the reported false positive rates are often on the order of 1%, which would be unacceptable in our environment.

In summary, neither does an appropriately realistic insider threat data set exist for comparison, nor are traditional anomaly detection systems applicable in our environment.

2.4 Self-Supervised Learning

In this section, we give a brief and non-security specific overview of the relevant machine learning techniques used by `Facade`. These techniques fall under the umbrella of *self-supervised learning* [6, 16], which forgoes the need for labeled and representative attack training data, while making use of the entire set of available but unlabeled data.

Contrastive learning [26] is one of the most effective approaches to self-supervised learning. It proceeds by synthetically generating (sampling) a new class of training examples from the existing data points, and thereby reduces an apriori unsupervised learning problem into a supervised one. A supervised classifier can then be trained to distinguish between the two classes, and possibly learn some intermediary representations for the data points, a.k.a. embeddings, in the process.

The two-tower recommendation system [15, 52] is a staple of the item retrieval domain. Each tower maps its input into a fixed-size high dimensional vector, a.k.a. an *embedding*. One tower would typically process a query (e.g. the raw text of the user's search query), the other an item (e.g. a video). For a given notion of distance in the embedding space, these two-tower models are trained such that the embeddings of "matching" query-item pairs are close to each other, while "mismatching" pairs are far from each other.

Negative sampling [41] is a specific form of contrastive learning that is widely used for training large-scale modern recommendation systems. Negative sampling requires a data set of matching query-item pairs (the positive class) and forms synthetic pairs (the negative class) by randomly sampling the elements of the pair. The two-tower model weights are then adjusted so as to push the matching embedding pairs together, and the mismatching embedding pairs away from each other.

3 Research Statement

Operating at a scale of 100,000 employees and billions of daily events, an organization like Google requires a solution that scales not only in data throughput but also in operational viability. At this magnitude, even a false-positive rate of 0.1% would generate millions of daily alerts, effectively burying security analysts in noise. For an anomaly detection system to be useful in such an environment, we need a detection system with low false-positive rates that detects both low-volume targeted attacks and novel threats, in the absence of malicious data for training.

We first approach the problem from the security analyst's perspective. During an investigation, security analysts need to collect all relevant facts surrounding an event. For example, a user viewing a sensitive file can only be properly understood by knowing the the user's role, team, and projects. If the user is viewing a file owned by a team they are collaborating with, the event could be deemed normal; however, if there is no collaboration history between the user or the user's own team, and the target team, the event might be deemed anomalous, justifying further investigation.

This approach is, intuitively, a contextual anomaly detection process: the analyst checks the validity of the event by looking at the context around it. We hypothesize that if we can replicate and scale up the work of the analyst by limiting the context to a relevant and appropriate subset of the state of the world, we will have developed an effective methodology.

To develop this system, we assume that the overwhelming majority of events within the organization are benign. This allows us to effectively apply self-supervised learning trick by creating synthetic, non-existent patterns that represent anomalies, using only the available unlabeled but assumed-benign data.

Our **threat model** for this insider detection system consists of two attack scenarios:

1. A rogue employee abuses their authorized access to critical resources.
2. A non-malicious employee is misled or tricked into accessing critical resources (e.g., when an external party compromises employee credentials and uses them to perform malicious actions).

In this threat model, the rogue or compromised accounts display behaviors that are atypical for an employee with the same role and access. For example, such a compromised account may search corporate documents for valuable digital assets such as development plans or trade secrets [10].

To ensure the practical applicability of such a system in an enterprise environment, we explore the following research question: **Is it possible to develop an insider detection system that detects insider threats across multiple scales, from single-event to days of activity, within an operational budget of ten audits per day, while only using benign data for training?**

In Section 4, we present the design of `Facade`, where we make the assumption that the problem at hand is a supervised learning task. Later, in Section 5 we clarify this point, and explain how we train the system in the absence of labeled attack data, by reducing the unsupervised problem into a supervised learning task. In Section 6, we provide details of our implemented architecture and featurization.

4 `Facade` Design

`Facade` finds anomalies by looking for events that are statistically unlikely given their context. For each event, it produces a score which can rank how unusual that event is.

Our system ingests raw activity logs, and creates featurized events which are then used to train the model. From the featurized events, the model generates anomaly scores. These scores are used in the detection of insider threats.

For the rest of the paper, we use the following definitions:

Principals. The set of all unique human identities in the system, including full-time employees, temporary workers, vendors, and contractors.

Resources. The set of all unique entities accessible by *principals* in the system, such as documents, SQL tables, and network endpoints.

Event. A *principal* interacting with a *resource* at a specific point in time. The environment consists of a continuous stream of *events*.

Action. The facts available to featurize a *resource* at the *event* time. The action is computed using the information about the resource, specifically the set of principals who have previously accessed the resource. Section 6.1 presents the featurization scheme in detail.

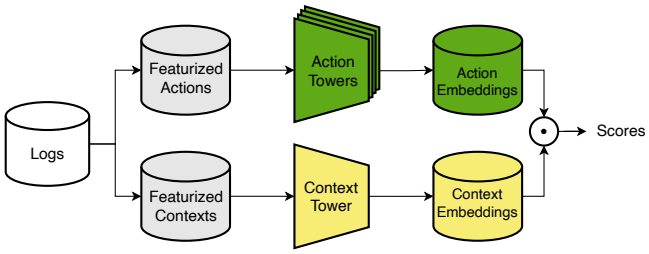

Fig. 1: The core idea behind `Facade`. Events are split into contexts and actions, the model computes embeddings independently for each, and the final anomaly score is obtained as the cosine distance, essentially a dot product, between the context and action embeddings.

Context. The facts available to featurize a *principal* at event time. The context is computed using the information about the principal's profile (e.g., role, team, project assignment) and social interactions (e.g., meetings). The details are provided in Section 6.2.

`Facade` produces anomaly scores for events, which are pairs formed by a context and an action. Let $\mathbb{A}$ be the set of all actions and $\mathbb{C}$ be the set of all contexts. Our goal is to train a machine learning system $f : \mathbb{A} \times \mathbb{C} \to \mathbb{R}$, which maps contexts and actions to anomaly scores. We adapt the two-tower model architecture [15] and decompose f into a *context* tower, $E_C : \mathbb{C} \to \mathbb{R}^d$, and several *action* towers, $E_A^t : \mathbb{A} \to \mathbb{R}^d$, one for each action type, t. While each action tower is a separate deep neural network, for brevity, we use E_A to refer to them collectively, applying the tower of the appropriate action type. These towers map contexts and actions to *context embeddings* and *action embeddings*.

We train the model to embed contexts and actions close to each other if and only if they correspond to a benign event, and define anomaly scores to be the cosine distance score between the embedded action and context, $f(a,c) = d_{cos}(E_A(a), E_C(c))$, so benign and suspicious events correspond to low and high scores respectively. The precise training procedure is explained in Section 5. As a natural consequence of this training, the embeddings capture behavioral semantics: pairs of principals who access similar resources, and pairs of resources accessed by similar sets of principals, will have a with a low cosine distance between their embeddings. Figure 1 summarizes the core idea behind `Facade`.

f can only produce a single score out of each context and action pair. This is useful when focusing on a specific principal and their most suspicious events, or for direct, single-event detection purposes. This detection mode is appropriate for fast detection of egregious-enough single accesses. However, because most attacks consist of multiple suspicious but not necessarily egregious events, attacks can become easier to detect when considering many events together. We thus extend `Facade`'s core single-event scoring capability to score an arbitrary set of events, such as all actions of a principal for a specific time period (e.g., a week).

We introduce two techniques that take advantage of our

embedding space, where cosine distance can be used to measure behavior similarity. First, we present *common event filtering* where we filter behaviorally similar events that were performed by distinct principals, as those are heuristically benign. Then, we discuss *per-principal aggregation* which groups events related to the same potential attacker – highlighting sets of actions containing behaviorally diverse anomalies. This not only reduces the number of tickets that need to be looked at but also collocates activity from a single principal when reviewed by an analyst. Taken together, these techniques reduce system level false positive rates.

4.1 Common Event Filtering

When distinct principals with similar coworkers access similar resources at similar times, there is often a common and justified business purpose that caused those events. We call such events *common events*. We expect f to learn to produce low scores for such events. However, even rare inference errors are visible in the tail at scale.

Since we are interested in malicious activity, we remove high scoring common events from the detection set. This is safe from adversarial manipulation, assuming it is significantly more difficult to compromise multiple principals than to compromise just one. Common event filtering shares this assumption with multi-party authorization systems.

We measure the similarity of coworkers and resources using the embedding distance of context and action embeddings respectively. Formally, given thresholds for similarity $T_a, T_c \in (0,1)$ and multiplicity $m \in \mathbb{N}$, we remove an event (a_0, c_0, p_0) if and only if there exists events $(a_1, c_1, p_1), ..., (a_m, c_m, p_m)$ such that the principals are distinct,

$$\forall p_i, p_j \in \{p_0 ..., p_m\}, \quad p_i \neq p_j$$

the contexts are behaviorally similar,

$$\forall c_j \in \{c_0 ..., c_m\}, \quad d_{cos}(E_c(c_0), E_c(c_j)) < T_c$$

and the actions are behaviorally similar,

$$\forall a_j \in \{a_0, ..., a_m\}, \quad d_{cos}(E_a(a_0), E_a(a_j)) < T_a$$

4.2 Per-principal Aggregation

Single action-context pairs may not be compelling enough to warrant further investigation, and evaluating large sets of related actions amortizes the cost of understanding the principal and resources involved. Hence, we develop the aggregator, $g : \wp(\mathbb{A}) \times \mathbb{C} \to \mathbb{R}$, where $\wp(\cdot)$ is the power set function, which scores arbitrary sets of actions performed by a single principal and facilitates comparisons across principals. An investigator may choose a family of actions to investigate, group the actions by principal, produce an aggregated score for each principal and group of actions, then use the scores to inform further investigation.

We now state three requirements for any such aggregator:
Monotonicity. We require that for a given context, no amount of benign activity can mask the presence of suspicious actions. We therefore introduce a monotonicity requirement on g. Specifically, for any fixed context c, we require the function $A \mapsto g(A, c)$ to be non-decreasing with respect to the set inclusion relation. Formally:

$$\forall c \in \mathbb{C}, \forall A', A \in \wp(\mathbb{A}),\ A' \subset A \Rightarrow g(A', c) \leq g(A, c)$$

This constraint ensures that an attacker cannot manipulate the aggregated score downward by introducing additional actions.
Consistency with Single-Action Scoring. For fixed $A \in \wp(\mathbb{A})$ and $c \in \mathbb{C}$, we require the function $a \mapsto g(A \cup \{a\}, c)$ to follow the score ordering induced by f. Formally:

$$\forall a, a' \in \mathbb{A}, f(a,c) \leq f(a',c) \Rightarrow g(A \cup \{a\}, c) \leq g(A \cup \{a'\}, c)$$

Approximate Idempotence to Behaviorally Redundant Actions. With the above requirements alone, if f always produces non-negative scores, g may be defined to be the sum of individual action-context scores: $g(A,c) = \sum_{a \in A} f(a,c)$. This scoring method is undesirably biased: large action sets $|A|$ will tend to score higher than small ones.

Low-level logs often contain large amounts of behavioral redundancy. For example, loading a single URL can result in a large number of subsequent implied requests. A detection system biased by $|A|$ would have an unacceptably high FPR. Therefore, we require g to discount redundant actions and highlight A when it is a diverse set of anomalous behaviors. Formally, there should exist small constants $\delta, \delta' > 0$ such that:

$$\forall A \subset \mathbb{A}, \forall a' \in , \forall c \in \mathbb{C}, \min_{a \in A} d_{cos}((E_A(a), E_A(a')) < \delta \Rightarrow g(A \cup \{a'\}, c) - g(A, c) \leq \delta'$$

A Clustering-Based Implementation. Our scoring function uses hierarchical agglomerative clustering [42]. We use cosine similarity over the single action embeddings, cosine distance between clusters centroids and specify fixed numbers for the maximum number of clusters and the maximal admissible distance for merging two clusters. Clustering organizes A into set of actions $X \subset G$, and we define the score as the sum of the max scores for each cluster: $g(A,c) = \sum_{X \in G} \max_{a \in X} f(a,c)$

Note that this technique does not perfectly adhere to the requirements above. While it is consistent with single-action scoring, one can construct counter-examples to the monotonicity and approximate idempotence property.

Despite its simplicity and imperfect adherence to the above principles, we find that the clustering approach is effective at detecting insider threats.

When presenting results to security analysts, the clusters are also useful for organizing large sets of actions by behavioral similarity.

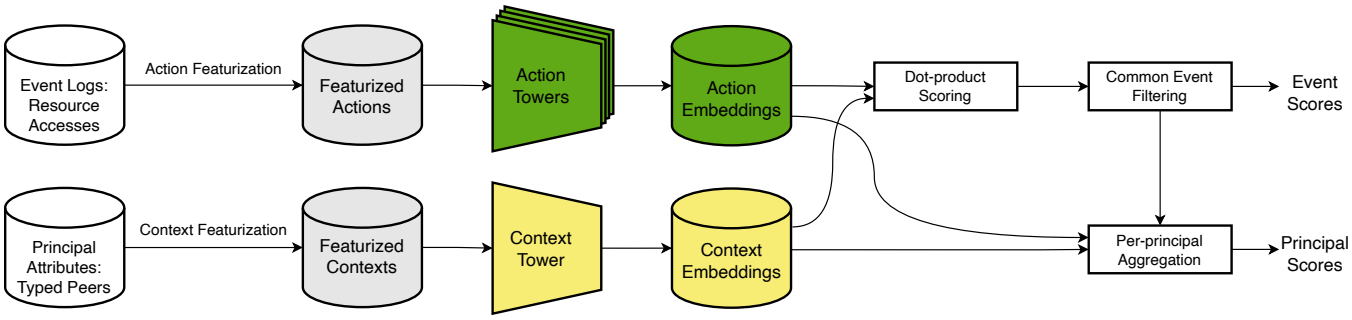

Fig. 2: `Facade` system at inference time. Cylinders are data sets, and trapezoids are (parts of) learned ML models.

4.3 System Summary

We now completed the details of our system for single-event and multi-event levels. Figure 2 depicts `Facade` in the inference time with full details. Event logs are featurized into context and actions. Then, we embed them with context and action towers, yielding context embeddings and actions embeddings. We score the event by taking the embeddings' dot-product. Finally, we apply common event filtering and apply per-principal aggregation to create per-principal scores.

5 `Facade` Training

In our environment, the overwhelming majority of events are benign. While it is possible to identify subsets of events that are considered suspicious or outright malicious, this subset is negligible in volume, and may be filtered when identified.

In what follows, our training data set is comprised only of unlabeled historical events, assumed to be benign. Despite the apparent unsupervised nature of the learning problem, we can cast it into a supervised one using contrastive learning.

Figure 3 presents the `Facade` training process. We now explain the details of, and theory behind, positive sampling.

5.1 Positive Sampling

Let $\mathcal{D} = \{(a_i, c_i, p_i)\}_{0 \le i < n}$ be the set of all available historical events. Each triple i is a single action $a_i \in \mathbb{A}$, from a principal p_i, with their context $c_i \in \mathbb{C}$. To train our classifier, $f : \mathbb{A} \times \mathbb{C} \to \mathbb{R}$, in a supervised way, we borrow an idea from Natural Language Processing (NLP) known as negative sampling [41]. Negative sampling in NLP aims to find the most similar pairs of elements. By contrast, our approach which we call *positive sampling* aims to find the most surprising, or dissimilar action-context pairs.

We create new action-context pairs by randomly mismatching actions and contexts belonging to different principals. Our synthetically labeled data set is defined as:

$$\tilde{\mathcal{D}} = \{(a_i, c_i, -1) \mid 0 \le i < n\} \cup \{(a_i, c_j, +1) \mid 0 \le i, j < n; p_i \ne p_j\} \qquad (\star)$$

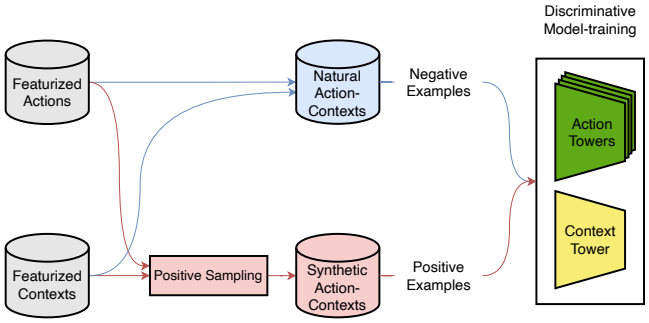

Fig. 3: `Facade` system at training time. `Facade` employs a self-supervised, contrastive training strategy: the model is optimized to differentiate between natural and positive sampled pairs of action-contexts.

We refer to the first set in the union as the negative pairs or natural instances and the second as the positive pairs or synthetic instances.

For example, for a toy data set containing two events $\{(a_1, c_1, \texttt{alice}), (a_2, c_2, \texttt{bob})\}$, we would generate exactly two natural pairs $\{(a_1, c_1, -1), (a_2, c_2, -1)\}$ and two synthetic positive pairs $\{(a_1, c_2, +1), (a_2, c_1, +1)\}$.

As there are generally $O(n^2)$ potential synthetic positive pairs for a data set of n natural events, and n exceeds 10^{10} in our application after data set simplification, we avoid explicitly constructing all positive pairs. Instead, we randomly sample n_p synthetic pairs per natural pair within each training minibatch. We sample $n_p = 10$ positives per negative at training time and $n_p = 1$ at validation time. This is computationally very efficient because the action and context embeddings are computed once over a batch of natural action-context pairs. The scores of synthetic positives are computed by sampling permutations over the context or action embeddings and computing the cosine distances for the resulting pairs.

5.2 Positive Sampling Learns Lift

We theoretically demonstrate that learning on positive sampling results in a semantically meaningful classifier f. Ranking events with f is equivalent to sorting by $\frac{p(a,c)}{p(a)p(c)}$, i.e., the ratio of the joint probability over the product of marginals. This quantity is know as *lift* in the data mining literature [14],

and the result holds independently of model architecture, featurization, and loss function under mild conditions.

For clarity of the proof, we use the following simplifications. First, we randomly and independently sample action and contexts from the natural pairs without verifying that the principals are distinct, i.e., without checking $p_i \neq p_j$ in ($\star$). We call that distribution $\tilde{\mathcal{D}}_{\text{simple}}$. Second, while in general, $\mathbb{A}$ and $\mathbb{C}$ may be infinite and richly-featurized action and contexts spaces, this part assumes they are finite and opaque.

Theorem 1 (positive sampling + pointwise loss $\Rightarrow$ lift)**.** *Let $\mathbb{A} = \{1,...,n_a\}$, $\mathbb{C} = \{1,...,n_c\}$, and P be probability distribution over $\mathbb{A} \times \mathbb{C}$. Let $\ell : \mathbb{R} \to \mathbb{R}$ be a decreasing, strictly convex differentiable function where $\ell'(t) \to 0$ at infinity. A binary classifier $f : \mathbb{A} \times \mathbb{C} \to \mathbb{R}$ which minimizes the expected risk under ℓ, e.g., $\mathbb{E}_{(a,c)\sim P}[\ell(f(a,c))]$ and simplified synthetic positives, $\tilde{\mathcal{D}}_{\text{simple}}$, produces scores such that $\forall a, a' \in \mathbb{A} \forall c, c' \in \mathbb{C}$:*

$$f(a,c) < f(a',c') \Leftrightarrow \frac{P(a,c)}{P(a)P(c)} > \frac{P(a',c')}{P(a')P(c')}$$

where we used the following notational shorthands to denote the marginals of actions and contexts: $P(a) = \sum_{c\in\mathbb{C}} P(a,c)$, $P(c) = \sum_{a\in\mathbb{A}} P(a,c)$.

See Appendix A for the proof and Appendix B for our choice of loss function.

The lift captures a notion of affinity between a context and an action that is independent of the underlying frequencies of either. Applying the definition of conditional probability, $\frac{P(a,c)}{P(a)P(c)} = \frac{P(a|c)}{P(a)}$. We can see that the lift is an action's conditional probability given its context, normalized by the probability of the action. An action being rare, in and of itself, is not a sufficient condition for anomalousness according to a perfectly trained model. It must be relatively rare given the context. Analogously, the lift equals $\frac{P(c|a)}{P(c)}$ so principals with uncommon contexts are not unfairly highlighted by the model, they have to have performed an unexpected action. What matters is the affinity between an action and context pair: The probability of sampling the pair from the joint distribution over sampling them independently from the marginals.

5.3 Robustness to False Positives and Negatives

Robustness to False Positives. Because of the random mismatching procedure, some synthetic pairs can mimic legitimate activity and should consequently be considered as negative pairs. This can happen even when care is taken to only cross pairs that belong to different principals. For instance, we may cross two negative pairs belonging to two employees working under the same team where members fulfill identical job roles. If the employees are functionally indistinguishable, there is no reason to expect a high score for the resulting synthetic pair, and we have effectively introduced a false positive label error. Theorem 1 shows that, asymptotically, these labeling errors do not matter and we still learn the lift.

Robustness to False Negatives. While our training strategy assumes the natural access logs are benign, we know there will be a subset of malicious accesses. Such events will be present in any non-trivial, sufficiently large system. Aside from malicious intent, human and software errors, such as configuration mistakes, induce false negatives in our training data. We show in our evaluation that small amounts of false negatives, both random noise and malicious activity, are empirically tolerable.

We postulate that even when subsets of known previous malicious events are available, a direct supervised learning approach on the labeled data is suboptimal compared to our synthetic positive approach. Our threat model is characterized by rare and novel abusive behavior. We do not expect the historically malicious subset, by itself, to provide enough behavioral diversity to teach a model to generalize to future, out of distribution, malicious events. This argument is independent from the challenge of learning on highly imbalanced labeled data.

Lastly, we describe the implementation of `Facade` and the features we use for contexts and actions.

6 Implementation

The data processing pipelines are implemented in C++ Dataflow [3] and the model is implemented in TensorFlow [5]. `Facade` contains on the order of 10,000 lines of code.

Altogether, creating the training set takes a couple of days and training the model from scratch takes a few hours with one NVIDIA H100 GPU. By contrast, inference is fast. Detection latency is dominated by log delays and batch job scheduling. We also provide on-demand low-latency evaluations of f with slightly stale data for arbitrary, previously seen, resources and principals by caching their most recent embeddings and computing d_{cos} on the fly.

Note that `Facade` can work with a variety of featurizations. We present it as it is implemented at Google.

6.1 Action Featurization

We model every action type of interest as an access to a resource (e.g., a document, a binary file, an HTTP request, a presentation), and require unique resource identifiers. If an event involves more than one resource, for instance a SQL query involving more than one table, we create an atomic action for each distinct resources involved. Because resources and their associated access modality (e.g., view, execute, share) are heterogeneous, we adopt a metadata-only featurization, ignoring both content and access modality, and obtain a general compatibility score between a user and a resource.

Resources, and to some extent access modalities, form an open universe. We cannot simply learn an embedding for each resource identifier, as there would be no such embedding for new resources, which we expect, continuously. Instead, given

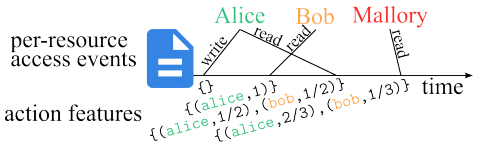

Fig. 4: History-based action featurization for a single resource. Timestamp and access modality are not featurized. Any action is represented by the weighted set of all previous accessors, where the weights are proportional to the frequency of accesses and sum to 1. Note that the features representing the resource changes every time a new access occurs.

a resource and at point in time, we create an embedding as a function of the weighted set of users who have previously accessed the resource, as illustrated in Figure 4.

The open universe of users is considerably smaller and more stable in time than the open universe of resources. This featurization strategy is effective at representing resources as their access control requirements evolve, and it gracefully handles new resources, not present at training time.

When the data is available, `Facade` and history featurization is sufficiently generic that implementing new action types can be remarkably easy. We need only implement a parser that extracts (resource, principal, time) access triples from some authoritative log source, update configuration, and retrain the model. We present results for an implementation which covers the following action types:

- **Cloud-based document accesses:** Documents include presentation slides, spreadsheets and any arbitrary files. Every such item has a unique identifier.
- **SQL-based data accesses:** This covers all registered data sets in a company-wide data lake. We ignore the SQL query statement and use the table's unique identifier. When there are multiple tables in the query, then we create one action per table.
- **HTTP/RPC requests to internal systems:** The resource identifier for HTTP requests is the hostname. For well-known internal sites we *specialize* the resource identifier by parsing the URL. For instance, in the internal issue management service, we use the ticket number. For Remote Procedure Calls (RPCs), the resource identifier is the RPC service and method name pair.

6.2 Context Featurization

Contexts represent what is known about the user who performed the action, at the time of the action. Collectively, these features enable inference of the principal's role, in the spirit of role-based access control systems [55]. They contain a few categorical features that describe non-sensitive, internally-public information about the user. This includes their job family type and their discretized employment duration. The main features are the user's weighted set of peers, representing the implicit social networks revealed by the logs of collaboration tools. For each user, we use:

- **Meeting peers.** The weights are proportional to the number of shared meetings and inversely proportional to the number of meeting participants. Meetings marked as private or confidential are ignored.
- **Code-review peers.**: The weights are proportional to the frequency and recency of code reviews provided or received. This feature describes whether a user is technical, and if so, who they work with.
- **Management peers.** The set of peers who share a manager or grand manager. Peers who only share a grand manager are assigned half the weight of those who share a manager.
- **Cost center peers.** Their cost center peers, weighted equally. Cost centers reflect high-level business areas, which often go beyond grand managers and job titles.

Context features evolve slower than action features. The largest change occurs when a user switches teams or job assignments. Notice that the model f does not rely on the identity of the principal associated with a context at training time. `Facade` trivially handles team changes and new users without model retraining: the model is always fed up-to-date, explicit facts about the principal's role and coworkers. In practice, we refresh the users' context features daily.

6.3 Model Architecture

Figure 5 presents our tower architecture. All towers share the same architecture and input types, though their specific number of layers and neurons are allowed to vary. Each resource type (documents, sql tables, urls) has its own *action* tower.

The context and action towers all take, as input, weighted sets of employee username tokens as presented in Section 6.2. For actions, the tokens represent the users who recently performed the action, weighted by frequency (see Section 6.1).

6.4 Data Set Simplification

Starting from the internal log sources, we separately compute the daily contexts and all actions for all users. To accelerate the data processing pipelines and the learning process itself, we store the context for each principal and one action per distinct accessed resource in each non-overlapping two hour period. We join the contexts and actions in a temporally consistent manner, that is, we pick the context with greatest timestamp earlier than the two hour period in which the actions occur.

We drop actions representing the access to company-wide resources (those accessed by over 2000 distinct principals a

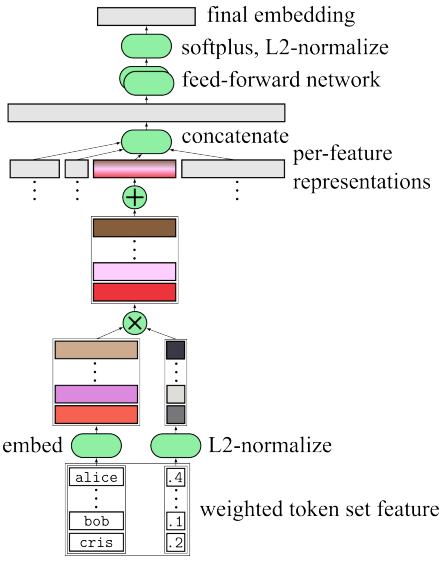

Fig. 5: `Facade` tower architecture. Green rounded shapes denote transformations, and rectangles denote concrete values. The subarchitecture that reduces a variable-length weighted token set feature to a fixed-size intermediary representation is expanded. By convention, our final embeddings are non-negative and L2 normalized.

day) as they are uninteresting from a security standpoint. Similarly, we drop actions with empty histories, which represent the first access to a resource.

Only the natural action-context features are stored on disk because synthetic pairs are sampled from mini-batches at training time.

Lastly, we apply hyperparameter optimization using a spatially and temporally split validation set. These details are presented in appendices F and E, respectively.

7 Evaluation

The novelty of our approach and the problem we are tackling makes it hard to find suitable public data set or related work to compare with. As mentioned in Section 6.1 and Section 6.2, `Facade` relies on histories of access to uniquely identifiable resources in addition to the implicit social network structure of the company employees. Unfortunately, to the best of our knowledge, no publicly available data set is available with such characteristics.

To perform an extensive evaluation of `Facade`, we tested it in the real-world Google environment. This environment is orders of magnitude larger in the number of users, resources, and access events than any public data set; with more than 100,000 full-time employees, additional temps, vendors, and contractors; and 10 billion yearly events. Most importantly, it displays the full, messy, long-tailed behavior of a large enterprise. We used a company internal program where volunteers spend two weeks performing an attack (under supervision). Our system is evaluated on the task of retrieving these attackers' activity against the background of all other activity in the company.

Table 1: Number of Attack-Related Events

Target \ Type	Docs	Tables	HTTP/RPC	Total
Media Sharing Platform	21	20	127	168
AI Research	60	5	260	325
Hardware Product	123	8	176	307
Security Research	54	3	123	180
Total	258	36	686	980

We asked volunteers to try to access sensitive information outside of their normal area of work, mimicking real-world insider threat scenarios. Participants were instructed to choose and emulate a profile from a playbook containing the following attacker goals. In all cases, the attackers seek to discover and access sensitive information about a particular topic.

- **Media Sharing Platform**: Attackers seek information about the inner workings of the platform. This information could include corporate financial data, earnings for individual creators, or ranking algorithms.
- **Hardware Product**: Attackers seek information about the company's next generation device, involving obtaining planning documents, design documents, timelines, pictures, or schematics.
- **AI Research**: Attackers seek information on next generation AI research and development projects. This includes documentation pertaining to current and future research projects, unpublished papers, source code, and model weights.
- **Security Research**: Attackers target internal security research. This includes undisclosed (zero-day) vulnerabilities, either within the company or in third-party systems, which they could potentially exploit or sell to nation-state adversaries; as well as internal security analysis tools and associated documentation.

We deliberately did not provide detailed attack plans nor specific resources to target. This approach relies on the participants' creativity and curiosity, while ensuring a minimum level of independence between our system and its evaluation.

All fifteen participants were full time employees at Google with a demonstrated interest in cybersecurity. Most of them were software engineers and all were familiar with internal tools and information storage practices. Each scenario involved up to four participants, and collaboration between attackers was both allowed and encouraged. As we are simulating insider threats, participants were not instructed to perform infiltration. They also were instructed not to exfiltrate any data, as per company policy. They simply began accessing sensitive data using their preexisting credentials, recorded their actions, and provided an after-action report.

Table 2: Training set characteristics.

Covered time period	1 year
Distinct human principals	$\in [10^5, 10^7)$
Distinct resource identifiers	$\approx 10^8$
Actions before simplification	$\approx 10^{13}$
Actions after simplification	$\approx 10^{10}$
Compressed TFExamples	≈ 3 Tb

To the best of our knowledge, an attack simulation and detection system evaluation with this level of environmental and attacker realism, and scale, is unprecedented in the published literature. Using the attack report, investigators manually discovered and recorded on the order of 1000 events in the `Facade` system that directly correspond to the attack. Though they did not need to, some attackers performed some actions related to their usual responsibilities. Actions such as these, and those that are only ambiguously related to the attack, are not counted here. Table 1 breaks the attack events down by target and action type. These are the same action types explained in Section 6.1.

7.1 Single Event Detection Performance

We train our baseline model using one year of data ending 90 days before the attack exercise begins. We select hyper parameters using a spatially and temporally split training and validation set, as per Appendix E and F. The validation set is the two week period before the attack exercise, i.e, there are 76 days of temporal separation. After hyper parameters are selected, we train a model without spatial splitting for evaluation. Table 2 summarizes key numerical characteristics of our training set.

We use data gathered from the attack simulation and evaluate a simplified model of our detection system: A team of security analysts who audit a fixed proportion of events, those with the highest `Facade` scores. We assume, for simplicity, that analysts will recognize attack events when they audit one.

The proportion of events audited is an upper bound on the false positive rate (FPR) of the system, and the prevalence of attack events is small, so they are approximately equal. Hence, we think of FPR and 'requisite audit proportion to discover an event' as interchangeable for practical purposes. We present receiver-operating characteristic (ROC) curves, which compare FPR and recall, to describe our retrieval of attack events.

While higher recall (ceteris paribus) is better, we focus on the low recall and low FPR region as successfully detecting a single attack event will trigger an investigation that can detect the rest of the events. Additionally, many of the low-scoring attack events are relatively common and innocuous. They include looking up potential victims who work in the target organization, using Google's internal company directory. This kind of reconnaissance activity, without additional information from other events, is difficult to identify as attack related for both analysts and `Facade` alike.

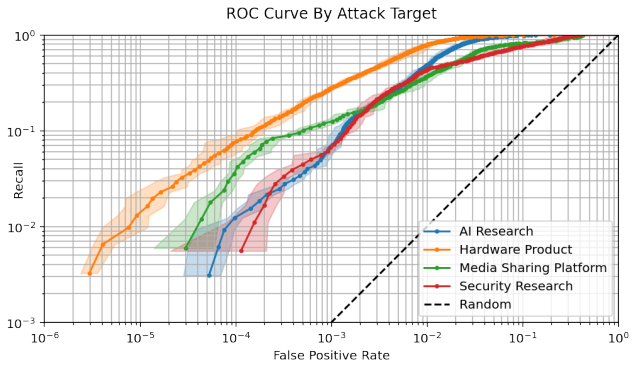

Fig. 6: Simulated Attack ROC curves, broken down by attack target. The baseline model is trained 5 times. We take the 5 ROC curves and for each TPR, plot the mean associated FPR and shade the region between the minimum and maximum FPRs.

Figure 6 demonstrates that analysts can detect four attack events, all targeting Hardware Product, while only auditing 1 in 10^5 events, as there are 4 points below 10^{-5} FPR. If 1 in 10^4 events are audited, we detect almost 10% of the Hardware Product attack events and detect three of the four attack targets. The highest ranked attack action has an FPR below 0.0003%. That is detectable with an extremely low, 3 in 10^6, audit proportion.

7.2 Common Event Filtering

Figure 7 illustrates the effect of common event filtering (introduced in Section 4.1) on single event detection varies by action type. Attack related document access events are already discoverable with f alone using a 1 in 10^5 audit proportion. Consequently, common event filtering yields no noticeable enhancement for this particular action type. Retrieval of attack-related SQL table access events is improved slightly, lowering the requisite audit proportion for detection by almost a factor of 2. Common event filtering most significantly affects retrieving attack-related HTTP/RPC requests, reducing FPRs at the low end by a factor over 20x. It is required to retrieve such events with a 1 in 10^4 proportion used by the other types.

7.3 Attacker Detection Performance

Given the scale of 10^{10} events per year outlined in Table 2, security analysts would need to audit thousands of events per day to reach the 1 in 10^4 audit proportion required to detect most targets of the simulated attack.

Additionally, because the top ranked events are generally unrelated, analysts are fatigued by frequent context switching. Overall, we find the continuous auditing of single events, by itself, prohibitively expensive for use in production. In practice, the audit volume may be reduced with additional rules and signals which are not considered here.

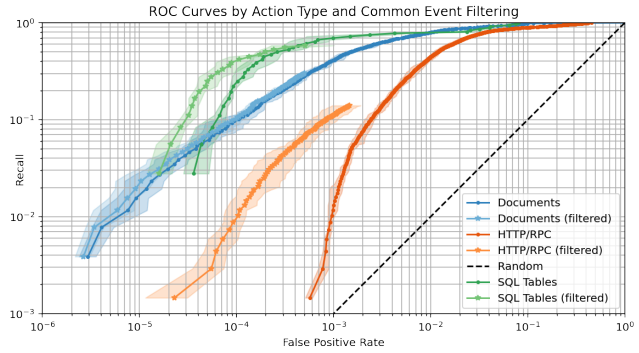

Fig. 7: Simulated Attack ROC curves, broken down by action type and whether common event filtering is applied.

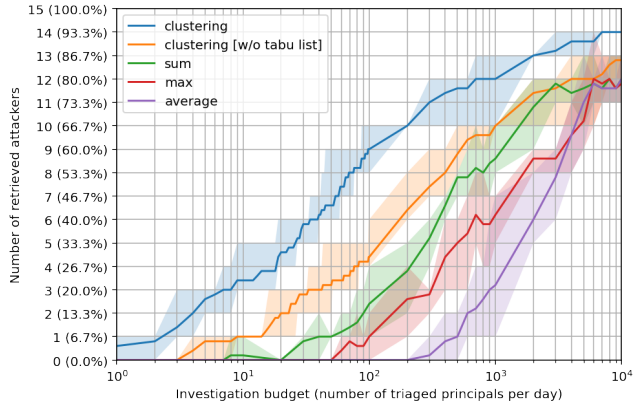

Fig. 8: Number of retrieved attackers as a function of the daily investigation budget. For each aggregation strategy, a solid line denotes the mean over five distinct models; the minimum-maximum band is also shown.

Hence, we evaluate another simplified model of our detection system that ranks *principals* every day. Security analysts audit the K highest ranked principals who were not audited in the preceding 7 days. For simplicity, we assume attackers are detected if they're audited. We rank a principal, p, using $g(c_p, A_p)$ where A_p is the set of all actions performed by p in the preceding 7 days, after common event filtering.

For comparison, we relax our idempotence to behaviorally redundant actions requirement (Section 4.2) and consider simpler aggregation methods of the form: $h_s(A,c) = s(\{f(a,c) : a \in A\})$ where $s \in \{\max, \text{average}, \text{sum}\}$, applied after post filtering. Notably, these aggregations do not leverage behavioral embeddings and averaging does not meet our monotonicity requirement either.

Figure 8 demonstrates that our clustering-based aggregator can detect 3 to 4 of the 15 attackers at a cost of 10 audits per day. This is remarkably low for a company with over 100,000 full time employees and additional temporary workers, vendors, and contractors. The alternative aggregation methods require orders of magnitude larger auditing budgets to achieve similar results. They are even sometimes non-monotonic as attackers are spuriously audited during a period without attack activity (which does not count as detection) and are prevented from being correctly audited for the subsequent week.

Preventing recently audited principals from being re-audited, à la Tabu Search [24], is usually helpful. Principals' ranks exhibit strong auto-correlation across days because we apply aggregated scoring over overlapping windows of activity. Preventing re-auditing uses security analyst time more efficiently. We see in Figure 8 that ablating the tabu list meaningfully reduces attacker recall at every investigation budget size. Though, it still outperforms the naive aggregation techniques with tabu lists.

7.4 Experiments

Finally, we conduct experiments to evaluate our various claims about f and show the results in Figure 9.

Resilience to Distribution Shift. We study the model's performance in the inductive setting and the effect of distribution shift in our model age experiment. We vary the length of the cooling-off period between when the model's training data ends and when the simulated attack begins. We train additional models whose cooling-off periods are 0, 180, and 365 days. Figure 9 demonstrates that, while newer models tend to perform slightly better in the FPR region between 10^{-4} and 10^{-3}, there is similar performance in the FPR region below 10^{-4}, where detection happens. It is remarkable that even in an environment as dynamic as Google with many team reorganizations, role transfers, new and cancelled projects, we can run a 1 year old model without significantly degraded detection performance.

Spurious Negatives in Synthetic Positives. We evaluate the claim that our model can tolerate spuriously generated natural pairs during positive sampling in our false positive noise experiment (Section 5.3). We intentionally convert random negative pairs into positive pairs until ϕ% of the final positive pairs are drawn from the negatives. Figure 9 demonstrates that false positive noise up to $\phi = 5\%$, not only does not diminish our ability to retrieve attack events, but seems to improve it. The mean FPR of the highest scoring attack event is below 10^{-6}. We believe the improvement is related to label smoothing [61], which regularizes the model's overconfidence in the negative events.

Next, we study the claim that `Facade` can tolerate some non-business-justified activity in the natural events (Section 5.3).

Robustness to Random False Negatives. In our false negative noise experiment, we study the effect of *random* false negatives by intentionally converting random positive pairs into negative pairs until θ% of the final negatives come from synthetic positives. Figure 9 shows `Facade` can detect some attack events while auditing just 1 in 10^5 events when $\theta < 1\%$. However, performance clearly decreases as θ increases. $\theta \geq 1\%$ is roughly equivalent to an environment in

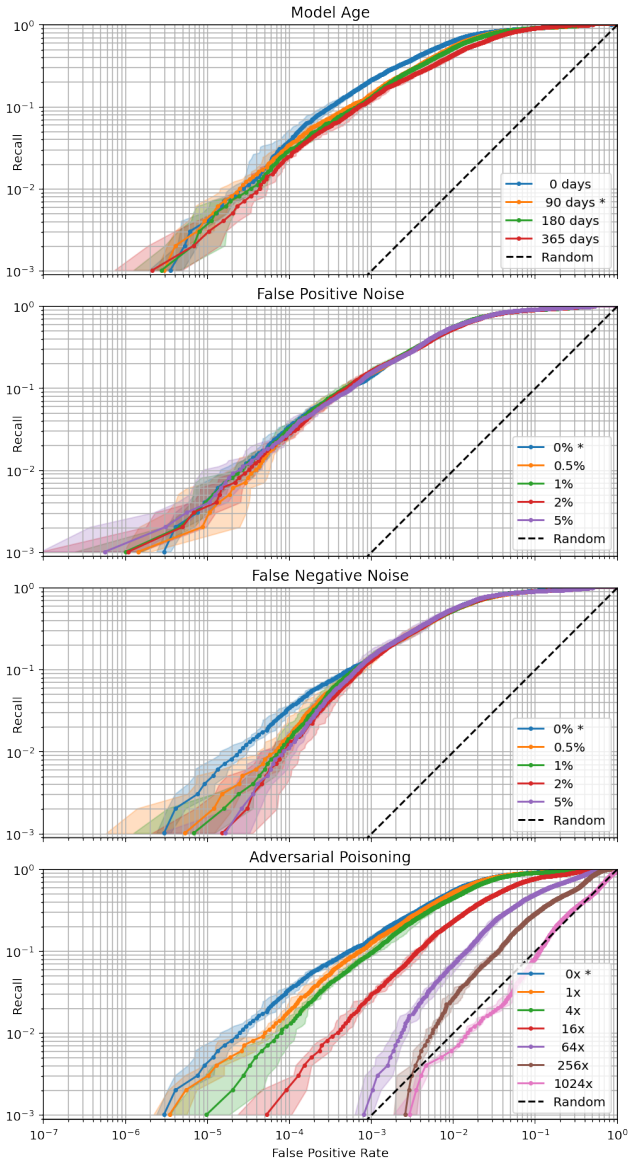

Fig. 9: Simulated Attack ROC curves, for each experiment and experiment variation. Each model variant is trained 5 times. We plot the mean ROC curve and shade the region between the minimum and maximum. The variations that correspond to the baseline model are marked with *.

which at least 1% of the logs incorrectly attribute events to random principals, which is generally considered unacceptable under normal operations.

Robustness to Adversarial Poisoning. In our adversarial poisoning data experiment, we anachronistically replicate the events from the simulated attack into our training data set and mislabel them as natural. This studies the worst-case effects of access abuse present in the historic access logs. Figure 9 shows that we can still detect the attack simulation with a 1 in 10^5 audit proportion if the attack events are replicated into in the training data once or even up to four times. At a 1 in 10^4 proportion, attack events may be replicated up to 16 times. However, retrieval is worse than random if the attack events are replicated 256x or more.

We conclude that our assumption that most events in the historical data are benign is critical to `Facade`'s retrieval capabilities. Some false negatives, random or malicious, may be present. So long as the proportion is sufficiently small, f can still be used to retrieve attack events.

7.5 Evaluation Summary

By using only historical data of normal business activity, contrasted against sampled synthetic positives, and no ground truth attack data, we can train a model capable of retrieving attack events in a real-world, large company environment.

`Facade` retrieves both events and principals associated with a simulated insider attack with an extremely low false positive rate, below 0.01%. This performance is maintained even 1 year after model training and when we intentionally violate our benign historical data assumption with small quantities of random noise or poison data. The false positive noise in our synthetic positive sampling is not only tolerable, but seems to even improve performance. The mean tail FPR for $\phi = 5\%$ is less than 10^{-6}. This suggests we should research incorporating label smoothing [61] and weak supervised learning [50]. In addition to improving baseline performance, we believe it may let us relax our benign data assumption.

We evaluated two uses of our behavioral embeddings. First, common event filtering over single events can dramatically lower false positive rates, as with HTTP/RPC actions which have relatively weak baseline performance. Second, we make our aggregator idempotent to behaviorally redundant actions, which enables the detection of *attackers* at extremely small auditing budgets, below 0.01% of our workforce per day.

8 Limitations

Recall vs. False Positive Rate. There is a trade-off between recall and false positive rate, as shown in our evaluation using ROC curves. Increasing recall at the expense of a higher false positive rate could increase the number of detected attacks, but at the cost of requiring more security analysts and financial resources. Ultimately, the decision depends on an organization's budget and its risk tolerance.

Benchmark. The sensitive nature of the data makes it impossible to share with the public. Furthermore, there are no suitable public benchmarks available. Consequently, we are unable to either share our data or utilize a benchmark for comparison. This is a limitation of a realistic evaluation in a production corporate environment rather than a constraint specific to this research.

Prior Work. Unfortunately, we do not have a point of comparison for our system. However, we believe this work will be a valuable contribution to the field of enterprise insider threat

detection. To this end, we open sourced Facade's model code, along with a reference implementation of our action and context featurization pipelines. This enables implementations of Facade outside of Google and establishes a baseline for future insider detection systems.

Contextual Detection Gap. Because events are contextualized per user, Facade's main limitation is that it is not able to detect malicious actions that are indistinguishable from everyday user actions. This makes it unsuitable for detecting a rogue employee's access to sensitive files that are directly related and relevant to their work. Google relies on other defense mechanisms to address such threats.

Generalizability of History Featurization. History featurization requires a unique key to identify actions on the same resource over time. In practice, we find Facade to be more successful for documents and SQL tables than for HTTP and RPC services. This may be because hostnames (and specialized URL parsing) are suboptimal choices for history key. Devising a better key system for those and for security-relevant actions which do not have obvious keys, such as ad-hoc CLI invocations, remains an open question.

Non-human identities. Facade primarily featurizes principals by their peers, as identified by meetings, code review patterns, and the HR reporting chain. This assumes that they are humans with coworkers and does not apply to non-human identities such as service accounts or AI agents. While we believe our technique, using contrastive learning with positive sampling, may be adapted to the non-human domain, effective featurization of such identities also remains an open question.

Adversarial Robustness. To decrease a single-event score, attackers may use training or testing time strategies. In the simplest training-time strategy, the attacker may repeatedly perform an action in the hopes of polluting the training set and in turn poisoning the model. Section 7.4 quantifies Facade 's robustness to this type of attack: the system tolerates some adversarial noise before eventually degrading. In testing-time attacks, the adversary manipulates their context and action features so as to decrease the score of the event. Our particular choice of context features make it possible for an attacker to partially manipulate their content, for instance by scheduling meetings or performing code reviews with specific, targeted peers. However, we posit that the publicly-visible and social nature of such actions create a barrier for the attacker: creating meetings with or performing code reviews for highly-unusual peers might raise questions and trigger additional scrutiny for the attacker. Finally, on the action side, an actor repeatedly accessing the same resource will decrease the anomaly score, as illustrated in Figure 4, though the first access still scores highly. As a last line of defense, the Facade system is one among many other internal detection mechanisms: attempts to evade detection may make the attacker paradoxically more detectable by more regular, rule-based systems.

9 Related Work

We now discuss the prior work that contributed similar techniques and capabilities in the field of anomaly detection applied to access abuse.

Single event detectors. Gafny *et al.* address the problem of anomalous resource access by learning invariant relationships between resource and principal features that are verified for every access [18]. This uses frequent itemset mining on categorical features [7].

Gelven *et al.* consider the problem of anomalous principal-resource access detection, where both principals and resources are opaque nodes of a bipartite graph [22]. The system uses matrix factorization on historical usage data to perform link prediction, in a classic collaborative filtering fashion. Facade draws from two-tower deep recommendation systems [15], allowing for rich featurizations of both principals and resources.

Zheng *et al.* also observe the structural similarity between anomaly detection and recommendation systems [67]. They apply graph neural network over provenance graphs and retrieve low probability interactions between system entities. Facade operates at a higher level of abstraction, reasoning about authenticated users and distributed resources in a corporate environment, rather than processes and files. Another difference is that Facade uses features entities with weighted sets of tokens, as opposed to per-entity embedding vectors, so it can better featurize out-of-distribution entities.

Detection of Anomalous Batches of Events. Many anomaly detection systems operate solely on batches of activity (e.g., login sessions, user-days, user-weeks) and rely on aggregate features, typically counts of events. These numerical features are in turn fed to traditional unsupervised learning techniques. Good examples include principal component analysis on network traffic patterns [29] and auto-encoders on network flows [31]. Closer to insider threat detection, several studies [54, 58] train classical statistical anomaly detection models, such as one-class SVM [39], on a per-user activity basis, using aggregated event counts and other volumetric behavioral features. The system described by Liu *et al.* detects anomalous user-day batches, and also produces per-event anomaly scores [36].

Contemporary work uses deep-learning models on either manually-defined count-like features describing the overall activity session [21, 32, 37] or directly on the sequence of actions types [38, 63, 66, 68], without considering the specific resources accessed.

Unlike the above, Facade can evaluate the anomalousness of *individual* events, and represents resources by their access history. It also can aggregate sets of events to achieve low FPR detection over batches of activity.

Detection by Aggregating Single Events. Hassan *et al.* designed an algorithm applied to arbitrary detection and alerting systems that, given a provenance graph, aggregates alerts using that graph [27]. Asaheel *et al.* applied sequence models

atop provenance graphs to extract an 'attack story' from the larger graph [9]. It requires labeled data of previous attacks. Facade does not require an explicit model of relationships between events. It can perform detection over both individual events and sets of events associated with a single principal.

Feature Representation. Gates *et al.* design an anomalous file access detector [20]. This work is file system specific, though like Facade, they use users' access history as features.

Some systems maintain per-user, variable size non-ML profiles, and compare the daily activities against those using ad-hoc hierarchical or graphical data representations [33, 45, 62]. Contemporary work uses deep-learning models on either manually-defined count-like features describing the overall activity session [21, 32, 37] or directly on the sequence of actions types [38, 63, 66, 68], without considering the specific resources accessed. Scores from Liu *et al.* 's system [35] are derived from the word2vec-learned [41] representations of the log entries and are not evaluated for separate uses.

Mathew *et al.* adopt a learned, per-role profile context for detecting improper data access at the database level [40]. Their features are derived from the SQL query response itself, which we consider to be content-based, as opposed to our history based featurization.

10 Conclusion

We have presented Facade, a deep contextual anomaly detection system that helps protect Google against malicious or unintentional insider threats with an extremely low false positive rate. Three techniques enable Facade to surface fundamentally novel insider threats with very high precision at scale.

First, positive sampling forgoes the need for expensive attack data collection for training purposes. Instead, training exclusively relies on cheap and readily available historical activity logs. Second, Facade works across resource types (arbitrary cloud documents, data-stores as they appear in SQL-based queries, and HTTP/RPC requests to internal network resources) and generalizes to unseen-at-training-time users and resources. This is achieved by using historical access-based and peers-based featurization strategies of actions and contexts, which in turn eliminates the need for frequent model retraining. Third, Facade embeddings enable measuring the behavioral similarity of principals and resources, and unlock simple yet powerful filtering and aggregation methods.

To our knowledge, Facade is the first proposed contextual anomaly detection system for insider threats with sufficient precision to be deployed in a large-scale environment.

Acknowledgments

We gratefully acknowledge the contributions of the following individuals, whose support, feedback, and dedication were essential to this research: Heather Adkins, Aman Ali, Nicholas Capalbo, Ceri Driskill, Marius Killinger, Louis Li, Peng Liu, Justin Ma, Tom MacGregor, Eric Morley, Michael Sinno, Johan Strumpfer, and Zachary Westrick.

Ethical Considerations

We are committed to the responsible and fair usage of machine learning. By detecting insider threats, the system participates in the responsible handling of the end user data of billions and the protection of the company's intellectual property.

Our commitment to employee privacy and fairness constrains what data we allow ourselves to use: In Section 6.1, only action types that involve potentially sensitive activities on the corporate systems are instrumented. In Section 6.2, we derive context features from internally-public information. In particular, the collected data is limited to the corporate environment and excludes any personal accounts or general browsing activity, and we only use meeting data where the meeting is set to be visible to anyone internally: private or hidden meetings are excluded.

Additionally, we check for the presence of bias by empirically validating that certain categories of employees are not over-represented in the final detection results. We make our methodology public inside the corporate environment. Model weights and feature values are kept internally private.

Open Science

We fully support the goals of open science and the principles of reproducibility and shared community advancement. To this end, we are committed to open sourcing the parts of Facade that we can, including the code that defines, trains, and evaluates our model. Therefore, we open source the core Facade model code and the code for constructing input features*. The open sourced version is implemented in the Python language as a reference implementation.

This paper also provides comprehensive descriptions of our techniques, model architecture, and feature engineering. We believe these artifacts will be valuable to the research community, enable others to implement similar systems, and facilitate future innovations.

However, Google's business considerations are such that we are unable to share the version of the Facade model that is evaluated in this paper, which is trained on Google's internal corporate data. The same considerations apply to the underlying training and evaluation data, as well as the data collection pipelines themselves. These artifacts cannot be directly used to secure systems outside of Google and contain sensitive information that could be exploited for purposes that do not align with our business.

* https://github.com/google/facade
https://zenodo.org/records/20316179 (historical record)

For example, while the number of full time employees is public information, Google does not disclose the number of temporary employees, vendors, or contractors that it hires. This makes the exact number of principals confidential information. As a result, in Table 2, we reveal the number as a two order of magnitude range. We featurize both actions and contexts with weighted sets of usernames. Releasing the trained model reveals this confidential information. Business restrictions aside, these usernames are internal to our company, limiting the public value of releasing the trained model.

`Facade` is trained on historic access logs. We cannot release this training data because such logs reveal sensitive internal information; including the number of documents we produce, their rate of production, and their rate of access. There are similar concerns for SQL tables and internal URL access data too. If the principals are deanonymized, access data reveals Google's relative investment and staffing into different business areas, which is considered highly confidential information.

Finally, our data collection pipelines are tightly coupled with Google's internal logging infrastructure. Sharing the code provides little value to the scientific community as it will be unusable outside of Google. Regardless, data pipelines are not what makes our contributions novel.

References

[1] Cert insider threat test datasets. https://insights.sei.cmu.edu/library/insider-threat-test-dataset/. Accessed: 2023-12-06.

[2] Defining insider threats. https://www.cisa.gov/topics/physical-security/insider-threat-mitigation/defining-insider-threats. Accessed 2024-09-03.

[3] Google cloud dataflow. https://cloud.google.com/dataflow. Accessed: 2024-09-03.

[4] YARA: The pattern matching swiss knife for malware researchers. http://virustotal.github.io/yara. Accessed: 2024-09-03.

[5] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

[6] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *Proceedings of the IEEE international conference on computer vision*, pages 37–45, 2015.

[7] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[8] Md Liakat Ali, Kutub Thakur, Suzanna Schmeelk, Joan Debello, and Denise Dragos. Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study. *Applied Sciences*, 15(4):1903, 2025.

[9] Abdulellah Alsaheel, Yuhong Nan, Shiqing Ma, Le Yu, Gregory Walkup, Z. Berkay Celik, Xiangyu Zhang, and Dongyan Xu. ATLAS: A sequence-based learning approach for attack investigation. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3005–3022. USENIX Association, August 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/alsaheel.

[10] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[11] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005. doi:10.1109/CVPR.2005.202.

[12] M Bishop Christopher. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[13] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8765–8775. Curran Associates, Inc., 2020.

[14] David S Coppock. Why lift? data modeling and mining. *Information Management Online*, pages 5329–1, 2002.

[15] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 191–198. Association for Computing Machinery (ACM), 2016.

[16] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[17] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2012–2020. PMLR, 2019.

[18] Ma'ayan Gafny, Asaf Shabtai, Lior Rokach, and Yuval Elovici. Poster: applying unsupervised context-based analysis for detecting unauthorized data disclosure. In *Proceedings of the 18th ACM conference on Computer and communications security (CCS)*, pages 765–768, 2011.

[19] Wei Gao and Zhi-Hua Zhou. On the consistency of auc pairwise optimization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 939–945. AAAI Press, 2015.

[20] Christopher Gates, Ninghui Li, Zenglin Xu, Suresh N Chari, Ian Molloy, and Youngja Park. Detecting insider information theft using features from file access logs. In *19th European Symposium on Research in Computer Security (ESORICS*, pages 383–400. Springer, 2014.

[21] RG Gayathri, Atul Sajjanhar, and Yong Xiang. Image-based feature representation for insider threat classification. *Applied Sciences*, 10(14), 2020.

[22] Grant Gelven and Shannon Strum. Graph-based user-entity behavior analytics for enterprise insider threat detection. In *6th Conference on applied machine learning for information security (CAMLIS)*, 2023.

[23] Joshua Glasser and Brian Lindauer. Bridging the gap: A pragmatic approach to generating insider threat data. In *2013 IEEE Security and Privacy Workshops*, pages 98–104. IEEE, 2013.

[24] Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[25] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley, editors. *Google Vizier: A Service for Black-Box Optimization*, 2017.

[26] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[27] Wajih Ul Hassan, Shengjian Guo, Ding Li, Zhengzhang Chen, Kangkook Jee, Zhichun Li, and Adam Bates. Nodoze: Combatting threat alert fatigue with automated provenance triage. *Network and Distributed Systems Security Symposium*, 2019. URL: https://par.nsf.gov/biblio/10085663.

[28] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag New York, 2001.

[29] Ling Huang, XuanLong Nguyen, Minos Garofalakis, Michael I Jordan, Anthony Joseph, and Nina Taft. In-network PCA and anomaly detection. In *Advances in Neural Information Processing Systems*, NIPS '07, pages 617–624, 2007.

[30] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi:10.1214/aoms/1177703732.

[31] G. Kathareios, A. Anghel, A. Mate, R. Clauberg, and M. Gusat. Catch it if you can: Real-time network anomaly detection with low false alarm rates. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*, ICMLA '17, pages 924–929, 2017.

[32] Duc C Le and Nur Zincir-Heywood. Anomaly detection for insider threats using unsupervised ensembles. *IEEE Transactions on Network and Service Management*, 18(2):1152–1164, 2021.

[33] Philip A. Legg, Oliver Buckley, Michael Goldsmith, and Sadie Creese. Automated insider threat detection system using user and role-based profile assessment. *IEEE Systems Journal*, 11(2):503–512, 2017.

[34] Ximing Li, Xiaoyong Li, Jia Jia, Linghui Li, Jie Yuan, Yali Gao, and Shui Yu. A high accuracy and adaptive anomaly detection model with dual-domain graph convolutional network for insider threat detection. *IEEE Transactions on Information Forensics and Security*, 18:1638–1652, 2023.

[35] Fucheng Liu, Yu Wen, Dongxue Zhang, Xihe Jiang, Xinyu Xing, and Dan Meng. Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (CCS)*, pages 1777–1794, 2019.

[36] Liu Liu, Chao Chen, Jun Zhang, Olivier De Vel, and Yang Xiang. Insider threat identification using the simultaneous neural learning of multi-source logs. *IEEE Access*, 7, 2019.

[37] Liu Liu, Olivier De Vel, Chao Chen, Jun Zhang, and Yang Xiang. Anomaly-based insider threat detection using deep autoencoders. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 39–48. IEEE, 2018.

[38] Martin Macak, Ivan Vanát, Michal Merjavỳ, Tomáš Jevočin, and Barbora Buhnova. Towards process mining utilization in insider threat detection from audit logs. In *2020 seventh international conference on social networks analysis, management and security (SNAMS)*, pages 1–6. IEEE, 2020.

[39] Larry M Manevitz and Malik Yousef. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2(Dec):139–154, 2001.

[40] Sunu Mathew, Michalis Petropoulos, Hung Q Ngo, and Shambhu Upadhyaya. A data-centric approach to insider attack detection in database systems. In *Proceedings of the 13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 382–401. Springer, 2010.

[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Advances in neural information processing systems*, NIPS '13, pages 3111–3119, 2013.

[42] Nick Monath, Avinava Dubey, Guru Prashanth Guruganesh, Manzil Zaheer, Amr Mahmoud El Houssieny Ahmed, Andrew McCallum, Gokhan Mergen, Marc Najork, Mert Terzihan, Bryon Tjanaka, Yuan Wang, and Yuchen Wu. Scalable hierarchical agglomerative clustering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 1245–1255, 2021.

[43] Arjun Mukherjee, Bing Liu, and Natalie Glance. Spotting fake reviewer groups in consumer reviews. In *Proceedings of the ACM international conference on World Wide Web*, WWW '12, pages 191–200, 2012.

[44] Preetam Pal, Pratik Chattopadhyay, and Mayank Swarnkar. Temporal feature aggregation with attention for insider threat detection from activity logs. *Expert Systems with Applications*, 224:119925, 2023.

[45] P. Parveen, J. Evans, B. Thuraisingham, K. W. Hamlen, and L. Khan. Insider threat detection using stream mining and graph mining. In *Proceedings of the IEEE International Conference on Privacy, Security, Risk and Trust*, pages 1102–1110, 2011.

[46] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999.

[47] Ricardo Ribeiro Pereira, Jacopo Bono, João Tiago Ascensão, David Aparício, Pedro Ribeiro, and Pedro Bizarro. The GANfather: Controllable generation of malicious activity to improve defence systems. In *4th ACM International Conference on AI in Finance*, pages 133–140, 2023.

[48] Ponemon Institute. 2022 cost of insider threats global report. https://protectera.com.au/wp-content/uploads/2022/03/The-Cost-of-Insider-Threats-2022-Global-Report.pdf, 2022. Accessed: 2024-09-03.

[49] Krunal Randive, R Mohan, and Ambairam Muthu Sivakrishna. An efficient pattern-based approach for insider threat classification using the image-based feature representation. *Journal of Information Security and Applications*, 73:103434, 2023.

[50] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282, November 2017. URL: http://dx.doi.org/10.14778/3157794.3157797, doi:10.14778/3157794.3157797.

[51] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021.

[52] Chuanwei Ruan, Allan Stewart, Han Li, Ryan Ye, David Vengerov, and Haixun Wang. Dynamic embedding-based retrieval for personalized item recommendations at Instacart. In *Companion Proceedings of the ACM Web Conference 2023*, page 983–987, New York, NY, USA, 2023. Association for Computing Machinery.

[53] Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 412–419. PMLR, 2007.

[54] Malek Ben Salem and Salvatore J. Stolfo. Modeling user search behavior for masquerade detection. In *Proceedings of the International Workshop Recent Advances in Intrusion Detection*, RAID '11, pages 181–200, 2011.

[55] Ravi S. Sandhu. *Role-based Access Control*, volume 46 of *Advances in Computers*. Elsevier, 1998. URL: https://www.sciencedirect.com/science/article/pii/S0065245808602065, doi:10.1016/S0065-2458(08)60206-5.

[56] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[57] D. Sculley, Matthew Eric Otey, Michael Pohl, Bridget Spitznagel, John Hainsworth, and Yunkai Zhou. Detecting adversarial advertisements in the wild. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, SIGKDD '11, pages 274–282, 2011.

[58] Ted E Senator, Henry G Goldberg, Alex Memory, William T Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, David A Bader, Edmond Chow, et al. Detecting insider threats in a real corporate database of computer usage activity. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1393–1401, 2013.

[59] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the IEEE Symposium on Security and Privacy*, S&P '10, pages 305–316, 2010.

[60] Tao Stein, Erdong Chen, and Karan Mangla. Facebook immune system. In *Proceedings of the ACM Workshop on Social Network Systems*, SNS '11, page 8, 2011.

[61] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. URL: https://arxiv.org/abs/1512.00567, arXiv:1512.00567.

[62] Flavio Toffalini, Ivan Homoliak, Athul Harilal, Alexander Binder, and Martin Ochoa. Detection of masqueraders based on graph partitioning of file system access events. In *IEEE Security and Privacy Workshops (SPW)*, pages 217–227. IEEE, 2018.

[63] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security*, AICS '17, 2017.

[64] Fangfang Yuan, Yanan Cao, Yanmin Shang, Yanbing Liu, Jianlong Tan, and Binxing Fang. Insider threat detection with deep neural network. In Yong Shi, Haohuan Fu, Yingjie Tian, Valeria V. Krzhizhanovskaya, Michael Harold Lees, Jack Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2018*, pages 43–54. Springer International Publishing, 2018.

[65] Fangfang Yuan, Yanmin Shang, Yanbing Liu, Yanan Cao, and Jianlong Tan. Data augmentation for insider threat detection with GAN. In *32nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 632–638. IEEE, 2020.

[66] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Qinghua Li. Insider threat detection via hierarchical neural temporal point processes. In *2019 IEEE International Conference on Big Data*, pages 1343–1350. IEEE, 2019.

[67] Jun Zengy, Xiang Wang, Jiahao Liu, Yinfang Chen, Zhenkai Liang, Tat-Seng Chua, and Zheng Leong Chua. Shadewatcher: Recommendation-guided cyber threat analysis using system audit records. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 489–506, 2022. doi:10.1109/SP46214.2022.9833669.

[68] Chunrui Zhang, Shen Wang, Dechen Zhan, Tingyue Yu, Tiangang Wang, and Mingyong Yin. Detecting insider threat from behavioral logs based on ensemble and self-supervised learning. *Security and Communication Networks*, 2021:1–11, 2021.

A Proof of Theorem 1

Proof. As $\mathbb{A}$ and $\mathbb{C}$ are assumed to be finite, the set of all possible models, $\mathcal{M}$, is $\mathbb{R}^{n_a \times n_c}$. Each model is parameterized by $n_a n_c$ real weights $\{w_{i,j}\}_{i\in\mathbb{A},j\in\mathbb{C}}$. Any classifier $f \in \mathcal{M}$ is of the form $f(a,c) = w_{a,c}$.

For generality, let $\alpha > 0$ be some weight assigned to synthetic positive instances. The risk of a given classifier $f \in \mathcal{M}$ is defined as:

$$R(f) = \sum_{a\in\mathbb{A},c\in\mathbb{C}} P_-(a,c)\ell(-f(a,c)) + \alpha P_+(a,c)\ell(f(a,c))$$

where P_-, P_+ respectively denote the probabilities of observing the pair as a negative and as a synthetic positive. By definition, we have $P_- = P$.

For P_+, notice that in the simplified version of positive sampling, the synthetic positive pairs are formed by randomly and independently drawing two natural action-context pairs (a,c) and (a',c') from the distribution P, resulting in the positive instance (a,c'). Hence, the positive distribution is the product of the marginal action and context distributions $P_+(a,c) = P(a)P(c)$.

Plugging in the model form, we get:

$$R(f) = \sum_{a\in\mathbb{A},c\in\mathbb{C}} P(a,c)\ell(-w_{a,c}) + \alpha P(a)P(c)\ell(w_{a,c})$$

Because of the simplicity of the model form, finding the model weights that minimize the risk is a separable optimization problem, by action-context pair. For a given pair (a,c), an optimal weight $w^*_{a,c}$ minimizes:

$$x \mapsto P(a,c)\ell(-x) + \alpha P(a)P(c)\ell(x)$$

Moreover, by convexity of the objective function, there exists a unique such weight, and the derivative of the above is such that:

$$-P(a,c)\ell'(-w^*_{a,c}) + \alpha P(a)P(c)\ell'(w^*_{a,c}) = 0$$

Rearranging the terms, we get:

$$\frac{\alpha\ell'(w^*_{a,c})}{\ell'(-w^*_{a,c})} = \frac{P(a,c)}{P(a)P(c)}$$

Finally, notice that the function

$$x \mapsto \frac{\alpha\ell'(x)}{\ell'(-x)}$$

is a ratio of negative strictly increasing and negative strictly decreasing functions, making this a positive strictly decreasing function. Therefore, letting f^* be the classifier which minimizes the risk, we have that for any two pairs (a,c) and (a',c'):

$$\begin{aligned}\frac{P(a,c)}{P(a)P(c)} > \frac{P(a',c')}{P(a')P(c')} &\Leftrightarrow \frac{\alpha\ell'(w^*_{a,c})}{\ell'(-w^*_{a,c})} > \frac{\alpha\ell'(w^*_{a',c'})}{\ell'(-w^*_{a',c'})} \\ &\Leftrightarrow w^*_{a,c} < w^*_{a',c'} \\ &\Leftrightarrow f^*(a,c) < f^*(a',c')\end{aligned}$$

□

B Choice of Loss Function

Having reduced the previously unsupervised problem into a supervised learning task, we turn our attention to the choice of loss function. While the task of scoring pairs of actions and contexts according to their affinity is drawn from contrastive learning [11], this application domain has specific issues to consider when designing an appropriate loss function.

We expect a small amount of label noise. Although Theorem 1 applies asymptotically, in practice, our training data is finite and our models do not have infinite capacity. The presence of labeling noise can, at best, delay training convergence and, at worst, introduce model bias [13]. Therefore, we avoid loss functions that place excessive emphasis on high-scoring negatives or low scoring positives. We prefer loss functions with a bounded gradient, ruling out the soft-nearest neighbors loss [17, 53] and the original contrastive loss [11]. Ideally, the loss function's emphasis on high scoring negatives would be parameterized. Label noise also restricts the applicability of hard instance mining [51].

Our goal is to identify the most anomalous action-context pair overall, without a priori fixing the action or the context. Therefore, scores must be commensurable among all context and action pairs, ruling out anchor-based loss functions like the triplet loss [56], as such loss functions may result in anchor-dependent score scales.

A final factor that influences our loss function design is the underlying data imbalance, with potentially orders of magnitude more synthetic positives than natural instances. Pairwise ranking loss functions behave well in such scenarios as they are, by definition, insensitive to such imbalance.

Taking into account the above requirements, we use the following pairwise loss function:

$$\mathcal{L}(B) = \left(\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{P}\sum_{j=1}^{P}\ell\left(h + \frac{y_j^+ - y_i^-}{s}\right)\right)^{\omega}\right)^{\frac{1}{\omega}}$$

Where $B = (\{y_i^-\}_{i=1}^N, \{y_j^+\}_{j=1}^P)$ are the N negative and P positive scores in a training minibatch and ℓ is a smooth pointwise loss function as in Theorem 1. The parameter $\omega \in \mathbb{R}^+$ controls the emphasis on high scoring negatives, and $s \in \mathbb{R}^+$ and $h \in \mathbb{R}^+$ are soft and hard margin parameters. We use the following Huber-like [30] expression for ℓ,

$$\ell(t) = \begin{cases} -t - \frac{1}{2} & \text{if } t < -1 \\ \frac{1}{2}t^2 & \text{if } t \in [-1;0] \\ 0 & \text{if } t > 0 \end{cases}$$

When $\omega = 1$, Theorem 2 in [19] shows that the loss is consistent with AUC minimization given our conditions on ℓ. In practice, we set ω using hyper parameter tuning, as per Appendix F.

C Shortcut Learning

Shortcut learning is a form of overfitting to our contrastive learning objective. It happens when the model learns to distinguish the natural pairs from synthetic pairs using information that is ultimately irrelevant to the security domain.

In Appendix D, we demonstrate a pathological featurization and model that by exploits the fact that the action and context principal are not equal for synthetic positive pairs. It can perfectly identify whether a given pair is natural or synthetic, but does not achieve the end goal of detecting malicious activity.

Rich featurization schemes increase the risk of inadvertent shortcuts. For instance, consider the action space of HTTP requests on internal websites, featurized with a bag-of-words scheme on the requests' parameter values and header data. It is extremely likely that these tokens contain some high-entropy, principal-identifying information that is irrelevant for behavioral analysis, for instance user-agent strings, authentication cookies, HTTP query parameters that directly identify the principal.

Shortcut features are not restricted to those that identify the principal. Shared temporal information between actions and contexts can also result in shortcut learning. For instance, if the action featurization contains the time of the event, and the context featurization contains the time at which the context

was computed, and provided context extraction is frequent enough, then positive sampling will focus the model on matching the action and context timestamps.

We take care to ensure that undesirable feature shortcuts between actions and contexts are not present. In Appendix E, we discuss an evaluation method that detect some instances of shortcut learning.

D Shortcut Learning Example

Unfortunately, we demonstrate that it is possible to achieve perfect separation with positive sampling, but not achieve the end goal of detecting malicious activity. Consider the following pathological model, f_s and featurization: Both actions and contexts are directly and exclusively featurized by the token of their principal, so that the labeled data set is:

$$\begin{aligned}\tilde{\mathcal{D}}_{\text{shortcut}} =&\{(p(a_i), p(c_i), -1)\}_i \\ &\cup\{(p(a_i), p(c_j), +1)\}_{i,j}\end{aligned}$$

where $p(x)$ denotes the username token of the principal associated with the action or context argument x.

Let *Users* be the set of distinct principal tokens $\{p_i | 0 \leq i < n\}$ and let $d \geq 2$ be the number of embedding dimensions. The model consists of a non-negative embedding matrix $w \in \mathbb{R}_+^{M \times d}$, where $M >> |Users|$. f_s hashes the principal token feature to select a row of w to represent the context and also for the action. It outputs their cosine distance, d_{cos}. Let h be a fixed pseudo-random function that maps principal tokens to $[0; M-1]$. The complete model is thus:

$$f(a,c) = d_{cos}(w_{h(p(a))}, w_{h(p(c))})$$

Even before any learning takes place, this model is already almost perfect from the standpoint of positive sampling. Provided any reasonable random initialization scheme for w,

1. any natural pair (a,c) is such that $f(a,c) = 0$, because both the action and context map to the same principal token, same embedding index and therefore same embedding, and

2. any synthetic positive pair (a',c') is such that $f(a',c') > 0$ with probability $1 - \frac{1}{M}$. This is because synthetic positives are by definition such that $p(a') \neq p(c')$, therefore the principal tokens map to different indexes with the above hash collision probability. The specific expected distance depends on the initialization scheme but increases with d.

While f almost perfectly separates natural from synthetic action-context pairs, it is also evident that f does not provide any information on whether a given document access is anomalous for a user. It uniformly scores 0 any actual access. In this scenario, we say that positive sampling is misaligned with the security application domain.

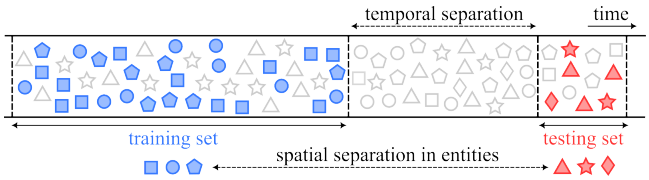

Fig. 10: Validation setup for tuning model architecture and hyperparameters. To properly assess model generalization performance in the presence of temporal distribution shift and of entities unseen at training time, we separate the training and testing sets in both time and "space" dimensions.

E Spatial-Temporal Splitting

In our experience, *spatially* and *temporally* splitting the training and validation sets is effective for revealing when a model has shortcut learning issues, as described in Section C.

Our validation set is constructed to be disjoint from the training set as follows: Temporally, the last event of training precedes the first event of testing set. Spatially, we randomly hold out 50% of the entities from the training set. This is achieved by dropping all events pertaining to either to held-out principal or to a held-out resource, using a deterministic pseudo random selection function applied to their identifier string. Usernames that were spatially held out are not present in the histories nor implicit social networks of actions and contexts in the training set. For the validation set, we only consider events where the user or the accessed resource was held-out from the training set. These "new" users and resources are featurized by the usernames that were *not* held out of the training set. Figure 10 summarizes the approach.

F Hyper Parameter Optimization

We make extensive use of Vizier [25] to select the various hyper-parameters present across our system. Vizier selects hyper parameters for f to maximize the AUC on a spatially and temporally split validation set.

After f is selected, a separate Vizier study selects hyper-parameters for common event filtering and the aggregator, g. It is tasked with retrieving a set of 100 synthetic 'attackers'. We sample a random principal, and for each action type, 0 to 33 of their actions. Those actions are interspersed amongst those of another random principal, which we call the synthetic 'attacker'. Vizier optimizes the mean inverse log rank of synthetic attackers. For computational efficiency, we set m, the number of principles required to filter an event, to 1 and have Vizier choose a minimum score threshold for each action type. Only actions from events with a higher score are presented to the aggregator, g.